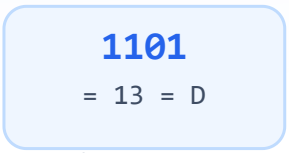
EES271 · DIGITAL CIRCUITS · SIIT

ติว MIDTERM

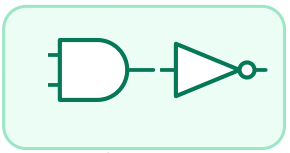
หนังสือติวกลางภาค
อ่านเล่มเดียวจบ ทุกเรื่องที่ออกสอบ

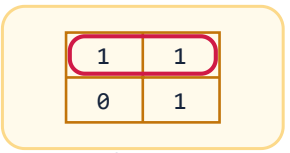
สอบวันที่ 23 ก.ย. 2026 · ขอบเขต Lecture 1–6

ทุกหัวข้อในสไลด์ Midterm Review ของอาจารย์ อยู่ในเล่มนี้ครบ ไม่ต้องเปิดเอกสารอื่นประกอบ

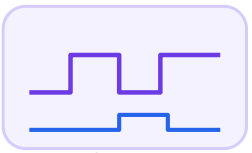


บทที่ 1–2 ตัวเลข·รหัส


บทที่ 3 บูลีน·เกต



บทที่ 4 K-Map


บทที่ 6 Timing

บทที่ 5 วงจรผสมที่เอาไปใช้งานจริง · ภาคผนวก เฉลย Assignment 1–6 · หน้าสรุปสูตร

เล่มนี้เขียนสำหรับคนที่ยังไม่เคยเรียนเรื่องนี้ ไม่ใช่คนทบทวน — ทุกขั้นตอนเดินให้ดูครบ ไม่ข้ามขั้น

เช็กลิสต์ขอบเขตสอบ

ทุกหัวข้อในสไลด์ Midterm Review → อยู่บทไหนของเล่มนี้

อ่านหน้านี้ก่อนหน้าอื่น

อาจารย์ใส่สไลด์ชื่อ `Midterm Review` ไว้ท้ายชุด **4 แผ่น** แล้วไล่หัวข้อที่ออกสอบไว้ทั้งหมด **65 หัวข้อย่อย** (นับทุกบูลเล็ตทุกระดับ) หน้านี้คือการเอาทั้ง 65 บรรทัดนั้นมาวางเรียง แล้วบอกว่าแต่ละบรรทัด**ไปอ่านที่บทไหนของเล่มนี้**ครับ

วิธีใช้: อ่านคอลัมน์กลางก่อน ถ้าอ่านแล้วตอบตัวเองได้ว่า "ข้อนี้ถ้าออกสอบ เราทำเป็น" ก็ขีดฆ่าทิ้ง · ถ้าตอบไม่ได้ ให้เปิดบทตามคอลัมน์ขวา **เล่มนี้มีครบทุกบรรทัด ไม่ต้องเปิดเอกสารอื่นประกอบ**

1 สไลด์แผ่นที่ 1 — ระบบเลขฐาน เลขคณิต คอมพลีเมนต์

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Numbering systems (positional numbering system)	ระบบเลขประจำหลัก — บอกได้ว่าเลขแต่ละหลักมี**น้ำหนัก**เท่าไร และเอาน้ำหนักคูณตัวเลขรวมกันเป็นค่าฐานสิบ	บทที่ 1
Binary (base 2) has 2 digits (0 or 1)	ฐานสอง ใช้เลข 0 กับ 1 · น้ำหนักหลักเป็น 2^n	บทที่ 1
Octal (base 8) has 8 digits (0-7)	ฐานแปด ใช้เลข 0–7 · ห้ามมีเลข 8 หรือ 9 โผล่มา	บทที่ 1
Decimal (base 10) has 10 digits (0-9)	ฐานสิบที่เราใช้ทุกวัน — ในข้อสอบคือ "ฐานกลาง" ที่ใช้แวะระหว่างแปลง	บทที่ 1
Hexadecimal (base 16) has 16 digits (0-9, A-F)	ฐานสิบหก · A=10 B=11 C=12 D=13 E=14 F=15 ต้องท่องได้โดยไม่ต้องนับนิ้ว	บทที่ 1
Fractional Numbers	เลขมีจุดทศนิยม — หลังจุดน้ำหนักเป็น $2^{-1}, 2^{-2}, \ldots$ · รวมถึง `floating point` ที่เก็บเลขเศษส่วนลงเครื่อง	บทที่ 1
Binary Arithmetics	บวก ลบ คูณ หาร ฐานสอง — ต้องเขียน**ตัวทดและตัวยืมทุกหลัก**ให้เห็น ไม่ใช่คิดในใจ	บทที่ 1
Base conversion	การแปลงฐาน — หัวข้อแม่ของ 5 บรรทัดถัดไป	บทที่ 1
Decimal number to base *r*	ฐานสิบ → ฐานอื่น · ใช้**หารสั้นเก็บเศษ** แล้วอ่านเศษจากล่างขึ้นบน	บทที่ 1
Whole numbers, Fractional Numbers	ส่วนหน้าจุดใช้หาร · ส่วนหลังจุดใช้**คูณเก็บตัวทด** แล้วอ่านจากบนลงล่าง — คนละทิศทางกัน ข้อสอบชอบดักตรงนี้	บทที่ 1

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Binary to/from octal or hexadecimal numbers	ฐานสอง ↔ ฐานแปด/สิบหก · จับกลุ่มบิต ไม่ต้องแวะฐานสิบ	บทที่ 1
3 binary digits = 1 octal digit	จับกลุ่มละ 3 บิต จากจุดออกไปทั้งสองทาง เติมศูนย์ให้ครบกลุ่ม	บทที่ 1
4 binary digits = 1 hexadecimal digit	จับกลุ่มละ 4 บิต วิธีเดียวกัน	บทที่ 1
Complements and signed numbers	คอมพลีเมนต์กับเลขมีเครื่องหมาย — หัวข้อแม่ของ 2 บรรทัดถัดไป	บทที่ 1
Sign-Magnitude Form, 1's complement form, 2's complement form	สามวิธีเก็บเลขติดลบ · **บิตซ้ายสุดคือเครื่องหมาย**ทั้งสามวิธี แต่ตีความส่วนที่เหลือไม่เหมือนกัน	บทที่ 1
Range of signed numbers	พิสัย — n บิตแบบ 2's complement เก็บได้ -2^{n-1} ถึง $+2^{n-1}-1$	บทที่ 1

สไลด์แผ่นนี้ **16 หัวข้อย่อย** → **บทที่ 1** ทั้งแผ่น

2 สไลด์แผ่นที่ 2 — รหัส การตรวจจับข้อผิดพลาด เกต และบูลีน

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Signed number arithmetic	บวกลบเลขมีเครื่องหมาย · ต้องเช็ก `overflow` ได้ว่าผลลัพธ์ล้นพิสัยหรือยัง	บทที่ 1
Binary-Coded-Decimal (BCD)	เก็บเลขฐานสิบทีละหลักด้วย 4 บิต · รหัส 1010–1111 **ใช้ไม่ได้** (invalid)	บทที่ 2
Gray code	รหัสเกรย์ — เลขที่ติดกันต่างกัน**บิตเดียว** ใช้กับเซนเซอร์หมุนเพื่อกันอ่านผิดตอนเปลี่ยนค่า	บทที่ 2
Binary-Gray code conversion	แปลงไบนารี ↔ เกรย์ · ไปใช้ XOR คู่ติดกัน · กลับใช้ XOR สะสม	บทที่ 2 (วงจรอยู่บทที่ 5)
Error detection, error correction codes	รหัสตรวจจับ/แก้ข้อผิดพลาด — หัวข้อแม่ของบรรทัดถัดไป	บทที่ 2
Even parity, Odd parity, CRC and **Hamming** codes	พาริตีคู่/คี่ (รู้ว่าผิดแต่ไม่รู้ผิดตรงไหน) · CRC (หารด้วยพหุนามเหลือศูนย์ = ผ่าน) · แฮมมิง (**บอกได้ว่าบิตไหนผิดและแก้ได้**)	บทที่ 2
Logic gate	ลอจิกเกต — หัวข้อแม่ของ 6 บรรทัดถัดไป	บทที่ 3
NOT, AND, OR, NAND, NOR, XOR, and XNOR	เกต 7 ตัว · ต้องวาดสัญลักษณ์ เขียนตารางค่าความจริง และเขียนนิพจน์ได้ทั้งสามอย่างจากความจำ	บทที่ 3
Opposing logic, Dual symbols	ตรรกะตรงข้าม (AND↔NAND · OR↔NOR · XOR↔XNOR) และ **สัญลักษณ์คู่** — เกตตัวเดียวกันวาดได้ 2 หน้าตา	บทที่ 3
Logic gates can be analyzed in three ways:	มองเกตได้ 3 ทาง และ**ทั้งสามทางบอกเรื่องเดียวกัน** ข้อสอบชอบให้ทางหนึ่งแล้วถามอีกทาง	บทที่ 3
Boolean expression	นิพจน์บูลีน — สูตรบรรทัดเดียว เช่น $X = AB$	บทที่ 3
Truth table	ตารางค่าความจริง — ไล่ครบ 2^n แถว	บทที่ 3
Timing diagrams	ไดอะแกรมเวลา — รูปคลื่นขาเข้า/ขาออกเทียบกับเวลา **นี่คือวิธีที่คนอ่านไม่เป็นเยอะที่สุด**	**บทที่ 6**
Simplification of Boolean Algebra	การย่อนิพจน์ — หัวข้อแม่ของ 3 บรรทัดถัดไป	บทที่ 3
Basic Boolean algebra laws	กฎพื้นฐาน commutative · associative · distributive · identity · complementary · idempotent · involution · null · absorption	บทที่ 3
De Morgan's Laws: Break the bar and change the sign!	เดอมอร์แกน — **หักขีดบน แล้วสลับเครื่องหมาย** · $\overline{XY} = \overline{X} + \overline{Y}$ และ $\overline{X+Y} = \overline{X}\,\overline{Y}$	บทที่ 3

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Consensus Theorem	ทฤษฎีบทคอนเซนซัส — $AB + \overline{A}C + BC = AB + \overline{A}C$ ตัดพจน์ที่ไม่ได้ทำอะไรทิ้ง	บทที่ 3

สไลด์แผ่นนี้ **17 หัวข้อย่อย** → บทที่ 1 = 1 · **บทที่ 2 = 5** · **บทที่ 3 = 10** · **บทที่ 6 = 1**

3 สไลด์แผ่นที่ 3 — มินเทอม SOP/POS วิเคราะห์วงจร และ K-Map

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Minterms (standard product terms)	มินเทอม — พจน์คูณที่มี**ตัวแปรครบทุกตัว** และเป็น 1 ได้แค่แถวเดียวในตาราง	บทที่ 3
Literal and product term	`literal` = ตัวแปรหนึ่งตัว (มีขีดบนหรือไม่ก็ได้) · `product term` = literal คูณกัน	บทที่ 3
Minterm numbers, Minterm notations	เลขประจำมินเทอม $m_0, m_1, \ldots$ และการเขียนย่อ $\sum m(1,3,5)$	บทที่ 3
Truth table to/from SOP and POS Boolean expression conversions	ตาราง ↔ SOP/POS · **SOP อ่านจากแถวที่ออก 1 · POS อ่านจากแถวที่ออก 0 และต้องกลับค่าตัวแปร**	บทที่ 3
Combinational logic analysis	วิเคราะห์วงจรผสม — ไล่จากขาเข้าไปขาออก ติดป้ายทุกจุดเชื่อม แล้วได้นิพจน์	บทที่ 3
NOT gate conversions	ย้าย/เพิ่ม/ตัดตัวกลับค่า · ใส่ NOT สองตัวติดกันแล้วค่าไม่เปลี่ยน (ใช้แปลงวงจรได้)	บทที่ 3
Universal gates (NAND and NOR)	เกตเอนกประสงค์ — ใช้ NAND อย่างเดียว หรือ NOR อย่างเดียว สร้างเกตอื่นได้ครบทุกตัว	บทที่ 3
Using dual symbols to simplify logic diagram	เปลี่ยนเกตเป็นสัญลักษณ์คู่ของมัน เพื่อให้วงจรอ่านง่ายขึ้นโดยค่าไม่เปลี่ยน	บทที่ 3
Bubble-to-bubble or nonbubble-to-nonbubble connections	**จุดกลม-ต่อ-จุดกลม** = ตัวกลับค่าสองตัวหักล้างกัน วงจรอ่านตรง ๆ ได้ทันที	บทที่ 3
Karnaugh Map (K-Map)	แผนผังคาร์โนห์ — ย่อนิพจน์ด้วยการมอง ไม่ต้องใช้กฎบูลีนทีละข้อ	**บทที่ 4**
Truth table or Boolean expression to/from K-Map conversion	ลงตาราง/นิพจน์ ↔ K-Map · หัวตารางต้องเรียงแบบ**เกรย์** (00 01 11 10) ไม่ใช่ 00 01 10 11	**บทที่ 4**
Grouping rules in K-Map	กฎจับกลุ่ม — กลุ่มต้องเป็นกำลังสอง (1,2,4,8) · ต่อขอบซ้าย-ขวา บน-ล่างได้ · กลุ่มยิ่งใหญ่ยิ่งดี · ซ้อนทับกันได้	**บทที่ 4**
Minimum SOP and POS expression using K-Map	อ่านคำตอบสั้นสุด · **จับ 1 ได้ SOP · จับ 0 ได้ POS**	**บทที่ 4**

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
K-Map tricks: "look left, look up, look for variable that do not change."	ประโยคกล — **มองซ้าย มองบน หาตัวแปรที่ไม่เปลี่ยนค่า** ตัวไหนเปลี่ยนค่าภายในกลุ่มให้ตัดทิ้ง	**บทที่ 4**
Don't care conditions **X**	ช่อง X = จะเป็น 0 หรือ 1 ก็ได้ · เลือกให้เป็น 1 เมื่อช่วยขยายกลุ่ม ไม่งั้นทิ้ง	**บทที่ 4**

สไลด์แผ่นนี้ **15 หัวข้อย่อย** → **บทที่ 3 = 9** · **บทที่ 4 = 6**

4 สไลด์แผ่นที่ 4 — วงจรผสมที่เอาไปใช้งานจริง

ทั้งแผ่นนี้เป็น **บทที่ 5** บทเดียว และเป็นแผ่นที่มีหัวข้อย่อยเยอะที่สุดพอ ๆ กับแผ่นที่ 2 แปลว่าอาจารย์ให้น้ำหนักกับ **วงจรที่เอาไปใช้งานจริง**สูงมากครับ

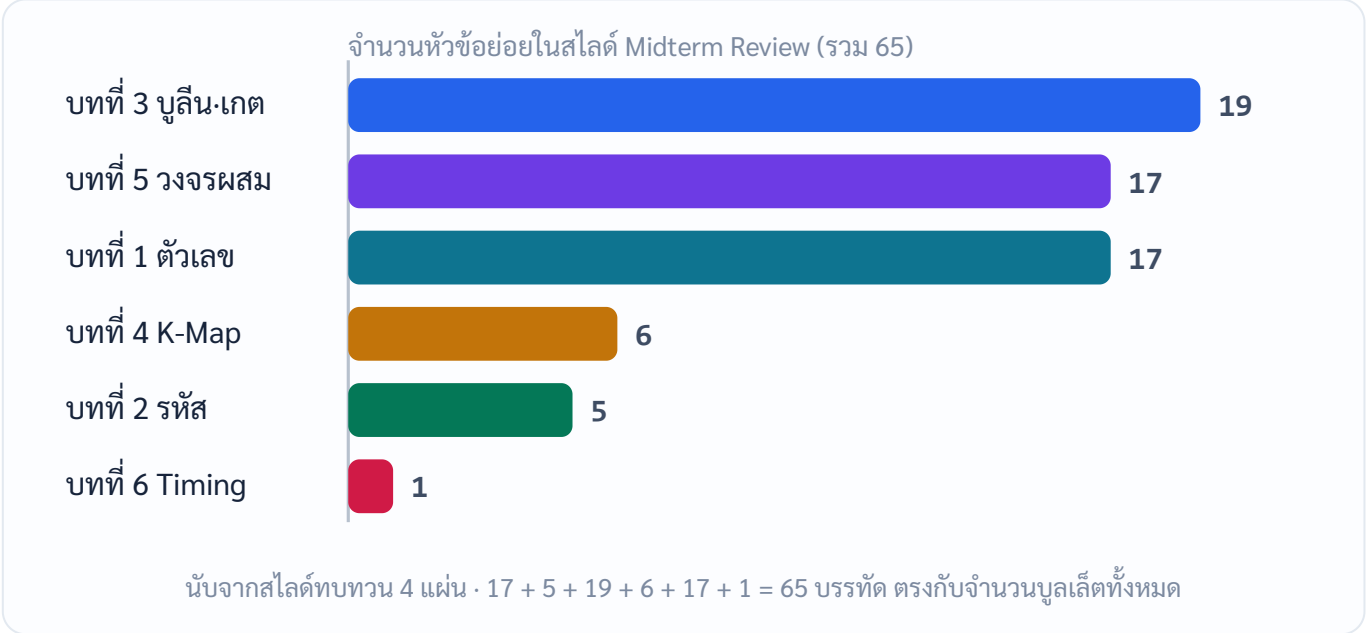
หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Applications of Combinational Logic	การใช้งานวงจรผสม — หัวข้อแม่ของทั้งแผ่น	บทที่ 5
Adders: half-adder, full-adder	วงจรบวก · **ครึ่ง**รับ 2 บิต ไม่มีตัวทดเข้า · **เต็ม**รับ 3 บิต มีตัวทดเข้า — ต้องเขียนสมการทั้งสองได้	บทที่ 5
Parallel binary adder	เอา full-adder มาต่อกันหลายตัวเพื่อบวกเลขหลายบิตพร้อมกัน	บทที่ 5
Ripple carry	ตัวทดไหลต่อกันเป็นทอด ๆ — ถูกและง่าย แต่**ช้า** เพราะหลักท้ายต้องรอหลักหน้า	บทที่ 5
Look-ahead carry: **carry generation, carry propagation**, output carry	ตัวทดแบบมองล่วงหน้า · $G = AB$ (สร้างตัวทดเอง) · $P = A + B$ (ส่งตัวทดผ่าน) · $C_{out} = G + PC_{in}$	บทที่ 5
Subtractors: half-subtractor, full-subtractor	วงจรลบ · ผลต่างใช้ XOR เหมือนวงจรบวก แต่**ตัวยืมไม่เหมือนตัวทด** — ตรงนี้คือกับดัก	บทที่ 5
Comparators	วงจรเปรียบเทียบ · เท่ากันใช้ XNOR · มากกว่า/น้อยกว่าไล่จากบิตซ้ายสุดลงมา	บทที่ 5
Encoders	ตัวเข้ารหัส — สายเข้าหลายเส้น ติด 1 เส้น ออกมาเป็นเลขประจำเส้นนั้น	บทที่ 5
Decimal-to-BCD encoder	ปุ่ม 0–9 → รหัส BCD 4 บิต	บทที่ 5
8-to-3 encoder	สายเข้า 8 เส้น → เลขฐานสอง 3 บิต	บทที่ 5
Decoders	ตัวถอดรหัส — ทำกลับทางกับ encoder · รับเลขเข้า ปลุกสายออก 1 เส้น	บทที่ 5
BCD-to-Decimal Decoder	รหัส BCD 4 บิต → สายออก 10 เส้น	บทที่ 5
3-to-8 Decoder	3 บิต → 8 เส้น · แต่ละเส้นคือ**มินเทอมหนึ่งตัว** (เชื่อมกลับบทที่ 3)	บทที่ 5
7-Segment LED Displays	จอเจ็ดขีด a–g ที่ใช้โชว์ตัวเลข	บทที่ 5
BCD-to-7-Segment Decoder	รหัส BCD → ขีดไหนติดบ้าง เพื่อให้เห็นเป็นเลข 0–9	บทที่ 5
Code converter: binary-gray conversion	วงจรแปลงรหัส — ไบนารี ↔ เกรย์ ด้วย XOR เรียงกัน (สูตรอยู่บทที่ 2 วงจรอยู่บทที่ 5)	บทที่ 5

หัวข้อตามสไลด์ (คำของอาจารย์)	ภาษาไทย — ต้องทำอะไรได้ถึงเรียกว่าผ่าน	อ่านที่ไหน
Multiplexer (MUX), Demultiplexer (DEMUX)	ตัวเลือกสาย · MUX = หลายเข้า 1 ออก (สวิตช์เลือกช่อง) · DEMUX = 1 เข้า หลายออก	บทที่ 5

สไลด์แผ่นนี้ **17 หัวข้อย่อย** → **บทที่ 5** ทั้งแผ่น

5 ถ้าเหลือเวลาน้อย อ่านตามลำดับนี้

สไลด์ทบทวนทั้ง 4 แผ่นรวมกันได้ **65 หัวข้อย่อย** ครับ ถ้าเอา 65 บรรทัดนั้นมานับว่าตกอยู่บทไหนกี่บรรทัด จะได้รูปข้างล่างนี้ — **ความยาวของแท่ง = จำนวนบรรทัดในสไลด์ที่บทนั้นรับผิดชอบ** ไม่ใช่การเดาว่าอะไรน่าจะออก แต่เป็นการนับสิ่งที่อาจารย์เขียนไว้เองครับ



1 **บทที่ 3 · พีชคณิตบูลีน มินเทอม และวงจร — 19 บรรทัด**
หนักที่สุด และยัง**เป็นฐานของบทที่ 4 กับบทที่ 5 ด้วย** ถ้าอ่านได้บทเดียว อ่านบทนี้ครับ

2 **บทที่ 5 · วงจรผสมที่เอาไปใช้งานจริง — 17 บรรทัด**
เสมอกับบทที่ 1 ที่ 17 บรรทัด แต่พี่ให้มาก่อน เพราะ 17 บรรทัดของบทนี้คือ**ชื่อวงจรเฉพาะตัว 17 ชื่อ** แต่ละชื่อกลายเป็นโจทย์ได้ 1 ข้อทันที ขณะที่ของบทที่ 1 มี 5 บรรทัดที่เป็นแค่นิยามว่าแต่ละฐานใช้เลขอะไรบ้าง

3 **บทที่ 1 · ตัวเลขในเครื่อง — 17 บรรทัด**
เยอะเท่ากัน แต่รวมนิยามพื้น ๆ ไว้ด้วย · ส่วนที่กินเวลาจริงคือ**คอมพลีเมนต์ พิสัย และ overflow**

4 **บทที่ 4 · K-Map — 6 บรรทัด**
บรรทัดน้อยแต่**ทำคะแนนได้แน่นอนที่สุด** เพราะมีขั้นตอนตายตัว · ต้องอ่านบทที่ 3 เรื่อง SOP/POS มาก่อน ไม่งั้นจับกลุ่มเสร็จก็อ่านคำตอบไม่ออก

5 **บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด — 5 บรรทัด**
BCD กับเกรย์ใช้เวลาน้อยมาก · ส่วนที่กินเวลาคือ **CRC กับแฮมมิง** ซึ่งเป็นขั้นตอนท่องจำล้วน ๆ

6

บทที่ 6 · Timing diagram — 1 บรรทัด

นับได้บรรทัดเดียว แต่**อย่าตัดทิ้ง** — อาจารย์ระบุชื่อไว้ตรง ๆ ว่าเป็น 1 ใน 3 วิธีวิเคราะห์เกต และบทนี้อ่านจบได้ในเวลาสั้นที่สุดของทั้งเล่ม คุ้มที่สุดต่อเวลาที่เสียไปครับ

อย่าอ่านสลับลำดับนี้

ลำดับข้างบนเรียงตาม**น้ำหนักในสไลด์** แต่มีข้อบังคับหนึ่งข้อทับอยู่ครับ: **บทที่ 4 และบทที่ 5 ใช้ของจากบทที่ 3** (เกต 7 ตัว · SOP/POS · มินเทอม) ถ้ายังไม่ได้อ่านบทที่ 3 แล้วกระโดดไปบทที่ 5 จะเจอคำว่า "XOR" กับ "มินเทอม" ในหน้าแรกแล้วไปต่อไม่ได้ เพราะฉะนั้น **บทที่ 3 ต้องมาก่อนเสมอ** ส่วนบทที่เหลือสลับได้ตามใจ

ภาคผนวก

เฉลย Assignment 1–6 ทุกข้อ อยู่ท้ายเล่ม และ**หน้าสรุปสูตร**อยู่หน้าสุดท้าย ทั้งสองส่วนไม่ได้เป็นหัวข้อในสไลด์ทบทวน แต่เป็นของที่ควรกวาดตาก่อนเข้าห้องสอบ 15 นาทีสุดท้ายครับ

บทที่ 1 · ตัวเลขในเครื่อง

บทนี้ตอบคำถามเดียว: **เครื่องจักรที่มีแต่สวิตช์ปิด-เปิด เก็บ "จำนวน" ไว้ได้ยังไง** — แล้วเราจะอ่านมันกลับออกมาเป็นเลขที่เราคุ้นเคยได้ยังไง

น้องยังไม่ต้องรู้อะไรมาก่อนเลย ไม่ต้องรู้ว่าฐานสองคืออะไร ไม่ต้องเคยเห็นคำว่าคอมพลีเมนต์ เราจะเริ่มจากแถวสวิตช์ไฟบนผนังจริง ๆ แล้วค่อย ๆ เดินไปทีละขั้นจนถึงเรื่องที่ข้อสอบกลางภาคถาม

บทนี้มีอะไรบ้าง

① ทำไมเครื่องนับด้วยเลข 2 ② อ่าน-เขียนฐานสอง ฐานแปด ฐานสิบหก ③ เลขมีจุด (เศษส่วน) ④ บวก ลบ คูณ หาร ในฐานสอง ⑤ คอมพลีเมนต์ ⑥ เลขติดลบสามแบบและพิสัยของมัน ⑦ บวกเลขมีเครื่องหมายและการล้น ⑧ ทศนิยมลอยตัว ⑨ เช็กลิสต์ก่อนเข้าห้องสอบ

1 ทำไมเครื่องถึงนับด้วยเลข 2

แถวสวิตช์บนผนัง

ลองนึกถึงผนังห้องแลบที่น้องเพิ่งไปเรียนมา บนผนังมีสวิตช์ไฟเรียงกัน 4 ตัว แต่ละตัวทำได้แค่สองอย่างเท่านั้นคือ **ปิด** กับ **เปิด** ไม่มี "เปิดครึ่งเดียว" ไม่มี "เปิดสามในสี่"

นี่คือข้อจำกัดของเครื่องทุกเครื่องที่เราจะเรียนในวิชานี้ ในสายไฟเส้นหนึ่งมันมีได้แค่ไฟแรงกับไฟอ่อน จะสั่งให้มันเก็บ "ค่ากลาง ๆ" ไม่ได้ เพราะไฟในสายมันไหวขึ้นลงตลอดเวลา ค่ากลางที่ตั้งไว้วันนี้พรุ่งนี้ก็เพี้ยน

เพราะฉะนั้นคำถามของเราคือ ในเมื่อแต่ละตัวมีได้แค่สองอย่าง เราจะทำให้ **แถว** ของมันแทนจำนวนได้ยังไง

ลองก่อน — คิดในหัวพอ ไม่ต้องเขียน

สวิตช์ 4 ตัว แต่ละตัวปิดหรือเปิดก็ได้ ทั้งแถวจะจัดได้กี่แบบที่ไม่ซ้ำกัน

ตอบ 16 ใช่ไหม — ตัวแรกเลือกได้ 2 ทาง ตัวที่สองก็ 2 ทางต่อจากนั้น ได้ 2×2 = 4 พอครบ 4 ตัวก็เป็น 2×2×2×2 = 16 แบบ

ที่น้องเพิ่งทำนั่นแหละคือกฎสำคัญข้อแรกของบทนี้ — **สวิตช์ n ตัว จัดได้ 2^n แบบ** สวิตช์ 8 ตัวจึงได้ 256 แบบ สวิตช์ 16 ตัวได้ 65,536 แบบ

สวิตช์หนึ่งตัวในภาษาเรามีชื่อเรียกว่า `bit` อ่านว่า **บิต** ย่อมาจาก `binary digit` แปลตรงตัวว่า "ตัวเลขที่มีสองค่า" ต่อจากนี้เราจะเรียกสวิตช์หนึ่งตัวว่าหนึ่งบิต และเขียนสถานะปิดเป็น **0** เปิดเป็น **1**

อ่านจำนวนออกมาจากแถวสวิตช์

รู้ว่ามี 16 แบบแล้ว แต่ยังไม่พอ เราต้องตกลงกันด้วยว่า **แบบไหนแปลว่าเลขอะไร** ถ้าไม่ตกลงกัน คนหนึ่งอ่านแถวหนึ่งได้ 5 อีกคนอ่านได้ 12 ก็จบกัน

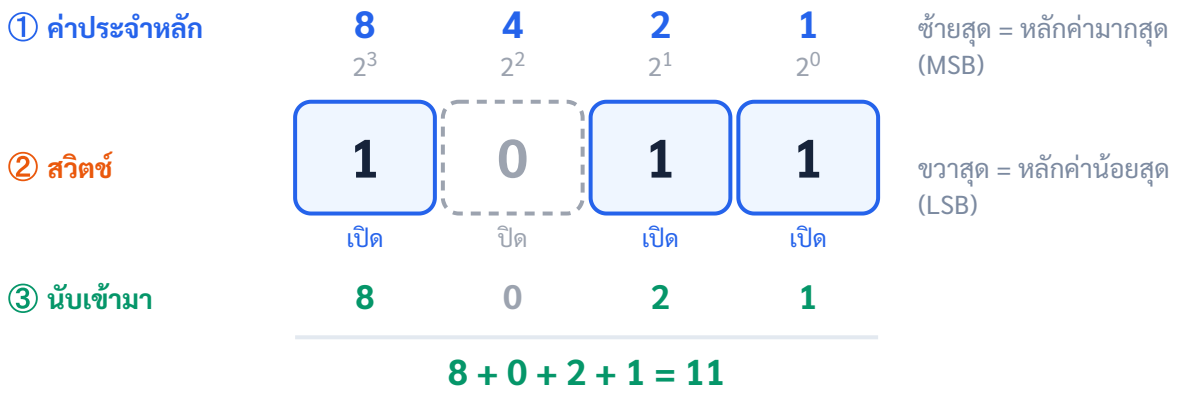
วิธีที่โลกตกลงกันคือ ให้สวิตช์แต่ละตำแหน่ง **มีค่าประจำตัวไม่เท่ากัน** นับจากขวาสุดไปซ้าย ค่าประจำตัวคือ 1, 2, 4, 8 — คือคูณสองขึ้นไปเรื่อย ๆ แล้วถ้าสวิตช์ตัวไหนเปิดอยู่ ก็เอาค่าประจำตัวของมันมาบวกรวมกัน

ลองก่อน

สวิตช์เรียงจากซ้ายไปขวาเป็น **เปิด ปิด เปิด เปิด** ค่าประจำตัวคือ 8 4 2 1 ตามลำดับ แถวนี้แปลว่าเลขอะไร

ตอบ 11 ใช่ไหม — ตัวที่เปิดคือค่า 8, 2 และ 1 บวกกันได้ 8 + 2 + 1 = 11 ส่วนตัวที่ปิด (ค่า 4) ไม่นับ

ที่น้องเพิ่งทำมีชื่อว่า **การอ่านเลขฐานสอง** (binary) แถวสวิตช์นั้นเราเขียนแทนด้วย **1011** แล้วห้อยเลข 2 ไว้บอกว่าอ่านแบบฐานสอง เป็น 1011_2 ซึ่งมีค่าเท่ากับ 11 ในเลขที่เราใช้กันทุกวัน



รูป 1.1 · อ่าน 1011_2 ออกมาเป็น 11 — ① วางค่าประจำหลักก่อน ② ดูว่าหลักไหนเป็น 1 ③ บวกเฉพาะหลักที่เป็น 1

สองคำที่จะใช้ต่อไปทั้งบท: หลักขวาสุดค่าน้อยที่สุด เรียกว่า LSB (least significant bit) และหลักซ้ายสุดค่ามากที่สุด เรียกว่า MSB (most significant bit) เวลาโจทย์บอกว่า "เริ่มจาก LSB" แปลว่าให้เริ่มจากตัวขวาสุด

ทำไมต้องเป็น 1 2 4 8 — ไม่ใช่ 1 2 3 4

ตรงนี้สำคัญ เพราะถ้าน้องเข้าใจเหตุผล น้องจะไม่ต้องท่องอะไรอีกเลยในบทนี้

เหตุผลคือ **หลักใหม่ต้องมีค่ามากกว่าทุกหลักทางขวารวมกัน** ดูสิ 1 + 2 = 3 หลักถัดไปจึงต้องเป็น 4 (มากกว่า 3 อยู่หนึ่ง) แล้ว 1 + 2 + 4 = 7 หลักถัดไปจึงเป็น 8 ถ้าเป็นแบบนี้ ทุกจำนวนตั้งแต่ 0 ถึง 15 จะเขียนได้แบบเดียวเท่านั้น ไม่ซ้ำ ไม่ขาด

ลองทำผิดวิธีให้ดู สมมติเราดื้อ ใช้ค่าประจำหลัก 4 3 2 1 แทน

แถวสวิตช์	ค่าประจำหลัก 4 3 2 1	อ่านได้
0 1 0 1	0 + 3 + 0 + 1	4
1 0 0 0	4 + 0 + 0 + 0	4
0 1 1 0	0 + 3 + 2 + 0	5
1 0 0 1	4 + 0 + 0 + 1	5

เห็นไหมว่าพัง — สองแถวที่ต่างกันอ่านได้เลขเดียวกัน ถ้าเครื่องเก็บเลข 4 ไว้ แล้วอีกวงจรหนึ่งมาอ่าน มันจะไม่มีทางรู้ว่าเจ้าของตั้งใจเขียนแบบไหน และเลขสูงสุดที่เขียนได้ก็เหลือแค่ 4+3+2+1 = 10 ทั้งที่มีตั้ง 16 แบบให้ใช้ เท่ากับเสียของไปเปล่า ๆ

จำไว้

ค่าประจำหลักต้องเป็น **กำลังของฐาน** เสมอ — ฐานสองก็ $2^0, 2^1, 2^2, \ldots$ ฐานสิบก็ $10^0, 10^1, 10^2, \ldots$ ไม่มีระบบเลขไหนในโลกที่ใช้ค่าประจำหลักมั่ว ๆ

กฎเดียวกันนี้ใช้กับฐานอะไรก็ได้

ที่จริงน้องใช้กฎนี้มาตั้งแต่ประถมแล้ว แค่ไม่มีใครบอกชื่อมัน เวลาน้องเขียน 6031 น้องหมายถึง 6 พัน 0 ร้อย 3 สิบ 1 หน่วย ใช่ไหม นั่นคือ 6×1000 + 0×100 + 3×10 + 1×1 = 6031 พอดี ค่าประจำหลักคือ $10^3, 10^2, 10^1, 10^0$

เลขฐานสิบใช้ตัวเลข 10 ตัว (0–9) ค่าประจำหลักจึงคูณสิบขึ้นไปทีละหลัก · เลขฐานสองใช้ตัวเลข 2 ตัว (0–1) ค่าประจำหลักจึงคูณสองขึ้นไปทีละหลัก · เห็นแบบแผนไหม **ใช้ตัวเลขกี่ตัว ค่าประจำหลักก็คูณเท่านั้น**

เขียนเป็นสูตรกลางที่ใช้ได้ทุกฐาน (อาจารย์ใช้สูตรนี้ทั้งคอร์ส):

สูตรกลาง

$$N = a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + \cdots + a_2r^2 + a_1r^1 + a_0r^0$$

โดย r คือ **ฐาน** (อีกชื่อคือ radix) · n คือจำนวนหลัก · a_i คือตัวเลขในหลักที่ i ซึ่งต้องอยู่ในช่วง $0 \le a_i < r$ เสมอ

ข้อความ "$0 \le a_i < r$" มีความหมายจริง ๆ นะ ไม่ได้เขียนไว้ประดับ มันแปลว่า **ฐานสองห้ามมีเลข 2** ฐานแปดห้ามมีเลข 8 ถ้าโจทย์ให้ 128_8 มา แสดงว่าโจทย์พิมพ์ผิด หรือไม่ก็เป็นข้อหลอก

ทำให้ดูทุกบรรทัด

โจทย์ 1.1 — แปลง 11101_2 เป็นฐานสิบ

ทำอะไร	บรรทัด
เขียนเลขออกมาทีละหลัก	1 1 1 0 1
วางค่าประจำหลักจากขวาไปซ้าย	16 8 4 2 1
คูณทีละหลัก	1×16 1×8 1×4 0×2 1×1
ได้	16 8 4 0 1

ทำอะไร	บรรทัด
บวกรวม	16 + 8 + 4 + 0 + 1 = **29**

คำตอบ

$11101_2 = 29_{10}$

เช็กกลับ — 29 น้อยกว่า 32 (2^5) แต่มากกว่า 16 (2^4) เพราะฉะนั้นมันต้องใช้ 5 หลักพอดี และหลักซ้ายสุดต้องเป็น 1 · ตรงกับที่โจทย์ให้มา

ข้อต่อไปเป็นเลขที่มีศูนย์ติดกันหลายตัว ลองดูว่าทำให้เร็วขึ้นได้ไหม

โจทย์ 1.2 แปลง 101000_2 เป็นฐานสิบ

ทำอะไร	บรรทัด
เขียนเลขออกมาทีละหลัก	**1 0 1 0 0 0**
วางค่าประจำหลัก	32 16 8 4 2 1
หยิบเฉพาะหลักที่เป็น 1	32 และ 8
บวกรวม	32 + 8 = **40**

คำตอบ

$101000_2 = 40_{10}$

เช็กกลับ — ท้ายมีศูนย์ติดกัน 3 ตัว แปลว่าหารด้วย 8 ลงตัว · 40 ÷ 8 = 5 ลงตัวจริง

สองข้อนี้ทำเหมือนกันทุกขั้น เพราะฉะนั้นสรุปเป็นลำดับที่ทำตามได้เลย

เช็กลิสต์หัวข้อ 1

1. เขียนเลขฐานสองออกมา แล้ว**วางค่าประจำหลักไว้เหนือทุกหลัก** เริ่มจากขวาสุดเป็น 1 แล้วคูณสองไปทางซ้าย
2. ขีดฆ่าหลักที่เป็น 0 ทิ้งไปเลย ไม่ต้องคิด
3. บวกค่าประจำหลักของตัวที่เหลือ
4. **หยุดตรงนี้** ได้คำตอบเป็นฐานสิบแล้ว
5. เช็กกลับด้วยจำนวนหลัก: เลข n หลักฐานสอง ต้องได้ค่าระหว่าง 2^{n-1} ถึง $2^n - 1$

2 อ่าน-เขียนฐานแปดกับฐานสิบหก

ปัญหาที่ทำให้ต้องมีฐานแปดกับฐานสิบหก

สมมติน้องทำแลบอยู่ แล้วต้องจดเลขที่อยู่ของหน่วยความจำลงสมุด เลขนั้นคือ

1011111001100011

ลองก่อน

ปิดหนังสือแล้วลองจดเลขข้างบนลงกระดาษดู แล้วเปิดกลับมาเทียบทีละตัว น้องคิดว่าจะจดถูกครบทั้ง 16 ตัวไหม

ส่วนใหญ่พลาดใช่ไหม — ปัญหาไม่ได้อยู่ที่เลขฐานสองผิด แต่อยู่ที่**ตาของคนนับ 1 กับ 0 ติดกันยาว ๆ ไม่ไหว** เราเลยต้องหาวิธีเขียนเลขตัวเดิมนี้ให้สั้นลง

ทางแก้คือ **จับบิตเป็นกลุ่ม แล้วตั้งชื่อย่อให้แต่ละกลุ่ม** เลขตัวเดิมเป๊ะ ๆ ไม่ได้เปลี่ยนค่าเลย แค่เขียนคนละหน้าตา เหมือนเขียนเบอร์โทร 0812345678 เป็น 081-234-5678 ค่าเท่ากัน แต่จดง่ายกว่ากันเยอะ

ฐานแปดกับฐานสิบหกคืออะไร

ฐานแปด (`octal`) ใช้ตัวเลข 8 ตัวคือ 0 1 2 3 4 5 6 7 · ค่าประจำหลักคือ 1, 8, 64, 512 คือคูณแปดขึ้นไปทีละหลัก ตามสูตรกลางในหัวข้อ 1 โดยให้ $r = 8$

ฐานสิบหก (`hexadecimal` เรียกสั้น ๆ ว่า เฮกซ์) ใช้ตัวเลข 16 ตัว แต่เลขอารบิกมีแค่ 0–9 ไม่พอ เลยยืมตัวอักษร A B C D E F มาใช้แทนค่า 10 11 12 13 14 15 · ค่าประจำหลักคือ 1, 16, 256, 4096

A–F ไม่ใช่ตัวอักษร

ในเลขฐานสิบหก **A แปลว่าสิบ** ไม่ใช่ตัว A · C แปลว่าสิบสอง · F แปลว่าสิบห้า ถ้าเห็น F8A ให้อ่านในใจว่า "สิบ ห้า แปด สิบ"

ฐานสิบ	ฐานสอง 3 บิต	ฐานสอง 4 บิต	ฐานแปด	ฐานสิบหก
0	000	0000	0	0
1	001	0001	1	1
2	010	0010	2	2
3	011	0011	3	3
4	100	0100	4	4
5	101	0101	5	5
6	110	0110	6	6
7	111	0111	7	7
8	—	1000	10	8
9	—	1001	11	9
10	—	1010	12	A
11	—	1011	13	B
12	—	1100	14	C
13	—	1101	15	D
14	—	1110	16	E
15	—	1111	17	F

ตารางนี้คือของที่ต้องคล่องที่สุดในบท ไม่ต้องท่อง แต่ต้อง**เขียนใหม่เองได้ใน 30 วินาที** วิธีคือเขียนคอลัมน์ฐานสองก่อน (นับ 0000 0001 0010 ...) แล้วคอลัมน์ที่เหลืออ่านออกมาจากมันได้ทั้งหมด

ทำไมต้องกลุ่มละ 3 กับกลุ่มละ 4 พอดี

ตัวเลขฐานแปดหนึ่งตัวเก็บค่าได้ 8 แบบ (0 ถึง 7) · บิตสามตัวเก็บได้ $2^3 = 8$ แบบพอดีเป๊ะ · **เท่ากันพอดี** จึงจับคู่กันได้แบบหนึ่งต่อหนึ่ง ไม่เหลือ ไม่ขาด

ตัวเลขฐานสิบหกหนึ่งตัวเก็บค่าได้ 16 แบบ (0 ถึง F) · บิตสี่ตัวเก็บได้ $2^4 = 16$ แบบพอดีเป๊ะ · จับคู่กันได้เหมือนกัน เพราะฉะนั้นมันไม่ใช่กฎที่ใครนึกอยากตั้งขึ้นมา แต่เป็นผลจากการที่ 8 กับ 16 เป็นกำลังของ 2 พอดี

ลองคิดกลับกันให้เห็นตอนพัง ถ้าเราอยากแปลงฐานสองเป็นฐานสิบด้วยวิธีจับกลุ่มบ้าง จะทำไม่ได้เลย เพราะ 10 ไม่ใช่กำลังของ 2 — บิตสามตัวได้ 8 แบบ (ขาดไป 2) บิตสี่ตัวได้ 16 แบบ (เกินมา 6) ไม่มีจำนวนบิตไหนลงตัว นี่คือเหตุผลที่การแปลงฐานสอง↔ฐานสิบต้องหารทีละรอบ ช้ากว่ากันมาก

กฎที่ต้องจำ

ฐานสอง 3 บิต = ฐานแปด 1 ตัว · ฐานสอง 4 บิต = ฐานสิบหก 1 ตัว

จับกลุ่มยังไง และจับจากด้านไหน

กฎด้านคือ **จับจากจุดทศนิยมออกไป** เลขจำนวนเต็มไม่มีจุดเขียนไว้ก็จริง แต่จุดของมันอยู่ท้ายสุดด้านขวาเสมอ เพราะฉะนั้นเลขจำนวนเต็มให้ **จับจากขวาสุดไปทางซ้าย**

ถ้ากลุ่มซ้ายสุดบิตไม่ครบ ให้เติมศูนย์ข้างหน้าจนครบ เติมศูนย์หน้าไม่ทำให้ค่าเปลี่ยน เหมือนเขียน 7 เป็น 007



รูป 1.2 · $1011111001100011_2 = BE63_{16}$ — บิตไม่ได้เปลี่ยนเลยสักตัว เปลี่ยนแค่วิธีเขียน (ทั้งสองฝั่งมีค่า 48,739)

ฐานแปดก็วิธีเดียวกันเป๊ะ แค่จับทีละ 3 และค่าประจำหลักในกลุ่มเป็น 4 2 1



รูป 1.3 · $101111010011_2 = 5723_8$ — เช็กกลับด้วยฐานสิบได้ ทั้งสองฝั่งมีค่า 3,027

เดินให้เห็นตอนพัง — จับกลุ่มผิดด้าน

นี่คือกับดักที่ข้อสอบชอบที่สุดในหัวข้อนี้ เพราะเวลาบิตครบพอดี (6 บิต 12 บิต 16 บิต) จับด้านไหนก็ได้คำตอบเดียวกัน คนเลยเผลอคิดว่าไม่สำคัญ แล้วพอเจอบิตไม่ครบก็ผิดทันที

เอาเลข $\mathbf{10111_2}$ มาแปลงเป็นฐานแปด เลขนี้มี 5 บิต หาร 3 ไม่ลงตัว

ถูก · จับจากขวาสุด แล้วเติมศูนย์ข้างหน้า

0 10111

010 111

2 7

ได้ 27_8

เช็ก: 2×8 + 7 = 23

และ 10111 = 16+4+2+1 = 23 ✓

ผิด · จับจากซ้ายสุด

10111

101 11

กลุ่มท้ายเหลือ 2 บิต เติมยังไงก็ผิด

เติมหน้าเป็น 011 → 53_8 = 43 ✗

เติมหลังเป็น 110 → 56_8 = 46 ✗

ทั้งคู่ไม่เท่ากับ 23

รูป 1.4 · เลขเดียวกัน จับคนละด้าน ได้คนละคำตอบ — ฝั่งขวาผิดทั้งสองแบบ

เหตุผลที่จับจากซ้ายไม่ได้ก็ตรงไปตรงมา — **ค่าประจำหลักนับจากขวา** บิตขวาสุดต้องมีค่า 1 เสมอ ถ้าเราเริ่มนับกลุ่มจากซ้าย บิตขวาสุดจะกลายเป็นบิตค่า 4 ของกลุ่มสุดท้าย ค่ามันเลยเพี้ยนไปทั้งเลข

กับดักข้อสอบ

ถ้าจำนวนบิตหาร 3 (หรือ 4) ไม่ลงตัว **ให้เติมศูนย์ที่ด้านซ้าย** ของเลขจำนวนเต็มเสมอ แล้วค่อยจับกลุ่ม · ห้ามเติมด้านขวา เพราะเติมศูนย์ด้านขวาของจำนวนเต็มคือการคูณสอง

แปลงกลับทางก็ทำเหมือนกัน

ถ้าโจทย์ให้ฐานแปดหรือฐานสิบหกมา แล้วให้ตอบเป็นฐานสอง ก็แค่ทำย้อนศร — แตกตัวเลขทีละตัวออกเป็น 3 บิต (ฐานแปด) หรือ 4 บิต (ฐานสิบหก) แล้วเขียนต่อกันตามลำดับเดิม **ห้ามสลับตำแหน่ง**

โจทย์ 2.1 — แปลง 4106_8 เป็นฐานสอง

ทำอะไร	บรรทัด
แยกตัวเลขออกมา	4 · 1 · 0 · 6
แตกแต่ละตัวเป็น 3 บิต (ค่าในกลุ่มคือ 4 2 1)	4 = 100 · 1 = 001 · 0 = 000 · 6 = 110
เขียนต่อกันตามลำดับเดิม	100 001 000 110

คำตอบ

$4106_8 = 100001000110_2$

เช็กกลับ — ฝั่งฐานแปด: 4×512 + 1×64 + 0×8 + 6×1 = 2048 + 64 + 6 = **2118** · ฝั่งฐานสอง: 2048 + 64 + 4 + 2 = **2118** ตรงกัน

ฐานสิบหกก็ทำแบบเดียวกัน แค่เปลี่ยนจากกลุ่มละ 3 เป็นกลุ่มละ 4

โจทย์ 2.2 — แปลง $A29C_{16}$ เป็นฐานสอง

ทำอะไร	บรรทัด
แยกตัวเลขออกมา	A · 2 · 9 · C
อ่านค่าของตัวอักษรก่อน	A = 10 · C = 12
แตกแต่ละตัวเป็น 4 บิต (ค่าในกลุ่มคือ 8 4 2 1)	10 = 1010 · 2 = 0010 · 9 = 1001 · 12 = 1100
เขียนต่อกันตามลำดับเดิม	1010 0010 1001 1100

คำตอบ

$A29C_{16} = 1010001010011100_2$

เช็กกลับ — นับบิตได้ 16 บิต ตรงกับฐานสิบหก 4 ตัว × 4 บิต · และทั้งสองฝั่งมีค่า 41,628

ฐานแปดกับฐานสิบหก แปลงเป็นฐานสิบ

อันนี้ไม่มีอะไรใหม่เลย ใช้สูตรกลางตัวเดิมจากหัวข้อ 1 แค่เปลี่ยน r

โจทย์ 2.3 — $37_8 \cdot 264_8 \cdot 5201_8$ เป็นฐานสิบ

โจทย์	กระจายตามค่าประจำหลัก	ได้
37_8	3×8 + 7×1 = 24 + 7	**31**
264_8	2×64 + 6×8 + 4×1 = 128 + 48 + 4	**180**
5201_8	5×512 + 2×64 + 0×8 + 1×1 = 2560 + 128 + 0 + 1	**2689**

เช็กกลับ

37_8 มี 2 หลัก ค่าสูงสุดที่ 2 หลักฐานแปดเก็บได้คือ $77_8 = 63 \cdot 31$ น้อยกว่า 63 จริง สมเหตุสมผล

5201_8 มี 4 หลัก ค่าต้องอยู่ระหว่าง 512 ถึง $4095 \cdot 2689$ อยู่ในช่วงจริง

ฐานสิบหกใช้สูตรเดียวกัน แต่ต้องแปลงตัวอักษรเป็นตัวเลขก่อน

โจทย์ 2.4 — $C9_{16} \cdot F8A_{16}$ เป็นฐานสิบ

โจทย์	แปลงตัวอักษรก่อน แล้วกระจาย	ได้
$C9_{16}$	$C = 12 \cdot 12\times16 + 9\times1 = 192 + 9$	**201**
$F8A_{16}$	$F = 15, A = 10 \cdot 15\times256 + 8\times16 + 10\times1 = 3840 + 128 + 10$	**3978**

เช็กกลับ

$F8A_{16}$ มี 3 หลัก ค่าต้องอยู่ระหว่าง 256 ถึง $4095 \cdot 3978$ อยู่ในช่วง และเกือบเต็มเพราะหลักแรกเป็น F (มากสุด)

ฐานสิบ ไปฐานอะไรก็ได้ — หารซ้ำ

ขามาเราทำได้แล้ว (เอาค่าประจำหลักมาบวก) ขากลับต้องใช้วิธีตรงข้าม คือ **หารด้วยฐานซ้ำ ๆ แล้วเก็บเศษ**

เหตุผลที่ต้องเก็บเศษ: เวลาเราหาร 47 ด้วย 2 เศษที่ได้ (1) คือส่วนที่ "ไม่ลงตัวกับหลักที่สูงกว่า" นั่นคือค่าของหลักขวาสุดพอดี ส่วนผลหาร (23) คือเลขที่เหลือสำหรับหลักที่สูงกว่า ทำซ้ำไปเรื่อย ๆ ก็ได้ทุกหลัก

จุดที่คนพลาดบ่อย

เศษตัวแรกคือหลักขวาสุด (LSB) เพราะฉะนั้นเวลาเขียนคำตอบต้อง**อ่านเศษจากล่างขึ้นบน** ถ้าเขียนจากบนลงล่างจะได้เลขกลับหัว

โจทย์ 2.5 แปลง 47_{10} เป็นฐานสอง

หารด้วย 2 ไปเรื่อย ๆ จนผลหารเป็น 0

รอบ	หาร	ผลหาร	เศษ
1	47 ÷ 2	23	**1** ← LSB
2	23 ÷ 2	11	**1**
3	11 ÷ 2	5	**1**
4	5 ÷ 2	2	**1**
5	2 ÷ 2	1	**0**
6	1 ÷ 2	0 ← หยุด	**1** ← MSB

อ่านเศษจากล่างขึ้นบน: 1 0 1 1 1 1

คำตอบ

$47_{10} = \mathbf{101111_2}$

เช็กกลับ — $101111_2 = 32 + 8 + 4 + 2 + 1 = 47$ ✓

ฐานสิบหกกับฐานแปดก็ใช้วิธีหารซ้ำเหมือนกัน แค่เปลี่ยนตัวหาร

โจทย์ 2.6 แปลง 47_{10} เป็นฐานสิบหก และฐานแปด

(ก) ฐานสิบหก — หารด้วย 16

รอบ	หาร	ผลหาร	เศษ
1	47 ÷ 16	2	**15** → เขียนเป็น **F** ← หลักขวา
2	2 ÷ 16	0 ← หยุด	**2** ← หลักซ้าย

อ่านจากล่างขึ้นบน: 2 F

(ข) ฐานแปด — หารด้วย 8

รอบ	หาร	ผลหาร	เศษ
1	47 ÷ 8	5	**7** ← หลักขวา

รอบ	หาร	ผลหาร	เศษ
2	5 ÷ 8	0 ← หยุด	**5** ← หลักซ้าย

อ่านจากล่างขึ้นบน: 5 7

คำตอบ

47_{10} = **$2F_{16}$** = **57_8** (และ = 101111_2 จากโจทย์ 2.5)

เช็กกลับทั้งสามอัน — 101111 = 32+8+4+2+1 = 47 ✓ · 2F = 2×16+15 = 47 ✓ · 57_8 = 5×8+7 = 47 ✓

เช็กอีกทางหนึ่ง — 101111 จับกลุ่มละ 4 จากขวา: 0010 1111 → 2 F ตรงกับคำตอบข้อ (ข) · จับกลุ่มละ 3: 101 111 → 5 7 ตรงกับข้อ (ค) · แปลว่าทำถูกทั้งสามทาง

หัวข้อนี้มีวิธีอยู่สามแบบคือ จับกลุ่ม · สูตรกลาง · หารซ้ำ — ใช้ให้ถูกกับทิศทางที่โจทย์ถาม

เช็กลิสต์หัวข้อ 2

1. **ฐานสอง ↔ ฐานแปด/สิบหก** ใช้จับกลุ่มเท่านั้น ห้ามอ้อมไปฐานสิบ เสียเวลาและผิดง่าย
2. จับกลุ่ม**จากจุดออกไป** — จำนวนเต็มคือจากขวาสุด · กลุ่มซ้ายสุดไม่ครบให้เติมศูนย์ด้านซ้าย
3. **ฐานแปด/สิบหก → ฐานสิบ** ใช้สูตรกลาง คูณค่าประจำหลักแล้วบวก
4. **ฐานสิบ → ฐานอะไรก็ได้** ใช้หารซ้ำ เก็บเศษ แล้ว**อ่านเศษจากล่างขึ้นบน**
5. **หยุดเมื่อผลหารเป็น 0** ไม่ใช่เมื่อเศษเป็น 0
6. เช็กกลับทุกครั้งโดยแปลงคำตอบย้อนเป็นฐานสิบ ถ้าไม่เท่าโจทย์แสดงว่าพลาด

3 เลขมีจุด — ฝั่งขวาของจุดคิดยังไง

ตาชั่งที่มีลูกตุ้มแค่ครึ่ง ๆ

ลองนึกถึงตาชั่งสองแขนในห้องแลบ กับกล่องลูกตุ้มที่มีแค่ 4 ก้อนคือ ครึ่งกิโล, หนึ่งส่วนสี่กิโล, หนึ่งส่วนแปดกิโล และหนึ่งส่วนสิบหกกิโล · ลูกตุ้มแต่ละก้อนมีก้อนเดียว จะใช้หรือไม่ใช้ก็ได้ แต่ใช้ซ้ำไม่ได้

ลองก่อน

น้องอยากชั่งของให้ได้ **0.625 กิโล** พอดี จะหยิบลูกตุ้มก้อนไหนบ้าง

ตอบ ครึ่งกิโล กับ หนึ่งส่วนแปดกิโล ใช่ไหม — 0.5 + 0.125 = 0.625 พอดี ส่วนก้อน 1/4 กับ 1/16 ไม่ได้ใช้

ที่น้องเพิ่งทำคือการเขียนเลขฐานสองฝั่งขวาของจุดพอดีเป๊ะ · ก้อนที่ใช้เขียนเป็น **1** ก้อนที่ไม่ใช้เขียนเป็น **0** เรียงจากก้อนใหญ่ไปก้อนเล็ก จะได้ **1, 0, 1, 0** หรือเขียนว่า $\mathbf{0.101_2}$ (ตัดศูนย์ท้ายทิ้งได้ เหมือน 0.50 กับ 0.5 ในฐานสิบ)

จุดที่คั่นกลางนั้นมีชื่อทางการว่า **จุดฐาน** (radix point) ในฐานสิบเราเรียกมันว่าจุดทศนิยม แต่ในฐานอื่นเรียกว่าจุดทศนิยมไม่ได้ เพราะ "ทศ" แปลว่าสิบ

ค่าประจำหลักฝั่งขวาของจุด

กฎเดิมไม่เปลี่ยนเลย — **เดินไปทางซ้ายคูณด้วยฐาน เดินไปทางขวาหารด้วยฐาน** · ฝั่งซ้ายของจุดคือ 1, 2, 4, 8 · พอข้ามจุดมาฝั่งขวาก็หารสองต่อไปเรื่อย ๆ เป็น 1/2, 1/4, 1/8, 1/16

เขียนด้วยเลขยกกำลังจะเห็นแบบแผนชัดกว่า: $\ldots, 2^2, 2^1, 2^0 \,.\, 2^{-1}, 2^{-2}, 2^{-3}, \ldots$ ตัวเลขยกกำลังลดลงทีละหนึ่งต่อเนื่องข้ามจุดไปเลย ไม่มีอะไรสะดุด



รูป 1.5 · $1011.101_2 = 11.625_{10}$ — ฝั่งขวาของจุดใช้กฎเดียวกับฝั่งซ้าย แค่ค่าประจำหลักหารสองลงไปเรื่อย ๆ

โจทย์ 3.1 — 10.01_2 และ $D3.2A_{16}$ เป็นฐานสิบ

(ก) 10.01_2

ทำอะไร	บรรทัด
แยกสองฝั่งของจุด	ฝั่งซ้าย 10 · ฝั่งขวา 01
ฝั่งซ้าย: ค่าประจำหลัก 2, 1	1×2 + 0×1 = 2
ฝั่งขวา: ค่าประจำหลัก 1/2, 1/4	0×(1/2) + 1×(1/4) = 0.25
บวกสองฝั่ง	2 + 0.25 = **2.25**

(ข) $D3.2A_{16}$

ทำอะไร	บรรทัด
แปลงตัวอักษรก่อน	D = 13 · A = 10
ฝั่งซ้าย: ค่าประจำหลัก 16, 1	13×16 + 3×1 = 208 + 3 = 211

ทำอะไร	บรรทัด
ฝั่งขวา: ค่าประจำหลัก 1/16, 1/256	2×(1/16) + 10×(1/256) = 0.125 + 0.0390625
ฝั่งขวารวม	0.1640625
บวกสองฝั่ง	211 + 0.1640625 = **211.1640625**

คำตอบ

10.01_2 = **2.25** · $D3.2A_{16}$ = **211.1640625**

เช็กกลับ — ฝั่งขวาของจุดต้องน้อยกว่า 1 เสมอ · 0.25 และ 0.1640625 น้อยกว่า 1 จริงทั้งคู่ ถ้าได้เกิน 1 แสดงว่าใช้ค่าประจำหลักผิดฝั่ง

ฐานสิบ ไปฐานสอง เมื่อมีเศษ — คูณสองซ้ำ

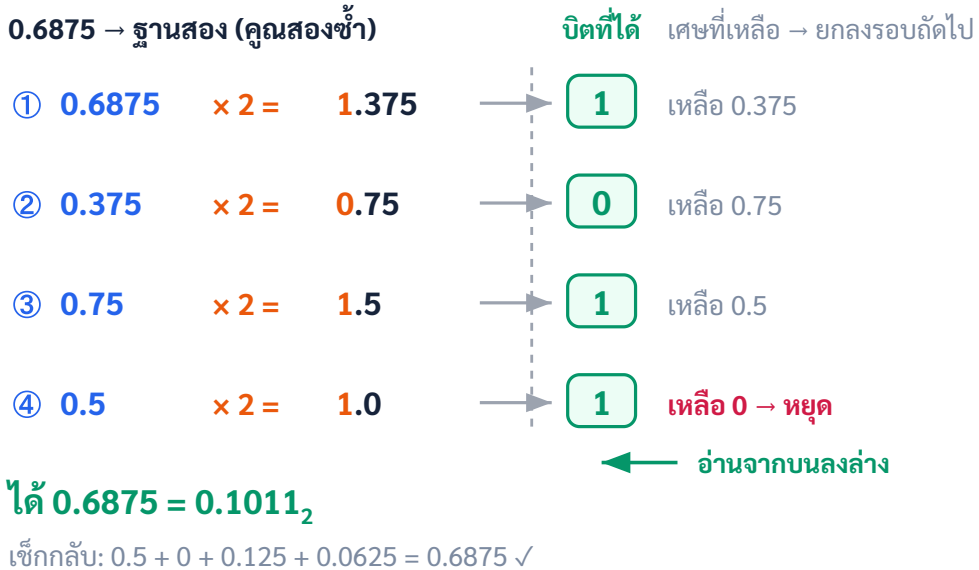
ฝั่งจำนวนเต็มเราใช้หารซ้ำแล้วเก็บเศษ · ฝั่งเศษส่วนต้องทำ**ตรงข้าม** คือ **คูณด้วยฐานซ้ำ ๆ แล้วเก็บส่วนที่โผล่ข้ามจุดออกมาหน้าจุด**

ทำไมถึงเป็นแบบนั้น: เอา 0.101_2 มาคูณสองสิ ทุกอย่างเลื่อนไปทางซ้ายหนึ่งหลัก กลายเป็น 1.01_2 · บิตที่เคยอยู่ติดจุดก็ถูกดันข้ามจุดมาเป็นส่วนจำนวนเต็มพอดี เราจึงอ่านมันออกมาได้ทีละตัว

ต่างจากฝั่งจำนวนเต็มตรงนี้

ฝั่งจำนวนเต็ม (หารซ้ำ) → **อ่านเศษจากล่างขึ้นบน**

ฝั่งเศษส่วน (คูณซ้ำ) → **อ่านจากบนลงล่าง** เพราะตัวแรกที่ได้คือบิตที่ติดจุดที่สุด



ได้ $0.6875 = 0.1011_2$

เช็กกลับ: 0.5 + 0 + 0.125 + 0.0625 = 0.6875 ✓

รูป 1.6 · คูณสองซ้ำ — เลขที่โผล่มาหน้าจุดในแต่ละรอบคือบิตคำตอบ เอาเฉพาะส่วนหลังจุดไปทำรอบถัดไป

สังเกตจุดหยุด — เราหยุดเมื่อ**เศษหลังจุดเป็น 0** ไม่ใช่เมื่อผลคูณเป็น 0 (ผลคูณไม่มีทางเป็น 0 อยู่แล้ว)

เดินให้เห็นตอนพัง — เศษที่ไม่มีวันจบ

เลข 0.1 ในฐานสิบเป็นเลขสวย ๆ ที่เราเขียนกันทุกวัน แต่พอเอาไปฐานสองแล้วจะเจอเรื่องแปลก ลองคูณสองซ้ำดู

รอบ	เอาไปคูณ 2	ได้	บิต	เหลือ
1	0.1 × 2	0.2	**0**	0.2
2	0.2 × 2	0.4	**0**	0.4
3	0.4 × 2	0.8	**0**	0.8
4	0.8 × 2	1.6	**1**	0.6
5	0.6 × 2	1.2	**1**	0.2 ← ซ้ำกับรอบ 2 แล้ว
6	0.2 × 2	0.4	**0**	0.4
7	0.4 × 2	0.8	**0**	0.8
8	0.8 × 2	1.6	**1**	0.6
9	0.6 × 2	1.2	**1**	0.2 ← วนอีกแล้ว

เศษกลับมาเป็น 0.2 ซ้ำแล้วซ้ำอีก แปลว่าถ้าเราทำต่อไปจะได้ 0011 0011 0011 ... ไม่มีวันจบ · เขียนได้ว่า $0.1_{10} = 0.000110011001\ldots_2$ โดยกลุ่ม **0011** ซ้ำไปเรื่อย ๆ

นี่ไม่ใช่ความผิดพลาดของเรา แต่เป็นธรรมชาติของฐาน — เหมือน 1/3 ในฐานสิบที่เขียนเป็น 0.3333... ไม่จบเหมือนกัน · เลขจะจบในฐานสองก็ต่อเมื่อตัวส่วนของมันเป็นกำลังของ 2 เท่านั้น (เช่น 1/2, 3/4, 11/16) · 0.1 = 1/10 ตัวส่วนมี 5 อยู่ด้วย จึงไม่มีวันจบ

เจอแบบนี้ในข้อสอบต้องทำยังไง

① ทำต่อจนครบจำนวนบิตที่โจทย์สั่ง (เช่น "ตอบ 8 บิตหลังจุด") แล้วหยุด

② ถ้าโจทย์ไม่ได้สั่ง ให้ทำจนเห็นเศษซ้ำ แล้วเขียนคำตอบพร้อมบอกว่ากลุ่มไหนซ้ำ

③ **ห้ามเขียนว่าหยุดเพราะเหลือ 0** ทั้งที่ยังไม่เหลือ 0 — นั่นคือตอบผิด ไม่ใช่ตอบย่อ

ถ้าตัดที่ 9 บิต จะได้ 0.000110011_2 ซึ่งมีค่า 0.099609375 ไม่ใช่ 0.1 พอดี · ความคลาดเคลื่อนตรงนี้เองคือสาเหตุที่โปรแกรมคอมพิวเตอร์บวก 0.1 สิบครั้งแล้วไม่ได้ 1 พอดี

คูณซ้ำด้วยฐานอื่นก็ทำเหมือนกัน

โจทย์ 3.2 — 0.65625_{10} เป็นฐานสิบหก · 0.5625_{10} เป็นฐานแปด

(ก) 0.65625 → ฐานสิบหก (คูณ 16 ซ้ำ)

รอบ	คูณ	ได้	หยิบส่วนหน้าจุด	เหลือ
1	0.65625 × 16	10.5	**10 → A**	0.5
2	0.5 × 16	8.0	**8**	0 → หยุด

อ่านจากบนลงล่าง: A แล้วก็ 8

(ข) 0.5625 → ฐานแปด (คูณ 8 ซ้ำ)

รอบ	คูณ	ได้	หยิบส่วนหน้าจุด	เหลือ
1	0.5625 × 8	4.5	**4**	0.5
2	0.5 × 8	4.0	**4**	0 → หยุด

คำตอบ

$0.65625_{10} = \mathbf{0.A8_{16}}$ · $0.5625_{10} = \mathbf{0.44_8}$

เช็กกลับ (ก) — 10/16 + 8/256 = 0.625 + 0.03125 = 0.65625 ✓

เช็กกลับ (ข) — 4/8 + 4/64 = 0.5 + 0.0625 = 0.5625 ✓

ระวัง — ถ้าส่วนหน้าจุดออกมาเป็น 10 ถึง 15 ในฐานสิบหก ต้องเขียนเป็น A–F อย่าปล่อยเป็นเลขสองหลัก

จับกลุ่มเศษส่วน — ต้องเริ่มจากจุด ไม่ใช่จากปลาย

เวลาแปลงเลขมีจุดจากฐานสองไปฐานสิบหก เราก็ยังใช้การจับกลุ่มเหมือนเดิม แต่คราวนี้กฎ "จับจากจุดออกไป" จะเริ่มมีความหมายจริง ๆ แล้ว

ฝั่งซ้ายของจุด จับจากจุดไปทางซ้าย ขาดก็เติมศูนย์ทางซ้าย · ฝั่งขวาของจุด จับจากจุดไปทางขวา ขาดก็**เติมศูนย์ทางขวา**



รูป 1.7 · $1011.101_2 = B.A_{16}$ — ฝั่งซ้ายเติมศูนย์ทางซ้าย ฝั่งขวาเติมศูนย์ทางขวา ทั้งคู่คือการเติมจาก "ปลายที่ไกลจุดที่สุด"

ทำไมฝั่งขวาถึงเติมทางขวา เหตุผลเดียวกับฝั่งซ้ายเป๊ะ — เติมศูนย์ตรงปลายที่ห่างจุดที่สุดไม่ทำให้ค่าเปลี่ยน เหมือนเขียน 0.5 เป็น 0.50 · แต่ถ้าเติมตรงที่ติดจุด ค่าจะหารสองไปทั้งเลข

ถูก · เติมศูนย์ปลายขวา

0. 101 0

1010 = 10 = A

ได้ $0.A_{16}$

เช็ก: 10/16 = 0.625

0.101 = 0.5+0.125 = 0.625 ✓

ผิด · เติมศูนย์ด้านที่ติดจุด

0. 0101

0101 = 5

ได้ 0.5_{16}

เช็ก: 5/16 = 0.3125 ✗

ค่าหายไปครึ่งหนึ่ง เพราะเลื่อนหลักไปหนึ่ง

รูป 1.8 · กับดักคลาสสิก — 0.101_2 ถ้าเติมศูนย์ผิดด้าน ค่าจะเหลือครึ่งเดียวทันที

โจทย์ 3.3 — แปลง 0.10111_2 เป็นฐานแปด

ทำอะไร	บรรทัด
นับจากจุดไปทางขวา ทีละ 3	101 | 11
กลุ่มท้ายเหลือ 2 บิต เติมศูนย์**ทางขวา**	101 | 110
อ่านค่าแต่ละกลุ่ม (4 2 1)	101 = 5 · 110 = 6
เขียนหลังจุด	0.56_8

คำตอบ

$0.10111_2 = \mathbf{0.56_8}$

เช็กกลับ — ฝั่งฐานแปด: 5/8 + 6/64 = 0.625 + 0.09375 = **0.71875**

ฝั่งฐานสอง: 0.5 + 0.125 + 0.0625 + 0.03125 = **0.71875** ตรงกัน

เลขมีจุดมีสองฝั่งที่ทำคนละวิธีกัน จึงต้องมีลำดับขั้นตอนไว้กันสับสน

เช็กลิสต์หัวข้อ 3

1. เจอเลขมีจุด ให้**ขีดเส้นแบ่งสองฝั่ง**แล้วทำทีละฝั่ง อย่าทำรวด
2. ฝั่งจำนวนเต็ม: หารซ้ำ เก็บเศษ อ่านจาก**ล่างขึ้นบน**
3. ฝั่งเศษส่วน: คูณซ้ำ เก็บส่วนหน้าจุด อ่านจาก**บนลงล่าง**
4. หยุดเมื่อเศษหลังจุดเป็น 0 · ถ้าเศษเริ่มวนซ้ำ แปลว่าไม่มีวันจบ ให้ตัดตามจำนวนหลักที่โจทย์สั่ง
5. ไปฐานแปด/สิบหก ใช้จับกลุ่ม · **นับออกจากจุดเสมอ** · ซ้ายเติมศูนย์ซ้าย ขวาเติมศูนย์ขวา
6. เช็กกลับทุกครั้ง: ฝั่งเศษส่วนต้องได้ค่าน้อยกว่า 1 และตรงกับโจทย์

4 บวก ลบ คูณ หาร ในฐานสอง

เริ่มจากสิ่งที่น้องทำได้อยู่แล้ว

เวลาน้องบวก 7 + 5 ในฐานสิบ น้องเขียน 2 แล้วทด 1 ใช่ไหม · เหตุผลคือ 12 มันเกินสิ่งที่หลักเดียวเก็บได้ (0–9) ส่วนที่เกินจึงต้องย้ายไปหลักซ้าย

ฐานสองก็ทำแบบเดียวกันทุกอย่าง แค่ **"เต็ม" เร็วกว่ามาก** — หลักหนึ่งเก็บได้แค่ 0 กับ 1 พอถึง 2 ก็เต็มแล้ว

ลองก่อน

ในฐานสอง 1 + 1 ได้เท่าไร · ระวัง คำตอบห้ามมีเลข 2

ตอบ 10 ใช่ไหม (อ่านว่า "หนึ่ง-ศูนย์" ไม่ใช่ "สิบ") — เพราะหลักนี้เต็มแล้ว ต้องทดไปหลักซ้าย เหมือน 9 + 1 = 10 ในฐานสิบ

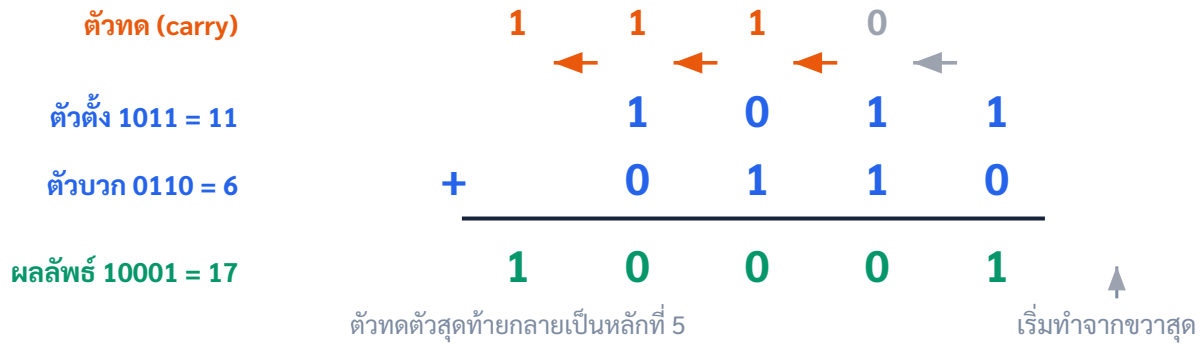
4.1 บวก

กฎมีแค่ 4 บรรทัด ท่องไม่ต้อง อ่านสองรอบก็จำได้

กฎการบวก

0 + 0 = **0** (ทด 0) · 0 + 1 = **1** (ทด 0) · 1 + 0 = **1** (ทด 0) · 1 + 1 = **0 ทด 1**

และเมื่อมีตัวทดเข้ามาด้วย: 1 + 1 + 1 = **1 ทด 1**



รูป 1.9 · $1011_2 + 0110_2 = 10001_2$ — ลูกศรส้มคือตัวทดที่วิ่งไปทางซ้ายทีละหลัก

โจทย์ 4.1 — 1011 + 0110 — เดินทีละหลัก

หลัก	ตัวทดเข้า	บิตบน	บิตล่าง	รวมได้	เขียน	ทดออก
1 (ขวาสุด)	0	1	0	1	**1**	0
2	0	1	1	10_2	**0**	1
3	1	0	1	10_2	**0**	1
4	1	1	0	10_2	**0**	1
5	1	—	—	1	**1**	0

คำตอบ

$1011_2 + 0110_2 = \mathbf{10001_2}$

เช็กกลับ — $1011_2 = 11 \cdot 0110_2 = 6 \cdot 11 + 6 = 17 \cdot$ และ $10001_2 = 16 + 1 = 17$ ✓

สังเกต — ผลบวกยาวกว่าตัวตั้งหนึ่งหลัก เพราะตัวทดจากหลักซ้ายสุดไม่มีที่ไป ต้องเปิดหลักใหม่

ลองอีกข้อ คราวนี้มีหลักที่ต้องบวกสามตัวพร้อมกัน

โจทย์ 4.2 — 1111 + 1100

หลัก	ตัวทดเข้า	บิตบน	บิตล่าง	รวมได้	เขียน	ทดออก
1	0	1	0	1	**1**	0
2	0	1	0	1	**1**	0
3	0	1	1	10_2	**0**	1

หลัก	ตัวทดเข้า	บิตบน	บิตล่าง	รวมได้	เขียน	ทดออก
4	1	1	1	11_2	**1**	1
5	1	—	—	1	**1**	0

คำตอบ

$1111_2 + 1100_2 = \mathbf{11011_2}$

เช็กกลับ — 15 + 12 = 27 · $11011_2 = 16 + 8 + 2 + 1 = 27$ ✓

ดูหลักที่ 4 ให้ดี — $1 + 1 + 1 = 11_2$ คือ "เขียน 1 ทด 1" ไม่ใช่ "เขียน 0 ทด 1" · ตรงนี้คนพลาดเยอะที่สุด

4.2 ลบ

ฝั่งลบก็เหมือนเดิม แต่เปลี่ยนจาก "ทด" เป็น "ยืม" · ในฐานสิบเวลา 3 − 7 ไม่ได้ เราไปยืมจากหลักซ้ายมา 1 ทำให้ 3 กลายเป็น 13 · ฐานสองก็ยืมแบบเดียวกัน แต่ยืมมาแล้ว 0 จะกลายเป็น $\mathbf{10_2}$ ซึ่งมีค่า 2

กฎการลบ

$0 - 0 = \mathbf{0}$ · $1 - 1 = \mathbf{0}$ · $1 - 0 = \mathbf{1}$ · $0 - 1 = \mathbf{1}$ **โดยยืม 1** (เพราะ $10_2 - 1 = 1$)

หลักที่ถูกยืมต้องลดลง 1 · ถ้าหลักนั้นเป็น 0 อยู่แล้ว ก็ต้องไปยืมต่ออีกหลัก

โจทย์ 4.3 — 1101 − 0110

หลัก	บิตบน (หลังโดนยืม)	บิตล่าง	ทำอะไร	เขียน
1 (ขวาสุด)	1	0	1 − 0 ได้เลย	**1**
2	0	1	0 − 1 ไม่ได้ → ยืมจากหลัก 3 · หลัก 3 เป็น 1 ลดเหลือ 0 · หลักนี้กลายเป็น 10_2 แล้ว $10_2 - 1 = 1$	**1**
3	0 (โดนยืมไปแล้ว)	1	0 − 1 ไม่ได้อีก → ยืมจากหลัก 4 · หลัก 4 เป็น 1 ลดเหลือ 0 · $10_2 - 1 = 1$	**1**
4	0 (โดนยืมไปแล้ว)	0	0 − 0	**0**

คำตอบ

$1101_2 - 0110_2 = \mathbf{0111_2}$

เช็กกลับ — $13 - 6 = 7 \cdot 0111_2 = 4 + 2 + 1 = 7$ ✓

เช็กอีกทาง — เอาคำตอบบวกกลับ: 0111 + 0110 ต้องได้ 1101 · 1 + 0 = 1, 1 + 1 = 0 ทด 1, 1 + 1 + 1 = 1 ทด 1, 0 + 0 + 1 = 1 → ได้ 1101 ✓

อีกข้อหนึ่งที่ง่ายกว่า เพราะไม่ต้องยืมเลย

โจทย์ 4.4 — 111 − 100

หลัก	บิตบน	บิตล่าง	ทำอะไร	เขียน
1	1	0	1 − 0	**1**
2	1	0	1 − 0	**1**
3	1	1	1 − 1	**0**

คำตอบ

$111_2 - 100_2 = \mathbf{011_2}$ (เขียนสั้น ๆ ว่า 11_2 ก็ได้)

เช็กกลับ — $7 - 4 = 3 \cdot 011_2 = 2 + 1 = 3$ ✓ · ข้อนี้ไม่ต้องยืมเลยสักหลัก

ก่อนไปเรื่องคูณ มีข้อจำกัดของวิธีลบแบบนี้ที่ต้องรู้ไว้ก่อนหนึ่งข้อ

ข้อจำกัดของวิธีลบตรง ๆ

วิธีข้างบนใช้ได้เฉพาะตอน **ตัวตั้งมากกว่าตัวลบ** เท่านั้น ถ้าตัวตั้งน้อยกว่า เช่น 0110 − 1101 จะยืมไม่ได้ ไม่มีหลักให้ยืมแล้ว

นั่นคือเหตุผลที่เครื่องจริง ๆ **ไม่ใช้วิธีลบนี้เลย** — มันเปลี่ยนการลบให้เป็นการบวกแทน ซึ่งเราจะเรียนในหัวข้อ 5

4.3 คูณ

ข่าวดีของฐานสอง — สูตรคูณมีแค่ 4 บรรทัด และไม่มีบรรทัดไหนต้องคิดเลย

กฎการคูณ

$0 \times 0 = \mathbf{0}$ · $0 \times 1 = \mathbf{0}$ · $1 \times 0 = \mathbf{0}$ · $1 \times 1 = \mathbf{1}$

แปลเป็นภาษาคน: คูณด้วย 1 คือ**ลอกมาทั้งแถว** · คูณด้วย 0 คือ**เขียนศูนย์ทั้งแถว**

วิธีทำเหมือนคูณเลขฐานสิบที่เรียนมาตั้งแต่ประถม — ทำทีละบิตของตัวคูณ เริ่มจากบิตขวาสุด ได้แถวหนึ่งเรียกว่า **ผลคูณย่อย** (partial product) แล้วแถวถัดไปเลื่อนไปทางซ้ายหนึ่งช่อง สุดท้ายบวกทุกแถวเข้าด้วยกัน

โจทย์ 4.5 — 1011 × 101

```
    1011        ตัวตั้ง = 11
x    101        ตัวคูณ  = 5
--------
    1011        (1) บิตขวาสุดของตัวคูณ = 1  ->  ลอก 1011 ลงมา
   0000         (2) บิตถัดไป = 0             ->  ศูนย์ทั้งแถว เลื่อนซ้าย 1
  1011          (3) บิตซ้ายสุด = 1           ->  ลอก 1011 เลื่อนซ้าย 2
--------
  110111        บวกสามแถว = 55
```

เขียนสามแถวนั้นให้ชิดขวาเท่ากันจะเห็นชัดขึ้นว่ากำลังบวกอะไร: **001011** + **000000** + **101100**

หลัก	ตัวทดเข้า	แถว 1	แถว 3	รวม	เขียน	ทดออก
1	0	1	0	1	**1**	0
2	0	1	0	1	**1**	0
3	0	0	1	1	**1**	0
4	0	1	1	10_2	**0**	1
5	1	0	0	1	**1**	0
6	0	0	1	1	**1**	0

(แถว 2 เป็นศูนย์ทั้งแถว เลยไม่ต้องใส่ในตาราง)

คำตอบ

$1011_2 \times 101_2 = \mathbf{110111_2}$

เช็กกลับ — $11 \times 5 = 55$ · $110111_2 = 32 + 16 + 0 + 4 + 2 + 1 = 55$ ✓

เช็กจำนวนหลัก — ตัวตั้ง 4 บิต ตัวคูณ 3 บิต ผลคูณต้องไม่เกิน 7 บิต · ได้ 6 บิต สมเหตุสมผล

ลองข้อที่ตัวคูณมีศูนย์สองตัว จะเห็นว่าเขียนแถวศูนย์ไว้ด้วยสำคัญแค่ไหน

โจทย์ 4.6 — 1101 × 1010 (ตอบเป็น 8 บิต)

```
    1101        ตัวตั้ง = 13
x   1010        ตัวคูณ  = 10
--------
    0000        (1) บิตขวาสุด = 0  -> ศูนย์ทั้งแถว
   1101         (2) บิตถัดไป = 1   -> ลอก 1101 เลื่อนซ้าย 1
  0000          (3) บิตถัดไป = 0   -> ศูนย์ทั้งแถว
 1101           (4) บิตซ้ายสุด = 1 -> ลอก 1101 เลื่อนซ้าย 3
--------
10000010        26 + 104 = 130
```

แถว (2) ที่เลื่อนแล้วคือ $\mathbf{11010_2 = 26}$ · แถว (4) ที่เลื่อนแล้วคือ $\mathbf{1101000_2 = 104}$ · อีกสองแถวเป็นศูนย์จึงไม่ต้องบวก

คำตอบ

$1101_2 \times 1010_2 = \mathbf{10000010_2}$

เช็กกลับ — $13 \times 10 = 130$ · $10000010_2 = 128 + 2 = 130$ ✓

ทางลัดที่ใช้ได้จริง — เลื่อนซ้าย 1 ช่องคือคูณ 2 · เลื่อนซ้าย 3 ช่องคือคูณ 8 · เพราะฉะนั้น $13 \times 10 = 13{\times}2 + 13{\times}8 = 26 + 104$ พอดี ตรงกับสองแถวที่ไม่เป็นศูนย์

4.4 หาร

หารยาวในฐานสองง่ายกว่าฐานสิบด้วยซ้ำ เพราะที่แต่ละขั้นเราไม่ต้องเดาว่า "ใส่ได้กี่ครั้ง" — ใส่ได้แค่ **1 ครั้ง หรือ 0 ครั้ง** เท่านั้น

ขั้นตอนการหาร

1. เอาตัวหารไปเทียบกับบิตหน้าสุดของตัวตั้ง (จำนวนบิตเท่าตัวหาร)
2. ถ้าตัวหาร**น้อยกว่าหรือเท่ากับ**ท่อนนั้น → เขียน **1** ในผลหาร แล้วลบออก · ถ้ามากกว่า → เขียน **0** แล้วไม่ต้องลบ
3. ชักบิตถัดไปของตัวตั้งลงมาต่อท้ายเศษ แล้วทำข้อ 2 ใหม่
4. ทำจนใช้บิตของตัวตั้งหมด · สิ่งที่เหลือคือ**เศษ**

โจทย์ 4.7 1111 ÷ 101

```
        11        <- ผลหาร
      ----
101 ) 1111        ตัวตั้ง = 15 , ตัวหาร = 5
      101         (1) 111 (=7) >= 101 (=5)  ->  เขียน 1  แล้วลบ
      ----
       101        (2) เหลือ 10 (=2) ชักบิตสุดท้ายลงมา ได้ 101 (=5)
       101        (3) 101 >= 101  ->  เขียน 1  แล้วลบ
       ----
         0        เศษ = 0
```

คำตอบ

$1111_2 \div 101_2 = \mathbf{11_2}$ เศษ **0**

เช็กกลับ — $15 \div 5 = 3$ เศษ $0 \cdot 11_2 = 3$ ✓

เช็กด้วยการคูณกลับ — $11_2 \times 101_2 = 3 \times 5 = 15 = 1111_2$ ตรงกับตัวตั้ง ✓

อีกข้อหนึ่งที่ตัวหารเป็นกำลังของสองพอดี

โจทย์ 4.8 1100 ÷ 100

```
        11          <- ผลหาร
      ----
100 ) 1100          ตัวตั้ง = 12 , ตัวหาร = 4
      100           (1) 110 (=6) >= 100 (=4)  ->  เขียน 1  แล้วลบ
      ----
       100          (2) เหลือ 10 (=2) ชักบิตสุดท้าย (0) ลงมา ได้ 100 (=4)
       100          (3) 100 >= 100  ->  เขียน 1  แล้วลบ
       ----
         0          เศษ = 0
```

คำตอบ

$1100_2 \div 100_2 = \mathbf{11_2}$ เศษ **0**

เช็กกลับ — $12 \div 4 = 3 \cdot 11_2 = 3$ ✓

ทางลัด — ตัวหารเป็น 100_2 คือ 4 ซึ่งเป็นกำลังของ 2 พอดี · หารด้วย 4 ก็แค่**เลื่อนขวา 2 ช่อง** · 1100 เลื่อนขวา 2 ได้ 11 ตรงกับคำตอบ (ใช้ได้เฉพาะตอนตัวหารเป็น 2, 4, 8, 16 ... เท่านั้น)

ทั้งสี่การดำเนินการใช้หลักเดียวกันคือทำทีละหลัก สรุปเป็นขั้นตอนได้ดังนี้

เช็กลิสต์หัวข้อ 4

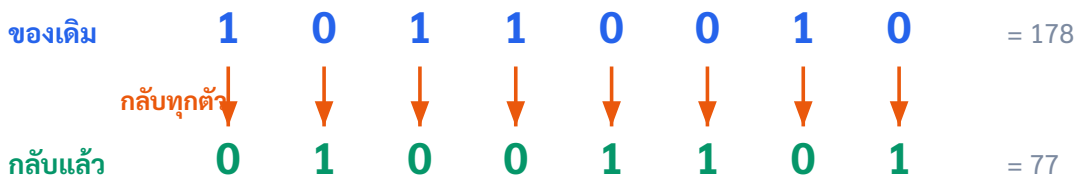
1. **ทุกการดำเนินการเริ่มจากขวาสุดเสมอ** ยกเว้นการหารที่เริ่มจากซ้าย
2. บวก: เขียนแถวตัวทดไว้ข้างบนก่อนลงมือ จะได้ไม่ลืม · 1+1+1 = เขียน 1 ทด 1
3. ลบ: ยืมมาแล้วต้อง**ขีดฆ่าหลักที่ถูกยืมทันที** ไม่งั้นจะเผลอใช้ค่าเดิม
4. คูณ: หนึ่งแถวต่อหนึ่งบิตของตัวคูณ · แถวถัดไปเลื่อนซ้ายหนึ่งช่อง · แถวศูนย์ก็ต้องเขียน อย่าข้าม
5. หาร: เทียบ → เขียน 1 หรือ 0 → ลบ → ชักลงมา · **หยุดเมื่อบิตของตัวตั้งหมด**
6. เช็กกลับทุกข้อ: ลบเช็กด้วยการบวกกลับ · หารเช็กด้วยการคูณกลับ · บวกและคูณเช็กด้วยการแปลงเป็นฐานสิบ

5 คอมพลีเมนต์ — วิธีทำให้เครื่องลบเป็น

หัวข้อนี้คือหัวใจของข้อสอบกลางภาคส่วนตัวเลข ถ้าเข้าใจหัวข้อนี้ หัวข้อ 6 กับ 7 จะง่ายไปเลย · เราจะเริ่มจากการลงมือทำก่อน แล้วค่อยมาคุยกันว่าทำไปทำไม

5.1 กลับบิตทั้งแถว

เอาเลขแถวนี้มาตั้ง: **10110010** · กติกาข้อเดียวคือ **เห็น 1 เขียน 0 เห็น 0 เขียน 1** ทำทุกตัว ไม่เว้นตัวไหน



รูป 1.10 · กลับบิตทั้งแถว 10110010 → 01001101

ลองก่อน

ลองกลับบิตของ **01001000** เองดู เขียนลงกระดาษเลย

ได้ 10110111 ใช่ไหม — ถ้าได้ ผ่าน · ถ้าไม่ตรง ให้ไล่ทีละตัวใหม่ อย่าไล่ทีละสอง

ที่น้องเพิ่งทำมีชื่อว่า **คอมพลีเมนต์แบบที่ 1** (1's complement) แปลเป็นไทยว่า "ตัวเติมเต็มฐานลบหนึ่ง" แต่ในห้องเรียนเรียกทับศัพท์กันว่า **วันส์คอมพลีเมนต์**

มีของแถมที่สังเกตได้ทันที: เอาของเดิมบวกกับตัวที่กลับแล้ว จะได้ **11111111** เสมอ (คือ 178 + 77 = 255) เพราะทุกหลักมี 1 อยู่ตัวเดียวพอดี ตัวหนึ่งเป็น 1 อีกตัวก็ต้องเป็น 0 · สำหรับ 8 บิต 255 คือ $2^8 - 1$

5.2 กลับบิตแล้วบวกหนึ่ง

คราวนี้เพิ่มอีกขั้นเดียว — **กลับบิตทั้งแถว แล้วบวก 1 ที่หลักขวาสุด**

ของที่ได้จากสองขั้นนี้เรียกว่า **คอมพลีเมนต์แบบที่ 2** (2's complement) ในห้องเรียนเรียกทับศัพท์ว่า **ทูส์คอมพลีเมนต์** · ชื่อนี้บอกแค่ว่าทำอะไร ยังไม่ได้บอกว่าทำไปทำไม — คำตอบอยู่ในหัวข้อ 5.3 ถัดไป

โจทย์ 5.1 — หา 2's complement ของ 01001000

ทำอะไร	ได้
ของเดิม	01001000
① กลับบิตทุกตัว (ได้ 1's complement)	10110111
② บวก 1 ที่หลักขวาสุด	10110111 + 1
ไล่ตัวทด: หลัก 1 คือ 1+1 = 0 ทด 1 · หลัก 2 คือ 1+1 = 0 ทด 1 · หลัก 3 คือ 1+1 = 0 ทด 1 · หลัก 4 คือ 0+1 = 1 หยุดทด	10111000

คำตอบ

2's complement ของ 01001000 คือ **10111000**

เช็กกลับ — 01001000 = 72 · 10111000 = 184 · 72 + 184 = 256 ซึ่งเท่ากับ 2^8 พอดี · นี่คือลายเซ็นของ 2's complement: **ของเดิมบวกกับคอมพลีเมนต์ได้ 2^n เสมอ**

วิธียาวข้างบนใช้ได้เสมอ แต่ในห้องสอบมีวิธีที่เร็วกว่านี้อยู่

ทางลัดที่อาจารย์สอน (เร็วกว่ามาก)

ไม่ต้องกลับแล้วบวก ทำทีเดียวจบ: ① **ไล่จากขวาสุดเข้ามา ลอกตัวเลขลงมาเหมือนเดิม จนกว่าจะเจอเลข 1 ตัวแรก (ลอกเลข 1 ตัวนั้นลงมาด้วย) ② ตั้งแต่ตัวถัดไปทางซ้าย กลับบิตให้หมด**

ลองกับ **01001000**: จากขวาเข้ามาเจอ 0 0 0 แล้วเจอ 1 → ลอก **1000** ลงมาทั้งก้อน · ที่เหลือคือ **0100** กลับเป็น **1011** · ต่อกันได้ **10111000** ตรงกับวิธียาว

ทางลัดนี้เร็วกว่าเพราะไม่ต้องไล่ตัวทด แต่**ต้องมั่นใจว่าหา "1 ตัวแรกจากขวา" ถูกตัว**

กรณีที่คนพลาด

ถ้าเลขลงท้ายด้วย 1 เช่น 00100111 ทางลัดจะลอกแค่ "1" ตัวเดียว แล้วกลับที่เหลือทั้งหมด · อย่าเผลอลอกเกิน

ถ้าเลขเป็น 00000000 ทั้งแถว จะไม่มี 1 ให้เจอเลย · กรณีนี้ 2's complement ของมันคือ 00000000 เหมือนเดิม (ตัวทดที่ล้นออกไปถูกทิ้ง) — เดี๋ยวเราจะเข้าใจว่าทำไมในหัวข้อถัดไป

5.3 ทำไมถึงต้องทำแบบนั้น — มาตรวัดที่เดินครบวง

ทีนี้มาถึงคำถามที่สำคัญกว่าวิธีทำ · เรากลับบิตแล้วบวกหนึ่งไปทำไม

ลองนึกถึงมาตรวัดระยะทางแบบเข็มกลมของรถรุ่นเก่า หรือตัวนับรอบแบบล้อหมุน · มันมีขีดอยู่จำนวนหนึ่งรอบวง พอหมุนครบรอบก็กลับมาที่ 0 ใหม่ ไม่มีที่ให้ไปต่อ

สวิตช์ 4 ตัวก็เป็นมาตรวัดแบบนั้น มันมี 16 ขีด (0000 ถึง 1111) พอนับถึง 1111 แล้วบวกอีกหนึ่ง มันก็วนกลับมาเป็น 0000

ลองก่อน

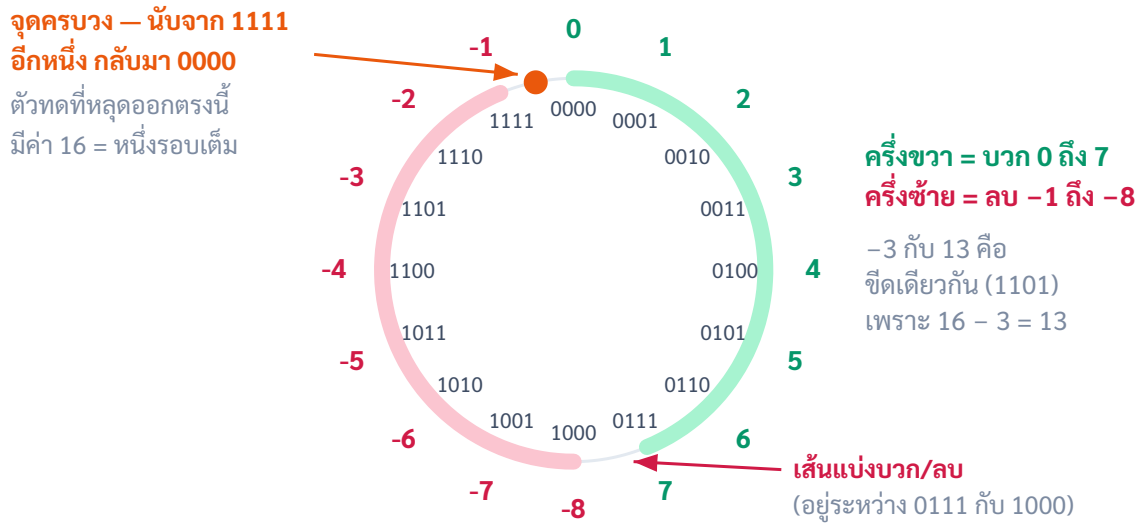
บนหน้าปัดที่มี 16 ขีด น้องอยู่ที่ขีดที่ 7 · ถ้า**ถอยหลัง 3 ขีด** จะถึงขีดที่เท่าไร · แล้วถ้า**เดินหน้า 13 ขีด** ล่ะ จะถึงขีดที่เท่าไร

ได้ขีดที่ 4 ทั้งสองแบบใช่ไหม — ถอยหลัง 3 กับ เดินหน้า 13 พาไปที่เดียวกันเป๊ะ เพราะ 3 + 13 = 16 คือครบหนึ่งรอบพอดี

ที่น้องเพิ่งค้นพบคือความคิดทั้งหมดของ 2's complement — **บนวงที่มี 2^n ขีด การลบด้วย x ให้ผลเหมือนการบวกด้วย $2^n - x$** เป๊ะทุกกรณี

แล้ว $2^n - x$ หาได้ยังไงโดยไม่ต้องลบ? ก็ใช้ของที่เราเพิ่งเจอในหัวข้อ 5.2 ไง — **2's complement ของ x มีค่าเท่ากับ $2^n - x$ พอดี** (เราเพิ่งเช็กไปว่า 72 + 184 = 256)

และ "กลับบิตแล้วบวกหนึ่ง" ก็คือวิธีคำนวณ $2^n - x$ แบบไม่ต้องยืมเลขสักครั้ง: กลับบิตได้ $(2^n - 1) - x$ (เพราะของเดิมบวกตัวกลับได้ 11111111 เสมอ) แล้วบวก 1 อีกทีก็ได้ $2^n - x$ พอดี



รูป 1.11 · หน้าปัด 4 บิต 16 ขีด — ขีดเดียวกันอ่านได้สองแบบ อ่านแบบไม่มีเครื่องหมายได้ 13 อ่านแบบมีเครื่องหมายได้ −3

5.4 ลองใช้จริง — ลบด้วยการบวก

เอา **7 − 3** ในระบบ 4 บิตมาทำดู · แต่คราวนี้เราจะไม่ลบเลย เราจะบวกอย่างเดียว

โจทย์ 5.2 — 7 − 3 ในระบบ 4 บิต โดยใช้ 2's complement

ขั้น	ทำอะไร	ได้
①	เขียน 7 เป็นฐานสอง 4 บิต	0111
②	เขียน 3 เป็นฐานสอง 4 บิต	0011
③	หา 2's complement ของ 3 — กลับบิตได้ 1100 แล้วบวก 1	1101

ขั้น	ทำอะไร	ได้
④	บวกกัน 0111 + 1101 (ไม่ลบแล้ว)	ดูตารางข้างล่าง

หลัก	ตัวทดเข้า	0111	1101	รวม	เขียน	ทดออก
1	0	1	1	10_2	**0**	1
2	1	1	0	10_2	**0**	1
3	1	1	1	11_2	**1**	1
4	1	0	1	10_2	**0**	1 ← ทดหลุดออกนอกแถว

คำตอบ

ผลรวมเต็ม ๆ คือ **10100** (5 บิต) แต่เรามีที่เก็บแค่ 4 บิต · **ทิ้งตัวทดที่หลุดออกไป** เหลือ **0100** = 4

เช็กกลับ — 7 − 3 = 4 ✓ ตรงเป๊ะ และเราไม่ได้ยืมเลขเลยสักครั้ง

5.5 ทำไมถึง "ทิ้ง" ตัวทดที่หลุดออกไปได้

คำถามนี้ต้องตอบให้ได้ เพราะข้อสอบชอบถามแบบอ้อม ๆ ว่า "ทำไมผลลัพธ์ถึงถูก ทั้งที่ทิ้งบิตไป"

ตัวทดที่หลุดออกจากหลักซ้ายสุดของระบบ 4 บิต มีค่าประจำหลักเท่ากับ $2^4 = 16$ · และ 16 คือ**จำนวนขีดทั้งหมดบนหน้าปัด** — คือหนึ่งรอบพอดี

ตอนเราบวก 13 เข้าไปแทนที่จะลบ 3 เราจงใจเดินเกินไปหนึ่งรอบเต็ม เข็มมันวนกลับมาที่จุดที่ถูกต้องแล้ว · ตัวทดตัวนั้นเป็นแค่ "ป้ายบอกว่าเข็มวนครบรอบ" ไม่ใช่ส่วนหนึ่งของคำตอบ · ทิ้งได้เลย

เทียบกับของใกล้ตัว

นาฬิกาเข็มก็ทำงานแบบเดียวกัน · ตอนนี้ 9 โมง อีก 5 ชั่วโมงเป็นกี่โมง · 9 + 5 = 14 แต่เราตอบ 2 โมง เพราะทิ้ง 12 (หนึ่งรอบ) ไป · ไม่มีใครบอกว่าเราคิดผิด เพราะหน้าปัดมันมีแค่ 12 ขีด

2's complement ก็คือเรื่องเดียวกัน แค่หน้าปัดมี 2^n ขีดแทนที่จะมี 12

ระวัง — "ทิ้งตัวทด" ไม่ได้แปลว่า "ไม่ต้องสนใจตัวทด"

ตัวทดที่หลุดออกไปยังมีประโยชน์อยู่ — ในหัวข้อ 7 เราจะใช้มันคู่กับตัวทดที่เข้าหลักซ้ายสุด เพื่อตรวจว่าคำตอบ**ล้นหรือเปล่า** · ตอนนี้แค่จำไว้ว่า "ไม่เอามาเขียนในคำตอบ" แต่ "อย่าเพิ่งลืมมัน"

5.6 แปลงกลับ — จากคอมพลีเมนต์ไปเป็นเลขจริง

ถ้าเครื่องเก็บ **10111000** ไว้ แล้วโจทย์บอกว่านี่คือเลขในรูป 2's complement · เราจะรู้ได้ยังไงว่ามันคือเลขอะไร

คำตอบง่ายมาก — **ทำ 2's complement ซ้ำอีกครั้ง** จะได้ขนาดของมันกลับมา (เพราะการทำสองครั้งคือการเดินไปครบวงแล้วกลับมาที่เดิม)

โจทย์ 5.3 — 10111000 ในรูป 2's complement มีค่าเท่าไร

ทำอะไร	ได้
ดูบิตซ้ายสุดก่อน — เป็น 1 แปลว่าเลขนี้ติดลบ	ติดลบแน่นอน
① กลับบิตทุกตัว	01000111
② บวก 1	01001000
③ อ่านค่าที่ได้เป็นเลขธรรมดา	64 + 8 = **72**
④ ใส่เครื่องหมายลบกลับเข้าไป	**−72**

คำตอบ

10111000 ในรูป 2's complement มีค่า **−72**

เช็กกลับ — ถ้ามันคือ −72 จริง เอา 72 (คือ 01001000) มาทำ 2's complement ต้องได้ 10111000 กลับมา · ซึ่งเราทำไปแล้วในโจทย์ 5.1 ✓

เช็กอีกทาง — 10111000 อ่านแบบไม่มีเครื่องหมายได้ 184 · 184 − 256 = −72 ✓ (ลบด้วยหนึ่งรอบเต็ม)

สำหรับ 1's complement การแปลงกลับยิ่งง่าย — **กลับบิตอย่างเดียว** ไม่ต้องบวกอะไร เพราะ 1's complement ของ 1's complement คือตัวเดิมเป๊ะ

เช็กลิสต์หัวข้อ 5

1. **1's complement** = กลับบิตทั้งแถว · เช็กได้ด้วย ของเดิม + ตัวกลับ = 1 ทั้งแถว
2. **2's complement** = กลับบิตแล้วบวก 1 · เช็กได้ด้วย ของเดิม + คอมพลีเมนต์ = 2^n
3. ทางลัด: ลอกจากขวาถึงเลข 1 ตัวแรก แล้วกลับที่เหลือ
4. **ลบ = บวกกับ 2's complement ของตัวลบ** แล้ว**ทิ้งตัวทดที่หลุดออกนอกแถว**
5. แปลงกลับ: 2's complement ทำซ้ำอีกครั้ง · 1's complement กลับบิตอีกครั้ง
6. ก่อนทำอะไรทุกครั้ง ให้**ดูว่าโจทย์กำหนดกี่บิต** แล้วเติมศูนย์หน้าให้ครบก่อน · ถ้าทำ complement บนจำนวนบิตผิด คำตอบจะผิดทั้งข้อ

6 เลขติดลบสามแบบ และพิสัยของแต่ละแบบ

ปัญหา: แถวสวิตช์เขียนเครื่องหมายลบไม่ได้

ตั้งแต่ต้นบทเราเขียนแต่เลขบวก แต่ของจริงมันมีเลขติดลบ · เทอร์โมมิเตอร์ในห้องเย็นอ่านได้ −5 องศา · เงินในบัญชีติดลบได้

ปัญหาคือแถวสวิตช์ไม่มีช่องให้เขียนขีดลบ มันมีแค่ 0 กับ 1 เท่านั้น

ลองก่อน

ถ้าน้องเป็นคนออกแบบเครื่อง มีสวิตช์ 8 ตัว จะทำยังไงให้มันบอกได้ว่าเลขนี้ติดลบ · คิดแบบง่ายที่สุดที่นึกออกเลย

ส่วนใหญ่ตอบว่า "เอาสวิตช์ตัวหนึ่งไว้บอกเครื่องหมาย" ใช่ไหม — ถูกต้อง และนั่นคือวิธีแรกในสามวิธีที่เราจะเรียนกันเลย · แต่มันมีปัญหาซ่อนอยู่ เดี๋ยวจะให้ดู

โลกมีวิธีเขียนเลขติดลบในฐานสองอยู่ **3 แบบ** ทั้งสามแบบใช้ข้อตกลงร่วมกันข้อหนึ่งคือ — **บิตซ้ายสุด (MSB) คือบิตเครื่องหมาย** เป็น 0 แปลว่าบวก เป็น 1 แปลว่าลบ · ต่างกันตรงที่ "บิตที่เหลือ" แปลว่าอะไร

สามแบบ

① **แบบเครื่องหมาย-ขนาด** (sign-magnitude) — บิตที่เหลือคือขนาดตรง ๆ ไม่ต้องทำอะไร

② **แบบ 1's complement** — เลขบวกเหมือนแบบ ① · เลขลบคือ 1's complement ของเลขบวกตัวนั้น

③ **แบบ 2's complement** — เลขบวกเหมือนแบบ ① · เลขลบคือ 2's complement ของเลขบวกตัวนั้น

สังเกตว่า **เลขบวกเขียนเหมือนกันทั้งสามแบบ** ต่างกันเฉพาะฝั่งลบ

ทำให้ดูทุกบรรทัด — เขียน −39 ทั้งสามแบบ

โจทย์ 6.1 — เขียน −39 เป็นเลข 8 บิต ทั้งสามแบบ

ขั้น	ทำอะไร	ได้
①	ทำเลขบวกก่อนเสมอ — เขียน 39 เป็นฐานสอง: 32 + 4 + 2 + 1	00100111
②	เช็กว่าครบ 8 บิตไหม (เติมศูนย์หน้าให้ครบ)	ครบแล้ว

แบบที่ 1 — เครื่องหมาย-ขนาด

เอา 00100111 มา เปลี่ยนเฉพาะบิตซ้ายสุดเป็น 1 · บิตที่เหลือไม่แตะเลย	1 0100111

แบบที่ 2 — 1's complement

กลับบิตของ 00100111 ทุกตัว	11011000

แบบที่ 3 — 2's complement

เอา 1's complement ข้างบนมาบวก 1: 11011000 + 1	11011001
(ทางลัด: ลอก 1 ตัวแรกจากขวาของ 00100111 คือ "1" แล้วกลับที่เหลือ 0010011 → 1101100 ต่อกันได้ 11011001 เหมือนกัน)	11011001

คำตอบ

−39 เขียนได้เป็น **10100111** (เครื่องหมาย-ขนาด) · **11011000** (1's complement) · **11011001** (2's complement)

สังเกต — ทั้งสามแบบขึ้นต้นด้วย 1 เหมือนกัน เพราะเป็นเลขลบทั้งคู่ · แต่บิตที่เหลือคนละเรื่องเลย · **ถ้าโจทย์ไม่บอกว่าใช้แบบไหน จะตอบไม่ได้**

เช็กกลับแบบที่ 3 — 00100111 (39) + 11011001 (217) = 256 = 2^8 ✓

อ่านกลับ — บิตชุดเดียวกัน อ่านได้สามค่า

ข้อสอบชอบถามกลับทาง: ให้บิตมา แล้วถามว่ามีค่าเท่าไรในแต่ละแบบ · วิธีอ่านของแต่ละแบบต่างกัน ต้องแยกให้ออก

โจทย์ 6.2 00110110 และ 10110110 มีค่าเท่าไร ในแต่ละแบบ

(ก) 00110110 — บิตซ้ายสุดเป็น 0 แปลว่าบวก

เลขบวกเขียนเหมือนกันทั้งสามแบบ เพราะฉะนั้นอ่านตรง ๆ ได้เลย: 32 + 16 + 4 + 2 = **54**

คำตอบ: **+54 ทั้งสามแบบ**

(ข) 10110110 — บิตซ้ายสุดเป็น 1 แปลว่าลบ ทั้งสามแบบให้คนละคำตอบ

แบบ	วิธีอ่าน	ได้
เครื่องหมาย-ขนาด	ตัดบิตเครื่องหมายทิ้ง เหลือ 0110110 = 32+16+4+2 = 54 แล้วใส่ลบ	**−54**
1's complement	กลับบิตทั้งแถวได้ 01001001 = 64+8+1 = 73 แล้วใส่ลบ	**−73**
2's complement	กลับบิตได้ 01001001 บวก 1 ได้ 01001010 = 64+8+2 = 74 แล้วใส่ลบ	**−74**

วิธีอ่านอีกแบบที่อาจารย์สอน (ใช้ค่าประจำหลักติดลบ)

ให้บิตซ้ายสุดมีค่าประจำหลักเป็น **ลบ** แล้วบวกทุกหลักตามปกติ

ค่าประจำหลักของ 10110110	−128, 64, 32, 16, 8, 4, 2, 1
หยิบเฉพาะหลักที่เป็น 1	−128 + 32 + 16 + 4 + 2
รวมได้	**−74** ← นี่คือคำตอบของแบบ 2's complement
แบบ 1's complement ให้เอาผลข้างบน **บวก 1**	−74 + 1 = **−73** ตรงกับตารางข้างบน

คำตอบ

10110110 = **−54** (เครื่องหมาย-ขนาด) · **−73** (1's complement) · **−74** (2's complement)

เช็กกลับ — สามคำตอบต้องเรียงแบบนี้เสมอเมื่อเป็นเลขลบ: 2's complement น้อยที่สุด แล้ว 1's complement มากกว่าอยู่ 1 · ถ้าได้ไม่เรียงแบบนี้ แสดงว่าคำนวณพลาด

เดินให้เห็นตอนพัง — ปัญหาศูนย์สองตัว

ทีนี้มาดูปัญหาที่ซ่อนอยู่ในแบบ "เครื่องหมาย-ขนาด" ที่น้องคิดเองได้ตอนแรก

ใช้ระบบ 4 บิต · เขียน **+0** ดู — เครื่องหมายบวกคือ 0 ขนาดคือ 0 ได้ **0000** · แล้วเขียน **−0** ดูบ้าง — เครื่องหมายลบคือ 1 ขนาดคือ 0 ได้ **1000**

รหัสสองอันนี้ต่างกันคนละบิต แต่หมายถึงจำนวนเดียวกันคือศูนย์ · เครื่องที่ต้องเช็กว่า "ผลลัพธ์เป็นศูนย์ไหม" จึงต้องเช็กสองแบบเสมอ วงจรใหญ่ขึ้น ช้าลง และเขียนโปรแกรมพลาดง่าย

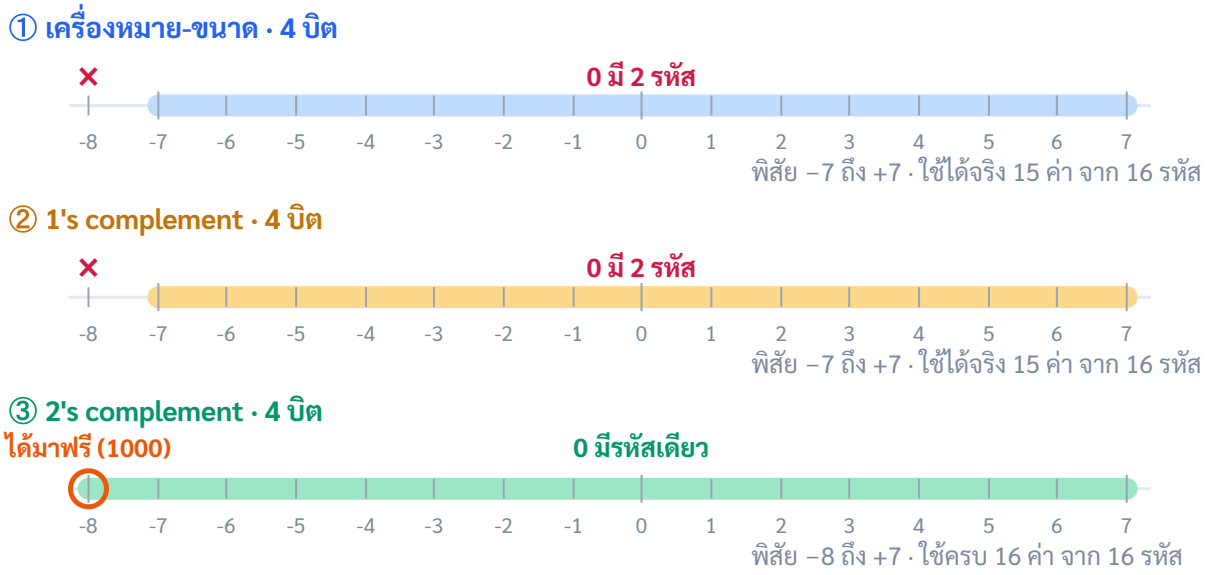
แบบ 1's complement ก็เป็นเหมือนกัน — **+0 = 0000** ส่วน **−0** คือ 1's complement ของ 0000 ก็คือ **1111** · มีสองรหัสอีกแล้ว

แต่แบบ 2's complement ไม่มีปัญหานี้ · ลองหา −0 ดู: กลับบิตของ 0000 ได้ 1111 บวก 1 ได้ 10000 ซึ่งเกิน 4 บิต · ทิ้งตัวทดที่หลุดออกไป เหลือ **0000** เท่าเดิม · **ศูนย์มีรหัสเดียว**

และนี่คือที่มาของ "ลบได้ไกลกว่าหนึ่งตัว"

4 บิตมีรหัสทั้งหมด 16 แบบ · สองแบบแรกเปลืองรหัสไปหนึ่งอันกับศูนย์ที่ซ้ำ เหลือแทนค่าได้จริง **15 ค่า** คือ −7 ถึง +7

2's complement ไม่เปลืองรหัสเลย ใช้ครบ **16 ค่า** · รหัส 1000 ที่แบบอื่นเอาไปทำ −0 ก็ว่างอยู่ จึงเอามาใช้เป็น **−8** ได้พอดี



รูป 1.12 · พิสัยของทั้งสามแบบในระบบ 4 บิต — ฝั่งบวกเท่ากันหมด ต่างกันแค่ปลายฝั่งลบ

สูตรพิสัย — ต้องจำให้ได้

พิสัยของเลข n บิต

จำนวนรหัสทั้งหมด = 2^n แบบ (เช่น 8 บิต ได้ 256 แบบ)

เครื่องหมาย-ขนาด และ **1's complement** · พิสัย $-(2^{n-1}-1)$ ถึง $+(2^{n-1}-1)$

2's complement · พิสัย $-(2^{n-1})$ ถึง $+(2^{n-1}-1)$

ฝั่งบวกสูงสุดเหมือนกันทั้งสามแบบ · ต่างกันแค่ฝั่งลบของ 2's complement ที่ไปได้ไกลกว่าอีกหนึ่งค่า

จำนวนบิต	รหัสทั้งหมด	เครื่องหมาย-ขนาด	1's complement	2's complement
4	16	−7 ถึง +7	−7 ถึง +7	**−8 ถึง +7**
8	256	−127 ถึง +127	−127 ถึง +127	**−128 ถึง +127**
16	65,536	−32,767 ถึง +32,767	−32,767 ถึง +32,767	**−32,768 ถึง +32,767**

โจทย์ 6.3 ต้องใช้กี่บิต ถึงจะเก็บ −100 ในรูป 2's complement ได้

ลอง n	พิสัย $-(2^{n-1})$ ถึง $+(2^{n-1}-1)$	เก็บ −100 ได้ไหม
7	−64 ถึง +63	ไม่ได้ (−100 ต่ำกว่า −64)
8	−128 ถึง +127	**ได้**

คำตอบ

ต้องใช้ **8 บิต**

เช็กกลับ — เขียน −100 ออกมาดูเลย: 100 = 01100100 · กลับบิตได้ 10011011 · บวก 1 ได้ **10011100** · ลองอ่านกลับด้วยค่าประจำหลักติดลบ: −128 + 16 + 8 + 4 = −100 ✓ พอดี 8 บิต

ตารางเทียบสามแบบ (4 บิต) — ดูให้เห็นความต่างทั้งภาพ

ค่าฐานสิบ	2's complement	1's complement	เครื่องหมาย-ขนาด
−8	**1000**	เขียนไม่ได้	เขียนไม่ได้
−7	1001	1000	1111
−6	1010	1001	1110
−5	1011	1010	1101
−4	1100	1011	1100
−3	1101	1100	1011
−2	1110	1101	1010
−1	1111	1110	1001
0	0000	0000 หรือ 1111	0000 หรือ 1000
1	0001	0001	0001
2	0010	0010	0010

ค่าฐานสิบ	2's complement	1's complement	เครื่องหมาย-ขนาด
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

ดูคอลัมน์ 2's complement แล้วอ่านจากล่างขึ้นบน จะเห็นว่ามันนับ 0000, 0001, ... , 0111 แล้วข้ามไป 1000, 1001, ... , 1111 ต่อเนื่องกันเป็นวงเดียวเหมือนหน้าปัดในรูป 1.11 เป๊ะ · นี่คือเหตุผลที่วงจรบวกตัวเดียวใช้ได้ทั้งเลขบวกและเลขลบ ไม่ต้องมีวงจรพิเศษ

ส่วนคอลัมน์เครื่องหมาย-ขนาด ลองไล่ดู — ฝั่งลบมันนับ**ถอยหลัง** (−1 คือ 1001 แต่ −7 คือ 1111) ถ้าเอาสองตัวนี้มาบวกกันตรง ๆ ด้วยวงจรบวกธรรมดา จะได้คำตอบมั่วทันที · นี่คือเหตุผลที่เครื่องจริงเลิกใช้แบบนี้ไปแล้ว

เช็กลิสต์หัวข้อ 6

1. อ่านโจทย์ให้ออกก่อนว่าถาม**แบบไหน** — ถ้าไม่บอก ให้ถือว่าเป็น 2's complement (เป็นแบบมาตรฐานของเครื่องจริง)
2. เขียนเลขลบ: **ทำเลขบวกให้ครบจำนวนบิตก่อนเสมอ** แล้วค่อยแปลง
3. อ่านเลขที่ให้มา: ดูบิตซ้ายสุด · เป็น 0 → อ่านตรง ๆ ได้เลย ทั้งสามแบบเท่ากัน · เป็น 1 → ต้องแปลงตามแบบที่โจทย์บอก
4. พิสัย: จำแค่ 2's complement คือ $-(2^{n-1})$ ถึง $+(2^{n-1}-1)$ · อีกสองแบบคือ $\pm(2^{n-1}-1)$
5. เจอโจทย์ "ใช้กี่บิต" ให้ไล่ n ขึ้นทีละตัวแล้วเทียบพิสัย · **หยุดที่ n ตัวแรกที่เก็บได้**
6. เช็กกลับเสมอ: อ่านคำตอบกลับด้วยค่าประจำหลักติดลบ (-2^{n-1} สำหรับบิตซ้ายสุด) ต้องได้เลขเดิม

7 บวกเลขมีเครื่องหมาย และการล้น

ตั้งแต่ตรงนี้ไปเราจะใช้ **2's complement อย่างเดียว** เหมือนที่อาจารย์กำหนดในสไลด์ เพราะเป็นแบบที่เครื่องจริงใช้

ข่าวดี: ไม่ต้องเรียนวิธีบวกใหม่เลย

เสน่ห์ของ 2's complement คือ **เอาสองตัวมาบวกกันตรง ๆ ได้เลย ไม่ต้องดูเครื่องหมาย** วงจรบวกตัวเดียวกับที่เราใช้ในหัวข้อ 4 ใช้ได้หมด · แล้วถ้ามีตัวทดหลุดออกนอกแถว ก็ทิ้งไป

โจทย์ 7.1 บวกเลข 8 บิต 2's complement สี่คู่ (จากสไลด์อาจารย์)

ข้อ	ตัวตั้ง	ตัวบวก	ผลรวมเต็ม	เก็บ 8 บิต	ตรวจด้วยฐานสิบ
(ก)	00000111 +7	00000100 +4	00001011	**00001011**	7 + 4 = **11** ✓
(ข)	00001111 +15	11111010 −6	**1**00001001 มีตัวทดหลุด	**00001001**	15 + (−6) = **9** ✓
(ค)	00010000 +16	11101000 −24	11111000	**11111000**	16 + (−24) = **−8** ✓
(ง)	11111011 −5	11110111 −9	**1**11110010 มีตัวทดหลุด	**11110010**	(−5) + (−9) = **−14** ✓

วิธีเช็กคำตอบแต่ละข้อ

(ก) 00001011 = 8 + 2 + 1 = 11 ✓

(ข) 00001001 = 8 + 1 = 9 ✓ — **มีตัวทดหลุดออกไป แต่คำตอบถูก**

(ค) 11111000 อ่านด้วยค่าประจำหลักติดลบ: −128 + 64 + 32 + 16 + 8 = −8 ✓

(ง) 11110010 อ่านด้วยค่าประจำหลักติดลบ: −128 + 64 + 32 + 16 + 2 = −14 ✓

สี่ข้อนี้มีข้อหนึ่งที่ดูแปลก ๆ อยู่ ลองสังเกตดูก่อนอ่านต่อ

ลองก่อน

ดูข้อ (ข) กับ (ง) ให้ดี — ทั้งสองข้อมีตัวทดหลุดออกนอกแถว 8 บิต แต่คำตอบถูกทั้งคู่ · แล้ว**ตัวทดหลุด** แปลว่าคำตอบผิดหรือเปล่า

ตอบว่าไม่ได้แปลว่าผิด ใช่ไหม — ถูกต้อง · ตัวทดที่หลุดออกไปคือ "เดินครบหนึ่งรอบ" ซึ่งเป็นเรื่องปกติของการบวกเลขลบ (เราเห็นมาแล้วในหัวข้อ 5) · **ตัวทดหลุด ≠ ล้น** ตรงนี้คือจุดที่คนสับสนมากที่สุดทั้งบท

แล้วเมื่อไรคำตอบถึงจะผิดจริง

คำตอบจะผิดเมื่อ **คำตอบจริงอยู่นอกพิสัยที่ 8 บิตเก็บได้** คือเกิน +127 หรือต่ำกว่า −128 · เรียกอาการนี้ว่า **การล้น** (`overflow`) · เมื่อล้น ผลลัพธ์ที่ได้จะ**ใช้ไม่ได้เลย** ไม่ใช่แค่คลาดเคลื่อน

โจทย์ 7.2 01111101 + 00111010 (8 บิต 2's complement) คำตอบใช้ได้ไหม

ทำอะไร	ได้
อ่านตัวตั้ง — บิตซ้ายสุดเป็น 0 แปลว่าบวก	01111101 = 64+32+16+8+4+1 = **+125**
อ่านตัวบวก — บิตซ้ายสุดเป็น 0 แปลว่าบวก	00111010 = 32+16+8+2 = **+58**
บวกฐานสิบไว้ก่อน เพื่อรู้ว่าควรได้เท่าไร	125 + 58 = **183**
183 เกิน +127 ไหม	**เกิน** → ต้องล้นแน่นอน
บวกฐานสองจริง	10110111
อ่านผลลัพธ์แบบมีเครื่องหมาย	−128+32+16+4+2+1 = **−73**

คำตอบ

ผลลัพธ์ **ใช้ไม่ได้** เพราะเกิดการล้น · บวกเลขบวกสองตัวแล้วได้เลขลบ เป็นไปไม่ได้ในโลกจริง

สังเกต — 183 กับ −73 ต่างกัน 256 พอดี ($183 - 256 = -73$) เพราะเข็มบนหน้าปัดเดินเลยจุดแบ่งบวก/ลบไปแล้ว

วิธีตรวจการล้นที่ใช้ได้ทุกกรณี — ดูตัวทดสองตัว

ในห้องสอบเราไม่อยากมานั่งแปลงเป็นฐานสิบทุกข้อ · มีวิธีตรวจที่เร็วกว่าและเป็นวิธีที่วงจรจริงใช้ — **เทียบตัวทดสองตัวที่บิตเครื่องหมาย**

กฎตรวจการล้น

ตัวทดที่เข้าบิตเครื่องหมาย (carry in) **ไม่เท่ากับ ตัวทดที่ออกจากบิตเครื่องหมาย** (carry out) ⇒ **ล้น**

ถ้าเท่ากัน (ทั้งคู่เป็น 0 หรือทั้งคู่เป็น 1) ⇒ **ไม่ล้น คำตอบใช้ได้**

กรณีล้น · +125 + 58

ตัวทด	0	1	1	1	1	0	0	0
+125		0	1	1	1	1	1	0	1
+ 58		0	0	1	1	1	0	1	0
ได้ −73		1	0	1	1	0	1	1	1
		บิตเครื่องหมาย							

ตัวทดเข้าบิตเครื่องหมาย = 1 (วงส้ม) · ตัวทดออก = 0 (วงแดง) · ไม่เท่ากัน ⇒ ล้น

รูป 1.13 · กรณีล้น — วงส้มคือตัวทดที่เข้าบิตเครื่องหมาย วงแดงคือตัวทดที่ออกนอกแถว สองตัวนี้ไม่เท่ากัน

กรณีมีตัวทดหลุด แต่ไม่ล้น · +15 + (−6)

ตัวทด	1	1	1	1	1	1	1	0	0
+15		0	0	0	0	1	1	1	1
+ (−6)		1	1	1	1	1	0	1	0
ได้ +9		0	0	0	0	1	0	0	1
		บิตเครื่องหมาย							

ตัวทดเข้า = 1 (วงส้ม) · ตัวทดออก = 1 (วงเขียว) · เท่ากัน ⇒ ไม่ล้น ทิ้งตัวทดไปได้เลย

รูป 1.14 · คู่เทียบ — ข้อนี้มีตัวทดหลุดออกไปเหมือนกัน แต่ตัวทดสองตัวเท่ากัน คำตอบจึงใช้ได้

กฎลัดที่อาจารย์เขียนในสไลด์ (ใช้ได้ผลเดียวกัน)

ถ้าเครื่องหมายของผลลัพธ์ต่างจากเครื่องหมายของตัวตั้งทั้งสองตัว ⇒ ล้น

ลองกับโจทย์ 7.2: บวก (+) กับ (+) แล้วได้ (−) ⇒ ล้น ✓ · ลองกับข้อ (ข): บวก (+) กับ (−) ได้ (+) — เครื่องหมายผลลัพธ์ตรงกับตัวตั้งตัวหนึ่ง ⇒ ไม่ล้น ✓

ผลพลอยได้ที่ใช้ตัดข้อสอบได้เร็วมาก — บวกเลขที่**เครื่องหมายต่างกัน** จะไม่มีทางล้นเลย เพราะผลลัพธ์ย่อมอยู่ระหว่างสองตัวนั้นเสมอ · การล้นเกิดได้เฉพาะตอนบวกเลขเครื่องหมายเดียวกันเท่านั้น

สามอย่างที่คนสับสนกันประจำ

① **ตัวทดหลุด ≠ ล้น** · ข้อ (ข) และ (ง) มีตัวทดหลุด แต่ไม่ล้น · ต้องเทียบกับตัวทดที่เข้าบิตเครื่องหมายเสมอ

② **ล้นได้โดยไม่มีตัวทดหลุด** · โจทย์ 7.2 ตัวทดออก = 0 (ไม่หลุด) แต่ล้น

③ **ล้นแล้วต้องตอบว่า "ใช้ไม่ได้"** ไม่ใช่ตอบเลขที่ได้ · ถ้าข้อสอบให้ช่องเดียว ให้เขียนบิตที่ได้พร้อมกำกับว่า overflow

ลบเลขมีเครื่องหมาย — แปลงเป็นบวกแล้วทำเหมือนเดิม

วิธีเดียวกับหัวข้อ 5 เป๊ะ — **ทำ 2's complement ของตัวลบ แล้วบวก ทิ้งตัวทดที่หลุด**

โจทย์ 7.3 — **−25 − 19 ในระบบ 8 บิต**

ขั้น	ทำอะไร	ได้
①	เขียน −25 · 25 = 00011001 → กลับบิต 11100110 → +1	11100111
②	เขียน +19	00010011
③	เปลี่ยนเป็นบวก: ทำ 2's complement ของ 19 · กลับบิต 11101100 → +1	11101101
④	บวก 11100111 + 11101101	**1**11010100 (มีตัวทดหลุด)
⑤	ทิ้งตัวทด เหลือ 8 บิต	11010100
⑥	ตรวจการล้น: ตัวทดเข้าบิตเครื่องหมาย = 1 · ตัวทดออก = 1	เท่ากัน ⇒ **ไม่ล้น**

คำตอบ

11010100 · อ่านด้วยค่าประจำหลักติดลบ: −128 + 64 + 16 + 4 = **−44**

เช็กกลับ — −25 − 19 = −44 ✓ และ −44 อยู่ในพิสัย −128 ถึง +127 จริง

เช็กเครื่องหมาย — บวกเลขลบสองตัวได้เลขลบ สมเหตุสมผล ถ้าได้เลขบวกแสดงว่าล้นแน่นอน

คูณและหารเลขมีเครื่องหมาย

ต่างจากบวกลบตรงที่ **คูณหารต้องทำกับเลขขนาดจริง (ไม่ติดลบ) ก่อน แล้วค่อยใส่เครื่องหมายทีหลัง** — เอาเลขติดลบไปคูณกันตรง ๆ ในรูปคอมพลีเมนต์จะได้คำตอบผิด

กฎเครื่องหมาย

เครื่องหมายเหมือนกัน → ผลลัพธ์เป็นบวก · **เครื่องหมายต่างกัน → ผลลัพธ์เป็นลบ** (ต้องทำ 2's complement ของผลลัพธ์ตอนท้าย)

ใช้ได้ทั้งการคูณและการหาร

โจทย์ 7.4 — 0110 × 1100 (4 บิต 2's complement) ตอบเป็น 8 บิต

ขั้น	ทำอะไร	ได้
①	อ่านตัวตั้ง — บิตซ้ายสุด 0 แปลว่าบวก	0110 = **+6**
②	อ่านตัวคูณ — บิตซ้ายสุด 1 แปลว่าลบ · ทำ 2's complement: กลับบิต 0011 บวก 1 ได้ 0100 = 4	1100 = **−4**
③	เครื่องหมายต่างกัน → ผลลัพธ์ต้องเป็นลบ (จดไว้ก่อน)	ลบ
④	คูณด้วยขนาดจริง 6 × 4 = 24 · เขียน 24 เป็น 8 บิต	00011000
⑤	ใส่เครื่องหมายลบ: ทำ 2's complement ของ 00011000 · กลับบิต 11100111 บวก 1	11101000

คำตอบ

0110 × 1100 = **11101000_2** = −24

เช็กกลับ — 11101000 อ่านด้วยค่าประจำหลักติดลบ: −128 + 64 + 32 + 8 = −24 ✓ และ 6 × (−4) = −24 ✓

ถ้าเผลอคูณตรง ๆ — 0110 × 1100 แบบไม่แปลงก่อน จะได้ 6 × 12 = 72 ซึ่งผิด เพราะ 1100 ไม่ได้แปลว่า 12 ในระบบมีเครื่องหมาย

ปิดท้ายด้วยการหาร ซึ่งเครื่องทำด้วยการลบซ้ำ

โจทย์ 7.5 — 01100100 ÷ 00011001 (8 บิต) ด้วยวิธีลบซ้ำ

วิธีที่อาจารย์สอนคือ **ลบตัวหารออกจากตัวตั้งซ้ำ ๆ แล้วนับว่าลบได้กี่ครั้ง** โดยการลบแต่ละครั้งทำด้วยการบวก 2's complement

ครั้งที่	เหลืออยู่	ลบ 25 (บวก 11100111)	ได้
1	100	01100100 + 11100111	75
2	75	01001011 + 11100111	50
3	50	00110010 + 11100111	25
4	25	00011001 + 11100111	0 → หยุด

คำตอบ

ลบได้ **4 ครั้ง** เหลือเศษ 0 · ผลหาร = **00000100_2** = 4

เครื่องหมายเหมือนกัน (บวกทั้งคู่) → ผลหารเป็นบวก ไม่ต้องทำ complement

เช็กกลับ — 100 ÷ 25 = 4 เศษ 0 ✓ · และ 4 × 25 = 100 ตรงกับตัวตั้ง ✓

ข้อเมื่อกี้หารลงตัวพอดี แล้วถ้าหารไม่ลงตัวล่ะ จะต้องหยุดตรงไหน

ถ้าหารไม่ลงตัว

ลองกับ 00110010 ÷ 00001111 (50 ÷ 15) · ลบครั้งที่ 1 เหลือ 35 · ครั้งที่ 2 เหลือ 20 · ครั้งที่ 3 เหลือ 5 · ครั้งที่ 4 จะได้ 5 − 15 = −10 ซึ่ง**ติดลบ**

กฎคือ **หยุดทันทีที่ผลติดลบ และไม่นับครั้งนั้น** · ผลหาร = 3 (00000011) เศษ = 5 (00000101) · เช็กกลับ: 3 × 15 + 5 = 50 ✓

วิธีดูว่าผลติดลบ: ดูบิตซ้ายสุดของผลลบ ถ้าเป็น 1 แปลว่าติดลบ

เช็กลิสต์หัวข้อ 7

1. เขียนเลขทั้งสองให้ครบจำนวนบิตที่โจทย์กำหนดก่อนเสมอ
2. **บวก**: บวกตรง ๆ ไม่ต้องดูเครื่องหมาย · **ลบ**: ทำ 2's complement ของตัวลบก่อน แล้วบวก
3. ทิ้งตัวทดที่หลุดออกนอกแถว แต่**อย่าทิ้งจากความจำ** — ต้องใช้ตรวจการล้น
4. **ตรวจการล้นทุกข้อ**: ตัวทดเข้าบิตเครื่องหมาย เทียบกับ ตัวทดออก · ไม่เท่ากัน = ล้น
5. เช็กซ้ำด้วยเครื่องหมาย: บวก+บวกต้องได้บวก · ลบ+ลบต้องได้ลบ · ถ้าไม่ใช่แปลว่าล้น
6. **คูณ/หาร**: แปลงเป็นขนาดจริงก่อน คำนวณ แล้วค่อยใส่เครื่องหมายตอนท้าย
7. **หยุดเมื่อ** ได้บิตครบตามที่โจทย์กำหนด และตรวจการล้นแล้ว

8 ทศนิยมลอยตัว

อ่านหัวข้อนี้เป็นอันดับสุดท้าย

หัวข้อนี้ **ไม่ได้อยู่ในรายการทบทวนก่อนสอบกลางภาคที่อาจารย์ให้ไว้** · รายการนั้นมีแค่ ระบบเลขฐาน · เลขมีจุด · เลขคณิตฐานสอง · การแปลงฐาน · คอมพลีเมนต์และเลขมีเครื่องหมาย · พิสัย · และเลขคณิตแบบมีเครื่องหมาย

ถ้าเวลาเหลือน้อย ให้ข้ามไปทำหัวข้อ 9 ก่อน แล้วค่อยกลับมาอ่านหัวข้อนี้ · แต่ถ้ามีเวลา ควรอ่าน เพราะมันอยู่ในสไลด์บทที่ 2 และอาจโผล่ในข้อสอบปลายภาค

ปัญหา: เลขใหญ่มากกับเล็กมากในที่เดียวกัน

ในวิชาฟิสิกส์น้องเคยเขียนเลขแบบนี้ใช่ไหม — มวลอิเล็กตรอนเขียนว่า 9.1×10^{-31} แทนที่จะเขียนศูนย์ 30 ตัว · เราไม่ได้เก็บตัวเลขทั้งหมด เราเก็บแค่ **ตัวเลขนัยสำคัญ** กับ **จุดควรอยู่ตรงไหน**

คอมพิวเตอร์ก็ทำแบบเดียวกันเป๊ะ แค่เปลี่ยนจากฐานสิบเป็นฐานสอง · แทนที่จะเก็บ "จุดอยู่หลักที่เท่าไร" ตายตัว มันเก็บว่า "ต้องเลื่อนจุดไปกี่ช่อง" · จุดมันเลย **ลอย** ได้ตามค่า — ที่มาของชื่อ **ทศนิยมลอยตัว** (floating point)

ทำให้เป็นรูปมาตรฐานก่อน

เอา 118.625 มาเขียนเป็นฐานสองก่อน — ฝั่งจำนวนเต็ม 118 = 1110110 · ฝั่งเศษส่วน 0.625 = 0.101 (จากหัวข้อ 3) · รวมได้ **1110110.101_2**

ลองก่อน

ถ้าต้องเลื่อนจุดใน 1110110.101 ให้เหลือเลข 1 ตัวเดียวหน้าจุด ต้องเลื่อนไปทางซ้ายกี่ช่อง

ตอบ 6 ช่องใช่ไหม — จาก 1110110.101 กลายเป็น 1.110110101 · จุดเดินผ่านเลขไป 6 ตัว

ที่น้องเพิ่งทำเรียกว่า **การทำให้เป็นรูปปกติ** (normalize) · รูปที่ได้คือ $1.m \times 2^e$ โดย m คือตัวเลขหลังจุด และ e คือจำนวนช่องที่เลื่อน

เลื่อนซ้าย e เป็นบวก · เลื่อนขวา (กรณีเลขน้อยกว่า 1) e เป็นลบ

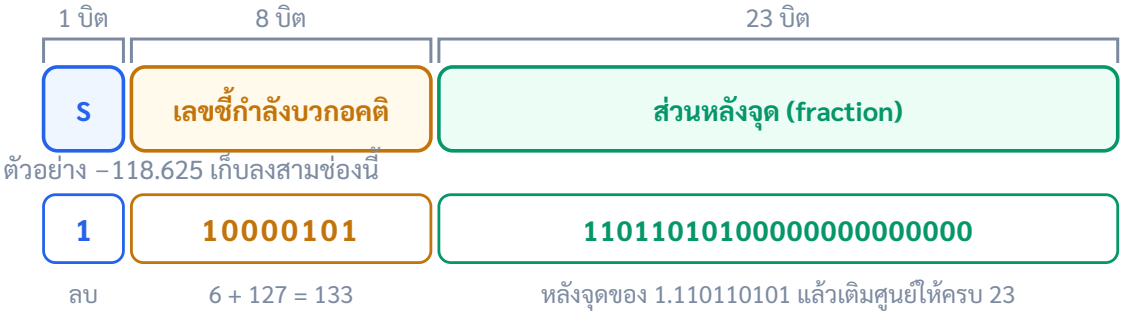
กรณีของเรา: $118.625 = 1.110110101_2 \times 2^6$

ทำไมไม่ต้องเก็บเลข 1 หน้าจุด

สังเกตว่ารูปปกติ**ขึ้นต้นด้วย 1 เสมอ** ไม่มีทางเป็นอย่างอื่น เพราะถ้าเป็น 0 แปลว่ายังเลื่อนไม่พอ

ในเมื่อมันเป็น 1 เสมอ เก็บไปก็เปลืองที่เปล่า ๆ · มาตรฐานจึง**ไม่เก็บบิตนี้** เอาที่ว่างไปเก็บตัวเลขที่มีประโยชน์แทน · เรียกบิตนี้ว่า **บิตซ่อน** (hidden bit) · เวลาแปลงกลับต้องอย่าลืมเติม "1." กลับเข้าไปเอง

สามช่องของทศนิยมลอยตัว



ความละเอียดคู่ (64 บิต) ใช้โครงเดียวกัน แต่เป็น 1 + 11 + 52 บิต และค่าอคติเป็น 1023
เขียนเป็นฐานสิบหกได้ C2ED4000 — จับกลุ่มละ 4 บิตจากซ้ายไปขวา 8 กลุ่มพอดี

รูป 1.15 · โครงสร้าง 32 บิต (ความละเอียดเดี่ยว ตามมาตรฐาน IEEE 754)

ค่าอคติ (bias) — ทำไมต้องบวก 127

เลขชี้กำลังต้องติดลบได้ (เลขที่น้อยกว่า 1 ต้องเลื่อนจุดไปขวา) แต่ช่องเลขชี้กำลังเป็นบิตธรรมดา 8 บิต ที่เก็บได้แค่ 0 ถึง 255

ทางแก้คือ **เลื่อนทั้งช่วงขึ้นมา** — บวก 127 เข้าไปก่อนเก็บ แล้วตอนอ่านก็ลบ 127 ออก · เลขชี้กำลังจริง −3 ก็เก็บเป็น 124 · เลขชี้กำลังจริง +6 ก็เก็บเป็น 133 · ค่า 127 นี้เรียกว่า **ค่าอคติ** (bias)

ทำไมต้องใช้วิธีนี้ ทั้งที่มี 2's complement อยู่แล้ว — เพราะการเก็บแบบบวกอคติทำให้**เปรียบเทียบเลขสองตัวได้ด้วยการดูบิตเรียงกันตรง ๆ** ซึ่งทำให้วงจรเปรียบเทียบง่ายกว่ามาก

โจทย์ 8.1 — แปลง −118.625 เป็นทศนิยมลอยตัวความละเอียดเดี่ยว

ขั้น	ทำอะไร	ได้
①	ดูเครื่องหมาย — เลขติดลบ บิตเครื่องหมายเป็น 1	S = 1
②	แปลงขนาดเป็นฐานสอง · 118 = 64+32+16+4+2 = 1110110	1110110
③	แปลงเศษส่วน · 0.625 × 2 = 1.25 (บิต 1) · 0.25 × 2 = 0.5 (บิต 0) · 0.5 × 2 = 1.0 (บิต 1) หยุด	.101
④	รวมสองฝั่ง	1110110.101

ขั้น	ทำอะไร	ได้
⑤	เลื่อนจุดให้เหลือ 1 ตัวหน้าจุด — เลื่อนซ้าย 6 ช่อง	1.110110101×2^6
⑥	คำนวณช่องเลขชี้กำลัง: 6 + 127 = 133 · เขียน 133 เป็น 8 บิต (128+4+1)	10000101
⑦	ช่องเศษส่วน: เอาเฉพาะหลังจุดของ 1.110110101 คือ 110110101 (9 บิต) เติมศูนย์ทางขวาให้ครบ 23 บิต	11011010100000000000000
⑧	ต่อกันทั้งสามช่อง	1 10000101 11011010100000000000000

คำตอบ

1 10000101 11011010100000000000000

เขียนเป็นฐานสิบหก (จับกลุ่มละ 4 จากซ้าย): 1100 0010 1110 1101 0100 0000 0000 0000 = **$C2ED4000_{16}$**

เช็กกลับ — นับบิตต้องได้ 1 + 8 + 23 = 32 พอดี · ช่องเศษส่วนต้องยาว 23 บิตเป๊ะ ถ้าเกินหรือขาดแปลว่าเติมศูนย์ผิด

เช็กอีกทาง — ประกอบกลับ: เลขชี้กำลัง 133 − 127 = 6 · ค่า = $1.110110101_2 \times 2^6 = 1110110.101_2$ = 118.625 · บิตเครื่องหมายเป็น 1 จึงได้ −118.625 ✓

ทีนี้ลองย้อนศร — ให้บิตมาแล้วถามว่าเป็นเลขอะไร

โจทย์ 8.2 แปลงกลับ — 0 10000011 01010000000000000000000 มีค่าเท่าไร

ขั้น	ทำอะไร	ได้
①	แยกสามช่อง — 1 บิต, 8 บิต, 23 บิต	S=0 · E=10000011 · F=0101000…
②	บิตเครื่องหมายเป็น 0 → เลขบวก	บวก
③	อ่านช่องเลขชี้กำลัง 10000011 = 128 + 2 + 1	133? ไม่ใช่ — **131**
④	ลบค่าอคติออก: 131 − 127	**e = 4**
⑤	เติมบิตซ่อน "1." กลับหน้าช่องเศษส่วน (ศูนย์ท้ายตัดทิ้งได้)	1.0101
⑥	เลื่อนจุดไปขวา 4 ช่องตามเลขชี้กำลัง	10101.
⑦	อ่าน 10101_2 เป็นฐานสิบ: 16 + 4 + 1	**21**

คำตอบ

ค่าคือ **+21.0**

เช็กกลับ — อีกวิธีคือคิดเป็น 1.0101_2 = 1 + 0.25 + 0.0625 = 1.3125 แล้วคูณ $2^4 = 16$ · 1.3125 × 16 = 21 ✓ ตรงกัน

เขียนทั้ง 32 บิตเป็นฐานสิบหกได้ **$41A80000_{16}$**

จุดที่พลาดบ่อย — ขั้น ③ ต้องอ่านค่าประจำหลักให้ถูก 10000011 คือ 128+2+1 = 131 ไม่ใช่ 133 · ถ้าอ่านผิดหนึ่งบิต คำตอบจะผิดไปเท่าตัว เพราะเลขชี้กำลังคูณด้วยกำลังสอง

สองข้อข้างบนเดินครบทุกขั้น ทีนี้มาดูว่าถ้าข้ามขั้นไหนไปแล้วจะพังยังไง

เดินให้เห็นตอนพัง

ถ้า**ลืมเติมบิตซ่อน** ในโจทย์ 8.2 จะได้ $0.0101_2 \times 2^4$ = 0.3125 × 16 = 5 แทนที่จะเป็น 21 · ผิดไปไกลมาก

ถ้า**ลืมลบค่าอคติ** จะได้ 1.0101×2^{131} ซึ่งเป็นเลขมหาศาล · เห็นคำตอบใหญ่ผิดปกติเมื่อไร ให้กลับไปเช็กขั้นลบ 127 ก่อนเลย

ถ้า**เติมศูนย์ผิดด้าน**ในช่องเศษส่วน (เติมซ้ายแทนขวา) ค่าจะเพี้ยนทั้งหมด — ช่องเศษส่วนคือเลขหลังจุด กฎเดียวกับหัวข้อ 3 คือเติมศูนย์ที่ปลายขวาเสมอ

เช็กลิสต์หัวข้อ 8

1. แปลงขนาดเป็นฐานสองให้ครบทั้งสองฝั่งของจุดก่อน
2. เลื่อนจุดให้เหลือ **1 ตัวเดียวหน้าจุด** · จำนวนช่องที่เลื่อนคือ e (ซ้ายเป็นบวก ขวาเป็นลบ)
3. ช่องเลขชี้กำลัง = $e + 127$ (ความละเอียดเดี่ยว) หรือ $e + 1023$ (ความละเอียดคู่)
4. ช่องเศษส่วน = เฉพาะหลังจุด **เติมศูนย์ทางขวา**ให้ครบ 23 บิต (หรือ 52 บิต)
5. แปลงกลับ: ลบค่าอคติ → **เติม "1." กลับเข้าไป** → เลื่อนจุดตาม e
6. เช็กจำนวนบิตทุกครั้ง: 1 + 8 + 23 = 32 · 1 + 11 + 52 = 64

9 เช็กลิสต์ก่อนเข้าห้องสอบ

สามบรรทัดที่ต้องเขียนก่อนลงมือทุกข้อ

ข้อสอบเรื่องตัวเลขที่ทำผิด ส่วนใหญ่ไม่ได้ผิดเพราะคิดเลขพลาด แต่ผิดเพราะ**ทำผิดระบบ** — ใช้จำนวนบิตผิด หรือใช้รูปแบบเลขลบผิดแบบ · เพราะฉะนั้นก่อนจะแตะตัวเลขแม้แต่ตัวเดียว ให้เขียนสามบรรทัดนี้ไว้มุมกระดาษก่อน

1 **ฐานอะไร**

ดูเลขห้อยของโจทย์ให้ครบทุกตัว · ถ้าโจทย์เขียน 1011 เฉย ๆ ให้ดูบริบทว่าหมายถึงฐานสองหรือฐานสิบ · ถ้าเห็นตัวอักษร A–F แปลว่าฐานสิบหกแน่นอน

2 **กี่บิต**

โจทย์บอก 4 บิต 8 บิต หรือไม่บอก · ถ้าไม่บอกแต่ให้เลขมา 8 ตัว ก็คือ 8 บิต · **เติมศูนย์หน้าให้ครบก่อนทำอะไรทั้งสิ้น** เพราะ complement บนจำนวนบิตต่างกันให้คำตอบต่างกัน

3 **มีเครื่องหมายไหม และแบบไหน**

ถ้าโจทย์พูดถึงเลขลบ ต้องรู้ว่าใช้ เครื่องหมาย-ขนาด, 1's complement หรือ 2's complement · ถ้าไม่ระบุและเป็นเรื่องเลขคณิต ให้ถือว่า **2's complement**

เจอโจทย์แบบไหน ให้ทำอะไรก่อน

โจทย์ถามแบบนี้	บรรทัดแรกที่เขียน	เช็กคำตอบยังไง
ฐานสอง/แปด/สิบหก → ฐานสิบ	วางค่าประจำหลักไว้เหนือทุกหลัก	นับจำนวนหลัก คำตอบต้องอยู่ในช่วง r^{n-1} ถึง $r^n - 1$
ฐานสิบ → ฐานอื่น	ตั้งหารซ้ำ (จำนวนเต็ม) และคูณซ้ำ (เศษส่วน) แยกกันคนละคอลัมน์	แปลงคำตอบย้อนกลับเป็นฐานสิบ ต้องได้เลขเดิม
ฐานสอง ↔ ฐานแปด/สิบหก	ขีดเส้นแบ่งกลุ่มจากจุดออกไป ทีละ 3 หรือ 4	นับบิตต่อกลุ่มให้ครบ · เช็กด้วยฐานสิบอีกทีถ้ามีเวลา
บวก ลบ คูณ หาร ฐานสอง	เขียนแถวตัวทด/ตัวยืมไว้ก่อนลงมือ	ลบ → บวกกลับ · หาร → คูณกลับ · บวก คูณ → แปลงเป็นฐานสิบ
หา 1's / 2's complement	เติมศูนย์หน้าให้ครบจำนวนบิตก่อน	ของเดิม + คอมพลีเมนต์ ต้องได้ 1 ทั้งแถว (1's) หรือ 2^n (2's)
อ่านค่าเลขมีเครื่องหมาย	ดูบิตซ้ายสุดก่อน แล้วเขียนว่า "บวก" หรือ "ลบ"	อ่านซ้ำด้วยค่าประจำหลักติดลบ (-2^{n-1} ที่บิตซ้ายสุด)
พิสัย / ใช้กี่บิต	เขียนสูตร $-(2^{n-1})$ ถึง $+(2^{n-1}-1)$ ลงไปเลย	ไล่ n ทีละตัวแล้วเทียบ หยุดที่ตัวแรกที่พอ
บวกเลขมีเครื่องหมาย	บวกตรง ๆ แล้ววงตัวทดสองตัวที่บิตเครื่องหมาย	ตัวทดเข้า = ตัวทดออก → ใช้ได้ · ไม่เท่า → ล้น
ทศนิยมลอยตัว	แยกกระดาษเป็น 3 ช่อง S / E / F ก่อน	นับบิตรวมต้องได้ 32 (หรือ 64) เป๊ะ

กับดักที่เสียคะแนนบ่อยที่สุด (อ่านซ้ำก่อนเข้าห้อง)

กับดัก	อาการ	ทางกัน
จับกลุ่มผิดด้าน	บิตไม่ครบกลุ่ม แล้วเติมศูนย์ผิดข้าง	นับ**ออกจากจุด**เสมอ · ซ้ายเติมซ้าย ขวาเติมขวา
อ่านเศษกลับหัว	หารซ้ำแล้วอ่านเศษจากบนลงล่าง	จำคู่กัน: หาร→อ่านขึ้น · คูณ→อ่านลง
ทำ complement บนจำนวนบิตผิด	ทำ 2's complement ของ 39 แบบ 6 บิต ทั้งที่โจทย์ให้ 8 บิต	เติมศูนย์หน้าให้ครบเป็นขั้นแรกเสมอ
1 + 1 + 1 ในการบวก	ตอบ 0 ทด 1 ทั้งที่ต้องเป็น 1 ทด 1	เขียนเป็น 11_2 ก่อน แล้วค่อยแยกว่าเขียนอะไร ทดอะไร
สับสนตัวทดหลุดกับการล้น	เห็นตัวทดหลุดแล้วตอบว่าล้น	ต้องเทียบ**สองตัว**เสมอ · ตัวทดหลุดตัวเดียวไม่บอกอะไร
คูณเลขลบตรง ๆ	เอา 1100 ไปคูณโดยไม่แปลงเป็น 4 ก่อน	คูณ/หาร ต้องแปลงเป็นขนาดจริงก่อนเสมอ
ลืมบิตซ่อนในทศนิยมลอยตัว	คำตอบเล็กกว่าที่ควรเป็นประมาณครึ่งหนึ่งหรือมากกว่า	เติม "1." กลับเข้าไปทุกครั้งตอนแปลงกลับ
เขียนเลขเกินฐาน	มีเลข 2 ในฐานสอง หรือเลข 8 ในฐานแปด	ตัวเลขในฐาน r ต้อง $0 \le a_i < r$ เสมอ

สรุปทั้งบท — ท่องเท่านี้พอ

ค่าประจำหลัก

$N = a_{n-1}r^{n-1} + \cdots + a_1 r^1 + a_0 r^0$ · ฝั่งขวาของจุดคือ $r^{-1}, r^{-2}, \ldots$ · **ฐานสอง: 1 2 4 8 16 32 64 128 · หลังจุด: 1/2 1/4 1/8 1/16**

แปลงฐาน

ฐานสิบ → ฐานอื่น: **หารซ้ำเก็บเศษ อ่านล่างขึ้นบน** · เศษส่วน: **คูณซ้ำเก็บหน้าจุด อ่านบนลงล่าง**

ฐานสอง ↔ ฐานแปด **กลุ่มละ 3** ($2^3 = 8$) · ฐานสอง ↔ ฐานสิบหก **กลุ่มละ 4** ($2^4 = 16$) · **นับออกจากจุดเสมอ**

คอมพลีเมนต์

1's = **กลับบิต** · 2's = **กลับบิตแล้ว +1** (หรือลอกจากขวาถึง 1 ตัวแรก แล้วกลับที่เหลือ)

เช็ก: เดิม + 1's = 1 ทั้งแถว · เดิม + 2's = 2^n

เลขมีเครื่องหมาย n บิต

เครื่องหมาย-ขนาด และ 1's: $-(2^{n-1}-1)$ **ถึง** $+(2^{n-1}-1)$ · ศูนย์มีสองรหัส

2's: $-(2^{n-1})$ **ถึง** $+(2^{n-1}-1)$ · ศูนย์มีรหัสเดียว · **8 บิต = −128 ถึง +127**

อ่านค่า 2's: ให้บิตซ้ายสุดมีค่า -2^{n-1} แล้วบวกทุกหลักตามปกติ · 1's ให้เอาผลนั้น **+1**

เลขคณิตมีเครื่องหมาย

ลบ = **บวกกับ 2's complement ของตัวลบ** · ทิ้งตัวทดที่หลุด

ล้นเมื่อ **ตัวทดเข้าบิตเครื่องหมาย ≠ ตัวทดออก** · หรือ **บวกเครื่องหมายเดียวกันแล้วได้เครื่องหมายตรงข้าม**

บวกเลข**คนละเครื่องหมาย**ไม่มีทางล้น · คูณ/หาร ต้องแปลงเป็นขนาดจริงก่อน แล้วใส่เครื่องหมายทีหลัง

ทศนิยมลอยตัว (ไม่อยู่ในรายการทบทวนกลางภาค)

32 บิต = **1 + 8 + 23** อคติ **127** · 64 บิต = **1 + 11 + 52** อคติ **1023**

รูปปกติ $1.m \times 2^e$ · ช่องเลขชี้กำลัง = e + อคติ · **ไม่เก็บเลข 1 หน้าจุด** ต้องเติมเองตอนแปลงกลับ

บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด

บทที่แล้วเราแปลงเลขไปมาระหว่างฐานสิบกับฐานสอง บทนี้เปลี่ยนคำถาม ไม่ใช่ "เลขนี้มีค่าเท่าไร" แต่เป็น **"เราจะจัดบิตยังไงให้งานที่ต้องทำมันง่ายขึ้น"**

คำว่า **รหัส** (code) ในบทนี้แปลว่า ข้อตกลงว่าบิตกลุ่มหนึ่งแทนอะไร แค่นั้นเอง ไม่ได้แปลว่าความลับหรือการเข้ารหัสลับอะไรทั้งนั้น

เราจะเจอรหัสห้าแบบ แต่ละแบบเกิดมาแก้ปัญหาคนละอย่าง ถ้าจำว่า "แบบนี้เกิดมาแก้ปัญหาอะไร" ได้ ข้อสอบจะเดาง่ายขึ้นมาก เพราะโจทย์มักจะบอกปัญหามาก่อนแล้วถามว่าใช้อะไร

แผนที่ของบทนี้

BCD — เกิดมาเพื่อให้เอาเลขฐานสองไปขึ้นจอตัวเลขได้ง่าย ๆ

Gray code — เกิดมาเพื่อให้เซนเซอร์อ่านตำแหน่งแล้วไม่เพี้ยน

Parity — เกิดมาเพื่อ *จับได้* ว่าบิตพลิกระหว่างทาง

CRC — เกิดมาเพื่อจับให้ได้แม้บิตพลิกหลายตัวติดกัน

Hamming code — เกิดมาเพื่อ *บอกตำแหน่ง* ที่พลิก แล้วแก้กลับได้เอง

1 BCD — เลขฐานสิบที่เก็บทีละหลัก

ของจริงก่อน: นาฬิกาดิจิทัลบนโต๊ะ

ลองนึกถึงนาฬิกาดิจิทัลที่ขึ้นเลข **12:45** หน้าจอมันมีช่องตัวเลขสี่ช่อง แต่ละช่องเป็นหลอดเจ็ดขีดที่ทำได้อย่างเดียวคือแสดงเลขหลักเดียว ตั้งแต่ 0 ถึง 9

ช่องแรกไม่รู้จักและไม่สนใจเลยว่าช่องอื่นแสดงอะไรอยู่ มันได้รับคำสั่งมาว่า "ขึ้นเลข 1" ก็ขึ้นเลข 1 เท่านั้น แต่ละช่องมีสายของตัวเอง มีวงจรขับของตัวเอง

ทีนี้ ถ้าเราเก็บเลข 1245 ไว้เป็นเลขฐานสองแบบปกติ มันคือ 10011011101 ซึ่งเป็นก้อนเดียวยาว ๆ พอจะเอาไปขึ้นจอ วงจรต้องหารเอา 1000 หารเอา 100 หารเอา 10 ทีละครั้งเพื่อแยกหลักออกมา วงจรหารในฮาร์ดแวร์นั้นแพงและช้า

ลองคิดในหัวก่อน ไม่ต้องเขียน — ถ้าช่องตัวเลขหนึ่งช่องต้องแสดงได้แค่ 0 ถึง 9 และเราจะส่งสายไฟเข้าไปบอกมันว่าให้ขึ้นเลขอะไร

เราต้องใช้สายกี่เส้น (กี่บิต) ต่อหนึ่งช่อง ถึงจะบอกครบสิบค่า?

ตอบ 4 เส้นใช่ไหม ที่น้องเพิ่งคิดนั่นแหละคือ BCD เต็ม ๆ ว่า `Binary-Coded Decimal` แปลตรงตัวว่า "เลขฐานสิบที่เขียนด้วยรหัสฐานสอง"

เหตุผลที่เป็น 4 เพราะ 3 บิตให้ได้ 8 แบบ ซึ่งไม่พอสำหรับสิบค่า ส่วน 4 บิตให้ได้ 16 แบบ ซึ่งพอแล้วและเหลือด้วย

กติกาของ BCD

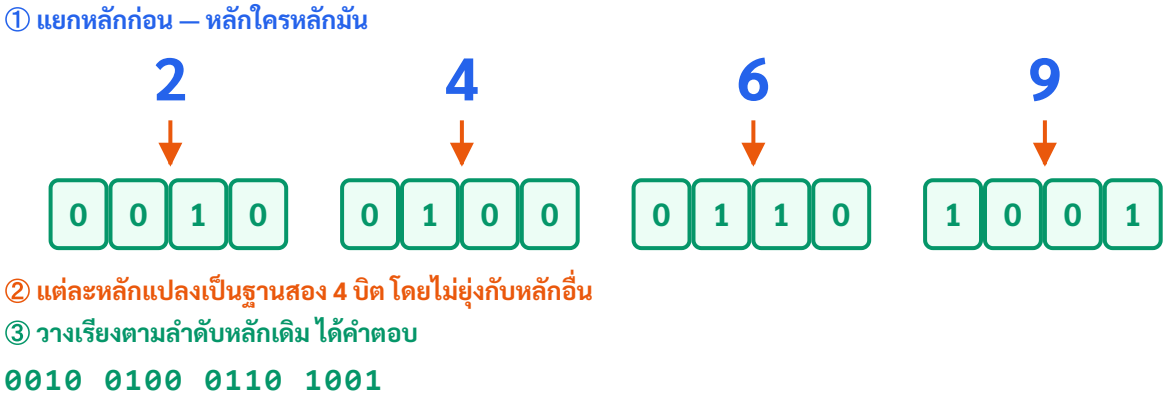
เขียนเลขฐานสิบทีละหลัก **หลักใครหลักมัน** แต่ละหลักแปลงเป็นฐานสอง 4 บิต แล้ววางเรียงกันตามลำดับหลักเดิม

หลักฐานสิบ	รหัส BCD	หลักฐานสิบ	รหัส BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

ตารางนี้ไม่ต้องท่อง เพราะมันคือเลขฐานสอง 4 บิตของ 0 ถึง 9 ตรง ๆ ที่เราแปลงเป็นอยู่แล้วจากบทที่แล้ว

ทำให้ดู: 2469 เป็น BCD

โจทย์นี้มาจากสไลด์ในคาบ เราจะเดินทีละหลัก ไม่ข้ามขั้นแม้แต่หลักเดียว



2469 เป็น BCD — สี่หลัก สี่กล่อง ไม่มีการทดข้ามกล่อง

สังเกตว่าตอนแปลง เราไม่เคยต้องรู้เลยว่าเลขทั้งก้อน 2469 มีค่าเท่าไร รู้แค่ว่าหลักนี้คือ 4 ก็เขียน 0100 ได้แล้ว นี่คือข้อดีทั้งหมดของ BCD

ถ้าเทียบกับฐานสองธรรมดา 2469 คือ 100110100101 ซึ่งต้องเอาทั้งก้อนมาหารสองไปเรื่อย ๆ ถึงจะได้ และถ้าเปลี่ยนแค่เลขหลักหน่วย บิตจะเปลี่ยนไปแทบทั้งก้อน

มีจุดทศนิยมก็ไม่ต่างกัน

BCD ไม่สนใจว่าหลักไหนอยู่ก่อนหรือหลังจุด มันแปลงทุกหลักด้วยกติกาเดียวกัน แล้วลอกจุดทศนิยมลงมาไว้ที่เดิม

17.1 เป็น BCD

```
หลัก 1   ->  0001
หลัก 7   ->  0111
จุด      ->  .        (ลอกลงมาที่เดิม)
หลัก 1   ->  0001
─────────────────────
ได้      0001 0111 . 0001
```

ตรวจกลับเข้าโจทย์เดิม อ่าน 0001 0111 . 0001 ทีละกลุ่มสี่บิต ได้ 1, 7, จุด, 1 นั่นคือ 17.1 ตรงกับโจทย์

อ่านกลับจาก BCD เป็นฐานสิบ

ทิศกลับกันง่ายกว่าอีก ตัดเป็นกลุ่มละ 4 บิต **โดยนับจากจุดทศนิยมออกไปทั้งสองทาง** แล้วแปลงทีละกลุ่ม

1001 0010 0110 . 0101 0001 (BCD) เป็นฐานสิบ

```
1001  ->  9
0010  ->  2
0110  ->  6
.     ->  .
0101  ->  5
0001  ->  1
──────────────
ได้   926.51
```

ตรวจกลับ เอา 926.51 แปลงเป็น BCD ใหม่ ได้ 1001 0010 0110 . 0101 0001 ตรงกับโจทย์ที่ให้มา

ลองทำสี่ข้อนี้ก่อนดูเฉลย (ชุดเดียวกับในสไลด์)

1) 8503 เป็น BCD · 2) 60.75 เป็น BCD

3) 0011 1000 0100 (BCD) เป็นฐานสิบ · 4) 1001 0111 . 0010 0101 (BCD) เป็นฐานสิบ

1) 1000 0101 0000 0011 (8→1000, 5→0101, 0→0000, 3→0011 — เลขศูนย์ต้องเขียน 0000 ห้ามข้าม)

2) 0110 0000 . 0111 0101 (6, 0, จุด, 7, 5)

3) 384 (0011→3, 1000→8, 0100→4)

4) 97.25 (1001→9, 0111→7, จุด, 0010→2, 0101→5)

ทำไม 1010 ถึง 1111 ถึงใช้ไม่ได้

กล่องหนึ่งกล่องมี 4 บิต เขียนได้ทั้งหมด 16 แบบ แต่เลขฐานสิบมีแค่ 10 ตัว คือ 0 ถึง 9

แปลว่ามีรูปแบบเหลือ 16 − 10 = 6 แบบที่ไม่มีเลขฐานสิบตัวไหนจับจอง ได้แก่ 1010, 1011, 1100, 1101, 1110 และ 1111

หกแบบนี้เรียกว่า **รหัสต้องห้าม** (`invalid / forbidden code`) ถ้าเจอกลุ่มสี่บิตหน้าตาแบบนี้ในสายที่บอกว่าเป็น BCD แปลว่ามีอะไรผิดแน่นอน

เก็บข้อสังเกตนี้ไว้ เพราะเดี๋ยวมันจะกลายเป็น **สัญญาณเตือน** ที่ทำให้วงจรบวก BCD รู้ตัวว่าต้องแก้ผลลัพธ์

ลองคิดก่อน — น้องเห็นสายข้อมูล BCD มาแบบนี้ **`0001 1001`**

ถ้าน้องเผลออ่านมันเป็นเลขฐานสองธรรมดาทั้งก้อน จะได้เท่าไร และค่าที่ถูกต้องจริง ๆ คือเท่าไร?

อ่านเป็นฐานสองธรรมดา 00011001 ได้ **25** แต่อ่านเป็น BCD คือ 0001→1, 1001→9 ได้ **19**

ต่างกัน 6 พอดี และถ้าอยากเขียน 25 เป็น BCD จริง ๆ ต้องเขียน 0010 0101 ซึ่งหน้าตาไม่เหมือนกันเลย

กับดัก

BCD กับเลขฐานสองธรรมดา **เขียนด้วยตัวเลข 0 กับ 1 เหมือนกันเป๊ะ** ดูจากหน้าตาไม่ออก โจทย์ต้องบอกมาเองว่าเป็นแบบไหน

ในข้อสอบจะมีคำว่า BCD หรือห้อยตัว `BCD` ไว้ ถ้าไม่มีให้ถือว่าเป็นฐานสองธรรมดา

เดินให้เห็นตอนพัง: เอา BCD ไปบวกตรง ๆ

วงจรบวกในเครื่องบวกฐานสองเป็น มันไม่รู้จักคำว่า BCD มันเห็นสี่บิตก็บวกสี่บิต ปัญหาจึงโผล่ทันทีที่ผลบวกเกิน 9

ลองเลขง่าย ๆ ก่อน 3 + 4 = 7 ซึ่งยังไม่เกิน 9

3 + 4 — กรณีปกติ

```
  0011      (3)
+ 0100      (4)
------------------
  0111      อ่านเป็น BCD ได้ 7   ถูกต้อง
```

0111 อยู่ในตารางของเรา แปลว่า 7 พอดี วงจรไม่ต้องทำอะไรเพิ่ม

ทีนี้ลองให้ผลบวกเกิน 9 บ้าง เอา 5 + 8 ซึ่งควรได้ 13

5 + 8 — กรณีที่พัง

```
  0101      (5)
+ 1000      (8)
------------------
  1101      อ่านเป็น BCD ไม่ได้ — นี่คือรหัสต้องห้าม
```

1101 ไม่มีอยู่ในตาราง BCD วงจรจึงอ่านออกมาเป็นตัวเลขหลักใดไม่ได้เลย ถ้าดันส่งไปที่จอ จอจะขึ้นอะไรก็ไม่รู้

สาเหตุอยู่ตรงนี้ เลข 4 บิตจะทดออกก็ต่อเมื่อเกิน 15 แต่เลขฐานสิบหนึ่งหลักต้องทดออกตั้งแต่เกิน 9 มันทดกันคนละจุด ห่างกันอยู่ 16 − 10 = **6**

วิธีแก้จึงตรงไปตรงมามาก **บวก 6 (คือ 0110) เพิ่มเข้าไป** เพื่อดันผลลัพธ์ให้ข้ามช่องต้องห้ามทั้งหกช่องไปพอดี



ช่องกากบาทหกช่องคือรหัสต้องห้าม การบวก 6 คือการดันผลลัพธ์ให้ข้ามหกช่องนั้นไปพอดี

5 + 8 — แก้ด้วยการบวก 6

```
  0101      (5)
+ 1000      (8)
------------------------------------
  1101      รหัสต้องห้าม -> ต้องแก้
+ 0110      (บวก 6)
------------------------------------
1 0011      ทดออก 1 = หลักสิบ, เหลือ 0011 = 3
```

อ่านผลลัพธ์เป็น BCD ได้ 0001 0011 ซึ่งคือ 13 และ 5 + 8 = 13 จริง ตรงกับโจทย์เดิม

กรณีที่หลอกตาที่สุด: ผลบวกดู "ถูก" แต่ผิด

เมื่อกี้เรารู้ว่าต้องแก้เพราะเห็นรหัสต้องห้าม แต่มีอีกกรณีที่ผลลัพธ์ออกมาเป็นรหัสที่ถูกกฎทุกอย่าง แล้วยังผิดอยู่ดี

ลอง 9 + 7 ซึ่งควรได้ 16

9 + 7 — ดูไม่ออกว่าผิด

```
  1001      (9)
+ 0111      (7)
------------------------------------
1 0000      สี่บิตล่างคือ 0000 ซึ่งแปลว่า 0 — อยู่ในตาราง ไม่ใช่รหัสต้องห้าม!
```

ถ้าดูแค่สี่บิตล่าง จะได้ 0 ซึ่งเป็นรหัสที่ใช้ได้สบาย ๆ ไม่มีอะไรน่าสงสัยเลย แต่คำตอบที่ถูกคือ 16 ไม่ใช่ 10 และไม่ใช่ 0

สิ่งที่ฟ้องว่ามีอะไรผิดคือ **ตัวทดที่โผล่ออกมาจากสี่บิต** ตัวทดนั้นบอกว่าผลบวกเกิน 15 ไปแล้ว ซึ่งแปลว่าเกิน 9 มานานแล้วแน่นอน

กติกาการแก้จึงมีสองเงื่อนไข ไม่ใช่เงื่อนไขเดียว

กติกาบวก BCD

บวกสี่บิตกับสี่บิตแบบฐานสองธรรมดาก่อน แล้วดูผล

ถ้าผลเป็นรหัสต้องห้าม (1010–1111) ให้บวก 0110 เพิ่ม

หรือถ้ามีตัวทดออกจากสี่บิต ให้บวก 0110 เพิ่มเหมือนกัน

ถ้าไม่เข้าสองข้อนี้ ผลบวกใช้ได้เลย ไม่ต้องทำอะไร

9 + 7 — แก้ให้ถูก

```
  1001      (9)
+ 0111      (7)
-------------------------------
1 0000      มีตัวทดออก -> ต้องแก้
  0000
+ 0110      (บวก 6 ที่สี่บิตล่าง)
-------------------------------
1 0110      ทดออก 1 = หลักสิบ, เหลือ 0110 = 6
```

อ่านเป็น BCD ได้ 0001 0110 ซึ่งคือ 16 และ 9 + 7 = 16 จริง ตรงกับโจทย์เดิม

เช็กลิสต์ BCD — ทำตามนี้แล้วหยุด

ฐานสิบ → BCD: แยกหลัก → แปลงหลักละ 4 บิต (ศูนย์ก็ต้องเขียน 0000) → ลอกจุดทศนิยมลงมา → **หยุด** ไม่ต้องรวมเป็นเลขเดียว

BCD → ฐานสิบ: ตัดกลุ่มละ 4 บิตโดยนับจากจุดออกไปทั้งสองทาง → แปลงทีละกลุ่ม → **หยุด**

บวก BCD: บวกฐานสองก่อน → ถ้าได้ 1010–1111 หรือมีตัวทดออก ให้ +0110 → **หยุด** อ่านผลทีละสี่บิต

ถ้าโจทย์ถามว่า "BCD ต่างจากฐานสองยังไง" ตอบว่า BCD เก็บทีละหลัก ใช้สี่บิตต่อหลัก เปลืองที่กว่า แต่แปลงขึ้นจอได้ทันที และมีรหัสต้องห้ามหกแบบ

2 Gray code — รหัสที่เปลี่ยนทีละบิต

ของจริงก่อน: จานวัดมุมของแขนกล

เครื่องจักรที่ต้องรู้ว่าแกนหมุนไปถึงไหนแล้ว มักใช้จานกลมติดไว้ที่แกน จานนั้นแบ่งเป็นช่อง ๆ เท่า ๆ กัน แล้วมีเซนเซอร์อ่านว่าตอนนี้อยู่ช่องไหน

สมมติจานของเราแบ่งเป็น 8 ช่อง ก็ต้องใช้เลข 3 บิตบอกหมายเลขช่อง จานจึงมีวงแหวน 3 วง วงละหนึ่งบิต และมีเซนเซอร์ 3 ตัวเรียงกันตามแนวรัศมี อ่านพร้อมกันทีเดียวได้เลข 3 บิต

ถ้าเราหมายเลขช่องด้วยเลขฐานสองธรรมดา ช่องที่ 3 คือ 011 และช่องที่ 4 คือ 100

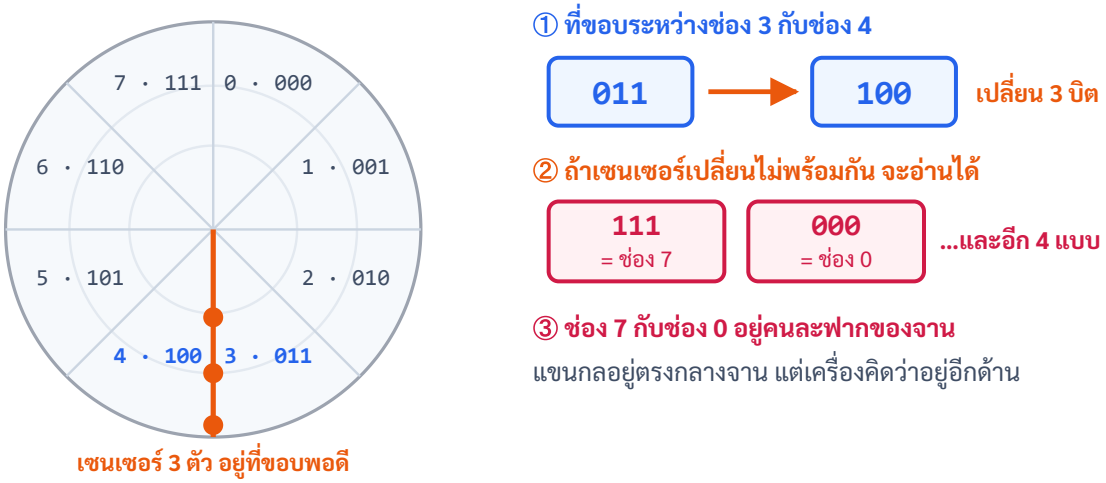
ลองคิดในหัวก่อน ไม่ต้องเขียน — เปรียบเทียบ 011 กับ 100 ทีละบิต

ตอนแกนหมุนข้ามจากช่อง 3 ไปช่อง 4 มีกี่บิตที่ต้องเปลี่ยนค่าพร้อมกัน?

แล้วถ้าเซนเซอร์สามตัววางเหลื่อมกันนิดเดียว ไม่ได้เปลี่ยนพร้อมกันเป๊ะ ๆ มันจะอ่านได้เลขอะไรออกมาบ้าง?

ตอบว่าเปลี่ยนทั้งสามบิตพร้อมกันใช่ไหม 0→1, 1→0 และ 1→0 ครบทั้งสามตำแหน่ง

และนี่คือปัญหา เพราะในโลกจริงไม่มีอะไรเปลี่ยนพร้อมกันเป๊ะ เซนเซอร์ตัวหนึ่งจะเห็นขอบก่อนอีกตัวเสมอ แม้จะเร็วกว่าแค่เศษเสี้ยวมิลลิเมตร



จานวัดมุม 8 ช่องที่ใช้เลขฐานสองธรรมดา — ขอบระหว่างช่อง 3 กับ 4 คือจุดที่อันตรายที่สุด

ลองนับดูว่าจริง ๆ แล้วมันอ่านผิดได้กี่แบบ เลข 3 บิตมี 8 แบบ ค่าที่ถูกต้องมีแค่ 2 แบบ คือ 011 กับ 100 เหลือ **6 แบบที่ผิด** และในหกแบบนั้นมี 000 กับ 111 รวมอยู่ด้วย

ความเสียหายอยู่ตรงนี้ แขนกลกำลังอยู่ตรงรอยต่อกลางจานพอดี แต่เครื่องได้ตัวเลขที่บอกว่ามันอยู่อีกฟากหนึ่งของจาน นี่ไม่ใช่ความคลาดเคลื่อนเล็กน้อย แต่เป็นการอ่านผิดแบบคนละเรื่องกันเลย

ทางแก้: ห้ามให้เปลี่ยนเกินหนึ่งบิต

ถ้าปัญหาเกิดจาก "หลายบิตเปลี่ยนพร้อมกัน" ทางแก้ก็คือ อย่าให้มันเปลี่ยนพร้อมกัน เลิกใช้เลขฐานสองธรรมดาเป็นหมายเลขช่อง แล้วไปคิดลำดับใหม่ที่ **ช่องติดกันต่างกันแค่บิตเดียวเสมอ**

ลำดับแบบนั้นมีจริงและมีชื่อว่า Gray code อ่านว่า เกรย์โค้ด

Gray code ไม่ใช่ระบบเลข มันไม่มีค่าประจำหลัก เอาไปบวกลบไม่ได้ มันเป็นแค่ **ลำดับการนับ** ที่ออกแบบมาให้เปลี่ยนทีละบิต

กลับไปที่จานของเรา ช่อง 3 กับช่อง 4 ในแบบ Gray คือ 010 กับ 110 ต่างกันแค่บิตซ้ายสุดบิตเดียว

ผลคือ ถ้าเซนเซอร์เหลื่อมกัน มันอ่านได้แค่ 010 (ช่องเดิม) หรือ 110 (ช่องใหม่) เท่านั้น ไม่มีทางได้ค่าที่สาม เพราะมีบิตเดียวที่กำลังเปลี่ยน อ่านผิดแย่ที่สุดคือช้าไปหรือเร็วไปหนึ่งช่อง

สมบัติเดียวที่ต้องจำ

Gray code ที่อยู่ติดกันในลำดับ **ต่างกันเพียงบิตเดียวเสมอ** และค่าสุดท้ายกับค่าแรกก็ต่างกันบิตเดียวด้วย (วนกลับได้)

ตาราง Gray code 4 บิต

ตารางนี้อยู่ในสไลด์ ไม่ต้องท่อง แต่ต้อง **สร้างเองได้** ซึ่งเดี๋ยวเราจะสร้างกันในหัวข้อถัดไป ตอนนี้ดูให้เห็นสมบัติก่อน

ฐานสิบ	ฐานสอง	Gray	ฐานสิบ	ฐานสอง	Gray
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

ลองตรวจสมบัติสักคู่ จาก 7 ไป 8 ฐานสองเปลี่ยนจาก 0111 เป็น 1000 คือเปลี่ยนครบสี่บิต แต่ Gray เปลี่ยนจาก 0100 เป็น 1100 คือเปลี่ยนบิตซ้ายสุดบิตเดียว

และลองวนกลับ จาก 15 (Gray = 1000) ไป 0 (Gray = 0000) ก็ต่างกันบิตเดียวเหมือนกัน จานหมุนครบรอบแล้วกลับมาเริ่มใหม่ได้โดยไม่มีจุดอันตราย

ฐานสอง → Gray: ทำให้ดูทุกบิต

กติกามีสองข้อ ข้อแรก **บิตซ้ายสุดของ Gray เท่ากับบิตซ้ายสุดของฐานสอง** ลอกมาตรง ๆ ไม่ต้องคิด

ข้อสอง ไล่จากซ้ายไปขวา **เอาบิตฐานสองที่ติดกันสองตัวมาบวกกัน แล้วทิ้งตัวทด** ผลที่ได้คือบิต Gray ตัวถัดไป

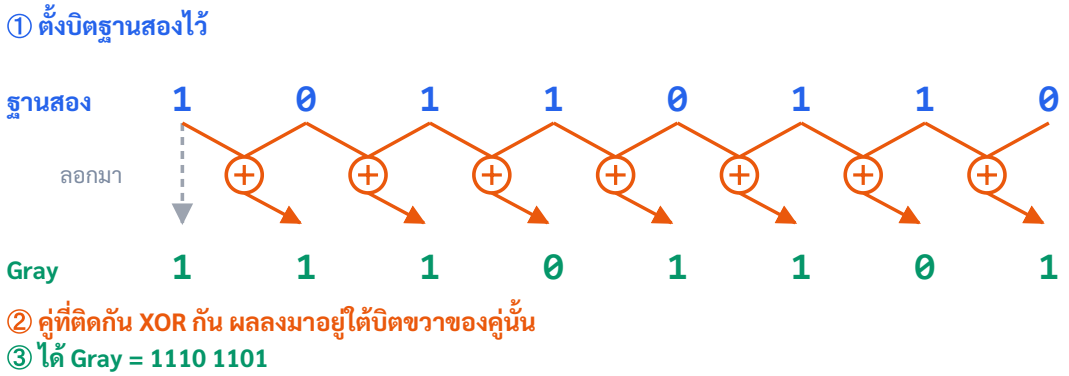
คำว่า "บวกแล้วทิ้งตัวทด" ของสองบิตนี้ มีชื่อเรียกสั้น ๆ ว่า XOR เขียนแทนด้วยเครื่องหมาย $\oplus$ และมันมีกฎง่ายมาก คือ **เหมือนกันได้ 0 ต่างกันได้ 1**

XOR สี่บรรทัด

`0 ⊕ 0 = 0` `1 ⊕ 1 = 0` (เหมือนกัน ได้ 0)

`0 ⊕ 1 = 1` `1 ⊕ 0 = 1` (ต่างกัน ได้ 1)

เอาโจทย์จากสไลด์มาทำ แปลง **`1011 0110`** เป็น Gray



โซ่ XOR ของการแปลงฐานสอง → Gray — ลูกศรเฉียงบอกว่าผลของคู่ไหนไปลงที่ตำแหน่งใด

ทีนี้ไล่เป็นตัวหนังสือทีละบรรทัด จะได้เห็นว่าไม่มีขั้นไหนถูกข้าม

1011 0110 -> Gray

```
บิตที่ 1   ลอก 1 มาตรง ๆ          ->  1
บิตที่ 2   1 XOR 0  ต่างกัน        ->  1
บิตที่ 3   0 XOR 1  ต่างกัน        ->  1
บิตที่ 4   1 XOR 1  เหมือนกัน      ->  0
บิตที่ 5   1 XOR 0  ต่างกัน        ->  1
บิตที่ 6   0 XOR 1  ต่างกัน        ->  1
บิตที่ 7   1 XOR 1  เหมือนกัน      ->  0
บิตที่ 8   1 XOR 0  ต่างกัน        ->  1
```

Gray = 1110 1101

สังเกตว่าบิตฐานสองตัวที่ 8 (ตัวขวาสุด) ถูกใช้แค่ครั้งเดียว และบิตตัวที่ 1 ก็ถูกใช้สองงานคือลอกมาเป็นบิตแรก และเป็นคู่ให้บิตที่สอง

ทำอีกข้อจากสไลด์ แปลง **`0101 0111`** เป็น Gray

0101 0111 -> Gray

```
บิตที่ 1   ลอก 0 มาตรง ๆ        ->  0
บิตที่ 2   0 XOR 1  ต่างกัน      ->  1
บิตที่ 3   1 XOR 0  ต่างกัน      ->  1
บิตที่ 4   0 XOR 1  ต่างกัน      ->  1
บิตที่ 5   1 XOR 0  ต่างกัน      ->  1
บิตที่ 6   0 XOR 1  ต่างกัน      ->  1
บิตที่ 7   1 XOR 1  เหมือนกัน    ->  0
บิตที่ 8   1 XOR 1  เหมือนกัน    ->  0
```

Gray = 0111 1100

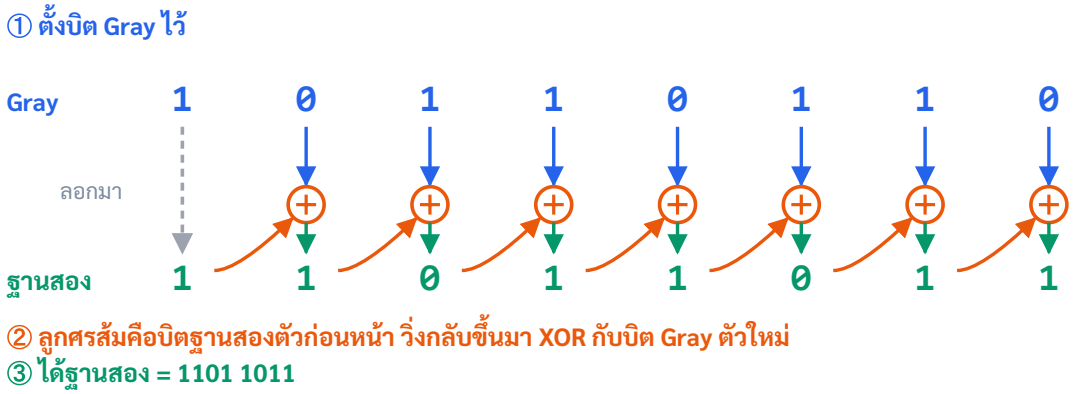
Gray → ฐานสอง: ทิศกลับ ซึ่งไม่เหมือนกัน

คนส่วนใหญ่พลาดตรงนี้ เพราะคิดว่าแปลงกลับก็ใช้กติกาเดิม แต่ไม่ใช่ ทิศกลับมีความต่างสำคัญอยู่หนึ่งจุด

กติกาข้อแรกเหมือนเดิม **บิตซ้ายสุดของฐานสองเท่ากับบิตซ้ายสุดของ Gray** ลอกมาตรง ๆ

กติกาข้อสองคือจุดที่ต่าง **เอาบิตฐานสองที่เราเพิ่งสร้างได้ มา XOR กับบิต Gray ตัวถัดไป** สังเกตคำว่า "ที่เราเพิ่งสร้างได้" นั่นคือหยิบมาจากแถวล่าง ไม่ใช่แถวบน

ขาไปเราหยิบสองตัวจากแถวบนทั้งคู่ ขากลับเราหยิบตัวหนึ่งจากแถวบน อีกตัวจากแถวล่างที่ทำเสร็จไปแล้ว มันจึงเป็นโซ่ที่ต้องทำเรียงกันไป ข้ามไม่ได้



ขากลับเป็นโซ่ต่อเนื่อง — ผลลัพธ์ตัวก่อนหน้าต้องเสร็จก่อน ถึงจะทำตัวถัดไปได้

Gray 1011 0110 -> ฐานสอง

```
บิตที่ 1   ลอก 1 มาตรง ๆ                    ->  1
บิตที่ 2   ได้ 1 มา XOR กับ Gray 0  ต่างกัน  ->  1
บิตที่ 3   ได้ 1 มา XOR กับ Gray 1  เหมือน   ->  0
บิตที่ 4   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
บิตที่ 5   ได้ 1 มา XOR กับ Gray 0  ต่างกัน  ->  1
บิตที่ 6   ได้ 1 มา XOR กับ Gray 1  เหมือน   ->  0
บิตที่ 7   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
บิตที่ 8   ได้ 1 มา XOR กับ Gray 0  ต่างกัน  ->  1
-----------------------------------------------------
ฐานสอง = 1101 1011
```

อีกข้อจากสไลด์ Gray **`0101 0111`** เป็นฐานสอง

Gray 0101 0111 -> ฐานสอง

```
บิตที่ 1   ลอก 0 มาตรง ๆ                    ->  0
บิตที่ 2   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
บิตที่ 3   ได้ 1 มา XOR กับ Gray 0  ต่างกัน  ->  1
บิตที่ 4   ได้ 1 มา XOR กับ Gray 1  เหมือน   ->  0
บิตที่ 5   ได้ 0 มา XOR กับ Gray 0  เหมือน   ->  0
บิตที่ 6   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
บิตที่ 7   ได้ 1 มา XOR กับ Gray 1  เหมือน   ->  0
บิตที่ 8   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
-----------------------------------------------------
ฐานสอง = 0110 0101
```

ทำสองทิศบนค่าเดียวกัน เพื่อตรวจคำตอบตัวเอง

วิธีที่ปลอดภัยที่สุดในห้องสอบคือ แปลงเสร็จแล้วแปลงกลับ ถ้ากลับมาเป็นของเดิมแปลว่าถูก ใช้เลขสั้น ๆ 4 บิต ทำให้ดูสักรอบ

ขาไป: ฐานสอง 1011 -> Gray

```
บิตที่ 1   ลอก 1 มา              ->  1
บิตที่ 2   1 XOR 0  ต่างกัน     ->  1
บิตที่ 3   0 XOR 1  ต่างกัน     ->  1
บิตที่ 4   1 XOR 1  เหมือน      ->  0
-----------------------------------------
Gray = 1110
```

ได้ Gray = 1110 เก็บไว้ก่อน ทีนี้แกล้งลืมไปว่าเรารู้คำตอบ แล้วเอา 1110 นี่แหละมาแปลงกลับด้วยกติกาขากลับ

ขากลับ: Gray 1110 -> ฐานสอง

```
บิตที่ 1   ลอก 1 มา                          ->  1
บิตที่ 2   ได้ 1 มา XOR กับ Gray 1  เหมือน   ->  0
บิตที่ 3   ได้ 0 มา XOR กับ Gray 1  ต่างกัน  ->  1
บิตที่ 4   ได้ 1 มา XOR กับ Gray 0  ต่างกัน  ->  1
-------------------------------------------------
ฐานสอง = 1011    ตรงกับของเดิม
```

ได้ 1011 กลับมาเท่าเดิม แปลว่าทั้งสองทิศถูก และถ้าเปิดตารางหน้าที่แล้วดูแถวเลข 11 จะเห็น ฐานสอง 1011 คู่กับ Gray 1110 พอดี

เดินให้เห็นตอนพัง: ใช้กติกาผิดทิศ

กับดักที่เจอบ่อยที่สุดคือ โจทย์ให้ Gray มาแล้วขอฐานสอง แต่เราเผลอใช้กติกาขาไป คือ XOR คู่ที่ติดกันของแถวบน

ทำผิดวิธี: Gray 1110 -> ฐานสอง (ใช้กติกาขาไป)

```
บิตที่ 1   ลอก 1 มา                      ->  1
บิตที่ 2   1 XOR 1  (สองตัวจากแถวบน)    ->  0
บิตที่ 3   1 XOR 1  (สองตัวจากแถวบน)    ->  0
บิตที่ 4   1 XOR 0  (สองตัวจากแถวบน)    ->  1
-----------------------------------------------
ได้ 1001   ซึ่งผิด
```

1001 คือ 9 แต่คำตอบที่ถูกคือ 1011 ซึ่งคือ 11 ผิดไปสองหน่วย และผิดแบบไม่มีอะไรเตือนเลย เพราะ 1001 ก็หน้าตาเป็นเลขฐานสองที่ใช้ได้ปกติ

ตรวจจับความผิดนี้ได้ง่ายมาก เอา 1001 แปลงกลับเป็น Gray ดู จะได้ 1101 ไม่ใช่ 1110 ที่โจทย์ให้มา

จำประโยคนี้ประโยคเดียวพอ

ขาไป (ฐานสอง → Gray) หยิบสองตัวจาก **แถวบน** ทั้งคู่ ทำพร้อมกันทุกบิตได้เลย

ขากลับ (Gray → ฐานสอง) หยิบตัวใหม่จาก **แถวบน** แต่หยิบอีกตัวจาก **แถวล่างที่เพิ่งทำเสร็จ** ต้องทำเรียงทีละตัว

ลองทำก่อนดูเฉลย (โจทย์ชุดเดียวกับในสไลด์)

ขาไป 1) 1011 2) 110010 3) 1000 0000 4) 0110 1101 → Gray

ขากลับ 5) 1110 6) 101011 7) 0101 8) 1101 0110 → ฐานสอง

---

1) 1110 **2)** 101011 **3)** 1100 0000 **4)** 0101 1011

5) 1011 **6)** 110010 **7)** 0110 **8)** 1001 1011

สังเกตข้อ 1 กับ 5 และข้อ 2 กับ 6 เป็นคู่กลับกันพอดี ใช้ตรวจกันเองได้

เช็กลิสต์ Gray — ทำตามนี้แล้วหยุด

ฐานสอง → Gray: ลอกบิตซ้ายสุด → ไล่ขวา XOR คู่ติดกันของแถวบน → **หยุด** จำนวนบิตเท่าเดิมเสมอ

Gray → ฐานสอง: ลอกบิตซ้ายสุด → ไล่ขวา เอาผลลัพธ์ตัวก่อนหน้า XOR กับ Gray ตัวใหม่ → **หยุด**

ตรวจเสมอ: แปลงกลับอีกทิศ ต้องได้ของเดิม ใช้เวลาไม่ถึงสิบวินาที

ถ้าโจทย์ถามว่าทำไมต้องใช้ Gray ตอบว่าช่องที่ติดกันต่างกันบิตเดียว ทำให้เซนเซอร์ที่เหลื่อมกันอ่านผิดได้อย่างมากแค่หนึ่งช่อง ไม่ใช่กระโดดไปอีกฟากของจาน

3 พาริตี — บิตเดียวที่คอยฟ้อง

ของจริงก่อน: ข้อมูลที่เดินทางแล้วเพี้ยน

สมมติเราส่งข้อมูล 7 บิตจากแผงวงจรหนึ่งไปอีกแผงหนึ่งตามสายแพ ระหว่างทางมีมอเตอร์ตัวหนึ่งสตาร์ต สัญญาณรบกวนกระชากขึ้นมาแวบเดียว แล้วบิตหนึ่งบิตก็พลิกจาก 0 เป็น 1

ปลายทางได้เลขมาชุดหนึ่ง และปัญหาคือ **มันไม่มีทางรู้เลยว่าเลขชุดนั้นคือของที่ส่งมาจริงหรือของที่เพี้ยนแล้ว** เพราะ 0110010 กับ 0110011 ต่างก็เป็นเลข 7 บิตที่ถูกต้องตามกฎทั้งคู่

ที่มันไม่รู้เพราะเราใช้รูปแบบบิตครบทุกแบบไปหมดแล้ว ทุกแบบมีความหมาย เลยไม่เหลือแบบไหนไว้เป็น "แบบที่ผิด" ให้สังเกตได้

ลองคิดในหัวก่อน ไม่ต้องเขียน — เราอยากให้ปลายทางจับได้ว่ามีบิตพลิก

ถ้าเราเติมอะไรลงไปกับข้อมูลได้ **แค่หนึ่งบิต** น้องจะให้บิตนั้นบอกอะไร ถึงจะพอใช้จับได้ว่ามีบิตพลิก?

ถ้าน้องคิดว่า "ให้มันบอกว่าจำนวนเลข 1 เป็นคู่หรือคี่" นั่นแหละใช่เลย เรียกว่า **บิตพาริตี** (`parity bit`) คำว่า `parity` แปลว่าความเป็นคู่-คี่

วิธีทำคือ ก่อนส่ง เรานับว่าในข้อมูลมีเลข 1 กี่ตัว แล้วเติมบิตพิเศษหนึ่งบิตเพื่อ **บังคับให้จำนวนเลข 1 ทั้งชุด (รวมบิตที่เติมแล้ว) เป็นเลขคู่เสมอ**

แบบนี้เรียกว่า **พาริตีคู่** (`even parity`) ถ้าบังคับให้เป็นเลขคี่แทนก็เรียก **พาริตีคี่** (`odd parity`)

กติกา

พาริตีคู่: นับเลข 1 ในข้อมูล ถ้าได้จำนวนคู่อยู่แล้วให้เติม 0 ถ้าได้จำนวนคี่ให้เติม 1

พาริตีคี่: ทำกลับกันทุกอย่าง ถ้านับได้จำนวนคู่ให้เติม 1 ถ้าได้จำนวนคี่ให้เติม 0

ระบบหนึ่งระบบเลือกใช้แบบเดียว จะสลับไปมาไม่ได้ และบิตพาริตีจะเอาไปแปะหัวหรือแปะท้ายก็ได้ ขอแค่ตกลงกันไว้

ทำให้ดู: เติมพาริตีให้ข้อมูล 7 บิต

โจทย์จากสไลด์ ข้อมูล 7 บิต ให้เติมพาริตีเป็น 8 บิต ทำทั้งแบบคู่และแบบคี่

ข้อมูล 0110010

```
นับเลข 1 ได้ 3 ตัว (คี่)
พาริตีคู่ ต้องการให้เป็นคู่ -> เติม 1 ได้ 1 0110010 รวมเป็น 4 ตัว คู่
พาริตีคี่ เป็นคี่อยู่แล้ว -> เติม 0 ได้ 0 0110010 รวมเป็น 3 ตัว คี่
```

ทำอีกแถวที่ต่างจากแถวแรกแค่บิตสุดท้าย จะได้เห็นว่าบิตพาริตีพลิกตามทันที

ข้อมูล 0110011

```
นับเลข 1 ได้ 4 ตัว (คู่)
พาริตีคู่ เป็นคู่อยู่แล้ว -> เติม 0 ได้ 0 0110011 รวมเป็น 4 ตัว คู่
พาริตีคี่ ต้องการให้เป็นคี่ -> เติม 1 ได้ 1 0110011 รวมเป็น 5 ตัว คี่
```

ลองทำสองแถวที่เหลือก่อนดูเฉลย

ข้อมูล 0000000 · ข้อมูล 0110111 (ทำทั้งพาริตีคู่และพาริตีคี่)

0000000 มีเลข 1 อยู่ 0 ตัว ซึ่งเป็นจำนวนคู่ → พาริตีคู่เติม **0** ได้ 00000000 · พาริตีคี่เติม **1** ได้ 10000000

0110111 มีเลข 1 อยู่ 5 ตัว ซึ่งเป็นจำนวนคี่ → พาริตีคู่เติม **1** ได้ 10110111 · พาริตีคี่เติม **0** ได้ 00110111

ข้อ 0000000 คือข้อที่คนพลาดบ่อย เพราะลืมว่า **ศูนย์ตัวเป็นจำนวนคู่**

อีกสองข้อจากสไลด์

ข้อ ก) ระบบใช้พาริตีคู่ ได้รับ 1101 0101 มา ระบบจะฟ้องว่าผิดไหม

ข้อ ข) ระบบใช้พาริตีคี่ ข้อมูลคือ 001 0101 ต้องเติมบิตพาริตีเป็นอะไร

ก) นับเลข 1 ใน 11010101 ได้ **5 ตัว** ซึ่งเป็นคี่ แต่ระบบนี้ต้องการคู่ → **ฟ้องว่าผิด**

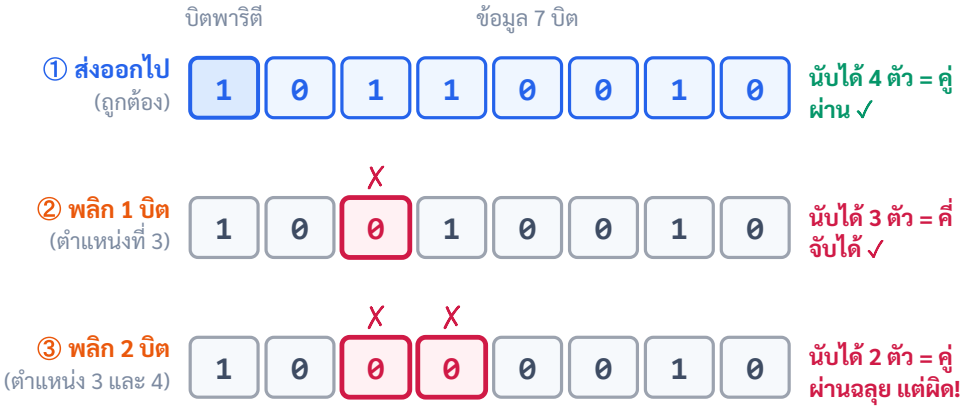
ข) นับเลข 1 ใน 0010101 ได้ **3 ตัว** ซึ่งเป็นคี่อยู่แล้ว ระบบนี้ต้องการคี่ → เติม **0**

ทำให้ดู: จับได้จริงไหม

เราจะส่งข้อมูล 0110010 ด้วยระบบพาริตีคู่ จากที่ทำไปเมื่อกี้ บิตพาริตีคือ 1 ดังนั้นสิ่งที่ออกจากสายคือ **`1011 0010`** ซึ่งมีเลข 1 อยู่ 4 ตัว

สมมติระหว่างทาง บิตที่ 3 นับจากซ้ายพลิกจาก 1 เป็น 0 ปลายทางจะได้ **`1001 0010`**

ปลายทางไม่ต้องรู้ว่าของเดิมคืออะไร มันแค่นับเลข 1 ในสิ่งที่ได้รับ นับได้ 3 ตัว ซึ่งเป็นคี่ ทั้งที่ระบบนี้ต้องได้คู่เสมอ จึงฟ้องได้ทันทีว่า **ข้อมูลชุดนี้เสีย**



พาริตีคู่บนคำเดียวกัน — พลิกหนึ่งบิตจับได้ แต่พลิกสองบิตจำนวนเลข 1 กลับมาเป็นคู่อีกครั้ง

เดินให้เห็นตอนพัง ข้อที่ 1: พลิกสองบิตแล้วรอด

ดูแถวล่างสุดของภาพ ถ้าบิตที่ 3 และบิตที่ 4 พลิกทั้งคู่ ปลายทางจะได้ **`1000 0010`**

นับเลข 1 ได้ 2 ตัว ซึ่งเป็นคู่พอดี ระบบพาริตีคู่จึงบอกว่า "ข้อมูลชุดนี้ดี" ทั้งที่มันเสียไปสองบิตแล้ว

สาเหตุตรงไปตรงมา การพลิกหนึ่งบิตทำให้จำนวนเลข 1 เปลี่ยนความเป็นคู่-คี่เสมอ พลิกสองครั้งก็เปลี่ยนสองครั้ง ซึ่งเท่ากับกลับมาที่เดิม

และมันไม่ใช่แค่คู่ 3 กับ 4 คู่ไหนก็ได้ ในคำ 8 บิตมีคู่บิตให้เลือกถึง 28 คู่ และ **ทั้ง 28 คู่หลุดรอดหมด**

เดินให้เห็นตอนพัง ข้อที่ 2: จับได้ แต่บอกไม่ได้ว่าตรงไหน

กลับไปดูแถวกลางของภาพ ปลายทางได้ 1001 0010 และรู้ว่าผิดแน่ แต่ลองถามต่อว่า "ผิดที่บิตไหน" จะตอบไม่ได้เลย

เพราะสัญญาณที่มันได้มีแบบเดียวคือ "จำนวนเลข 1 เป็นคี่" ไม่ว่าจะพลิกบิตที่ 1, บิตที่ 2 หรือบิตไหนก็ตาม ก็ได้สัญญาณหน้าตาเหมือนกันเป๊ะทั้ง 8 ตำแหน่ง

ผลคือปลายทางทำได้อย่างเดียว คือขอให้ส่งมาใหม่ ถ้าเป็นข้อมูลจากยานอวกาศที่ส่งกลับมาใช้เวลาครึ่งชั่วโมง การขอส่งใหม่เป็นเรื่องใหญ่มาก

เช็กลิสต์พาริตี — ทำตามนี้แล้วหยุด

เติมบิต: นับเลข 1 ในข้อมูล → ดูว่าระบบต้องการคู่หรือคี่ → เติมบิตให้ได้ตามนั้น → **หยุด**

ตรวจรับ: นับเลข 1 ทั้งชุดรวมบิตพาริตี → ถ้าไม่ตรงกับที่ระบบต้องการ แปลว่าผิด → **หยุด** ห้ามพยายามบอกว่าผิดตรงไหน

ประโยคที่ต้องเขียนได้ในข้อสอบ: พาริตีตรวจจับข้อผิดพลาด **หนึ่งบิต** ได้ แต่แก้ไม่ได้ และบอกตำแหน่งไม่ได้ · ถ้าผิด **สองบิต** จะตรวจไม่พบ

4 CRC — หารเอาเศษไว้เป็นลายเซ็น

ของจริงก่อน: สัญญาณรบกวนไม่ได้มาทีละบิต

ตอนที่แล้วเราเห็นว่าพาริตีหลุดทันทีที่มีบิตพลิกสองตัว และในโลกจริงนั่นคือสิ่งที่เกิดบ่อยกว่าด้วยซ้ำ

ฟ้าผ่าลงใกล้สาย มอเตอร์สตาร์ต หรือสายหลวมแวบหนึ่ง สิ่งที่เกิดขึ้นไม่ใช่บิตเดียวพลิก แต่เป็นช่วงสั้น ๆ ที่ข้อมูลทั้งท่อนเละ ติดกันสามสี่บิต เรียกว่า **เบิร์สต์เออเรอร์** (burst error)

เราจึงอยากได้ตัวตรวจที่ **ผูกกับข้อมูลทั้งชุด** ไม่ใช่แค่นับจำนวนเลข 1 เพราะการนับจำนวนมันหยาบเกินไป ข้อมูลคนละชุดนับได้เท่ากันเต็มไปหมด

ลองคิดก่อน — ถ้าให้น้องทำ "ลายเซ็น" สั้น ๆ ของเลขก้อนหนึ่ง โดยใช้เลขคณิตที่เรียนมาตั้งแต่ประถม การกระทำอะไรที่เอาเลขทั้งก้อนมาบด แล้วได้ของสั้น ๆ ออกมา ซึ่งถ้าเลขก้อนนั้นเปลี่ยนนิดเดียว ของสั้น ๆ นั้นก็เปลี่ยนตาม?

ถ้าน้องนึกถึง "การหารแล้วเอาเศษ" ก็ตรงแล้ว วิธีนี้ชื่อว่า CRC เต็ม ๆ ว่า Cyclic Redundancy Check

ไอเดียหลักมีประโยคเดียว **ผู้ส่งจัดแจงให้ข้อมูลที่ส่งออกไปหารด้วยเลขตัวหนึ่งลงตัวพอดี ถ้าปลายทางหารแล้วยังลงตัว แปลว่าของมาถึงครบ**

ของสามอย่างที่ต้องรู้จักก่อน

อย่างแรก **ตัวหารร่วม** หรือ generator เขียนย่อว่า G เป็นเลขฐานสองที่ผู้ส่งกับผู้รับตกลงกันไว้ล่วงหน้า ไม่ใช่ความลับ ทั้งสองฝ่ายรู้เหมือนกัน

ในบทนี้เราใช้ **G = 1010** ตามสไลด์ ซึ่งยาว 4 บิต

อย่างที่สอง จำนวนศูนย์ที่ต้องเติมท้ายข้อมูล เรียกว่า r และกติกาคือ **r = ความยาวของ G ลบ 1** ในที่นี้ r = 4 − 1 = **3** เราจึงเติมศูนย์สามตัว

อย่างที่สาม และเป็นจุดที่คนพลาดมากที่สุด **การหารของ CRC ไม่ได้ลบแบบปกติ แต่ใช้ XOR** คือลบโดยไม่มีการยืม เหมือนกันได้ 0 ต่างกันได้ 1 ตรง ๆ ตามที่เราใช้ตอนทำ Gray code

กับดักใหญ่ที่สุดของเรื่องนี้

ในการหารแบบ CRC **1 − 0 = 1 และ 0 − 1 = 1** ไม่มีการยืมหลักหน้า

ถ้าเผลอยืมเมื่อไร เศษจะผิดทันทีและผิดตั้งแต่บรรทัดนั้นไปจนจบ

วิธีที่ปลอดภัยคือเลิกคิดว่ามันเป็นการลบ ให้คิดว่าเป็น **XOR ทีละคอลัมน์** อย่างเดียว

มองข้อมูลเป็นพหุนาม

ชื่อเต็มของวิธีนี้มีคำว่า polynomial อยู่ในตำราหลายเล่ม เพราะเลขฐานสองหนึ่งชุดถูกมองเป็นพหุนามได้ โดย **บิตตำแหน่งไหนเป็น 1 ก็มีพจน์กำลังนั้น** นับกำลังจากขวาสุดเป็น 0

G = 1010

ตำแหน่ง	3	2	1	0
บิต	1	0	1	0
พจน์	x^3	-	x^1	-

$G = x^3 + x$

ข้อมูลก็เขียนแบบเดียวกัน ต่างแค่ยาวกว่า

D = 1101 0011

ตำแหน่ง	7	6	5	4	3	2	1	0
บิต	1	1	0	1	0	0	1	1

$D = x^7 + x^6 + x^4 + x + 1$

ที่ต้องรู้เรื่องนี้เพราะข้อสอบอาจให้ generator มาเป็นพหุนามแทนที่จะให้เป็นบิต เช่นเขียนว่า $x^3 + x$ ซึ่งแปลว่า 1010 นั่นเอง

แต่เวลาคำนวณจริงเราไม่ได้ใช้พหุนามเลย เราใช้บิตกับ XOR ล้วน ๆ ดังนั้นแปลงเป็นบิตแล้วลงมือหารได้เลย

CRC สี่ขั้น ฝั่งผู้ส่ง

① ตกลง G กันไว้ก่อน ทั้งสองฝ่ายรู้เหมือนกัน

② เติมศูนย์ r ตัวต่อท้ายข้อมูล โดย r = ความยาว G − 1

③ หารด้วย G แบบ XOR เก็บ **เศษ** ไว้ (เศษจะยาว r บิตเสมอ)

④ เอาเศษไปแทนที่ศูนย์ที่เติมไว้ แล้วส่งออกไป

ทำให้ดู: หารยาวแบบ XOR ทุกบรรทัด

โจทย์จากสไลด์ ข้อมูล D = **`1101 0011`** และ G = **`1010`** ให้หา CRC ที่ต้องส่งออกไป

ขั้นแรก เติมศูนย์ 3 ตัวท้ายข้อมูล ได้ตัวตั้งยาว 11 บิต คือ **`11010011000`**

จากนั้นไล่จากคอลัมน์ซ้ายสุดไปขวา ทุกครั้งที่ **บิตบนสุดของคอลัมน์นั้นเป็น 1** ให้วาง G ลงไปแล้ว XOR ถ้าบิตนั้นเป็น 0 ให้ข้ามไปเฉย ๆ

ตัวตั้ง = ข้อมูล 1101 0011 ต่อท้ายด้วยศูนย์ 3 ตัว · ตัวหาร G = 1010

	1	2	3	4	5	6	7	8	9	10	11	
ตัวตั้ง	1	1	0	1	0	0	1	1	0	0	0	
① XOR	1	0	1	0								
	0	1	1	1	0	0	1	1	0	0	0	คอลัมน์ 2 เป็น 1 → XOR ต่อ
② XOR		1	0	1	0							
	0	0	1	0	0	0	1	1	0	0	0	คอลัมน์ 3 เป็น 1 → XOR ต่อ
③ XOR			1	0	1	0						
	0	0	0	0	1	0	1	1	0	0	0	คอลัมน์ 4 เป็น 0 → ข้าม
④ XOR					1	0	1	0				
	0	0	0	0	0	0	0	1	0	0	0	คอลัมน์ 6,7 เป็น 0 → ข้าม
⑤ XOR								1	0	1	0	
	0	0	0	0	0	0	0	0	0	1	0	

เศษ = 010 (ยาว 3 บิต เสมอ)

ทุกบรรทัดเป็น XOR ไม่ใช่การลบ · คอลัมน์ที่บิตบนเป็น 0 ให้ข้าม ไม่ต้องวาง G

เศษที่ได้คือ **010** ยาว 3 บิตพอดีตามที่ r บอกไว้ ถ้าเศษออกมายาวไม่เท่า r แปลว่าคำนวณผิดที่ไหนสักแห่ง ให้กลับไปดูใหม่

ขั้นสุดท้าย เอาเศษไปแทนที่ศูนย์สามตัวที่เราเติมไว้ ไม่ใช่ต่อท้ายเพิ่ม

ประกอบเฟรมที่จะส่ง

```
ข้อมูลเดิม    1101 0011
เติมศูนย์     1101 0011 000    <- ใช้ตอนหารเท่านั้น
เอาเศษแทน     1101 0011 010    <- นี่คือสิ่งที่ส่งออกไปจริง
-------------------------------------------------
เฟรมที่ส่ง = 11010011010    ยาว 11 บิต
```

สังเกตว่าเฟรมที่ส่งยาว 11 บิตเท่ากับตัวตั้ง ไม่ได้ยาวขึ้นอีก เพราะเศษไป **แทนที่** ศูนย์ที่เติมไว้ ไม่ได้ต่อเพิ่ม

ฝั่งผู้รับ: หารซ้ำด้วย G ตัวเดิม

ผู้รับไม่ต้องแยกข้อมูลกับเศษออกจากกัน มันเอา **ทั้งเฟรม 11 บิต** มาหารด้วย G ตัวเดิม แล้วดูเศษอย่างเดียว

กติกาการอ่านผลมีแค่สองบรรทัด **เศษเป็น 000 แปลว่าไม่พบข้อผิดพลาด รับไว้** และ **เศษไม่เป็น 000 แปลว่ามีข้อผิดพลาด ขอให้ส่งใหม่**

ผู้รับหาร 11010011010 ด้วย 1010 — กรณีข้อมูลมาถึงครบ

```
11010011010      เฟรมที่รับได้
1010             ① XOR ที่คอลัมน์ 1
-----------------------------------------------
01110011010
 1010            ② XOR ที่คอลัมน์ 2
-----------------------------------------------
00100011010
  1010           ③ XOR ที่คอลัมน์ 3
-----------------------------------------------
00001011010      คอลัมน์ 4 บิตบนเป็น 0 -> ข้าม
    1010         ④ XOR ที่คอลัมน์ 5
-----------------------------------------------
00000001010      คอลัมน์ 6 และ 7 บิตบนเป็น 0 -> ข้าม
       1010      ⑤ XOR ที่คอลัมน์ 8
-----------------------------------------------
00000000000      เศษ = 000   ไม่พบข้อผิดพลาด
```

ได้เศษ 000 พอดี ซึ่งไม่ใช่เรื่องบังเอิญ เพราะผู้ส่งจงใจแทนที่สามบิตท้ายด้วยเศษ เพื่อบังคับให้ทั้งเฟรมหารลงตัว

พูดอีกแบบคือ การเอาเศษไปแทนที่ท้ายเฟรม เท่ากับ "หักเศษทิ้ง" ล่วงหน้า ของที่ออกจากผู้ส่งจึงหารลงตัวตั้งแต่ยังไม่ออกเดินทาง

เดินให้เห็นตอนพัง: ข้อมูลเสียระหว่างทาง

ทีนี้สมมติว่าบิตที่ 2 นับจากซ้ายโดนรบกวนจนพลิก เฟรมที่ผู้รับได้จึงกลายเป็น **`10010011010`** แทน (โจทย์ข้อนี้อยู่ในสไลด์)

ผู้รับไม่รู้ว่ามีอะไรเกิดขึ้น มันทำสิ่งเดิมเป๊ะ คือหารด้วย G ตัวเดิม

ผู้รับหาร 10010011010 ด้วย 1010 — กรณีข้อมูลเสีย

```
10010011010      เฟรมที่รับได้ (บิตที่ 2 พลิกแล้ว)
1010             ① XOR ที่คอลัมน์ 1
----------
00110011010      คอลัมน์ 2 บิตบนเป็น 0 -> ข้าม
  1010           ② XOR ที่คอลัมน์ 3
----------
00011011010
   1010          ③ XOR ที่คอลัมน์ 4
----------
00001111010
    1010         ④ XOR ที่คอลัมน์ 5
----------
00000101010
     1010        ⑤ XOR ที่คอลัมน์ 6
----------
00000000010      คอลัมน์ 7 และ 8 บิตบนเป็น 0 -> ข้าม
----------
เศษ = 010  ไม่เป็นศูนย์ -> มีข้อผิดพลาด ขอส่งใหม่
```

เศษออกมา 010 ไม่ใช่ 000 ผู้รับจึงทิ้งเฟรมนี้แล้วขอให้ส่งมาใหม่ จบหน้าที่ของ CRC

สังเกตด้วยว่าลำดับคอลัมน์ที่ต้อง XOR เปลี่ยนไปคนละเรื่องเลย จากเดิม 1, 2, 3, 5, 8 กลายเป็น 1, 3, 4, 5, 6 บิตเดียวที่พลิกทำให้การหารทั้งชุดเดินคนละทาง

สิ่งที่ CRC ทำไม่ได้

CRC บอกได้แค่ว่า "มีอะไรผิด" **บอกไม่ได้ว่าผิดที่บิตไหน และแก้ให้ไม่ได้**

ลองดูจากเฟรมนี้เอง ถ้าพลิกบิตที่ 1 จะได้เศษ 100 และถ้าพลิกบิตที่ 3 ก็ได้เศษ 100 เหมือนกัน เศษเดียวกัน แต่คนละตำแหน่ง จึงย้อนกลับไปหาตัวที่พังไม่ได้

ข้อดีของมันคือ **จับได้ครอบคลุมกว่าพาริตีมาก** ทั้งผิดหนึ่งบิต ผิดสองบิต และผิดเป็นช่วงติดกัน

ข้อควรรู้: G ที่ใช้ฝึกมือกับ G ที่ใช้งานจริง

ประโยคข้างบนเป็นคุณสมบัติของ CRC **เมื่อเลือก G ดี ๆ** และนั่นคือสิ่งที่ข้อสอบถามและสิ่งที่ต้องตอบ

แต่ G = 1010 ที่เราใช้ทำมือในบทนี้เป็นตัวเล็กมาก เลือกมาเพื่อให้หารด้วยมือได้ในหนึ่งหน้า ไม่ได้เลือกมาเพราะมันแข็งแรง

ลองทดสอบกับเฟรม 11010011010 ของเราดู พลิกทีละหนึ่งบิตครบทั้ง 11 ตำแหน่ง **จับได้ทั้ง 11 ตำแหน่ง** และถ้าเสียติดกันเป็นช่วงยาวไม่เกิน 3 บิต ซึ่งเท่ากับ r ก็ **จับได้ทุกกรณีเหมือนกัน**

แต่พอพลิกสองบิตพร้อมกัน ในคู่ที่เป็นไปได้ทั้งหมด 55 คู่ มี **20 คู่ที่หลุดรอด** และทุกคู่ที่หลุดล้วนเป็นคู่ที่อยู่ห่างกันเป็นจำนวนคู่

ตัวอย่างคู่ที่หลุดรอดของ G = 1010

```
เฟรมจริง       11010011010
พลิกบิต 1 และบิต 3 (ห่างกัน 2)
เฟรมที่รับ    01110011010
หารด้วย 1010 -> เศษ = 000    ผู้รับคิดว่าข้อมูลดี  ทั้งที่เสียไปสองบิต
```

สรุปสั้น ๆ สำหรับข้อสอบ ตอบตามที่เรียนว่า **CRC จับข้อผิดพลาดได้ทั้งหนึ่งบิต สองบิต และแบบเป็นช่วง** ส่วนเรื่องนี้เก็บไว้รู้ว่า ความแข็งแรงจริง ๆ ขึ้นกับว่าเลือก G ตัวไหน G ที่ใช้งานจริงยาว 16 ถึง 32 บิต

เช็กลิสต์ CRC — ทำตามนี้แล้วหยุด

ผู้ส่ง: หา r = ความยาว G − 1 → เติมศูนย์ r ตัว → หารแบบ XOR → เอาเศษแทนที่ศูนย์ → **หยุด** ส่งได้

ผู้รับ: เอาทั้งเฟรมหารด้วย G ตัวเดิม → เศษ 0 คือผ่าน เศษไม่ 0 คือเสีย → **หยุด** ห้ามเดาว่าผิดบิตไหน

ทุกบรรทัดเป็น XOR ห้ามยืม · **คอลัมน์ที่บิตบนเป็น 0 ให้ข้าม** ไม่ต้องวาง G · **เศษยาว r บิตเสมอ** ถ้าไม่เท่าแปลว่าผิด

ถ้าโจทย์ให้ generator มาเป็นพหุนาม ให้แปลงเป็นบิตก่อน เช่น $x^3 + x$ คือ 1010

5 Hamming code — รหัสที่บอกได้ว่าผิดตรงไหน

ของจริงก่อน: ปัญหาที่ค้างไว้จากพาริตี

ย้อนกลับไปที่พาริตี ปลายทางรู้ว่า "มีบิตพลิก" แต่ไม่รู้ว่าบิตไหน เลยทำได้แค่ขอให้ส่งใหม่

สาเหตุที่มันบอกไม่ได้ เพราะบิตพาริตีตัวเดียว **ดูแลทุกบิตพร้อมกันหมด** ผลที่ได้จึงมีแค่สองแบบคือผ่านกับไม่ผ่าน ซึ่งไม่พอจะแยกว่าเป็นตำแหน่งไหนในเจ็ดแปดตำแหน่ง

ลองคิดก่อน — มีกล่อง 7 ใบเรียงกัน หนึ่งใบมีของปลอมอยู่ เราถามคำถามที่ตอบได้แค่ "ใช่" หรือ "ไม่ใช่" เท่านั้น

อย่างน้อยต้องถามกี่คำถาม ถึงจะระบุได้แน่ ๆ ว่าใบไหน และคำถามควรถามว่าอะไร?

ตอบ 3 คำถามใช่ไหม เพราะคำตอบใช่/ไม่ใช่ 3 ครั้งให้ผลได้ 2 × 2 × 2 = 8 แบบ ซึ่งพอสำหรับ 7 ตำแหน่งและเหลือไว้อีกหนึ่งแบบสำหรับ "ไม่มีอะไรผิด"

และคำถามต้องเป็นแบบที่ **แบ่งกล่องคนละแบบกันในแต่ละครั้ง** ถ้าถามซ้ำแบบเดิมสามรอบก็ได้ข้อมูลเท่าเดิม

นี่คือทั้งหมดของ Hamming code ใช้บิตพาริตีหลายตัว แต่ให้แต่ละตัว **ดูแลคนละกลุ่มที่เหลื่อมกัน** แล้วเอาผลตรวจทั้งชุดมาอ่านรวมกันเป็นตำแหน่ง

ต้องใช้บิตพาริตีกี่ตัว

สมมติข้อมูลมี d บิต และเราจะเติมพาริตี p บิต รวมเป็นรหัสยาว d + p บิต

ผลตรวจ p ตัวให้ค่าได้ 2^p แบบ และเราต้องการให้มันบอกได้ทั้ง **ตำแหน่งที่ผิดทั้ง d + p ตำแหน่ง** บวกกับกรณี **ไม่มีอะไรผิด** อีกหนึ่งแบบ

สูตรที่ต้องใช้

```
2^p  >=  d + p + 1
```

ลองกับข้อมูล 4 บิต ซึ่งเป็นกรณีในสไลด์ ไล่ค่า p จากน้อยไปมาก

d = 4 ต้องใช้ p เท่าไร

```
p = 2   2^2 = 4    แต่ d + p + 1 = 4 + 2 + 1 = 7    4 >= 7 ไม่จริง -> ไม่พอ
p = 3   2^3 = 8    และ d + p + 1 = 4 + 3 + 1 = 8    8 >= 8 จริง   -> พอดี
-------------------------------------------------------------------------
ใช้ p = 3  ได้รหัสยาว 4 + 3 = 7 บิต
```

ทำอีกกรณีที่สไลด์ยกไว้ ข้อมูล 7 บิต

d = 7 ต้องใช้ p เท่าไร

```
p = 3   2^3 = 8    แต่ d + p + 1 = 7 + 3 + 1 = 11   8 >= 11 ไม่จริง -> ไม่พอ
p = 4   2^4 = 16   และ d + p + 1 = 7 + 4 + 1 = 12   16 >= 12 จริง -> พอ
-------------------------------------------------------------------------
ใช้ p = 4  ได้รหัสยาว 7 + 4 = 11 บิต
```

บิตไหนอยู่ตำแหน่งไหน

Hamming นับตำแหน่ง **เริ่มที่ 1 ไม่ใช่ 0** และนับจากซ้ายไปขวา ข้อนี้ต้องจำให้แม่น เพราะถ้านับผิดฐาน ทุกอย่างหลังจากนี้เลื่อนหมด

กติกาการวางมีข้อเดียว **ตำแหน่งที่เป็นกำลังของสอง คือ 1, 2, 4, 8 ให้บิตพาริตีอยู่ ตำแหน่งที่เหลือทั้งหมดให้ข้อมูลอยู่**

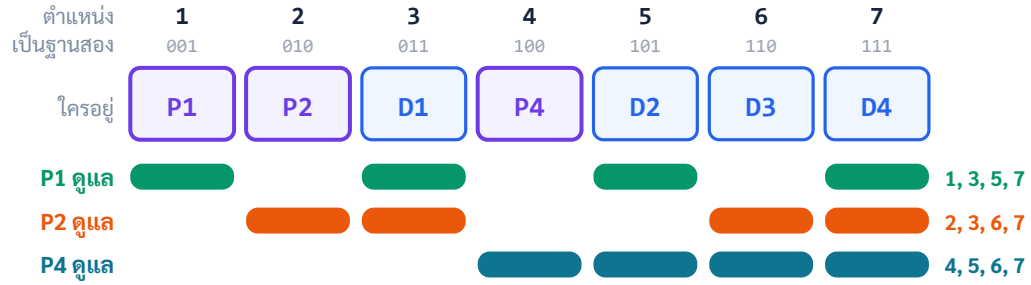
สำหรับรหัส 7 บิต พาริตีอยู่ที่ตำแหน่ง 1, 2, 4 และข้อมูลอยู่ที่ตำแหน่ง 3, 5, 6, 7

ทีนี้มาถึงส่วนที่สวยที่สุดของเรื่องนี้ **ใครดูแลใคร ดูจากเลขประจำตำแหน่งเขียนเป็นฐานสอง**

บิตพาริตีที่ตำแหน่ง 1 ดูแลทุกตำแหน่งที่เลขฐานสองของมัน **มีบิตค่า 1 ติดอยู่** ได้แก่ตำแหน่ง 1 (001), 3 (011), 5 (101) และ 7 (111)

บิตพาริตีที่ตำแหน่ง 2 ดูแลทุกตำแหน่งที่มี **บิตค่า 2** ได้แก่ 2 (010), 3 (011), 6 (110) และ 7 (111)

บิตพาริตีที่ตำแหน่ง 4 ดูแลทุกตำแหน่งที่มี **บิตค่า 4** ได้แก่ 4 (100), 5 (101), 6 (110) และ 7 (111)



กล่องม่วงคือที่อยู่ของบิตพาริตี (ตำแหน่ง 1, 2, 4 ซึ่งเป็นกำลังของสอง) กล่องฟ้าคือที่อยู่ของข้อมูล

ทุกตำแหน่งถูกดูแลด้วยชุดที่ไม่ซ้ำกับใคร — ตำแหน่ง 7 ถูกดูแลทั้งสามตัว ตำแหน่ง 4 ถูกดูแลตัวเดียว

ลองดูให้ละเอียดว่าทำไมมันถึงบอกตำแหน่งได้ เพราะ **ไม่มีสองตำแหน่งไหนถูกดูแลด้วยชุดเดียวกัน**

ตำแหน่ง 3 ถูกดูแลโดย P1 กับ P2 ส่วนตำแหน่ง 5 ถูกดูแลโดย P1 กับ P4 ต่างกัน ถ้าตำแหน่ง 3 พัง P1 กับ P2 จะฟ้อง แต่ถ้าตำแหน่ง 5 พัง P1 กับ P4 จะฟ้องแทน แยกออกจากกันได้ทันที

เขียนเป็นตารางแบบในสไลด์จะเห็นชัดขึ้น คอลัมน์ R1 R2 R4 คือเลขประจำตำแหน่งเขียนเป็นฐานสอง และ **ช่องไหนเป็น 1 แปลว่าพาริตีตัวนั้นดูแลตำแหน่งนี้**

ตำแหน่ง	R4	R2	R1	ใครดูแลบ้าง	บิตอะไรอยู่
1	0	0	1	P1	พาริตี P1
2	0	1	0	P2	พาริตี P2
3	0	1	1	P1, P2	ข้อมูล D1
4	1	0	0	P4	พาริตี P4
5	1	0	1	P1, P4	ข้อมูล D2
6	1	1	0	P2, P4	ข้อมูล D3
7	1	1	1	P1, P2, P4	ข้อมูล D4

ถ้าข้อมูลยาวขึ้นจนต้องใช้ P8 ด้วย กติกาก็ตัวเดิม P8 ดูแลตำแหน่ง 8, 9, 10, 11 และต่อไปเรื่อย ๆ ตามบิตค่า 8

ทำให้ดู: เข้ารหัสข้อมูล 1100

โจทย์จากสไลด์ เข้ารหัสข้อมูล 4 บิต **`1100`** แบบพาริตีคู่

ขั้นที่ ① วางข้อมูลลงในตำแหน่งของมันก่อน ข้อมูลสี่ตัวเรียงจากซ้ายคือ 1, 1, 0, 0 ลงที่ตำแหน่ง 3, 5, 6, 7 ตามลำดับ

① วางข้อมูลก่อน (ช่องพาริตียังว่าง)

ตำแหน่ง	1	2	3	4	5	6	7
บิต	?	?	1	?	1	0	0
	P1	P2	D1	P4	D2	D3	D4

ขั้นที่ ② หา P1 โดยดูเฉพาะตำแหน่งที่ P1 ดูแล คือ 1, 3, 5, 7 แต่ตำแหน่ง 1 คือตัวมันเอง จึงนับเฉพาะ 3, 5, 7

② หา P1 (ดูตำแหน่ง 3, 5, 7)

```
ตำแหน่ง 3 = 1
ตำแหน่ง 5 = 1
ตำแหน่ง 7 = 0
─────────────────────────
นับเลข 1 ได้ 2 ตัว   เป็นคู่อยู่แล้ว
พาริตีคู่ -> เติม 0      P1 = 0
```

ขั้นที่ ③ หา P2 โดยดูตำแหน่ง 2, 3, 6, 7 นับเฉพาะ 3, 6, 7

③ หา P2 (ดูตำแหน่ง 3, 6, 7)

```
ตำแหน่ง 3 = 1
ตำแหน่ง 6 = 0
ตำแหน่ง 7 = 0
─────────────────────────
นับเลข 1 ได้ 1 ตัว   เป็นคี่
พาริตีคู่ -> เติม 1      P2 = 1
```

ขั้นที่ ④ หา P4 โดยดูตำแหน่ง 4, 5, 6, 7 นับเฉพาะ 5, 6, 7

④ หา P4 (ดูตำแหน่ง 5, 6, 7)

```
ตำแหน่ง 5 = 1
ตำแหน่ง 6 = 0
ตำแหน่ง 7 = 0
─────────────────────────
นับเลข 1 ได้ 1 ตัว   เป็นคี่
พาริตีคู่ -> เติม 1      P4 = 1
```

ครบสามตัวแล้ว เอาไปเติมลงช่องที่เว้นไว้ตั้งแต่ขั้นแรก โดยไม่ขยับข้อมูลเลยสักบิต

⑤ ประกอบเป็นรหัส

```
ตำแหน่ง   1   2   3   4   5   6   7
บิต       0   1   1   1   1   0   0
─────────────────────────────────────
รหัส Hamming = 0111100
```

ยังไม่จบแค่นี้ ตรวจคำตอบกลับก่อนเสมอ คราวนี้เรานับ **รวมตัวพาริตีเองด้วย** เพราะมันมีค่าแล้ว ทุกกลุ่มต้องออกมาเป็นจำนวนคู่

ตรวจกลับ — ต้องเป็นคู่ทั้งสามกลุ่ม

```
กลุ่ม P1 (ตำแหน่ง 1,3,5,7)  =  0 1 1 0  ->  นับได้ 2  คู่  ผ่าน
กลุ่ม P2 (ตำแหน่ง 2,3,6,7)  =  1 1 0 0  ->  นับได้ 2  คู่  ผ่าน
กลุ่ม P4 (ตำแหน่ง 4,5,6,7)  =  1 1 0 0  ->  นับได้ 2  คู่  ผ่าน
```

ผ่านครบสามกลุ่ม แปลว่ารหัสถูกต้อง ส่งออกไปได้

ทำให้ดู: หาบิตที่พลิกให้เจอ

เราส่ง 0111100 ออกไป สมมติระหว่างทางบิตที่ตำแหน่ง 5 พลิกจาก 1 เป็น 0 ปลายทางจึงได้ **`0111000`**

ปลายทางไม่รู้ว่าเกิดอะไรขึ้น มันแค่ทำสิ่งเดียวคือ **ตรวจทั้งสามกลุ่มว่ายังเป็นคู่อยู่ไหม** แล้วจดผลเป็น 0 ถ้าผ่าน และ 1 ถ้าไม่ผ่าน

ลองตรวจเองก่อนดูเฉลย — ปลายทางได้ **`0111000`** (ตำแหน่ง 1 ถึง 7 จากซ้าย)

นับเลข 1 ในกลุ่ม P1 (ตำแหน่ง 1,3,5,7) · กลุ่ม P2 (ตำแหน่ง 2,3,6,7) · กลุ่ม P4 (ตำแหน่ง 4,5,6,7)

กลุ่มไหนเป็นคี่บ้าง?

ตรวจทั้งสามกลุ่มของ 0111000

```
กลุ่ม P1 (ตำแหน่ง 1,3,5,7)  =  0 1 0 0  ->  นับได้ 1  คี่   ไม่ผ่าน  C1 = 1
กลุ่ม P2 (ตำแหน่ง 2,3,6,7)  =  1 1 0 0  ->  นับได้ 2  คู่   ผ่าน     C2 = 0
กลุ่ม P4 (ตำแหน่ง 4,5,6,7)  =  1 0 0 0  ->  นับได้ 1  คี่   ไม่ผ่าน  C4 = 1
```

ผลตรวจสามตัวนี้เรียกว่า **ซินโดรม** (`syndrome`) และตรงนี้คือหัวใจของทั้งบท

เอาน้ำหนักของกลุ่มที่ไม่ผ่านมาบวกกัน ผลลัพธ์คือเลขตำแหน่งของบิตที่พลิก ไม่ต้องเดา ไม่ต้องลองผิดลองถูก



ซินโดรมไม่ได้เป็นแค่สัญญาณเตือน มันเป็นเลขตำแหน่งโดยตรง

จะมองเป็นเลขฐานสองก็ได้ เขียน C4 C2 C1 เรียงกันได้ 101 ซึ่งอ่านเป็นฐานสองคือ 5 พอดี เหมือนกัน ไม่ต้องจำสองวิธี

เหตุผลที่มันออกมาเป็นเลขตำแหน่งเป๊ะ ๆ ย้อนกลับไปที่ตารางเมื่อกี้ ตำแหน่ง 5 เขียนเป็นฐานสองคือ 101 แปลว่ามันอยู่ในกลุ่มของ P1 และ P4 แต่ไม่อยู่ในกลุ่ม P2

ดังนั้นถ้าบิตตำแหน่ง 5 พลิก กลุ่มที่จะเสียคือ P1 กับ P4 พอดี และผลตรวจที่ออกมาจึงเป็น 101 ซึ่งคือเลขประจำตำแหน่งของมันเอง

แก้ให้ถูกต้องก็แค่พลิกบิตตำแหน่ง 5 กลับ ได้ 0111100 ตรงกับที่ส่งมาทุกบิต จบ

ประโยคเดียวที่ต้องจำสำหรับ Hamming

ซินโดรมคือเลขตำแหน่ง ถ้าซินโดรมเป็น 0 แปลว่าไม่มีอะไรผิด ถ้าไม่เป็น 0 ให้ไปพลิกบิตที่ตำแหน่งนั้นกลับ

อีกข้อหนึ่ง คราวนี้ไม่รู้ว่าของเดิมคืออะไร

โจทย์จากสไลด์ ปลายทางได้รหัส Hamming มาเป็น **`1000001`** ระบบใช้พาริตีคู่ ให้หาว่าผิดตรงไหนและข้อมูลจริงคืออะไร

ข้อนี้เหมือนของจริงมากกว่า เพราะเราไม่มีของเดิมให้เทียบเลย มีแค่สิ่งที่รับมา และต้องพึ่งการตรวจสามกลุ่มอย่างเดียว

ลองทำเองก่อนดูเฉลย — เขียนตำแหน่ง 1 ถึง 7 ลงไปก่อน แล้วค่อยตรวจทีละกลุ่ม

ตำแหน่ง 1 = 1, ตำแหน่ง 2 = 0, ตำแหน่ง 3 = 0, ตำแหน่ง 4 = 0, ตำแหน่ง 5 = 0, ตำแหน่ง 6 = 0, ตำแหน่ง 7 = 1

① ตรวจทั้งสามกลุ่มของ 1000001

```
กลุ่ม P1 (ตำแหน่ง 1,3,5,7)  =  1 0 0 1  ->  นับได้ 2  คู่   ผ่าน      C1 = 0
กลุ่ม P2 (ตำแหน่ง 2,3,6,7)  =  0 0 0 1  ->  นับได้ 1  คี่   ไม่ผ่าน  C2 = 1
กลุ่ม P4 (ตำแหน่ง 4,5,6,7)  =  0 0 0 1  ->  นับได้ 1  คี่   ไม่ผ่าน  C4 = 1
```

สองกลุ่มไม่ผ่าน คือ P2 กับ P4 เอาน้ำหนักของสองตัวนั้นมาบวกกันได้เลย

② อ่านซินโดรม

```
บวกน้ำหนักของกลุ่มที่ไม่ผ่าน    4 + 2 = 6
หรือเขียนเรียง  C4 C2 C1 = 110 ซึ่งเป็นฐานสองของ 6
บิตที่พลิกอยู่ที่ตำแหน่ง  6
```

รู้ตำแหน่งแล้วก็แค่พลิกบิตนั้นกลับ แล้วค่อยหยิบเฉพาะช่องข้อมูลออกมา

③ แก้แล้วอ่านข้อมูลออกมา

```
ตำแหน่ง     1   2   3   4   5   6   7
รับมา       1   0   0   0   0   0   1
แก้แล้ว     1   0   0   0   0   1   1
-------------------------------------
รหัสที่ถูกต้อง = 1000011
ข้อมูลอยู่ตำแหน่ง 3, 5, 6, 7  ->  0, 0, 1, 1
ข้อมูล = 0011
```

ตรวจคำตอบกลับ เอาข้อมูล 0011 ไปเข้ารหัสใหม่ตั้งแต่ต้น ถ้าได้ 1000011 กลับมาแปลว่าเราแก้ถูก

ตรวจกลับ — เข้ารหัส 0011 ใหม่

```
วางข้อมูล    ตำแหน่ง 3 = 0, ตำแหน่ง 5 = 0, ตำแหน่ง 6 = 1, ตำแหน่ง 7 = 1
P1 ดู 3,5,7 = 0,0,1   นับได้ 1   คี่    -> P1 = 1
P2 ดู 3,6,7 = 0,1,1   นับได้ 2   คู่    -> P2 = 0
P4 ดู 5,6,7 = 0,1,1   นับได้ 2   คู่    -> P4 = 0
-------------------------------------------------------
ได้ 1 0 0 0 0 1 1 = 1000011    ตรงกับที่แก้ไว้
```

เดินให้เห็นตอนพัง: พลิกสองบิต

กลับไปที่รหัสของเรา 0111100 คราวนี้ให้พลิกสองบิตพร้อมกัน คือตำแหน่ง 5 และตำแหน่ง 6 ปลายทางจะได้ **0111010**

ตรวจทั้งสามกลุ่มของ 0111010

```
กลุ่ม P1 (ตำแหน่ง 1,3,5,7)  =  0 1 0 0  ->  นับได้ 1  คี่   ไม่ผ่าน  C1 = 1
กลุ่ม P2 (ตำแหน่ง 2,3,6,7)  =  1 1 1 0  ->  นับได้ 3  คี่   ไม่ผ่าน  C2 = 1
กลุ่ม P4 (ตำแหน่ง 4,5,6,7)  =  1 0 1 0  ->  นับได้ 2  คู่   ผ่าน     C4 = 0
-------------------------------------------------------------------------
ซินโดรม = 0 + 2 + 1 = 3    ชี้ไปที่ตำแหน่ง 3
```

ซินโดรมไม่เป็นศูนย์ แปลว่ามันรู้ว่ามีอะไรผิด ซึ่งถูกต้อง แต่มันชี้ไปที่ **ตำแหน่ง 3** ซึ่งเป็นบิตข้อมูลที่ไม่ได้ถูกแตะต้องเลย

ถ้าเราเชื่อมันแล้วไปพลิกตำแหน่ง 3 กลับ จะได้ 0101010 ซึ่งตอนนี้ผิดจากของเดิมไป **3 บิต** แทนที่จะเป็น 2 บิต การแก้ทำให้แย่ลง

เหตุผลอยู่ตรงนี้ ผลของการพลิกสองบิตคือการเอาผลของทั้งสองตำแหน่งมา XOR กัน ตำแหน่ง 5 คือ 101 ตำแหน่ง 6 คือ 110 เอามา XOR กันได้ 011 ซึ่งคือ 3 พอดี

ซินโดรมจึงไม่ใช่ตำแหน่งไหนเลยในสองตำแหน่งที่พัง มันเป็นเลขตัวที่สามที่ไม่เกี่ยวอะไรด้วย

ขอบเขตของ Hamming แบบพื้นฐาน

ผิด **1 บิต** — หาเจอและแก้ได้ถูกต้องทุกกรณี

ผิด **2 บิต** — รู้ว่าผิด แต่ชี้ผิดที่เสมอ และถ้าแก้ตามจะกลายเป็นผิด 3 บิต

ข้อสอบชอบถามตรงนี้ คำตอบสั้น ๆ คือ **Hamming แก้ได้แค่หนึ่งบิต** ถ้าอยากรู้ทันว่าผิดสองบิตต้องเติมพาริตีรวมอีกหนึ่งตัว ซึ่งเป็นคนละเรื่องกับที่เรียนในบทนี้

เช็กลิสต์ Hamming — ทำตามนี้แล้วหยุด

เข้ารหัส: หา p จาก $2^p \geq d + p + 1$ → วางข้อมูลในตำแหน่งที่ไม่ใช่กำลังของสอง → หา P1 P2 P4 ทีละตัวจากกลุ่มของมัน → **หยุด**

ถอดรหัส: ตรวจสามกลุ่ม จด 0 ถ้าผ่าน 1 ถ้าไม่ผ่าน → บวกน้ำหนักของกลุ่มที่ไม่ผ่าน → ได้เลขตำแหน่ง → พลิกบิตนั้นกลับ → อ่านข้อมูลจากตำแหน่ง 3, 5, 6, 7 → **หยุด**

ถ้าซินโดรมเป็น 0 แปลว่าไม่มีอะไรผิด อ่านข้อมูลออกมาได้เลย ห้ามไปแก้อะไร

สองข้อที่พลาดบ่อย: นับตำแหน่งเริ่มที่ 1 ไม่ใช่ 0 · ตอนหา P1 ห้ามนับตัว P1 เอง เพราะมันยังว่างอยู่

6 ก่อนเข้าห้องสอบ: ดูออกว่าโจทย์ถามอะไร

ปัญหาในห้องสอบไม่ค่อยใช่ว่าทำไม่เป็น แต่เป็นการอ่านโจทย์แล้วหยิบวิธีผิดตัว ตารางนี้จึงเรียงตาม **คำที่โผล่ในโจทย์** ไม่ได้เรียงตามชื่อรหัส

เห็นคำพวกนี้ในโจทย์	คือเรื่องอะไร	ลงมือทำอะไรเป็นอย่างแรก
BCD · แต่ละหลัก · จอตัวเลข · 8421	BCD	ตัดเป็นกลุ่มละ 4 บิต นับจากจุดทศนิยมออกไป
1010 ถึง 1111 · invalid code · บวก 6	BCD และการบวก	บวกฐานสองก่อน แล้วดูว่าต้อง +0110 ไหม
Gray · เปลี่ยนทีละบิต · shaft encoder · จานหมุน	Gray code	ดูว่าโจทย์ให้อะไรมา แล้วเลือกทิศให้ถูก
parity · even · odd · เติมหนึ่งบิต	พาริตี	นับจำนวนเลข 1 ก่อนเลย
CRC · generator · G = ... · polynomial · $x^3 + x$	CRC	หา r = ความยาว G − 1 แล้วเติมศูนย์
Hamming · correct the error · ตำแหน่ง 1, 2, 4	Hamming code	เขียนเลขตำแหน่ง 1 ถึง n กำกับไว้ก่อน

แยกให้ออกว่าโจทย์ขอ "จับได้" หรือ "แก้ได้"

คำถามพวกตรวจจับข้อผิดพลาดชอบถามกลับไปกลับมาอยู่สามแบบ คือ จับได้ไหม · บอกตำแหน่งได้ไหม · แก้ได้ไหม

วิธี	ผิด 1 บิต	ผิด 2 บิต	บอกตำแหน่ง	แก้เองได้
พาริตี	จับได้	จับไม่ได้	ไม่ได้	ไม่ได้
CRC	จับได้	จับได้ (ถ้า G ดี)	ไม่ได้	ไม่ได้
Hamming	จับได้	จับได้	ได้ (เฉพาะผิด 1 บิต)	ได้ (เฉพาะผิด 1 บิต)

ช่องที่คนตอบผิดบ่อยที่สุดคือช่อง Hamming กับ "ผิด 2 บิต" คำตอบคือ **รู้ว่าผิด แต่ชี้ผิดที่** ไม่ใช่ "ไม่รู้เลย"

สรุปหน้าเดียว — บทที่ 2

BCD

หลักละ 4 บิต ฐานสิบ 0–9 เท่านั้น · **1010–1111 คือรหัสต้องห้าม**

บวกแล้วได้รหัสต้องห้าม **หรือ** มีตัวทดออก → บวก 0110 เพิ่ม

ตัวอย่าง 5 + 8 → 1101 → +0110 → 1 0011 = 13 · 9 + 7 → 1 0000 → +0110 → 1 0110 = 16

Gray code

ฐานสอง → Gray: ลอก MSB แล้ว **XOR คู่ติดกันของแถวบน**

Gray → ฐานสอง: ลอก MSB แล้ว **เอาผลตัวก่อนหน้า XOR กับ Gray ตัวใหม่**

ตัวอย่างคู่ตรวจ 1011 ↔ 1110 · 110010 ↔ 101011 · แปลงกลับเสมอเพื่อเช็ก

พาริตี

พาริตีคู่ = จำนวนเลข 1 ทั้งชุดต้องเป็นคู่ · พาริตีคี่ = ต้องเป็นคี่

จับได้ 1 บิต · หลุด 2 บิต · บอกตำแหน่งไม่ได้

อย่าลืมว่า "ศูนย์ตัว" เป็นจำนวนคู่

CRC

r = ความยาว G − 1 · เติมศูนย์ r ตัว · หารแบบ **XOR ห้ามยืม** · เศษไปแทนที่ศูนย์

ผู้รับหารทั้งเฟรม เศษ 0 = ผ่าน เศษไม่ 0 = เสีย · **จับได้ แต่บอกตำแหน่งไม่ได้**

ตัวอย่าง D = 11010011, G = 1010 → เศษ 010 → ส่ง 11010011010

Hamming code

$2^p \geq d + p + 1$ · พาริตีอยู่ตำแหน่ง 1, 2, 4, 8 · นับตำแหน่งเริ่มที่ 1

P1 ดู 1,3,5,7 · P2 ดู 2,3,6,7 · P4 ดู 4,5,6,7 · P8 ดู 8,9,10,11

ซินโดรมคือเลขตำแหน่ง — บวกน้ำหนักของกลุ่มที่ไม่ผ่าน แล้วพลิกบิตนั้นกลับ

ตัวอย่าง ข้อมูล 1100 → รหัส 0111100 · รับ 1000001 → ซินโดรม 6 → แก้เป็น 1000011 ข้อมูล 0011

สามข้อสุดท้ายก่อนวางปากกา

เขียนเลขตำแหน่งกำกับเสมอ ทั้ง Hamming และ CRC พังเพราะนับตำแหน่งเลื่อนมากกว่าพังเพราะคิดเลขผิด

ตรวจย้อนกลับทุกข้อที่ทำได้ Gray แปลงกลับ · Hamming เข้ารหัสใหม่ · CRC หารซ้ำต้องได้เศษ 0 · BCD อ่านกลับเป็นฐานสิบ

ถ้าโจทย์ถามว่า "ทำไม" ให้ตอบด้วยปัญหาที่รหัสนั้นเกิดมาแก้ ไม่ใช่ท่องนิยาม

บทที่ 3 · พีชคณิตบูลีน มินเทอม และวงจร

บทนี้จะได้อะไร

ตอนแลบ EES270 น้องต่อสาย IC แล้วไฟ LED ติด — นั่นคือ **ของจริง** แต่ข้อสอบกลางภาคไม่ให้ต่อสาย มันให้กระดาษกับปากกา แล้วถามว่า "ลดรูปนิพจน์นี้" "เปลี่ยนตารางนี้เป็น SOP" "แปลงวงจรนี้ให้เหลือ NAND อย่างเดียว"

บทนี้คือการย้ายสิ่งที่น้องต่อได้อยู่แล้ว ลงมาอยู่บนกระดาษ เราจะเริ่มจากสวิตช์กับหลอดไฟ แล้วค่อย ๆ ขยับไปจนอ่านวงจร NAND หกตัวออกได้ด้วยสายตา

ทุกตัวเลขและทุกนิพจน์ในบทนี้ถูกตรวจด้วยโปรแกรม `check_boolean.py` ที่กางค่าอินพุตครบทุกชุด รวม 213 ข้อตรวจ

สัญลักษณ์ที่ใช้

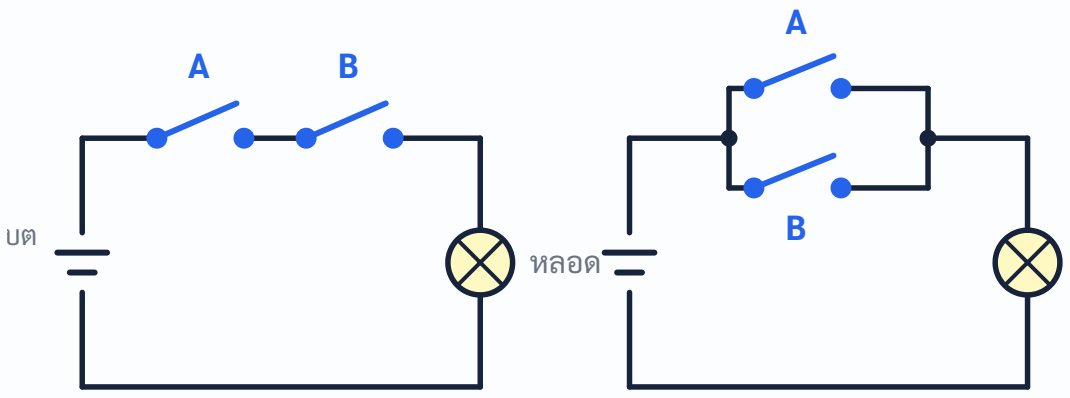
สไลด์ของอาจารย์ใช้ **ขีดบน** เช่น $\overline{A}$ และบางสไลด์ใช้ **เครื่องหมายไพรม์** เช่น A' — สองอย่างนี้คือสิ่งเดียวกัน แปลว่า "กลับค่าของ A"

เครื่องหมายคูณ · คือ AND และมักละไว้ (AB หมายถึง A AND B) · เครื่องหมายบวก $+$ คือ OR

1 เกตเจ็ดตัว = ตารางเจ็ดตาราง

ลืมคำว่า "ลอจิกเกต" ไปก่อนหนึ่งนาที เรามาดูของที่น้องเคยเห็นตั้งแต่เด็ก คือ **สวิตช์ไฟกับหลอดไฟ**

รูปข้างล่างมีวงจรสองแบบ แบบซ้ายเอาสวิตช์สองตัวมา **ต่ออนุกรม** คือต่อเรียงกันบนสายเส้นเดียว แบบขวาเอาสวิตช์สองตัวมา **ต่อขนาน** คือแยกเป็นสองทางเดินแล้วมาบรรจบกัน



ต่ออนุกรม — ไฟติดเมื่อ...? **ต่อขนาน — ไฟติดเมื่อ...?**

สวิตช์วาดในท่า "เปิดอยู่" คือยกก้านขึ้น กระแสยังผ่านไม่ได้

ลองคิดในหัวก่อน ไม่ต้องเขียน — วงจรซ้าย (อนุกรม) น้องต้องกดสวิตช์กี่ตัวไฟถึงจะติด? แล้ววงจรขวา (ขนาน) ล่ะ?

ถ้าตอบได้แล้ว ลองเติมช่องว่างในตารางนี้ในใจ โดยให้ **1 = กดสวิตช์ลง (วงจรต่อถึงกัน)** และ **0 = ยกสวิตช์ขึ้น** ส่วน X คือหลอดไฟ 1 = ติด 0 = ดับ

A	B	X อนุกรม	X ขนาน
0	0	?	?
0	1	?	?
1	0	?	?
1	1	?	?

ตอบว่า "อนุกรมต้องกดทั้งสองตัว ขนานกดตัวใดตัวหนึ่งก็พอ" ใช่ไหม ถ้าใช่ น้องเพิ่งเขียน **ตารางค่าความจริง (truth table)** ของเกตสองตัวแรกเสร็จไปแล้ว โดยไม่ต้องมีใครสอน

ตั้งชื่อให้ของที่น้องเพิ่งทำ

ตารางค่าความจริง คือตารางที่ไล่อินพุตทุกความเป็นไปได้ครบทุกแถว แล้วบอกว่าเอาต์พุตเป็นอะไรในแต่ละแถว อินพุต n ตัว ได้ 2^n แถวเสมอ ($n = 2$ ได้ 4 แถว, $n = 3$ ได้ 8 แถว)

วงจรอนุกรมของน้อง มีชื่อว่า **AND** เขียนเป็นนิพจน์ว่า $X = AB$

วงจรขนานของน้อง มีชื่อว่า **OR** เขียนเป็นนิพจน์ว่า $X = A + B$

เหตุผลที่ AND ใช้เครื่องหมายคูณ และ OR ใช้เครื่องหมายบวก ไม่ใช่เรื่องบังเอิญ ลองแทนค่าดู: $0 \cdot 0 = 0, 0 \cdot 1 = 0, 1 \cdot 1 = 1$ — เหมือนตาราง AND เป๊ะ และ $0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 1$ — เหมือนตาราง OR ทุกแถว ยกเว้นบรรทัดสุดท้ายที่คณิตศาสตร์ปกติจะได้ 2 แต่ในโลกบูลีนมีแค่ 0 กับ 1 เลยตัดเป็น 1

ศัพท์ที่อาจารย์ใช้

สไลด์เรียกการ AND ว่า **Boolean multiplication** (การคูณแบบบูลีน) และเรียกการ OR ว่า **Boolean addition** (การบวกแบบบูลีน) เจอคำนี้ในข้อสอบให้แปลกลับเป็น AND กับ OR ได้ทันที

ตัวที่สาม: สวิตช์ที่ทำตรงข้าม

ทีนี้ลองนึกถึงสวิตช์อีกแบบหนึ่ง ที่ **กดแล้วไฟดับ ปล่อยแล้วไฟติด** — เหมือนปุ่มตรงกรอบประตูตู้เย็น พอปิดประตู ปุ่มถูกกด ไฟในตู้ดับ พอเปิดประตู ปุ่มเด้ง ไฟติด

ของแบบนี้มีอินพุตตัวเดียว และมันแค่ **กลับค่า** ชื่อของมันคือ **NOT** หรือ **inverter** (ตัวกลับค่า) เขียนว่า $X = \overline{A}$ อ่านว่า "เอ-บาร์"

เอา NOT ไปแปะท้าย AND ก็ได้ตัวใหม่ เอาไปแปะท้าย OR ก็ได้อีกตัวหนึ่ง สองตัวนี้ชื่อ **NAND** (NOT-AND) และ **NOR** (NOT-OR) ตารางของมันคือตาราง AND และ OR ที่กลับค่าคอลัมน์สุดท้ายทั้งคอลัมน์

ตัวที่หก: สวิตช์บันได

บ้านสองชั้นที่มีสวิตช์ไฟบันไดสองจุด — จุดหนึ่งข้างล่าง จุดหนึ่งข้างบน — น้องกดจุดไหนก็ได้ ไฟจะสลับสถานะ เปิดเป็นปิด ปิดเป็นเปิด

ลองคิดในหัวก่อน — ถ้าสวิตช์ล่าง A และสวิตช์บน B อยู่ที่ตำแหน่ง "เหมือนกัน" (0,0 หรือ 1,1) ไฟติดหรือดับ? แล้วถ้าอยู่ตำแหน่ง "ต่างกัน" (0,1 หรือ 1,0) ล่ะ?

ตอบ: ต่างกัน → ไฟติด · เหมือนกัน → ไฟดับ นี่คือเกต **XOR** (exclusive-OR) เขียนว่า $X = A \oplus B$ และถ้าเอา NOT ไปแปะท้าย ก็ได้ **XNOR** ที่ "เหมือนกันแล้วติด"

คำว่า exclusive แปลว่า "แบบกีดกัน" — OR ธรรมดายอมให้จริงทั้งคู่ได้ แต่ XOR **กีดกัน** กรณีจริงทั้งคู่ออกไป เหลือแค่ "จริงอย่างใดอย่างหนึ่ง"

รวมเจ็ดตัวไว้ที่เดียว — ท่องไม่ต้อง อ่านจากรูปได้

เจ็ดตัวนี้คือทั้งหมดที่ข้อสอบใช้ แถวที่เอาต์พุตเป็น 1 ถูกระบายเขียวไว้ให้แล้ว เวลาอ่านให้ดูว่า "เขียวอยู่แถวไหน" อย่าไปท่องเป็นตัวเลขสี่ตัวเรียงกัน

NOT — กลับค่า



$$X = \overline{A}$$

A	X
0	1
1	0

AND — ครบทุกขาถึงเป็น 1



$$X = AB$$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

OR — มีขาไหนเป็น 1 ก็พอ



$$X = A + B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

NAND — AND แล้วกลับค่า



$$X = \overline{AB}$$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NOR — OR แล้วกลับค่า



$$X = \overline{A + B}$$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

XOR — สองขาต่างกันถึงเป็น 1



$$X = A \oplus B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

XNOR — สองขาเหมือนกันถึงเป็น 1



$$X = \overline{A \oplus B}$$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

เกตพวกนี้มีมากกว่าสองขาได้ อาจารย์วาดไว้ทั้งแบบ 3 ขาและ 4 ขา กฎยังเหมือนเดิม: AND 3 ขา $X = ABC$ เป็น 1 **แถวเดียว** จาก 8 แถว (แถวที่อินพุตเป็น 111) ส่วน OR 3 ขา $X = A + B + C$ เป็น 0 **แถวเดียว** (แถว 000)

ทำผิดให้ดู

คนส่วนใหญ่พลาดตรงนี้: เห็น $X = \overline{AB}$ แล้วอ่านในใจว่า "A ต้องไม่เป็น 1 และ B ต้องไม่เป็น 1"

ถ้าอ่านแบบนั้น น้องจะได้ตาราง 1,0,0,0 ซึ่ง **คือ NOR ไม่ใช่ NAND** — ผิดไป 2 แถวจาก 4 แถว

ขีดบนคลุม **ทั้ง** AB แปลว่า "ทำ AND ให้เสร็จก่อน แล้วค่อยกลับค่าผลลัพธ์" ไม่ใช่ "กลับค่า A กับ B แยกกัน" ตรงนี้คือหัวใจของ De Morgan ที่จะเจอในหัวข้อ 3

อ่านผิด (ได้ NOR)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

NAND ของจริง

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

1

เจอชื่อเกต ให้นึกถึงประโยคเดียว

AND = "ต้องครบทุกขา" · OR = "ขาไหนก็ได้" · XOR = "ต้องต่างกัน" · ตัวที่ขึ้นต้นด้วย N = เอาผลของตัวที่ไม่มี N มากลับค่า

2 **เจอสัญลักษณ์วงจร ให้ดูสองอย่าง**

① รูปทรงตัวกาต์ — ท้ายตัด = AND, ท้ายแหลม = OR, สามเหลี่ยม = NOT · ② มีวงกลมเล็ก (บับเบิล) ที่ขาออกไหม ถ้ามี แปลว่ากลับค่า

3 **ไม่แน่ใจ ให้กางตาราง**

อินพุต n ตัว เขียน 2^n แถว นับ 0 ขึ้นไปเป็นเลขฐานสอง แล้วเติมเอาต์พุตทีละแถว — ช้ากว่าแต่ไม่เคยผิด **และหยุดตรงนี้** อย่าเพิ่งลดรูปอะไรทั้งนั้น

2 เกตตัวเดียว วาดได้สองรูป

เครื่องปั๊มโลหะในโรงงานมีปุ่มสองปุ่ม ต้องกดพร้อมกันสองมือเครื่องถึงจะทำงาน — ออกแบบมาเพื่อไม่ให้มือใครค้างอยู่ใต้แท่นปั๊ม

ทีนี้วิศวกรสองคนเขียนข้อกำหนดของ "ไฟเตือนสีแดง" ไว้คนละประโยค

สองประโยค วงจรเดียวกัน

คนที่หนึ่งเขียนว่า: **"ไฟแดงติด เมื่อ ไม่ได้กดครบทั้งสองปุ่ม"**

คนที่สองเขียนว่า: **"ไฟแดงติด เมื่อ ปุ่มซ้ายไม่ถูกกด หรือ ปุ่มขวาไม่ถูกกด"**

ลองคิดในหัวก่อน — สองประโยคนี้สั่งวงจรให้ทำต่างกันไหม? ลองไล่ทีละกรณี: กดครบสองปุ่ม / กดแค่ซ้าย / กดแค่ขวา / ไม่กดเลย

ทั้งสี่กรณีให้คำตอบตรงกันหมด — เป็นวงจรเดียวกัน ประโยคแรกคือ NAND ("ไม่ใช่ (ซ้าย AND ขวา)") ประโยคที่สองคือ OR ของขาที่ถูกกลับค่าไว้แล้ว

คำแรก: opposing logic

ก่อนจะไปต่อ ต้องแยกให้ออกระหว่างสองคำที่หน้าตาคล้ายกันมากในข้อสอบ คำแรกคือ **opposing logic** (ลอจิกที่ตรงข้ามกัน)

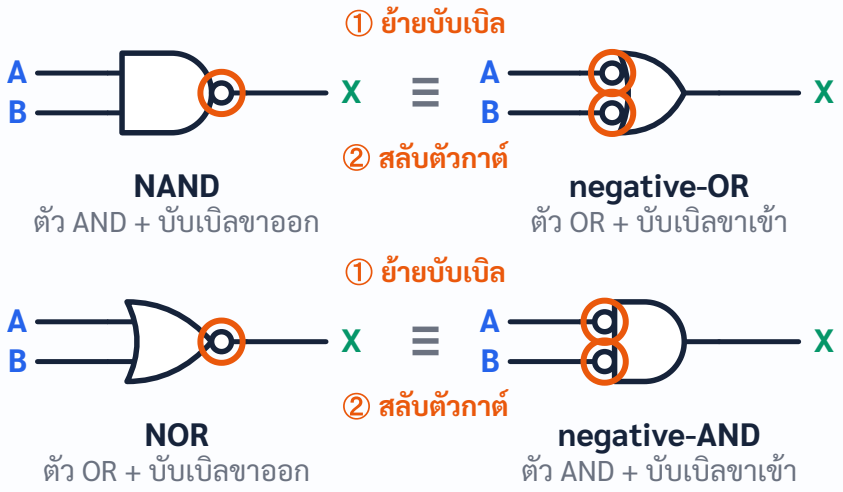
opposing logic หมายถึงเกตสองตัวที่ **คอลัมน์เอาต์พุตกลับค่ากันทุกแถว** มีสามคู่พอดี คือ AND กับ NAND, OR กับ NOR, XOR กับ XNOR ตรวจง่าย ๆ ว่าเอา 1 ลบด้วยค่าเดิมได้ค่าของอีกตัวไหม

ขอย้ำว่า opposing logic คือ **เกตคนละตัว ทำงานคนละอย่าง** AND ไม่ใช่ NAND แค่เขียนคู่กันเพราะจำง่าย

คำที่สอง: dual symbol

dual symbol (สัญลักษณ์คู่) เป็นคนละเรื่องเลย มันคือ **เกตตัวเดียวกัน วาดได้สองรูป** ตัว IC ยังเป็นตัวเดิม ขายังต่อเหมือนเดิม เปลี่ยนแค่รูปที่เราวาดลงกระดาษ

วิธีเปลี่ยนจากรูปหนึ่งไปอีกรูปหนึ่งมีสองขั้น ทำพร้อมกันทั้งสองขั้นเสมอ ทำขั้นเดียวไม่ได้



วงส้ม = บับเบิลที่ย้ายที่ · แถวบน NAND กับ negative-OR คือ IC ตัวเดียวกัน · แถวล่าง NOR กับ negative-AND ก็เช่นกัน

กฎการวาดรูปคู่ (ทำสองขั้นพร้อมกัน)

ขั้นที่ ① ย้ายบับเบิล — ถ้าเดิมอยู่ขาออก ให้ย้ายไปอยู่ที่ขาเข้าทุกขา ถ้าเดิมอยู่ขาเข้า ให้ย้ายไปอยู่ขาออก

ขั้นที่ ② สลับตัวกาต์ — ตัว AND (ท้ายตัด) เปลี่ยนเป็นตัว OR (ท้ายแหลม) และตัว OR เปลี่ยนเป็นตัว AND

พิสูจน์ด้วยตาราง ว่าสองรูปนี้คือเกตเดียวกันจริง

อย่าเชื่อเพราะหนังสือบอก เรากางตารางดูเองทีละแถว ทางซ้ายคือรูป NAND (AND แล้วค่อยกลับค่า) ทางขวาคือรูป negative-OR (กลับค่าขาเข้าก่อน แล้วค่อย OR)

A	B	AB	NAND $\overline{AB}$	$\overline{A}$	$\overline{B}$	neg-OR $\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

คอลัมน์ที่ 4 กับคอลัมน์ที่ 7 เหมือนกันทั้งสี่แถว

แถว 00: AND ได้ 0 กลับค่าเป็น 1 · อีกทางคือ $1+1=1$

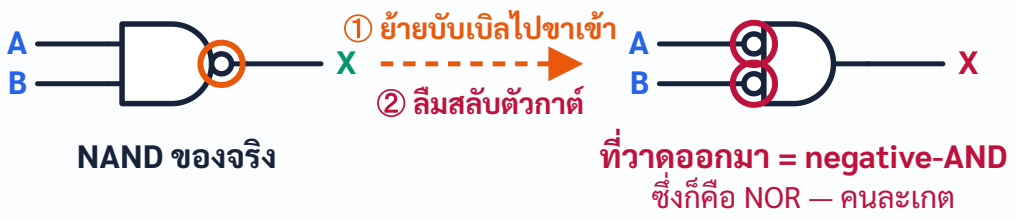
แถว 11: AND ได้ 1 กลับค่าเป็น 0 · อีกทางคือ $0+0=0$

เหมือนกันครบทุกแถว จึงเป็นเกตเดียวกัน **ไม่ใช่แค่คล้าย**

คู่ NOR กับ negative-AND ก็ตรวจแบบเดียวกัน คอลัมน์ $\overline{A+B}$ กับ $\overline{A}\,\overline{B}$ ออกมาเป็น 1,0,0,0 เท่ากันทั้งคู่

ทำผิดให้ดู: ย้ายบับเบิลแต่ลืมสลับตัวกาต์

ข้อสอบชอบวางกับดักตรงนี้มาก นักศึกษาจำได้ว่า "บับเบิลย้ายไปขาเข้าได้" แล้วก็ย้ายบับเบิลจากขาออกของ NAND ไปไว้ที่ขาเข้าทั้งสองขา — แต่ปล่อยตัวกาต์ไว้เป็นตัว AND เหมือนเดิม



ตัวกาต์ยังท้ายตัด (AND) เหมือนเดิม รูปนี้จึงไม่ใช่ NAND อีกต่อไป

รูปที่วาดออกมาคือ negative-AND ซึ่งเราเพิ่งพิสูจน์ไปเมื่อกี้ว่ามันคือ **NOR** ลองเทียบคอลัมน์กัน

A	B	NAND ที่ต้องการ $\overline{AB}$	ที่วาดผิด $\overline{A}\,\overline{B}$	ตรงกันไหม
0	0	1	1	ตรง
0	1	1	0	ผิด
1	0	1	0	ผิด
1	1	0	0	ตรง

ผิดไปสองแถวจากสี่แถว ถ้านี่เป็นวงจรจริงในแลบ LED จะติดผิดครึ่งหนึ่งของกรณีทั้งหมด

วิธีกันพลาด

ทุกครั้งที่ย้ายบับเบิล ให้ถามตัวเองออกเสียงว่า **"แล้วสลับตัวกาต์หรือยัง"** ถ้ารูปใหม่ยังท้ายตัดเหมือนรูปเก่า แปลว่าลืมแน่นอน

ตรวจซ้ำได้ในสิบวินาที: ลองแทน $A = 0, B = 1$ ลงทั้งสองรูป ถ้าได้คนละค่า แปลว่าวาดผิด

3 De Morgan — กฎที่อยู่เบื้องหลังรูปคู่

หัวข้อที่แล้วเราพิสูจน์ด้วยตารางว่า NAND กับ negative-OR เป็นตัวเดียวกัน แต่ตารางบอกได้แค่ว่า "มันเท่ากัน" ไม่ได้บอกว่า "ทำไม" และที่สำคัญกว่านั้น ตารางช่วยไม่ได้เวลาเจอโจทย์ที่มีตัวแปรห้าตัว

สิ่งที่ต้องมีคือกฎที่ใช้กับนิพจน์ได้โดยตรง กฎนั้นชื่อ **De Morgan's laws** และอาจารย์สรุปไว้เป็นประโยคเดียวว่า **"Break the bar and change the sign!"** — หักขีดแล้วเปลี่ยนเครื่องหมาย



ขีดเดียวคลุมทั้งก้อน

คูณอยู่ใต้ขีดยาว

ขีดสั้นสองเส้น คนละตัว

กลายเป็นบวกของสองตัวที่มีขีด

กฎข้อที่หนึ่ง: $\overline{AB} = \overline{A} + \overline{B}$ — ขีดยาวเส้นเดียวถูกหักเป็นขีดสั้นสองเส้น และ $\cdot$ กลายเป็น $+$

ลองคิดในหัวก่อน — ถ้ากฎนี้เดินย้อนทางได้ แล้ว $\overline{A + B}$ ควรจะเท่ากับอะไร?

หักขีดเหมือนเดิม แล้วเปลี่ยน $+$ เป็น $\cdot$ ได้ $\overline{A + B} = \overline{A}\,\overline{B}$ — นี่คือกฎข้อที่สอง

De Morgan's Laws

$$\overline{AB} = \overline{A} + \overline{B}$$

ข้อที่หนึ่ง — ขีดคลุมการคูณ หักแล้วได้บวก

$$\overline{A + B} = \overline{A}\,\overline{B}$$

ข้อที่สอง — ขีดคลุมการบวก หักแล้วได้คูณ

ทั้งสองข้อใช้ได้ **ทั้งสองทิศทาง** คือจะกางออก (ซ้ายไปขวา) หรือยุบกลับ (ขวาไปซ้าย) ก็ได้

ตรวจทั้งสองข้อด้วยตาราง

เขียนคอลัมน์กลางทุกคอลัมน์ให้ครบ อย่าข้าม เพราะตอนสอบถ้าไม่แน่ใจน้องจะต้องกางแบบนี้จริง ๆ

ข้อที่หนึ่ง: $\overline{AB} = \overline{A} + \overline{B}$

A	B	AB	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

ข้อที่สอง: $\overline{A + B} = \overline{A}\,\overline{B}$

A	B	A+B	$\overline{A + B}$	$\overline{A}\,\overline{B}$
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0

สองคอลัมน์ขวาสุดของแต่ละตารางเท่ากันทุกแถว กฎนี้จึงใช้ได้จริง ไม่ใช่กฎที่ครูบอกให้ท่อง

ใช้กับสามตัวแปร

กฎนี้ไม่ได้จำกัดที่สองตัว ขีดคลุมกี่ตัวก็หักได้หมด ขอแค่เปลี่ยนเครื่องหมายทุกตัวที่อยู่ใต้ขีดเดิม

$$\overline{ABC} = \overline{A} + \overline{B} + \overline{C}$$

ตรวจแล้วเท่ากันครบ 8 แถว

$$\overline{A + B + C} = \overline{A}\,\overline{B}\,\overline{C}$$

ตรวจแล้วเท่ากันครบ 8 แถว

เหตุผลที่ขยายได้ ไม่ใช่เพราะ "มันน่าจะได้" แต่เพราะจัดกลุ่มแล้วใช้กฎสองตัวซ้ำสองรอบ: $\overline{ABC} = \overline{(AB)C} = \overline{(AB)} + \overline{C} = \overline{A} + \overline{B} + \overline{C}$ — รอบแรกมอง AB เป็นก้อนเดียว รอบที่สองค่อยแตกก้อนนั้น

ทำให้ดูทุกบรรทัด: ถอดขีดออกจนไม่มีขีดไหนคลุมเกินหนึ่งตัวแปร

นี่คือรูปแบบโจทย์ที่อาจารย์ให้ไว้ตรง ๆ คำสั่งคือ "Remove every bar so that no bar covers more than one variable" แปลว่า พอทำเสร็จ ขีดทุกเส้นต้องเหลือสั้น ๆ คลุมตัวแปรเดียว

เครื่องมือเพิ่มอีกหนึ่งชิ้น

ระหว่างทางจะเจอขีดซ้อนขีด เช่น $\overline{\overline{A}}$ กฎที่ใช้คือ **involution** หรือ **กฎขีดสองชั้น**: $\overline{\overline{A}} = A$

ที่มาง่ายมาก: $A = 0$ กลับเป็น 1 กลับอีกทีเป็น $0 \cdot A = 1$ กลับเป็น 0 กลับอีกทีเป็น 1 — ครบสองแถว กลับสองครั้งได้ของเดิมเสมอ

โจทย์ที่ 1

$(\overline{A} + B + \overline{C})'$

$\overline{\overline{A} + B + \overline{C}}$	โจทย์ — ขีดยาวคลุมการบวกของสามก้อน
$= \overline{\overline{A}} \cdot \overline{B} \cdot \overline{\overline{C}}$	หักขีดยาว เปลี่ยน $+$ เป็น $\cdot$ แล้วแปะขีดให้ทุกก้อน (De Morgan ข้อสอง)
$= A\,\overline{B}\,C$	ขีดสองชั้นหายไปสองที่ (involution) — ตอบ

ตรวจกลับ: เอา $A = 1, B = 0, C = 1$ ใส่โจทย์เดิม ได้ $\overline{0 + 0 + 0} = \overline{0} = 1$ และคำตอบได้ $1 \cdot 1 \cdot 1 = 1$ ตรงกัน

โจทย์ที่ 2

$((A + B)C)'$ — นิพจน์สามตัวแปร

$\overline{(A + B)C}$	โจทย์ — ใต้ขีดยาวเป็นการคูณของสองก้อน คือก้อน $(A + B)$ กับก้อน C
$= \overline{(A + B)} + \overline{C}$	หักขีดยาว เปลี่ยน $\cdot$ เป็น $+$ (De Morgan ข้อหนึ่ง) — ก้อน $(A + B)$ ยังมีขีดคลุมอยู่
$= \overline{A}\,\overline{B} + \overline{C}$	หักขีดที่เหลือ เปลี่ยน $+$ เป็น $\cdot$ (De Morgan ข้อสอง)
$= \overline{A}\,\overline{B} + \overline{C}$	ทุกขีดคลุมตัวแปรเดียวแล้ว — ตอบ

ตรวจกลับทั้ง 8 แถว เพราะมีสามตัวแปร:

A	B	C	A+B	(A+B)C	โจทย์ $\overline{(A+B)C}$	คำตอบ $\overline{AB}+\overline{C}$
0	0	0	0	0	1	1
0	0	1	0	0	1	1
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
1	1	0	1	0	1	1
1	1	1	1	1	0	0

สองคอลัมน์ขวาสุดเท่ากันครบ 8 แถว

โจทย์ที่ 3 และ 4 ฝึกเอง แล้วเทียบเฉลย

$(AB+CD)'$ และ $(\overline{A}\,\overline{B})'$

เฉลย

$\overline{AB+CD}=\overline{AB}\cdot\overline{CD}=(\overline{A}+\overline{B})(\overline{C}+\overline{D})$ — ใต้ขีดเป็นการบวก หักแล้วได้คูณ แล้วหักต่ออีกชั้นในแต่ละก้อน (ตรวจแล้วเท่ากันครบ 16 แถว)

$\overline{\overline{A}\,\overline{B}}=\overline{\overline{A}}+\overline{\overline{B}}=A+B$ — นี่คือเหตุผลว่าทำไม negative-AND ถึงเป็น NOR: ขาเข้ากลับค่าแล้ว AND กัน มีค่าเท่ากับ $\overline{A+B}$

ทำผิดให้ดู

ความผิดที่เจอบ่อยที่สุดคือ "หักขีดแต่ลืมเปลี่ยนเครื่องหมาย" เช่น เขียนว่า $\overline{A+B}=\overline{A}+\overline{B}$

ลองแทน $A=0, B=1$ ดู: ข้างซ้าย $\overline{0+1}=\overline{1}=0$ · ข้างขวา $\overline{0}+\overline{1}=1+0=1$ — **ได้ 0 กับ 1 คนละค่า** จึงไม่เท่ากันแน่นอน

อีกอันที่พลาดบ่อยคือหักขีดแค่บางตัว เช่น $\overline{ABC}=\overline{A}+BC$ — ขีดเดิมคลุมทั้งสามตัว เวลาหักต้องแจกขีดให้ครบทั้งสามตัว ห้ามทิ้งตัวไหนไว้เปล่า ๆ

1 **หาขีดที่ยาวที่สุดก่อน**
ถ้ามีขีดซ้อนกันหลายชั้น ให้เริ่มจากขีด **นอกสุด** เสมอ อย่าเริ่มจากขีดข้างใน

2 **ดูว่าใต้ขีดนั้นเป็นบวกหรือคูณ**
ใต้ขีดเป็นคูณ → หักแล้วได้บวก · ใต้ขีดเป็นบวก → หักแล้วได้คูณ · แล้วแปะขีดใหม่ให้ **ทุกก้อน** ที่อยู่ใต้ขีดเดิม

3 **ทำซ้ำจนขีดทุกเส้นคลุมตัวแปรเดียว**
ระหว่างทางเจอ $\overline{\overline{X}}$ เมื่อไหร่ ตัดทิ้งเป็น X ได้เลย · **หยุดเมื่อ** ไม่มีขีดไหนคลุมเกินหนึ่งตัวแปร

4 กฎพื้นฐานที่ข้อสอบใช้จริง

อาจารย์เปิดหัวข้อนี้ด้วยวงจรที่ดูน่ากลัวมากวงจรหนึ่ง มันใช้ NOT สองตัว AND สองขาสามตัว และ OR สามขาอีกหนึ่งตัว รวมหกตัว นิพจน์ของมันคือ $X = \overline{A}B + A\overline{B} + AB$

ลองเติมตารางนี้ในใจก่อน — แต่ละแถวดูทีละพจน์ว่าพจน์ไหนเป็น 1 บ้าง ถ้ามีพจน์ใดพจน์หนึ่งเป็น 1 ผลรวมก็เป็น 1

A	B	$\overline{A}B$	$A\overline{B}$	AB	X
0	0	0	0	0	?
0	1	1	0	0	?
1	0	0	1	0	?
1	1	0	0	1	?

ได้ 0, 1, 1, 1 — **นี่คือตาราง OR เป๊ะ ๆ** วงจรหกตัวนั้นทำงานเหมือน OR สองขาตัวเดียว

น้องเพิ่งเห็นเหตุผลที่ต้องเรียนพีชคณิตบูลีน: วงจรหกตัวกับวงจรตัวเดียว ให้ผลเหมือนกันทุกแถว แต่ราคา พื้นที่บนแผ่นวงจร และไฟที่กินต่างกันหกเท่า

การกางตารางแบบเมื่อกี้ใช้ได้เมื่อมีสองตัวแปร แต่ถ้ามีห้าตัวแปรจะได้ 32 แถว และข้อสอบไม่มีเวลาให้ทำแบบนั้น เราจึงต้องมีกฎที่ลดรูปได้ตรง ๆ บนบรรทัดเดียว

กฎตัวแปรเดียว — ตรวจครบในตารางเดียวสองแถว

กฎกลุ่มแรกมีตัวแปรตัวเดียว ตัวแปรตัวเดียวมีแค่สองค่า ตารางจึงมีแค่สองแถว และตรวจได้หมดในตารางเดียว

A	$\overline{A}$	A+0	A·1	A+1	A·0	A+A	A·A	$A+\overline{A}$	$A\overline{A}$	$\overline{\overline{A}}$
0	1	0	0	1	0	0	0	1	0	0
1	0	1	1	1	0	1	1	1	0	1

อ่านตารางนี้ทีละคอลัมน์แล้วเทียบกับคอลัมน์ A ทางซ้ายสุด จะได้กฎทั้งหมดนี้ทันที

เอกลักษณ์ — identity

$A+0=A \;\cdot\; A\cdot 1=A$

คอลัมน์ A+0 และ A·1 ลอกคอลัมน์ A มาทั้งสองแถว — 0 กับ OR และ 1 กับ AND ไม่เปลี่ยนอะไรเลย

ศูนย์และหนึ่ง — null

$A+1=1 \;\cdot\; A\cdot 0=0$

คอลัมน์ A+1 เป็น 1 ทั้งสองแถว และ A·0 เป็น 0 ทั้งสองแถว — ค่าของ A ไม่มีผลเลย

ซ้ำตัวเอง — idempotent

$A+A=A \;\cdot\; A\cdot A=A$

$0+0=0, 1+1=1$ และ $0\cdot 0=0, 1\cdot 1=1$ — เขียนพจน์เดิมซ้ำกี่ครั้งก็เท่ากับเขียนครั้งเดียว

ส่วนเติมเต็ม — complement

$A+\overline{A}=1 \;\cdot\; A\overline{A}=0$

A กับ $\overline{A}$ เป็นคนละค่ากันเสมอ — OR จึงเจอ 1 ทุกแถว และ AND จึงเจอ 0 ทุกแถว

กฎสองข้อที่เหลือ อ่านง่ายจนเกือบไม่ต้องพิสูจน์: **สลับที่ (commutative)** $A+B=B+A$ และ $AB=BA$ · **จัดกลุ่ม (associative)** $A+(B+C)=(A+B)+C$ และ $A(BC)=(AB)C$ — สองข้อนี้บอกว่าจะสลับขาเข้าของ AND หรือ OR ยังไง หรือจะจับคู่กันก่อนคู่ไหน ผลก็เท่าเดิม

กฎสองตัวแปร — ต้องดูที่มา ไม่งั้นจำสลับ

กฎกลุ่มนี้คือกลุ่มที่ทำคะแนนหายจริง เพราะหน้าตามันคล้ายกันมาก เราจะไม่ท่อง แต่จะสร้างมันขึ้นมาจากกฎตัวแปรเดียวที่เพิ่งตรวจไป

กฎการกระจาย แบบที่หนึ่ง (distributive)

$$A(B+C)=AB+AC$$

อันนี้เหมือนการแจกแจงในคณิตศาสตร์ปกติเป๊ะ แจก A เข้าไปคูณทุกก้อนในวงเล็บ อ่านย้อนทางก็คือ "ดึงตัวร่วมออกมา"

กฎการกระจาย แบบที่สอง

$$(A+B)(A+C) = A+BC$$

อันนี้ **ไม่มีในคณิตศาสตร์ปกติ** ($(2+3)(2+4) \neq 2+3\cdot 4$) จึงต้องสร้างให้เห็น:

$(A+B)(A+C) = AA + AC + BA + BC$ กระจายแบบที่หนึ่งสองรอบ

$= A + AC + AB + BC$ $AA = A$ (ซ้ำตัวเอง)

$= A(1 + C + B) + BC$ ดึง A ออก

$= A\cdot 1 + BC = A + BC$ $1+$อะไรก็เป็น 1 (null) แล้ว $A\cdot 1 = A$ (เอกลักษณ์)

กฎการดูดกลืน (absorption) — สามหน้าตา

① $A + AB = A$ ที่มา: $A + AB = A(1+B) = A\cdot 1 = A$

② $A(A+B) = A$ ที่มา: $AA + AB = A + AB = A$ (ใช้ข้อ ① ต่อ)

③ $A + \overline{A}B = A + B$ ที่มา: $A + \overline{A}B = (A+\overline{A})(A+B) = 1\cdot(A+B) = A + B$ (กระจายแบบที่สอง แล้วใช้ส่วนเติมเต็ม)

ให้สังเกตความต่างของ ① กับ ③ ให้ดี: ตัวที่คูณอยู่กับ B ถ้าเป็น A ตัวเดียวกัน พจน์หลัง **หายไปเลย** แต่ถ้าเป็น $\overline{A}$ ขีดจะ **หายไปเหลือแต่** B

เรื่องชื่อกฎ

สไลด์ของวิชานี้เรียกกฎบางข้อด้วยชื่อที่ไม่ตรงกันในแต่ละหน้า เช่น $A + \overline{A} = 1$ ถูกเรียกทั้ง Complementary Law และ Null Law ในคนละหน้า

เวลาสอบ ให้ยึด **สูตร** เป็นหลัก ถ้าข้อสอบให้ระบุชื่อกฎ ให้เขียนชื่อพร้อมสูตรกำกับไว้ด้วยเสมอ ผู้ตรวจจะเห็นว่าน้องใช้กฎถูกข้อ

ทำให้ดูทุกบรรทัด: ลดรูป $AB + A(B+C) + B(B+C)$

โจทย์นี้มาจากสไลด์ของอาจารย์ตรง ๆ ทุกบรรทัดจะมีชื่อกฎกำกับไว้ข้าง ๆ ถ้าบรรทัดไหนเขียนไม่ได้ว่าใช้กฎอะไร แปลว่าบรรทัดนั้นมั่ว

ทำให้ดู — ลดรูปทีละบรรทัด

$AB + A(B+C) + B(B+C)$	โจทย์ — สามพจน์ มีวงเล็บสองอัน

$= AB + AB + AC + BB + BC$	กระจายแบบที่หนึ่ง — แจก A เข้าวงเล็บแรก แจก B เข้าวงเล็บที่สอง
$= AB + AB + AC + B + BC$	ซ้ำตัวเอง — $BB = B$
$= AB + AC + B + BC$	ซ้ำตัวเอง — $AB + AB = AB$
$= AB + AC + B$	ดูดกลืน ① — $B + BC = B$
$= B + AC$	ดูดกลืน ① อีกครั้ง — $B + AB = B$ (ตอบ)

ตรวจคำตอบกลับเข้าโจทย์เดิม เลือกสองแถวที่ผลต่างกัน

$A = 1, B = 0, C = 1 \rightarrow$ โจทย์เดิม $= 1 \cdot 0 + 1(0 + 1) + 0(0 + 1) = 0 + 1 + 0 = 1 \cdot$ คำตอบ $= 0 + 1 \cdot 1 = 1$ **ตรง**

$A = 0, B = 0, C = 1 \rightarrow$ โจทย์เดิม $= 0 + 0(0 + 1) + 0(0 + 1) = 0 \cdot$ คำตอบ $= 0 + 0 \cdot 1 = 0$ **ตรง**

ตรวจด้วยโปรแกรมครบทั้ง 8 แถว ทุกบรรทัดกลางทางก็เท่ากับโจทย์เดิมหมด — ผลลัพธ์เป็น 1 ที่แถว 2, 3, 5, 6, 7

วงจรเดิมต้องใช้เกต 5 ตัว (AND สามตัว, OR สองขาหนึ่งตัว, OR สามขาหนึ่งตัว) วงจรใหม่ใช้ 2 ตัว (AND หนึ่ง, OR หนึ่ง) และให้ผลเท่ากันทุกแถว

ทำให้ดูอีกข้อ: เมื่อมีขีดบนมาปนกับกฎพื้นฐาน

ข้อนี้รวมสองหัวข้อเข้าด้วยกัน ต้อง **ถอดขีดก่อน** แล้วค่อยลดรูป เพราะกฎพื้นฐานทำงานกับพจน์ที่กางออกแล้วเท่านั้น

ทำให้ดู $\overline{AB + AC} + \overline{A}\,\overline{BC}$

$\overline{AB + AC} + \overline{A}\,\overline{BC}$	โจทย์ — ขีดยาวคลุมการบวกของสองพจน์
$= \overline{AB} \cdot \overline{AC} + \overline{A}\,\overline{BC}$	De Morgan ข้อสอง — ใต้ขีดเป็นบวก หักแล้วได้คูณ
$= (\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A}\,\overline{BC}$	De Morgan ข้อหนึ่ง สองครั้ง — หักขีดที่เหลือ
$= \overline{A} + \overline{B}\,\overline{C} + \overline{A}\,\overline{BC}$	กระจายแบบที่สอง — มองว่า $\overline{A}$ คือตัวร่วม: $(X + Y)(X + Z) = X + YZ$
$= \overline{A} + \overline{B}\,\overline{C}$	ดูดกลืน ① — $\overline{A} + \overline{A}(\overline{B}C) = \overline{A}$ (ตอบ)

ตรวจกลับ $A = 1, B = 1, C = 0 \rightarrow$ โจทย์ $= \overline{1 + 0} + 0 = \overline{1} + 0 = 0 \cdot$ คำตอบ $= 0 + 0 \cdot 1 = 0$ **ตรง**

ตรวจด้วยโปรแกรมครบ 8 แถว ทุกบรรทัดเท่ากัน ผลลัพธ์เป็น 1 อยู่ 5 แถว

ทำผิดให้ดู — สองท่าที่เสียคะแนนบ่อยที่สุด

ท่าที่หนึ่ง: ดูดกลืนกลับข้าง เขียนว่า $A + AB = AB$

ลอง $A = 1, B = 0$: ข้างซ้าย $= 1 + 0 = 1 \cdot$ ข้างขวา $= 0$ — คนละค่า ที่ถูกคือพจน์ **ยาว** หายไป เหลือพจน์ **สั้น** เพราะพจน์สั้นครอบคลุมพจน์ยาวอยู่แล้ว

ท่าที่สอง: ตัด A ทิ้งทั้งตัว เขียนว่า $A + \overline{A}B = B$

ลอง $A = 1, B = 0$: ข้างซ้าย $= 1 + 0 \cdot 0 = 1 \cdot$ ข้างขวา $= 0$ — คนละค่าอีก ที่ถูกคือ **ขีด** หายไป ได้ $A + B$ ไม่ใช่ A หายไป

กันพลาดด้วยนิสัยเดียว: **ลดรูปเสร็จทุกครั้ง ให้สุ่มค่ามาสองสามแถว แทนกลับทั้งของเดิมและของใหม่** ถ้าต่างกันแถวเดียวก็คือผิด

1 **ถอดขีดก่อนเสมอ**

ถ้ามีขีดคลุมเกินหนึ่งตัวแปร ใช้ De Morgan จนไม่เหลือ แล้วค่อยเริ่มลดรูป

2 **กางวงเล็บให้หมด**

ใช้กระจายแบบที่หนึ่งจนได้รูปผลบวกของผลคูณ (พจน์บวกกันล้วน ๆ) — ตอนนี้จะเห็นพจน์ซ้ำและพจน์ที่ถูกดูดกลืนชัดขึ้นมาก

3 **ไล่เก็บสามอย่างตามลำดับ**

① พจน์ซ้ำ (ซ้ำตัวเอง) → ② พจน์ที่มีพจน์สั้นกว่าครอบอยู่ (ดูดกลืน) → ③ พจน์ที่มีทั้ง X และ $\overline{X}$ อยู่ด้วยกัน (ได้ 0 ทิ้งได้เลย)

4 **หยุดเมื่อไหร่**

หยุดเมื่อ **ทุกพจน์ที่เหลือ ตัดตัวแปรออกอีกตัวไม่ได้ และตัดพจน์ทิ้งทั้งพจน์ก็ไม่ได้** แล้วแทนค่าตรวจกลับสองแถวก่อนส่ง

5 Consensus theorem — กฎที่ตัดพจน์ที่ไม่ได้ทำอะไรเลย

โรงหนังแห่งหนึ่งประกาศเงื่อนไขเข้าฟรีไว้สามข้อ ให้ A แปลว่า "เป็นสมาชิก" B แปลว่า "มาก่อนบ่ายสอง" C แปลว่า "มีคูปอง"

ประกาศหน้าโรงหนัง

ข้อ 1 เป็นสมาชิก และ มาก่อนบ่ายสอง → เข้าฟรี (AB)

ข้อ 2 ไม่เป็นสมาชิก แต่ มีคูปอง → เข้าฟรี $(\overline{A}C)$

ข้อ 3 มาก่อนบ่ายสอง และ มีคูปอง → เข้าฟรี (BC)

ลองคิดในหัวก่อน — มีใครสักคนไหมที่เข้าฟรีได้ เพราะข้อ 3 เท่านั้น คือถ้าลบข้อ 3 ทิ้งแล้วเขาจะหมดสิทธิ์?

ใบ้ให้: คนที่เข้าข่ายข้อ 3 คือคนที่มาก่อนบ่ายสองและมีคูปอง คนคนนั้นจะเป็นสมาชิกหรือไม่เป็นสมาชิกก็ได้ ลองแยกดูทั้งสองกรณี

ถ้าเขา **เป็นสมาชิก** เขาเข้าข่ายข้อ 1 อยู่แล้ว (สมาชิก + มาก่อนบ่ายสอง) · ถ้าเขา **ไม่เป็นสมาชิก** เขาเข้าข่ายข้อ 2 อยู่แล้ว (ไม่เป็นสมาชิก + มีคูปอง) — ไม่มีใครเลยที่พึ่งข้อ 3 เพียงอย่างเดียว ข้อ 3 จึงลบทิ้งได้

Consensus Theorem

$$AB + \overline{A}C + BC \;=\; AB + \overline{A}C$$

พจน์ BC เรียกว่า **consensus term** (พจน์พ้อง) มันคือพจน์ที่ "ซ้ำซ้อน" กับอีกสองพจน์จนไม่เพิ่มอะไรเลย

ดูจากตารางว่ามันไม่เพิ่มอะไรจริง

รูปข้างล่างคือ 8 แถวของ A, B, C เรียงกัน แต่ละพจน์ครอบคลุมแถวไหนบ้าง ระบายไว้ให้แล้ว



แถบเทาประ (BC) ไม่ได้ครอบแถวใหม่เลยสักแถว → ลบทิ้งได้

AB ครอบแถว 6,7 · $\overline{A}C$ ครอบแถว 1,3 · BC ครอบแถว 3,7 ซึ่งถูกครอบไปหมดแล้วทั้งคู่

ถ้าอยากเห็นเป็นตารางเต็ม ก็ไล่ทั้ง 8 แถวได้ คอลัมน์ขวาสุดสองคอลัมน์เหมือนกันหมด แปลว่าพจน์ BC ไม่เคยเปลี่ยนคำตอบเลยสักแถว

A	B	C	AB	$\overline{A}C$	BC	รวมทั้งสามพจน์	เอาแค่สองพจน์แรก
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

ให้สังเกตสองแถวที่ $BC = 1$ คือแถว 3 กับแถว 7 · แถว 3 มี $\overline{A}C = 1$ อยู่แล้ว · แถว 7 มี $AB = 1$ อยู่แล้ว — ตรงกับที่เราคิดจากเรื่องโรงหนังทุกประการ

ทำไมถึงเป็นอย่างนั้น — พิสูจน์สามบรรทัด

ตารางบอกว่า "มันจริง" แต่เหตุผลอยู่ตรงนี้ เราแยก BC ออกเป็นสองกรณีตามค่าของ A

$BC = BC \cdot 1 = BC(A + \overline{A}) = ABC + \overline{A}BC$ เอกลักษณ์ แล้วใช้ส่วนเติมเต็ม $A + \overline{A} = 1$ แล้วกระจาย

ABC ถูก AB ดูดกลืน เพราะ $AB + ABC = AB$ ดูดกลืน ① โดยมอง AB เป็นตัวเดียว

$\overline{A}BC$ ถูก $\overline{A}C$ ดูดกลืน เพราะ $\overline{A}C + \overline{A}CB = \overline{A}C$ ดูดกลืน ① อีกครั้ง

ทั้งสองชิ้นถูกดูดหมด BC จึงหายไปทั้งพจน์

วิธีมองหา consensus term ในข้อสอบ

กฎที่อาจารย์ให้ไว้ แปลเป็นขั้นตอน

ขั้นที่ ① หาคู่พจน์ที่มีตัวแปรตัวหนึ่ง **โผล่แบบไม่มีขีดในพจน์หนึ่ง และโผล่แบบมีขีดในอีกพจน์หนึ่ง**

ขั้นที่ ② คูณสองพจน์นั้นเข้าด้วยกัน แล้ว **ตัดตัวแปรคู่นั้นกับขีดของมันทิ้ง** สิ่งที่เหลือคือ consensus term

ขั้นที่ ③ ถ้า consensus term นั้น **มีอยู่ในนิพจน์อยู่แล้ว** ให้ลบทิ้งได้เลย

ตัวอย่างการทำขั้นที่ ②: คู่ AB กับ $\overline{A}C$ — ตัวแปรที่เข้าคู่คือ A กับ $\overline{A}$ · คูณกันได้ $AB\overline{A}C$ · ตัด A กับ $\overline{A}$ ทิ้ง เหลือ BC

ทำให้ดู ตัดพจน์พ้องออกจาก $\overline{A}\,\overline{B} + AC + B\overline{C} + \overline{B}C + AB$

นิพจน์นี้มีห้าพจน์ เราไล่หาคู่ที่มีตัวแปรเข้าคู่กันทีละคู่

คู่ที่ ① $\overline{A}\,\overline{B}$ กับ AC	ตัวแปรที่เข้าคู่คือ $\overline{A}$ กับ A → คูณแล้วตัดทิ้ง เหลือ $\overline{B}C$ — **พจน์นี้มีอยู่ในโจทย์! ลบได้**
คู่ที่ ② AC กับ $B\overline{C}$	ตัวแปรที่เข้าคู่คือ C กับ $\overline{C}$ → คูณแล้วตัดทิ้ง เหลือ AB — **พจน์นี้ก็มีอยู่ในโจทย์! ลบได้**

เหลือ $\overline{A}\,\overline{B} + AC + B\overline{C}$ **(ตอบ)**

ตรวจกลับด้วยโปรแกรม ทั้งห้าพจน์รวมกันเป็น 1 ที่แถว 0, 1, 2, 5, 6, 7 และสามพจน์ที่เหลือก็เป็น 1 ที่แถวเดียวกันเป๊ะ ทั้ง 8 แถวตรงกันหมด

รูปคู่ของ consensus (เจอน้อยกว่า แต่ออกได้)

ถ้าโจทย์อยู่ในรูปผลคูณของผลบวก กฎนี้ก็มีหน้าตาคู่ขนานกัน:

$$(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$$

สังเกตว่ารูปร่างเหมือนเดิมทุกอย่าง เปลี่ยนแค่ $+$ กับ $\cdot$ สลับที่กัน (ตรวจแล้วเท่ากันครบ 8 แถว)

ทำผิดให้ดู

กับดักคือ "เห็นสามพจน์แล้วลบพจน์ที่สามทิ้งเลย" โดยไม่ตรวจว่ามันเป็น consensus term จริงไหม

ตัวอย่าง $AB + \overline{A}C + B\overline{C}$ — หน้าตาคล้าย consensus มาก แต่พจน์ที่สามคือ $B\overline{C}$ ไม่ใช่ BC ดังนั้น **ลบไม่ได้**

วิธีกันพลาดคือทำขั้นที่ ② ให้ครบทุกครั้ง: คูณสองพจน์แรกแล้วตัด $A, \overline{A}$ ทิ้ง ได้ BC — เทียบกับพจน์ที่สามคือ $B\overline{C}$ แล้วจะเห็นทันทีว่า **คนละพจน์**

ใช้ตอนไหน

ใช้ตอนที่ **ลดรูปด้วยกฎอื่นจนตันแล้ว** แต่นิพจน์ยังยาวผิดปกติ โดยเฉพาะนิพจน์ที่มีทั้ง X และ $\overline{X}$ กระจายอยู่หลายพจน์

อีกที่ที่จะเจอคือบท K-Map ข้างหน้า — วงที่ทุกช่องของมันถูกวงอื่นคลุมไปหมดแล้ว ก็คือ consensus term ในภาพนั่นเอง ลบวงนั้นทิ้งได้โดยคำตอบไม่เปลี่ยน

6 มินเทอม — ชื่อเรียกแถวของตาราง

กุญแจรหัสสามหลักแบบหมุน ถ้ารหัสคือ 1-0-1 ก็มีเลขชุดเดียวเท่านั้นที่เปิดได้ ชุดอื่นทั้ง 7 ชุด เปิดไม่ออก

ในตารางค่าความจริงสามตัวแปรก็มี 8 แถว เหมือนมีล็อก 8 ชุด และเราอยากมีนิพจน์ที่ "เปิดได้แถวเดียว" ไว้เรียกแต่ละแถวโดยเฉพาะ

ลองคิดในหัวก่อน — นิพจน์ $A\overline{B}C$ เป็น 1 ที่แถวไหนบ้างของตาราง A, B, C (8 แถว)?

ใบ้: ผลคูณจะเป็น 1 ก็ต่อเมื่อทุกตัวที่คูณกันเป็น 1 พร้อมกัน ดังนั้นต้องการ $A = 1$, $\overline{B} = 1$, $C = 1$

$\overline{B} = 1$ แปลว่า $B = 0$ · จึงต้องการ $A = 1, B = 0, C = 1$ พอดีหนึ่งชุด นั่นคือแถว **101 = แถวที่ 5** แถวเดียว แถวอื่นอีก 7 แถวเป็น 0 หมด

สิ่งที่น้องเพิ่งทำ คือการอ่าน **มินเทอม (minterm)** นิพจน์แบบนี้มีสมบัติพิเศษคือเป็น 1 ได้แถวเดียวเสมอ ไม่มากไม่น้อยกว่านั้น

ศัพท์สามคำ ต้องแยกให้ออก

literal — ตัวอักษรหนึ่งตัว

literal คือการปรากฏตัวของตัวแปรหนึ่งครั้ง จะมีขีดหรือไม่มีก็นับเป็นหนึ่ง literal เช่น A เป็นหนึ่ง literal และ $\overline{A}$ ก็เป็นหนึ่ง literal

ถ้า A โผล่สามที่ในนิพจน์ ก็นับเป็นสาม literal — **นับทุกครั้งที่เห็น** จำนวน literal ใช้วัด "ความซับซ้อน" ของนิพจน์

product term — พจน์ผลคูณ

product term คือ literal ตั้งแต่หนึ่งตัวขึ้นไปที่เชื่อมกันด้วย AND เช่น $A\overline{B}$, $BC\overline{D}$ · literal ตัวเดียวโดด ๆ เช่น $\overline{E}$ ก็นับเป็น product term หนึ่งพจน์

minterm — พจน์ผลคูณมาตรฐาน

minterm คือ product term ที่ **ตัวแปรทุกตัวในโจทย์โผล่ครบพอดีตัวละหนึ่งครั้ง** จะมีขีดหรือไม่มีก็ได้ แต่ห้ามขาด และห้ามโผล่ซ้ำ

ถ้าโจทย์มีสี่ตัวแปร A, B, C, D : $AB\overline{C}D$ เป็น minterm · $\overline{A}B\overline{D}$ **ไม่ใช่** เพราะ C หายไป

อาจารย์เรียกอีกชื่อว่า **standard product term** เป็นคำเดียวกัน

ทำให้ดู นับของในนิพจน์ $A\overline{B} + B\overline{C}D + \overline{A}D + \overline{E}$

ตัวแปร	A, B, C, D, E → **5 ตัว**
product term	$A\overline{B}$ / $B\overline{C}D$ / $\overline{A}D$ / $\overline{E}$ → **4 พจน์** (นับจากจำนวนเครื่องหมายบวกบวกหนึ่ง)
literal	2 + 3 + 2 + 1 → **8 ตัว**
minterm	**0 พจน์** — โจทย์มี 5 ตัวแปร แต่พจน์ที่ยาวที่สุดมีแค่ 3 literal ยังไม่ครบ 5

กฎการให้หมายเลขมินเทอม

แต่ละมินเทอมเป็น 1 ได้แถวเดียว เราจึง **ตั้งชื่อมินเทอมด้วยหมายเลขแถวนั้น** เขียนว่า m_i

วิธีแปลง ใช้กฎเดียวสองบรรทัด: **ตัวไหนไม่มีขีด เขียน 1 · ตัวไหนมีขีด เขียน 0** แล้วอ่านเลขฐานสองที่ได้เป็นเลขฐานสิบ



ไล่ให้ครบทั้ง 8 แถว จะได้ตารางนี้ ซึ่งควรเขียนเองได้ภายในหนึ่งนาทีในห้องสอบ

แถว	A B C	มินเทอม m_i	แมกซ์เทอม M_i
0	0 0 0	$\overline{A}\,\overline{B}\,\overline{C}$	$A+B+C$
1	0 0 1	$\overline{A}\,\overline{B}C$	$A+B+\overline{C}$
2	0 1 0	$\overline{A}B\overline{C}$	$A+\overline{B}+C$
3	0 1 1	$\overline{A}BC$	$A+\overline{B}+\overline{C}$
4	1 0 0	$A\overline{B}\,\overline{C}$	$\overline{A}+B+C$
5	1 0 1	$A\overline{B}C$	$\overline{A}+B+\overline{C}$
6	1 1 0	$AB\overline{C}$	$\overline{A}+\overline{B}+C$
7	1 1 1	ABC	$\overline{A}+\overline{B}+\overline{C}$

แมกซ์เทอม — กระจกบานตรงข้าม

มินเทอมคือนิพจน์ที่เป็น **1 แถวเดียว** ถ้าเราอยากได้นิพจน์ที่เป็น **0 แถวเดียว** ล่ะ?

เอามินเทอมมาตีขีดทั้งก้อนก็ได้เลย เช่น $\overline{m_5} = \overline{A\overline{B}C}$ แล้วใช้ De Morgan ถอดขีดออก จะได้ $\overline{A}+B+\overline{C}$ — นี่คือ **แมกซ์เทอม (maxterm)** M_5

แมกซ์เทอม

maxterm M_i คือผลบวก (sum term) ที่ตัวแปรทุกตัวโผล่ครบตัวละหนึ่งครั้ง และมันเป็น **0 ที่แถว i แถวเดียว** แถวอื่นเป็น 1 หมด

กฎการเขียนจึง **กลับด้านกับมินเทอม**: ตัวไหนในแถวเป็น **0 เขียนแบบไม่มีขีด** · ตัวไหนเป็น **1 เขียนแบบมีขีด** · แล้วเชื่อมทุกตัวด้วย $+$

เหตุผลตรงไปตรงมา: ผลบวกจะเป็น 0 ก็ต่อเมื่อทุกตัวที่บวกกันเป็น 0 พร้อมกัน ที่แถว 101 เราจึงต้องการ $\overline{A}=0, B=0, \overline{C}=0$ ซึ่งก็คือ $A=1, B=0, C=1$ พอดี

ทำผิดให้ดู

ที่ผิดกันมากคือลอกตัวอักษรจากมินเทอมมาแล้วเปลี่ยนแค่เครื่องหมายคูณเป็นบวก เช่น เขียน $M_5 = A + \overline{B} + C$

ลองแทนแถว 5 คือ $A=1, B=0, C=1$ ลงไป: $1+1+1=1$ — **ได้ 1 ไม่ใช่ 0** แปลว่านิพจน์นี้ไม่ใช่ M_5

ที่ถูกคือ $M_5 = \overline{A} + B + \overline{C}$ แทนแถว 5 ได้ $0+0+0=0$ ถูกต้อง — **ต้องกลับขีดทุกตัวด้วย ไม่ใช่เปลี่ยนแค่เครื่องหมาย**

สัญกรณ์ Σm และ ΠM — เขียนทั้งตารางด้วยตัวเลขไม่กี่ตัว

พอทุกแถวมีชื่อแล้ว เราก็เขียนฟังก์ชันทั้งตัวด้วยการ **ไล่ชื่อแถวที่เอาต์พุตเป็น 1** ได้ทันที

สองสัญกรณ์

Σm **(ซิกมา-เอ็ม)** = ผลบวกของมินเทอม ให้ไล่ **แถวที่เอาต์พุตเป็น 1** — ได้นิพจน์รูป SOP

ΠM **(พาย-เอ็ม)** = ผลคูณของแมกซ์เทอม ให้ไล่ **แถวที่เอาต์พุตเป็น 0** — ได้นิพจน์รูป POS

ตัวหนึ่งไล่แถวหนึ่ง อีกตัวไล่แถวศูนย์ — รวมกันจึงครบทุกแถวพอดี ถ้าตารางมี 8 แถว แล้ว Σm มี 5 ตัว ΠM ต้องมี 3 ตัวเสมอ

ทำให้ดู — แปลง $X(A,B)=\Sigma m(0,2)$ กลับเป็นนิพจน์

โจทย์มีสองตัวแปร แต่ละมินเทอมจึงมี 2 literal

m_0: แถว 00 → ทั้งคู่เป็น 0 → ใส่ขีดทั้งคู่ → $\overline{A}\,\overline{B}$

m_2: แถว 10 → A เป็น 1 ไม่ใส่ขีด, B เป็น 0 ใส่ขีด → $A\overline{B}$

ตอบ $X = \overline{A}\,\overline{B} + A\overline{B}$

ลดรูปต่อได้อีก: ดึง $\overline{B}$ ออกมา ได้ $\overline{B}(\overline{A} + A) = \overline{B} \cdot 1 = \overline{B}$ — ตรวจแล้ว $\overline{B}$ เป็น 1 ที่แถว 0 และ 2 พอดี

ลองเอง $D(A_2, A_1, A_0) = \Sigma m(3, 5, 7)$

เฉลย

แถว 3 = 011 → $\overline{A_2}A_1A_0$ · แถว 5 = 101 → $A_2\overline{A_1}A_0$ · แถว 7 = 111 → $A_2A_1A_0$

$D = \overline{A_2}A_1A_0 + A_2\overline{A_1}A_0 + A_2A_1A_0$ — ตรวจแล้วเป็น 1 ที่แถว 3, 5, 7 พอดีสามแถว

ให้สังเกตว่าชื่อตัวแปรเปลี่ยนจาก A, B, C เป็น A_2, A_1, A_0 ไม่มีผลอะไรเลย ตัวซ้ายสุดคือบิตที่มีน้ำหนักมากสุดเสมอ

7 ตาราง → SOP และ POS และเดินกลับ

เวลาส่งข้อมูลผ่านสาย บิตอาจเพี้ยนได้ วิธีตรวจจับแบบถูกที่สุดคือเติมบิตพิเศษท้ายข้อมูลหนึ่งบิต เรียกว่า **บิตพาริตี (parity bit)** โดยเลือกค่าของมันให้ **จำนวนเลข 1 ทั้งหมดรวมกันเป็นเลขคู่**

สมมติข้อมูลมีสามบิต ชื่อ A, B, C เราต้องสร้างวงจรที่คำนวณบิตพาริตี X ให้อัตโนมัติ

ลองเติมในใจก่อน — ข้อมูล 0 1 1 มีเลข 1 อยู่สองตัว เป็นเลขคู่อยู่แล้ว ต้องเติม X เป็นอะไร? แล้วข้อมูล 1 0 0 ที่มีเลข 1 ตัวเดียวล่ะ?

0 1 1 มี 1 อยู่ 2 ตัว (คู่แล้ว) → เติม $X = 0$ · 1 0 0 มี 1 อยู่ 1 ตัว (คี่) → ต้องเติม $X = 1$ ให้ครบเป็นสองตัว — สรุปสั้น ๆ: $X = 1$ **เมื่อข้อมูลมีเลข 1 เป็นจำนวนคี่**

ไล่ให้ครบทุกแถว จะได้ตารางนี้ นี่คือ "โจทย์" ของหัวข้อนี้ — เรามีตาราง แต่เรายังไม่มีนิพจน์และยังต่อวงจรไม่ได้

ทางที่หนึ่ง: อ่านตารางเป็น SOP (ไล่แถวที่ได้ 1)

หลักคิดมีบรรทัดเดียว: **แต่ละแถวที่เอาต์พุตเป็น 1 เขียนเป็นมินเทอมหนึ่งพจน์ แล้วเอาทุกพจน์มาบวกกัน**

เหตุผลที่ใช้ได้ เพราะมินเทอมแต่ละตัวเป็น 1 ได้แถวเดียว เมื่อเอามาบวก (OR) กัน ผลรวมจึงเป็น 1 เฉพาะแถวที่เราเลือกมา และเป็น 0 ที่แถวอื่นทั้งหมดโดยอัตโนมัติ

แถว	A	B	C	X	มินเทอมของแถวนี้
0	0	0	0	0	—
1	0	0	1	1	$\overline{A}\,\overline{B}C$
2	0	1	0	1	$\overline{A}B\overline{C}$
3	0	1	1	0	—
4	1	0	0	1	$A\overline{B}\,\overline{C}$
5	1	0	1	0	—
6	1	1	0	0	—
7	1	1	1	1	ABC

แถวที่ระบายเขียวคือแถวที่ $X = 1$ มีสี่แถว คือแถว **1, 2, 4, 7**

เขียนสัญกรณ์สั้นได้ว่า

$$X(A, B, C) = \Sigma m(1, 2, 4, 7)$$

กางออกเป็นนิพจน์ SOP ได้

$$X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$

สี่พจน์ ตรงกับสี่แถวเขียวพอดี — ตรวจด้วยโปรแกรมแล้วให้คอลัมน์เดิมครบทั้ง 8 แถว

เตือนเรื่องชื่อ

SOP ย่อจาก sum of products แปลตรงตัวว่า "ผลบวกของผลคูณ" — คูณก่อนแล้วค่อยบวก หน้าตาคือ $\square\square + \square\square + \ldots$

ถ้าทุกพจน์เป็นมินเทอมครบทุกตัวแปร เรียกว่า **standard SOP** ซึ่งเป็นรูปที่อ่านจากตารางได้เสมอ

ทางที่สอง: อ่านตารางเป็น POS (ไล่แถวที่ได้ 0)

ทางนี้เดินสวนทางกัน: **แต่ละแถวที่เอาต์พุตเป็น 0 เขียนเป็นแมกซ์เทอมหนึ่งวงเล็บ แล้วเอาทุกวงเล็บมาคูณกัน**

เหตุผลก็กลับด้านกันพอดี แมกซ์เทอมแต่ละตัวเป็น 0 ได้แถวเดียว เมื่อเอามาคูณ (AND) กัน ผลคูณจึงเป็น 0 เฉพาะแถวที่เราเลือกมา และเป็น 1 ที่แถวอื่นทั้งหมด

แถว	A	B	C	X	แมกซ์เทอมของแถวนี้
0	0	0	0	**0**	$(A+B+C)$
1	0	0	1	**1**	—
2	0	1	0	**1**	—
3	0	1	1	**0**	$(A+\overline{B}+\overline{C})$
4	1	0	0	**1**	—
5	1	0	1	**0**	$(\overline{A}+B+\overline{C})$
6	1	1	0	**0**	$(\overline{A}+\overline{B}+C)$
7	1	1	1	**1**	—

แถวที่ระบายชมพูคือแถวที่ $X=0$ มีสี่แถว คือแถว **0, 3, 5, 6**

$$X(A,B,C)=\Pi M(0,3,5,6)$$

$$X=(A+B+C)(A+\overline{B}+\overline{C})$$
$$(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)$$

ให้สังเกตว่าตัวอักษร **กลับขีดหมด** เทียบกับตอนเขียนมินเทอม เช่นแถว 3 คือ 0 1 1 มินเทอมจะเป็น $\overline{A}BC$ แต่แมกซ์เทอมเป็น $(A+\overline{B}+\overline{C})$

ทำไม POS ถึงต้องกลับขีด — ที่มาสามบรรทัด

อย่าท่องว่า "POS ต้องกลับขีด" เพราะมันมีที่มา และที่มานั้นคือสิ่งที่อาจารย์ให้ทำในสไลด์: **สร้างคอลัมน์ $\overline{X}$ ขึ้นมาก่อน**

$\overline{X}=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}BC+A\overline{B}C+AB\overline{C}$ เขียน SOP ของ $\overline{X}$ โดยไล่แถวที่ X เป็น 0 (แถว 0,3,5,6)

$X=\overline{\overline{X}}=\overline{\overline{A}\,\overline{B}\,\overline{C}}\cdot\overline{\overline{A}BC}\cdot\overline{A\overline{B}C}\cdot\overline{AB\overline{C}}$ ตีขีดกลับ (involution) แล้วใช้ De Morgan: ขีดคลุมผลบวก → กลายเป็นผลคูณ

$X=(A+B+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)$ De Morgan อีกชั้นในแต่ละวงเล็บ ขีดของตัวอักษรจึงกลับหมด

ขีดที่กลับ ไม่ได้มาจากกฎท่องจำ แต่มาจาก De Morgan ที่เราพิสูจน์ไปแล้วในหัวข้อ 3

ตรวจว่าสองนิพจน์ให้คอลัมน์เดียวกันจริง

SOP หน้าตาไม่เหมือน POS เลยสักนิด ต้องพิสูจน์ ไม่ใช่เชื่อ

A	B	C	SOP	$(A+B+C)$	$(A+\overline{B}+\overline{C})$	$(\overline{A}+B+\overline{C})$	$(\overline{A}+\overline{B}+C)$	POS (คูณกัน)
0	0	0	0	0	1	1	1	0
0	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	0	1	0	1	1	0
1	0	0	1	1	1	1	1	1
1	0	1	0	1	1	0	1	0
1	1	0	0	1	1	1	0	0
1	1	1	1	1	1	1	1	1

คอลัมน์ SOP กับคอลัมน์ POS เท่ากันครบ 8 แถว · และให้สังเกตว่าในแต่ละแถวที่ผลเป็น 0 มีวงเล็บ **เพียงวงเดียว** ที่เป็น 0 — เพราะแมกซ์เทอมแต่ละตัวออกแบบมาให้ล้มแถวเดียวพอดี

ทำผิดให้ดู — สองท่าคลาสสิก

ท่าที่หนึ่ง ใช้แถวที่เป็น 1 ไปเขียน POS — ถ้าทำแบบนั้น แถวที่ควรเป็น 1 จะกลายเป็น 0 หมด ได้ฟังก์ชันกลับหัวพอดี

ท่าที่สอง ใช้แถวที่เป็น 0 ถูกแล้ว แต่ลอกตัวอักษรโดยไม่กลับขีด เช่นแถว 3 (011) เขียนเป็น $(\overline{A}+B+C)$ แทนที่จะเป็น $(A+\overline{B}+\overline{C})$

ตรวจสิบวินาที: **เอาค่าในแถวนั้นแทนลงวงเล็บที่เพิ่งเขียน ต้องได้ 0** แถว 3 คือ $A=0, B=1, C=1$ — ที่ถูก $0+0+0=0$ ✓ · ที่ผิด $1+1+1=1$ ✗

เดินกลับ: มีนิพจน์อยู่แล้ว ต้องการตาราง

ทิศนี้ง่ายกว่า แต่มีหลุมอยู่หนึ่งหลุม ลองดูโจทย์นี้ — $X=\overline{A}B\overline{C}+A\overline{B}C+AB$

ลองคิดในหัวก่อน — พจน์แรกกับพจน์ที่สองมีสามตัวอักษรครบ แต่พจน์ที่สามมีแค่ AB ไม่มี C เลย แล้วพจน์นี้จะทำให้แถวไหนเป็น 1 บ้าง?

AB ไม่ได้บอกอะไรเกี่ยวกับ C แปลว่า C จะเป็น 0 หรือ 1 ก็ได้ทั้งคู่ — พจน์นี้จึงทำให้ **สองแถว** เป็น 1 คือแถว 110 และ 111

กฎสามข้อที่อาจารย์ให้ไว้

① แต่ละ **มินเทอม** (พจน์ที่มีตัวแปรครบ) ทำให้ **หนึ่งแถว** เป็น 1

② พจน์ที่ **ไม่ครบตัวแปร** ทำให้ **มากกว่าหนึ่งแถว** เป็น 1 — ขาดไป k ตัวแปร จะครอบ 2^k แถว

③ แถวที่เหลือทั้งหมดเป็น 0

ทำให้ดู $X = \overline{A}B\overline{C} + A\overline{B}C + AB$ → ตาราง

$\overline{A}B\overline{C}$	ครบสามตัว → อ่านเป็นบิต: $\overline{A}$=0, B=1, $\overline{C}$=0 → **แถว 010 = แถว 2**
$A\overline{B}C$	ครบสามตัว → 1, 0, 1 → **แถว 101 = แถว 5**
AB	ขาด C ไปหนึ่งตัว → ครอบ $2^1 = 2$ แถว คือ $AB\overline{C}$ และ ABC → **แถว 110 และ 111 = แถว 6 และ 7**

เติม 1 ที่แถว 2, 5, 6, 7 ที่เหลือเป็น 0 หมด — เขียนสั้นได้ว่า $X = \Sigma m(2, 5, 6, 7)$

แถว	A	B	C	X	มาจากพจน์ไหน
0	0	0	0	0	—
1	0	0	1	0	—
2	0	1	0	1	$\overline{A}B\overline{C}$
3	0	1	1	0	—
4	1	0	0	0	—
5	1	0	1	1	$A\overline{B}C$
6	1	1	0	1	AB
7	1	1	1	1	AB

ตรวจกลับ ที่มาของการกาง AB คือ $AB = AB \cdot 1 = AB(C + \overline{C}) = ABC + AB\overline{C}$ — ใช้ส่วนเติมเต็มกับกฎการกระจาย ไม่ใช่การเดา

ทำผิดให้ดู

คนที่รีบจะเห็น AB แล้วกา 1 แค่แถวเดียว (แถว 6) เพราะเคยชินว่า "หนึ่งพจน์ = หนึ่งแถว"

ตารางที่ได้จะขาดแถว 7 ไป และถ้าเอาไปเขียน SOP กลับ จะได้นิพจน์ที่ **ไม่เท่ากับโจทย์เดิม** — ต่างกันหนึ่งแถวก็คือคนละฟังก์ชัน

กันพลาด: ก่อนกาตาราง ให้นับก่อนว่าพจน์นี้มีกี่ literal ถ้าน้อยกว่าจำนวนตัวแปร ให้หยุดแล้วกางพจน์ให้ครบก่อน

ลองเอง ตารางนี้ใช้ชื่อตัวแปร A_2, A_1, A_0

A_2	A_1	A_0	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

เขียนทั้งแบบ SOP และแบบ POS

เฉลย

SOP — แถวที่เป็น 1 คือ 1, 2, 4, 5, 7 → $X = \Sigma m(1, 2, 4, 5, 7)$

$X = \overline{A_2}\,\overline{A_1}A_0 + \overline{A_2}A_1\overline{A_0} + A_2\overline{A_1}\,\overline{A_0} + A_2\overline{A_1}A_0 + A_2A_1A_0$ (5 พจน์)

POS — แถวที่เป็น 0 คือ 0, 3, 6 → $X = \Pi M(0, 3, 6)$

$X = (A_2 + A_1 + A_0)(A_2 + \overline{A_1} + \overline{A_0})(\overline{A_2} + \overline{A_1} + A_0)$ (3 วงเล็บ)

5 + 3 = 8 ครบทุกแถวพอดี — ตรวจด้วยโปรแกรมแล้วทั้งสองนิพจน์ให้คอลัมน์เดียวกันครบ 8 แถว

ข้อสังเกตที่อาจารย์เขียนไว้ท้ายสไลด์: **ถ้า SOP ยาว POS จะสั้น และกลับกัน** เพราะจำนวนพจน์ของสองฝั่งรวมกันเท่ากับจำนวนแถวเสมอ

1 **โจทย์ให้ตาราง อยากได้ SOP**
วงแถวที่เอาต์พุตเป็น **1** → แต่ละแถวเขียนมินเทอม (บิต 1 ไม่ใส่ขีด บิต 0 ใส่ขีด) → เอามาบวกกัน

2 **โจทย์ให้ตาราง อยากได้ POS**
วงแถวที่เอาต์พุตเป็น **0** → แต่ละแถวเขียนแมกซ์เทอม (บิต 0 ไม่ใส่ขีด บิต 1 ใส่ขีด — **กลับกัน**) → เอามาคูณกัน

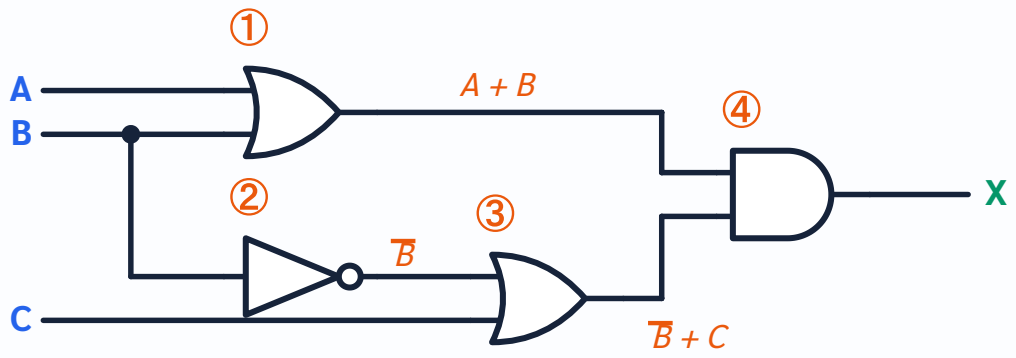
3 **โจทย์ให้นิพจน์ อยากได้ตาราง**
ดูทีละพจน์ → พจน์ที่ครบตัวแปรกา 1 หนึ่งแถว → พจน์ที่ขาด k ตัวแปรกา 1 จำนวน 2^k แถว → ที่เหลือเป็น 0

4 **ตรวจก่อนส่งทุกครั้ง**
นับจำนวนพจน์ SOP บวกจำนวนวงเล็บ POS **ต้องเท่ากับจำนวนแถวพอดี** ถ้าไม่เท่าแปลว่าตกแถวไปแล้ว **หยุดตรงนี้** อย่าเพิ่งลดรูป — โจทย์ข้อนี้ไม่ได้ขอให้ลดรูป

8 อ่านวงจร เขียนวงจร และเกตสารพัดประโยชน์

ตอนแลบ น้องอ่านวงจรด้วยการต่อสายแล้วดู LED แต่ข้อสอบให้รูปวงจรมาแล้วถามว่า "write the output expression" ต้องอ่านด้วยตา

วิธีอ่านมีกฎเดียว: **ไล่จากซ้ายไปขวา เขียนนิพจน์ของทุกจุดที่สายออกจากเกต แล้วเอาไปเป็นขาเข้าของเกตถัดไป** ห้ามข้ามเกต ห้ามเดาทั้งวงจร



ป้ายส้มคือนิพจน์ของแต่ละจุด เขียนไล่ไปตามลำดับ ①②③④

ลองอ่านเองก่อน — เกต ① เอา A กับ B มา OR กัน ได้ $A + B$ · เกต ② กลับค่า B ได้ $\overline{B}$ · แล้วเกต ③ กับ ④ ได้อะไร?

เกต ③ เอา $\overline{B}$ กับ C มา OR กัน ได้ $\overline{B} + C$ · เกต ④ เอาผลของ ① กับ ③ มา AND กัน ได้ $X = (A + B)(\overline{B} + C)$

ถ้าข้อสอบขอแค่ "output expression" จบตรงนี้ได้เลย แต่ถ้าขอให้ลดรูปด้วย ก็ใช้ของจากหัวข้อ 4 และ 5 ต่อได้ทันที

ทำให้ดู ลดรูป $(A + B)(\overline{B} + C)$

$(A + B)(\overline{B} + C)$	นิพจน์ที่อ่านได้จากวงจร
$= A\overline{B} + AC + B\overline{B} + BC$	กระจายแบบที่หนึ่ง — คูณทุกคู่
$= A\overline{B} + AC + BC$	ส่วนเติมเต็ม — $B\overline{B} = 0$ และ 0 บวกอะไรก็หายไป
$= A\overline{B} + BC$	consensus — คู่ $A\overline{B}$ กับ BC มี $\overline{B}$ กับ B เข้าคู่กัน คูณแล้วตัดทิ้งได้ AC ซึ่งมีอยู่แล้ว จึงลบได้ (ตอบ)

ตรวจกลับ ทั้งสองรูปเป็น 1 ที่แถว 3, 4, 5, 7 เท่ากันครบ 8 แถว และตรวจด้วยโปรแกรมแล้ว $A\overline{B} + BC$ สั้นที่สุดแล้ว (2 พจน์ 4 literal ไม่มีทางได้สั้นกว่านี้)

เรื่องจำนวนเกต ต้องนับให้ครบ: วงจรเดิมใช้ 4 ตัว (OR สองตัว, NOT หนึ่งตัว, AND หนึ่งตัว) ส่วน $A\overline{B} + BC$ ก็ใช้ 4 ตัวเหมือนกัน (NOT หนึ่งตัวสำหรับ $\overline{B}$, AND สองตัว, OR หนึ่งตัว) — **จำนวนเกตเท่ากัน** สิ่งที่ได้มาคือรูป SOP ซึ่งเอาไปแปลงเป็น NAND ล้วนได้ตรง ๆ ในหัวข้อถัดไป

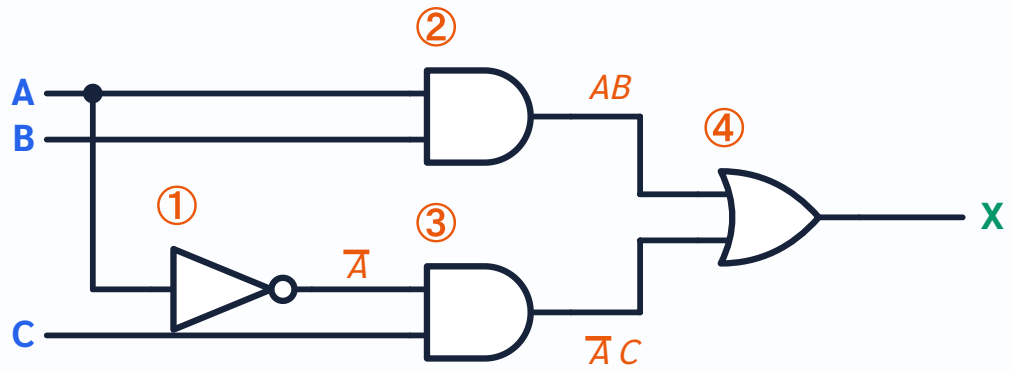
ทางกลับ: มีนิพจน์ ต้องวาดวงจร

ทางนี้อ่านนิพจน์จาก **ในออกนอก** คือทำสิ่งที่อยู่ในวงเล็บลึกสุดก่อน แล้วค่อยไล่ออกมา สำหรับรูป SOP ง่ายมาก เพราะโครงสร้างตายตัว

AND-OR logic — โครงสร้างของ SOP ทุกอัน

นิพจน์ SOP ทุกอันวาดได้ด้วยแบบเดียวกันเสมอ: **AND หนึ่งตัวต่อหนึ่งพจน์ แล้ว OR หนึ่งตัวรวบทุกพจน์** · ถ้าพจน์ไหนมีตัวแปรที่มีขีด ก็เติม inverter ที่ขาเข้าตัวนั้น

อาจารย์เรียกโครงสร้างนี้ว่า **AND-OR logic** และเพราะตารางทุกตารางเขียนเป็น SOP ได้ จึงแปลว่า **วงจรผสมทุกวงจรสร้างได้ด้วย AND กับ OR กับ NOT เท่านั้น**



$X = AB + \overline{A}C$ — ① inverter สร้าง $\overline{A}$ · ② AND ทำพจน์แรก · ③ AND ทำพจน์ที่สอง · ④ OR รวบสองพจน์ · ใช้เกต 4 ตัว

ตรวจกลับได้เหมือนเดิม: วงจรนี้ให้ $X = 1$ ที่แถว 1, 3, 6, 7 ซึ่งตรงกับที่คำนวณจากนิพจน์ $AB + \overline{A}C$ ทุกแถว

แถมอีกหนึ่งชื่อที่อาจโผล่ในข้อสอบ

ถ้าเอา inverter ไปแปะที่ **ขาออก** ของวงจร AND-OR จะได้สิ่งที่อาจารย์เรียกว่า **AND-OR-Invert (AOI)** ซึ่งใช้สร้างนิพจน์รูป POS ได้

วิธีคิด: สร้าง $\overline{X}$ ด้วย AND-OR ตามปกติก่อน แล้วค่อยส่งผ่าน NOT หนึ่งตัวเพื่อให้ได้ X — เป็นเหตุผลเดียวกับที่เราใช้หา POS ในหัวข้อ 7

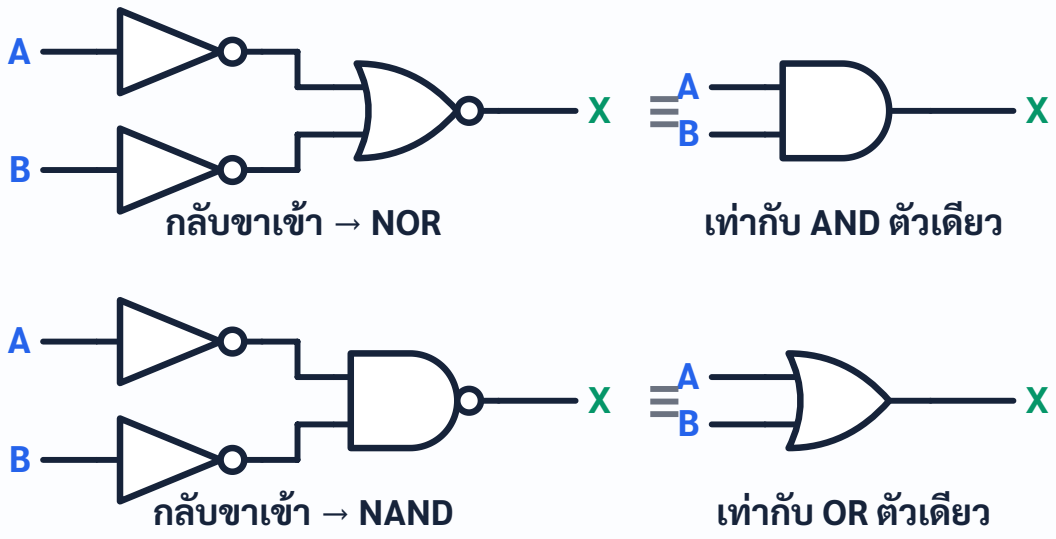
NOT gate conversions — เปลี่ยนเกตที่ไม่มี ให้เป็นเกตที่มี

ในแลบ ถ้าในลิ้นชักเหลือแต่ IC เบอร์ NOR กับ inverter แต่แบบวงจรต้องใช้ AND น้องจะทำยังไง?

คำตอบอยู่ในกฎที่เรียนไปแล้วทั้งหมด ไม่มีของใหม่เลย: เอา inverter ไปแปะที่ **ขาเข้าทุกขา** ของ NOR แล้วมันจะกลายเป็น AND

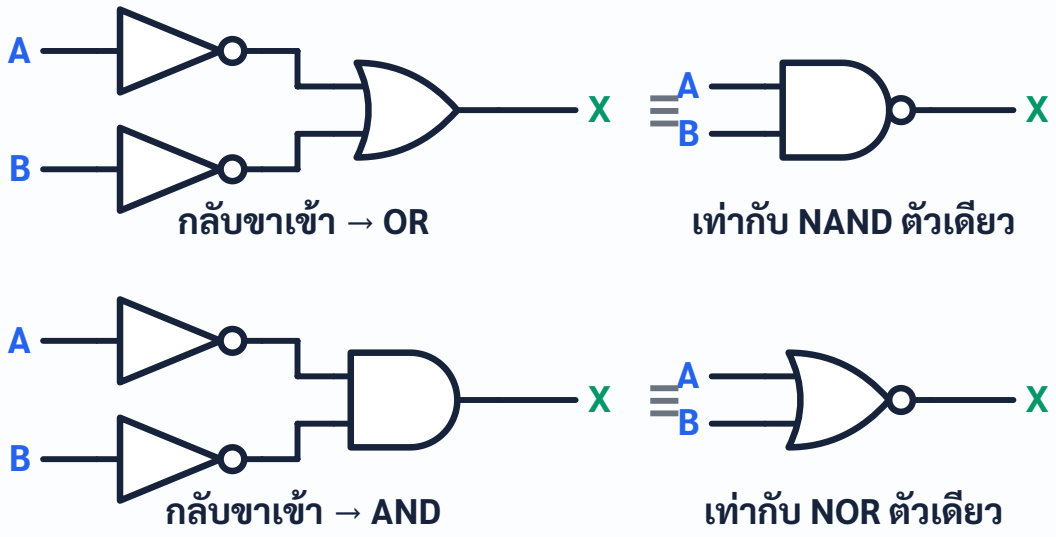
ลองคิดในหัวก่อน — ถ้ากลับค่า A กับ B ก่อน แล้วเอาเข้า NOR นิพจน์ที่ได้คือ $\overline{\overline{A} + \overline{B}}$ ลองใช้ De Morgan ถอดดูว่าได้อะไร

ขีดนอกสุดคลุมการบวก หักแล้วได้คูณ: $\overline{\overline{A}} \cdot \overline{\overline{B}}$ แล้วขีดสองชั้นหายไป เหลือ AB — **ได้ AND จริง ๆ**



แถวบน $\overline{\overline{A} + \overline{B}} = AB$ · แถวล่าง $\overline{\overline{A}\,\overline{B}} = A + B$ — ตรวจแล้วเท่ากันครบสี่แถวทั้งคู่

สองรูปข้างบนคือการ "สร้างเกตที่ไม่มี" อีกสองรูปข้างล่างคือมุมกลับ: เอาเกตธรรมดามาทำเป็นเกตที่มี N นำหน้า



แถวบน $\overline{A}+\overline{B}=\overline{AB}$ · แถวล่าง $\overline{A}\,\overline{B}=\overline{A+B}$ — สองบรรทัดนี้คือ De Morgan ที่วาดเป็นรูป

จำทั้งสี่รูปด้วยประโยคเดียว

กลับค่าขาเข้าทุกขา แล้วตัวกาต์จะสลับหน้าที่ — AND กลายเป็น NOR, OR กลายเป็น NAND, NOR กลายเป็น AND, NAND กลายเป็น OR

ซึ่งก็คือ dual symbol ในหัวข้อ 2 นั่นเอง เพียงแต่คราวนี้เราวาดบับเบิลเป็น inverter จริง ๆ แทนที่จะวาดเป็นวงกลมเล็กติดตัวกาต์

Universal gates — NAND ตัวเดียวสร้างได้ทุกเกต

ในโรงงานจริง การสั่ง IC เบอร์เดียวจำนวนมาก ถูกกว่าการสั่งห้าเบอร์อย่างละน้อย และบนชิปก็เช่นกัน — ถ้าใช้เกตชนิดเดียวทั้งชิป การผลิตจะง่ายและถูกกว่า

NAND ทำได้ครบทุกอย่าง จึงถูกเรียกว่า **universal gate** (เกตสารพัดประโยชน์) NOR ก็เป็นอีกตัวหนึ่งที่ทำได้เช่นกัน

ลองคิดในหัวก่อน — ถ้าเรามี NAND ตัวเดียว แล้วเอาขาเข้าทั้งสองขา **ต่อเข้าด้วยกัน** ให้รับสัญญาณ A เหมือนกันทั้งคู่ จะได้อะไรออกมา?

นิพจน์คือ $\overline{A \cdot A}$ และ $A \cdot A = A$ (ซ้ำตัวเอง) เหลือ $\overline{A}$ — **ได้ NOT ฟรี ๆ ด้วย NAND ตัวเดียว**

พอมี NOT แล้ว ที่เหลือก็ต่อยอดได้ทันที เพราะ NAND คือ AND ที่ตามด้วย NOT ถ้าเอา NOT ไปกลับผลลัพธ์อีกที ก็ได้ AND กลับคืนมา

NAND 1 ตัว → NOT



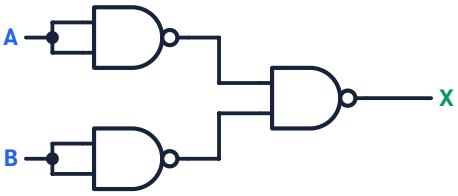
$X = \overline{AA} = \overline{A}$ — ตรวจแล้วได้ตาราง 1,0 ตรงกับ NOT

NAND 2 ตัว → AND



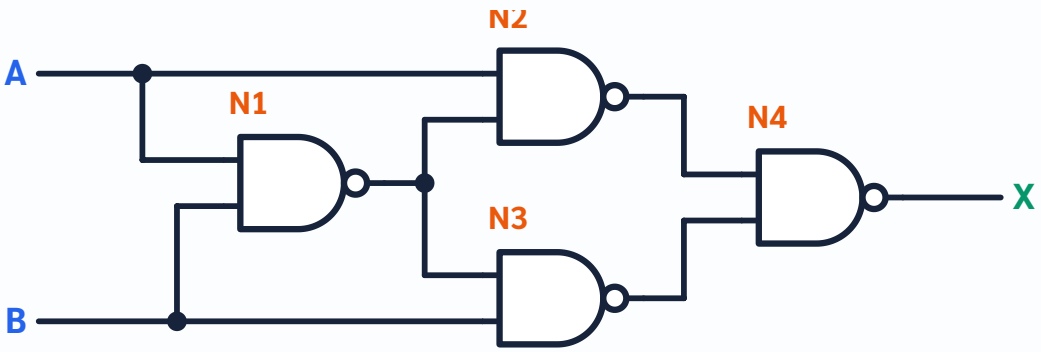
$X = \overline{\overline{AB}} = AB$ — ตัวที่สองทำหน้าที่เป็น NOT

NAND 3 ตัว → OR



สองตัวแรกทำเป็น NOT ได้ $\overline{A}, \overline{B}$ แล้วตัวที่สาม: $\overline{\overline{A}\,\overline{B}} = A + B$ (De Morgan)

NAND 4 ตัว → XOR



N1 รับ A, B · N2 รับ A กับผลของ N1 · N3 รับ B กับผลของ N1 · N4 รวบ N2 กับ N3

ทำให้ดู พิสูจน์ว่า NAND สี่ตัวได้ XOR จริง

N1	$\overline{AB}$ NAND ธรรมดา
N2	$\overline{A \cdot \overline{AB}} = \overline{A} + AB$ (De Morgan + involution) $= \overline{A} + B$ ดูดกลืน ③: $\overline{A} + AB = \overline{A} + B$
N3	$\overline{B \cdot \overline{AB}} = \overline{B} + AB = A + \overline{B}$ ทำแบบเดียวกัน
N4	$\overline{(\overline{A} + B)(A + \overline{B})}$ → กางในวงเล็บก่อน: $\overline{A}A + \overline{A}\,\overline{B} + BA + B\overline{B} = \overline{A}\,\overline{B} + AB$
	แล้วตีขีด: $\overline{\overline{A}\,\overline{B} + AB} = (A + B)(\overline{A} + \overline{B}) = A\overline{B} + \overline{A}B$ **= XOR**

ตรวจด้วยโปรแกรม จำลองวงจรสี่เกตนี้ครบทั้งสี่แถว ได้ 0, 1, 1, 0 ตรงกับตาราง XOR ทุกแถว และค่าของโหนดกลางทางทั้งสามก็ตรงกับที่คำนวณไว้ข้างบน

สรุปจำนวนเกตที่ต้องใช้ (จากวงจรที่ตรวจแล้วในบทนี้)

อยากได้	ใช้ NAND กี่ตัว	ใช้ NOR กี่ตัว
NOT	1	1
AND	2	3
OR	3	2
NAND	1	4
NOR	4	1
XOR	4	5
XNOR	5	4

ให้สังเกตความสมมาตร: ตัวเลขของ NOR คือตัวเลขของ NAND ที่สลับ AND กับ OR และสลับ XOR กับ XNOR — เพราะ NOR คือ "รูปคู่" ของ NAND นั่นเอง

วิธีต่อด้วย NOR ก็คิดแบบเดียวกัน: NOR ที่ขาเข้าต่อรวมกันได้ NOT · NOR แล้วตามด้วย NOT ได้ OR · กลับค่าขาเข้าทั้งสองก่อนเข้า NOR ได้ AND

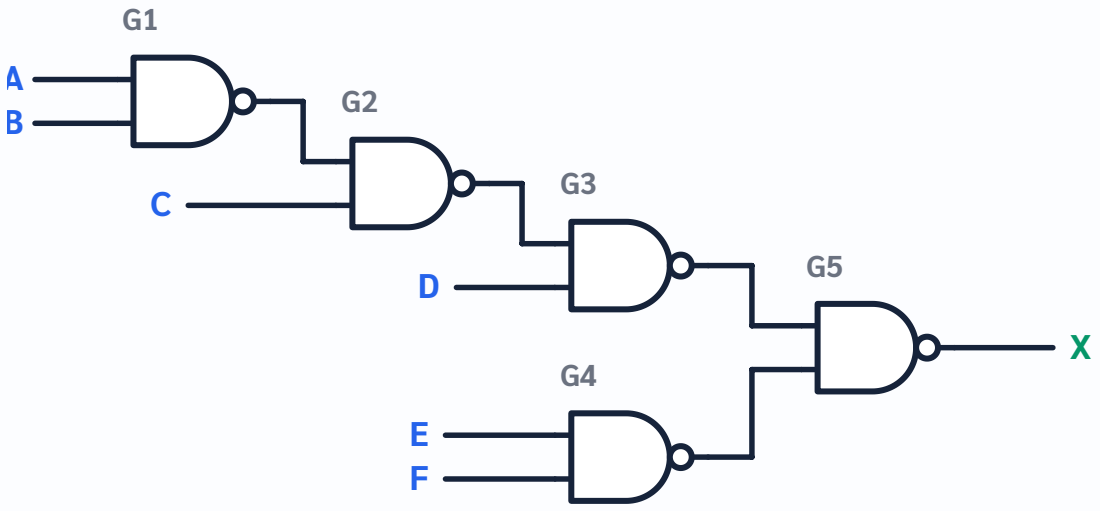
ทำผิดให้ดู

กับดักคือ "แปลง AND เป็น NAND เฉย ๆ" แล้วลืมว่าผลลัพธ์กลับหัวไปแล้ว — วงจรที่ได้จะให้คำตอบตรงข้ามทุกแถว

ถ้าโจทย์บอกว่า "convert to NAND only" แล้วน้องนับเกตได้ **เท่าเดิม** กับวงจรต้นฉบับ ให้สงสัยไว้ก่อนว่าลืมใส่ตัวกลับค่า เพราะการแปลงมักทำให้จำนวนเกตเพิ่ม ไม่ใช่เท่าเดิม

9 วาดใหม่ให้บับเบิลหักล้างกัน แล้ววงจรจะอ่านออกด้วยตา

ถึงตรงนี้น้องอ่านวงจรเป็นแล้ว แต่ลองดูวงจรข้างล่างนี้ มันเป็น NAND ห้าตัวต่อกันเป็นลูกโซ่ การอ่านมันตรง ๆ ต้องเขียนขีดซ้อนขีดหลายชั้นและใช้ De Morgan หลายรอบกว่าจะได้คำตอบ



รูปที่ ① — วงจร NAND ล้วน ห้าตัว ตามที่โจทย์ให้มา

ลองอ่านตรง ๆ ดูสักสองเกตก่อน — G1 ได้ $\overline{AB}$ · G2 รับผลของ G1 กับ C แล้ว NAND กัน ได้ $\overline{\overline{AB} \cdot C}$ · แล้ว G3 ล่ะ หน้าตาจะยาวแค่ไหน?

G3 ได้ $\overline{\overline{\overline{AB} \cdot C} \cdot D}$ — ขีดสามชั้นตั้งแต่เกตที่สาม และยังเหลืออีกสองเกต การอ่านแบบนี้ผิดง่ายมาก

วิธีที่อาจารย์สอนไม่ได้แก้ที่การคำนวณ แต่แก้ที่ **รูปที่เราวาด** โดยใช้ dual symbol จากหัวข้อ 2 มาเปลี่ยนรูปของบางเกต ให้บับเบิลมาชนกันแล้วหักล้างกัน

กฎการวาดใหม่ (อาจารย์เขียนไว้ตรง ๆ)

ให้วาดใหม่จนกระทั่ง **ทุกจุดเชื่อมระหว่างขาออกของเกตหนึ่ง กับขาเข้าของเกตถัดไป** เป็นหนึ่งในสองแบบนี้เท่านั้น

bubble-to-bubble — มีบับเบิลทั้งสองฝั่ง แล้วมันหักล้างกัน (กลับค่าสองครั้ง = ไม่กลับ)

nonbubble-to-nonbubble — ไม่มีบับเบิลทั้งสองฝั่ง ก็ไม่มีอะไรต้องหักล้าง

ห้ามเหลือแบบ "มีข้างเดียว" เพราะนั่นคือจุดที่การอ่านด้วยตาจะผิด

ลงมือวาดใหม่ทีละเกต

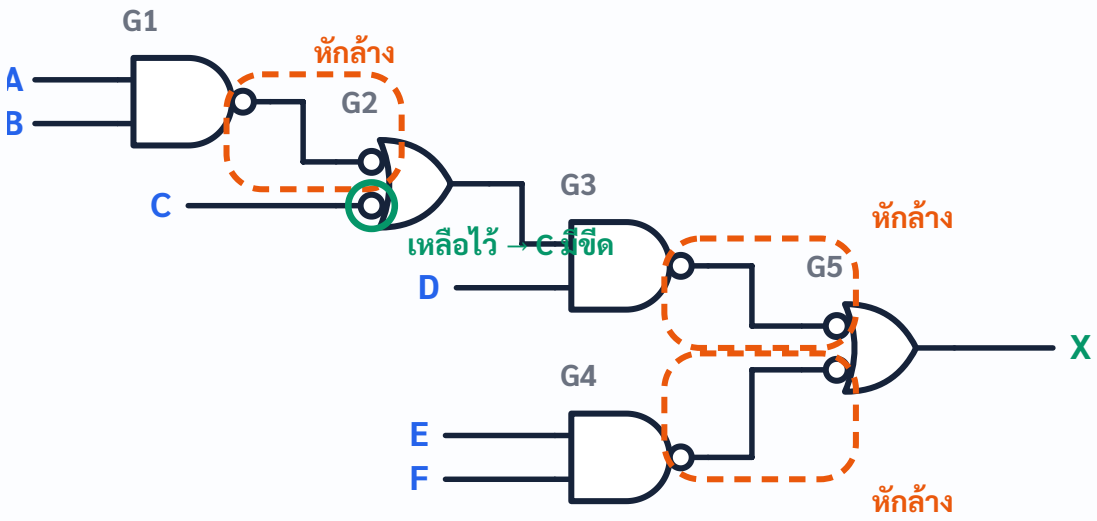
เริ่มจากเกตสุดท้ายแล้วย้อนกลับ เพราะขาออกของ G5 คือ X ที่เราอยากได้ตรง ๆ ไม่อยากให้มีบับเบิลคาอยู่

G5 เปลี่ยนเป็นรูป negative-OR (ตัว OR บับเบิลอยู่ขาเข้า) ขาออกจึงสะอาด · ทีนี้ขาเข้าทั้งสองของมันมีบับเบิลแล้ว ดังนั้น **G3 กับ G4 ต้องมีบับเบิลที่ขาออก** ซึ่งก็คือรูป NAND เดิม ไม่ต้อง

เปลี่ยน — ได้ bubble-to-bubble สองจุด

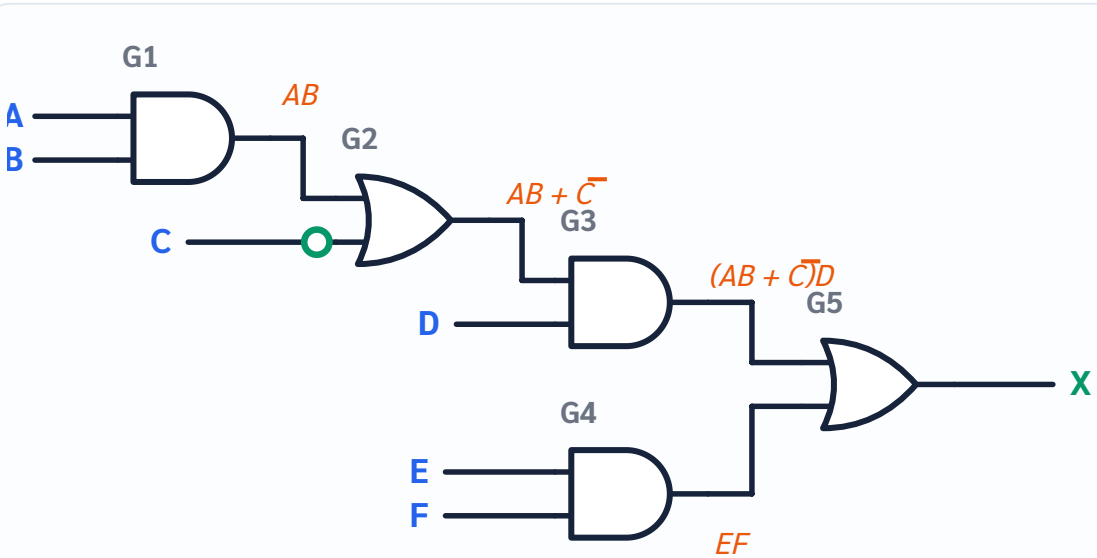
G3 เป็น NAND ขาเข้าไม่มีบับเบิล ดังนั้นสิ่งที่ป้อนเข้ามาต้องไม่มีบับเบิลด้วย · **G2** จึงต้องเปลี่ยนเป็นรูป negative-OR เพื่อให้ขาออกสะอาด — ได้ nonbubble-to-nonbubble

G2 รูปใหม่มีบับเบิลที่ขาเข้าทั้งสองขา · ขาบนรับจาก G1 ซึ่งเป็น NAND มีบับเบิลที่ขาออกอยู่แล้ว — ได้ bubble-to-bubble อีกจุด · แต่ **ขาล่างรับ C มาตรง ๆ จากภายนอก ไม่มีอะไรมาหักล้าง บับเบิลนั้นจึงอยู่ต่อ** และแปลว่าสัญญาณที่เข้า OR คือ $\overline{C}$



รูปที่ ② — กรอบส้มประคือบับเบิลสามคู่ที่ชนกันแล้วหักล้าง · วงเขียวคือบับเบิลที่ไม่มีคู่ จึงต้องเก็บไว้เป็น $\overline{C}$

ทีนี้ลบบับเบิลที่หักล้างกันทั้งสามคู่ออกไปได้เลย เพราะกลับค่าสองครั้งเท่ากับไม่กลับ (involution) เหลือรูปสะอาด ๆ ที่อ่านได้ด้วยตาทันที



รูปที่ ③ — AND-OR อ่านออกทันที: $X = (AB + \overline{C})D + EF$ · เหลือบับเบิลเขียวที่ขา C เพียงจุดเดียว

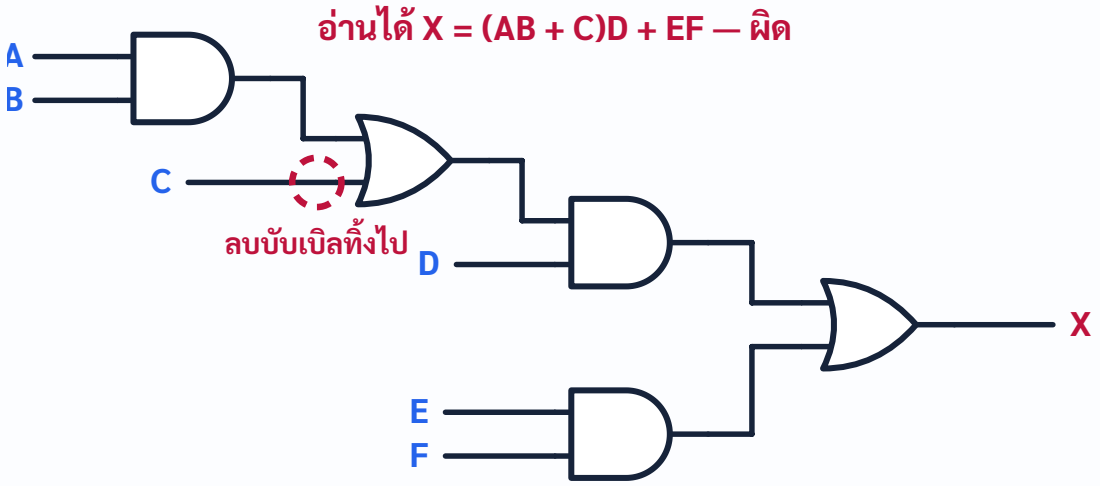
อ่านรูปที่ ③ จากซ้ายไปขวา: G1 คูณ A กับ B ได้ AB · G2 บวกกับ $\overline{C}$ ได้ $AB + \overline{C}$ · G3 คูณด้วย D · G4 คูณ E กับ F · G5 บวกสองก้อนสุดท้าย ได้

$$X = (AB + \overline{C})\,D + EF$$

สามรูปนี้คือ **วงจรเดียวกัน** ไม่ได้เปลี่ยนอะไรบนแผ่นวงจรเลย IC ที่ต่อยังเป็น NAND ห้าตัวเหมือนเดิม เราเปลี่ยนแค่รูปบนกระดาษ — ตรวจด้วยโปรแกรมแล้ว วงจร NAND ห้าตัวกับนิพจน์นี้ให้ค่าตรงกันครบทั้ง 64 แถว

ทำผิดให้ดู: วาดใหม่แบบลืมบับเบิลที่ไม่มีคู่

ความผิดที่เจอบ่อยที่สุดคือ เห็นว่า "NAND สลับกับ negative-OR ไปเรื่อย ๆ แล้วบับเบิลก็หักล้างกันหมด" แล้วก็ลบบับเบิลทุกอันทิ้ง รวมถึงอันที่ขา C ที่ **ไม่มีคู่ให้หักล้าง**



รูปที่วาดผิด ต่างจากรูปที่ ③ แค่จุดเดียว คือไม่มีบับเบิลที่ขา C

ต่างกันแค่วงกลมเล็ก ๆ วงเดียว แต่ฟังก์ชันเปลี่ยนไปเยอะมาก — ตรวจด้วยโปรแกรมครบทั้ง 64 แถว พบว่า **ให้คำตอบต่างกันถึง 18 แถว**

จับผิดด้วยหนึ่งแถว

ไม่ต้องตรวจทั้ง 64 แถว แค่เลือกแถวเดียวให้ถูกจุดก็พอ เลือกแถวที่ C มีผล คือ $A = 0, B = 0, C = 0, D = 1, E = 0, F = 0$

รูปที่ ③ (ถูก): $AB = 0 \cdot \overline{C} = 1 \cdot AB + \overline{C} = 1 \cdot$ คูณ $D = 1$ ได้ $1 \cdot EF = 0 \cdot X = 1 + 0 = \mathbf{1}$

รูปที่วาดผิด: $AB = 0 \cdot C = 0 \cdot AB + C = 0 \cdot$ คูณ $D = 1$ ได้ $0 \cdot EF = 0 \cdot X = 0 + 0 = \mathbf{0}$

ได้ 1 กับ 0 คนละค่า — ยืนยันว่ารูปที่วาดผิดเป็นคนละวงจร

วิธีวาดใหม่ที่ไม่พลาด

① เริ่มจากเกตขาออกสุดท้าย เลือกรูปที่ขาออก **ไม่มีบับเบิล**

② ถอยหลังทีละเกต: ดูว่าขาเข้าของเกตถัดไปมีบับเบิลไหม ถ้ามี เกตก่อนหน้าต้องมีบับเบิลที่ขาออกด้วย ถ้าไม่มี ก็ต้องไม่มีทั้งคู่

③ ลบเฉพาะบับเบิลที่ **ชนกันเป็นคู่** — บับเบิลที่ติดอยู่กับสัญญาณที่มาจากภายนอก (ไม่ได้มาจากเกตอื่น) **ห้ามลบ** ให้เขียนตัวแปรนั้นเป็นแบบมีขีดแทน

④ อ่านรูปสุดท้ายจากซ้ายไปขวาได้เลย แล้วแทนค่าหนึ่งแถวเทียบกับวงจรเดิมเพื่อยืนยัน

ฝั่ง NOR ก็ทำแบบเดียวกัน

ถ้าโจทย์ให้วงจร NOR ล้วน ก็ใช้คู่ NOR กับ negative-AND แทน ผลที่ได้จะอ่านเป็น **OR-AND** (ตรงข้ามกับฝั่ง NAND ที่อ่านเป็น AND-OR)

วงจร NOR ห้าตัวที่ต่อแบบเดียวกับรูปที่ ① จะอ่านได้ว่า $X = \big((A+B)\overline{C}+D\big)(E+F)$ — ตรวจด้วยโปรแกรมครบ 64 แถวแล้ว

10 เช็กลิสต์ก่อนเข้าห้องสอบ

บทนี้ยาว แต่สิ่งที่ต้องจำจริง ๆ มีไม่กี่อย่าง ที่เหลือสร้างขึ้นมาใหม่ได้จากของไม่กี่อย่างนั้น

ของที่ต้องจำจริง ๆ

① ตารางเจ็ดเกต — หรือจำแค่ประโยค: AND ครบทุกขา, OR ขาไหนก็ได้, XOR ต้องต่างกัน, ตัวที่มี N นำหน้าคือกลับค่าของตัวที่ไม่มี N

② De Morgan — **หักขีดแล้วเปลี่ยนเครื่องหมาย** $\overline{AB}=\overline{A}+\overline{B}$ และ $\overline{A+B}=\overline{A}\,\overline{B}$

③ ดูดกลืนสามหน้า — $A+AB=A\cdot A(A+B)=A\cdot A+\overline{A}B=A+B$

④ กระจายแบบที่สอง — $(A+B)(A+C)=A+BC$

⑤ Consensus — $AB+\overline{A}C+BC=AB+\overline{A}C$

⑥ มินเทอมใช้แถวที่เป็น 1 · แมกซ์เทอมใช้แถวที่เป็น 0 และ **กลับขีดทุกตัว**

นอกนั้น — รูปคู่ การแปลงเกต universal gate การวาดใหม่แบบ bubble-to-bubble — ทั้งหมดเป็น De Morgan ในรูปวาดทั้งสิ้น ถ้าจำข้อ ② ได้ ก็สร้างที่เหลือขึ้นมาเองได้ในห้องสอบ

เจอโจทย์แบบนี้ ให้เดินแบบนี้

① "Simplify this Boolean expression"

1. มีขีดคลุมเกินหนึ่งตัวแปรไหม → ถ้ามี ใช้ **De Morgan** ถอดจากขีดนอกสุดเข้าไปข้างในจนไม่เหลือ

2. กางวงเล็บทั้งหมดด้วย **กฎกระจาย** ให้เหลือแต่พจน์บวกกัน

3. ทิ้งพจน์ที่มี $X\overline{X}$ อยู่ข้างใน (เป็น 0) · ยุบพจน์ซ้ำ ($AB + AB = AB$)

4. ไล่ **ดูดกลืน**: มีพจน์สั้นตัวไหนที่ตัวอักษรของมันอยู่ครบในพจน์ยาวไหม ถ้ามี ลบพจน์ยาวทิ้ง

5. ยังยาวอยู่และมี X กับ $\overline{X}$ กระจายหลายพจน์ → ลอง **consensus**

6. **หยุด** เมื่อตัดตัวอักษรหรือพจน์ออกอีกไม่ได้ แล้วสุ่มสองแถวแทนค่าเทียบกับโจทย์เดิม

② "Convert this truth table to SOP / POS"

1. นับก่อนว่ามีกี่ตัวแปร → ต้องมี 2^n แถว ถ้าไม่ครบแปลว่าอ่านโจทย์ตก

SOP วงแถวที่ได้ **1** → แต่ละแถวเขียนมินเทอม (บิต 1 ไม่ใส่ขีด · บิต 0 ใส่ขีด) → บวกกันทั้งหมด

POS วงแถวที่ได้ **0** → แต่ละแถวเขียนแมกซ์เทอม (บิต 0 ไม่ใส่ขีด · บิต 1 ใส่ขีด — **กลับกัน**) → คูณกันทั้งหมด

2. ตรวจ: จำนวนพจน์ SOP + จำนวนวงเล็บ POS **ต้องเท่ากับจำนวนแถว**

3. ตรวจอีกชั้น: หยิบแมกซ์เทอมมาหนึ่งวงเล็บ แทนค่าแถวของมันลงไป **ต้องได้ 0**

ทางกลับ นิพจน์ → ตาราง: พจน์ครบตัวแปรกา 1 หนึ่งแถว · พจน์ที่ขาดไป k ตัว กา 1 จำนวน 2^k แถว

③ "Convert this circuit to NAND only"

1. อ่านวงจรเดิมให้เป็นนิพจน์ก่อน แล้วจัดให้อยู่รูป SOP (AND-OR)

2. แทนทีละเกตด้วยวงจร NAND: NOT ใช้ 1 ตัว (ต่อขาเข้ารวมกัน) · AND ใช้ 2 ตัว · OR ใช้ 3 ตัว

3. ทางลัดสำหรับ SOP: **เปลี่ยน AND ทุกตัวและ OR ตัวรวบให้เป็น NAND ได้เลย** เพราะบับเบิลที่เพิ่มมาชนกันเองแล้วหักล้างพอดี — เช่น $AB + \overline{A}C$ ต่อเป็น NAND(AB) กับ NAND$(\overline{A}C)$ แล้ว NAND สองอันนั้นเข้าด้วยกัน ได้ค่าเดิมครบทุกแถว

ข้อควรระวัง: ตัวแปรที่มีขีด (เช่น $\overline{A}$) ยังต้องมีตัวกลับค่าของตัวเองอยู่ดี ใช้ NAND ที่ต่อขาเข้ารวมกันหนึ่งตัว

4. ตรวจ: ถ้านับเกตแล้วได้เท่าวงจรเดิมเป๊ะ ให้สงสัยว่าลืมตัวกลับค่า

5. แทนค่าหนึ่งหรือสองแถวเทียบกับวงจรเดิมก่อนส่ง

④ "Write the output expression" (ให้รูปวงจรมา)

1. ถ้าวงจรเป็น NAND ล้วนหรือ NOR ล้วน → **วาดใหม่ก่อน** ให้บับเบิลชนกันเป็นคู่ อย่ารีบเขียนขีดซ้อนขีด

2. ลบเฉพาะบับเบิลที่ชนกันเป็นคู่ · บับเบิลที่ติดกับอินพุตจากภายนอก **ห้ามลบ** ให้เขียนตัวแปรนั้นแบบมีขีด

3. อ่านรูปใหม่จากซ้ายไปขวา เขียนนิพจน์ที่ขาออกของทุกเกต ห้ามข้าม

4. โจทย์ขอแค่ expression ก็ **หยุด** ตรงนั้น ไม่ต้องลดรูปถ้าไม่ได้สั่ง

⑤ "Opposing logic" กับ "Dual symbol" ต่างกันตรงไหน

opposing เกต **คนละตัว** ที่เอาต์พุตกลับค่ากันทุกแถว — AND/NAND · OR/NOR · XOR/XNOR

dual เกต **ตัวเดียวกัน** วาดสองรูป — NAND/negative-OR · NOR/negative-AND

วาด ย้ายบับเบิล **และ** สลับตัวกาต์ ต้องทำทั้งสองอย่าง ทำอย่างเดียวได้เกตผิดตัวทันที

นิสัยสุดท้ายที่ควรติดตัวไปห้องสอบ

ทุกครั้งที่เขียนเครื่องหมายเท่ากับบนกระดาษ แปลว่าน้องกำลังอ้างว่า "สองข้างนี้ให้ค่าเท่ากันทุกแถว"

ถ้าไม่แน่ใจแม้แต่นิดเดียว ให้หยุดแล้วสุ่มค่ามาหนึ่งชุด แทนลงทั้งสองข้าง — ใช้เวลาสิบวินาที แต่กันคะแนนหายทั้งข้อ

บทที่ 4 · K-Map

แผนที่คาร์นอห์ — วิธีย่อสมการลอจิกด้วยการมองภาพ แทนการจัดพีชคณิต · อ่านบทนี้จบแล้วต้องทำข้อสอบ K-map ได้เองทุกแบบ

บทนี้พาไปถึงไหน

บทนี้เริ่มจากศูนย์ครับ — ถือว่าน้อง**ไม่เคยเห็น K-map มาก่อนเลย** ขอแค่น้องอ่านตารางค่าความจริงเป็น และรู้ว่า A กับ $\overline{A}$ ต่างกันตรงไหน

สิ่งที่อาจารย์บอกว่าออกสอบมีห้าเรื่อง: แปลงตารางค่าความจริงหรือนิพจน์บูลีนไปเป็น K-map และแปลงกลับ · กฎการวงกลุ่ม · หา minimum SOP · หา minimum POS · และเงื่อนไข don't-care ที่เขียนด้วยตัว X

ทั้งห้าเรื่องอยู่ในบทนี้หมด และทุกตัวอย่างจะทำให้ดู**ทุกบรรทัด** ตั้งแต่ตารางค่าความจริงจนถึงคำตอบสุดท้าย ไม่มีข้ามขั้นครับ

1 ทำไมโลกนี้ต้องมี K-map

เรื่องจริงก่อน: กรรมการสามคนยกมือโหวต

ที่ประชุมมีกรรมการสามคนครับ เรียกว่าคนที่ 1 คนที่ 2 คนที่ 3 แต่ละคนมีสองทางเลือกเท่านั้นคือยกมือหรือไม่ยกมือ

กติกาของที่ประชุมคือ **ถ้ามีคนยกมืออย่างน้อยสองคน ญัตติผ่าน** ไฟเขียวหน้าห้องจะติด ถ้ายกมือคนเดียวหรือไม่มีใครยกเลย ไฟไม่ติด

ทีนี้สมมติว่าน้องต้องต่อวงจรไฟเขียวดวงนี้จริง ๆ ครับ วงจรไม่รู้จักคำว่า "อย่างน้อยสองคน" มันรู้จักแค่ไฟเข้า–ไฟออก เพราะฉะนั้นงานแรกคือเขียนให้ครบว่า**ทุกกรณีที่เป็นไปได้** ไฟต้องติดหรือไม่ติด

ตั้งชื่อตัวแปรก่อน: A คือคนที่ 1 · B คือคนที่ 2 · C คือคนที่ 3 โดย 1 แปลว่ายกมือ 0 แปลว่าไม่ยก และ X คือไฟเขียว 1 แปลว่าติด

สามคน แต่ละคนมีสองทาง จึงมีทั้งหมด $2^3 = 8$ กรณี ตารางข้างล่างนี้คือทั้งแปดกรณีครับ คอลัมน์ขวาสุดคือเลขแถว ซึ่งเดี๋ยวจะได้ใช้ตลอดทั้งบท

A	B	C	X	แถวที่
0	0	0	0	0
0	0	1	0	1
0	1	0	0	2
0	1	1	**1**	3
1	0	0	0	4
1	0	1	**1**	5
1	1	0	**1**	6
1	1	1	**1**	7

ลองก่อน ไม่ต้องเขียน

มองตารางข้างบนแล้วตอบในใจสองข้อครับ

หนึ่ง — มีกี่แถวที่ไฟติด

สอง — ถ้าน้องต้องอธิบายให้เพื่อนฟังโดย**ห้ามพูดคำว่า “อย่างน้อยสองคน”** น้องจะพูดว่าอะไร

เฉลย — ไฟติด 4 แถวครับ คือแถวที่ 3, 5, 6, 7 ส่วนข้อสองคนส่วนใหญ่จะตอบประมาณว่า “คนที่ 1 กับ 2 ยกมือ หรือ คนที่ 1 กับ 3 ยกมือ หรือ คนที่ 2 กับ 3 ยกมือ” — ที่น้องเพิ่งพูดไปนั่นแหละคือคำตอบสุดท้ายของทั้งหัวข้อนี้ครับ เหลือแค่เขียนให้เป็นสัญลักษณ์

เขียนตารางให้เป็นสมการ — ตรง ๆ ก่อน

วิธีแปลงตารางเป็นสมการมีขั้นเดียวครับ: **หยิบเฉพาะแถวที่ผลลัพธ์เป็น 1 มาทีละแถว** แถวที่เป็น 0 ไม่ต้องสนใจเลย

ในแต่ละแถวที่หยิบมา ให้เขียนตัวแปรเรียงกันทุกตัว ตัวไหนในแถวนั้นเป็น 1 เขียนตัวเปล่า ตัวไหนเป็น 0 เขียนแบบมีขีดบนหัว เช่นแถวที่ 3 คือ $A = 0,\ B = 1,\ C = 1$ จึงเขียนเป็น $\overline{A}BC$

ของที่เพิ่งเขียนนั้นเรียกว่า `product term` หรือ **พจน์ผลคูณ** ครับ และเพราะมันมาจากแถวเดียวของตาราง มันจึงเป็น 1 ได้แค่กรณีเดียวเท่านั้น ชื่อเฉพาะของมันคือ `minterm` — จำคำนี้ไว้ เดี๋ยวเจออีกเยอะ

ทำครบทั้งสี่แถวแล้วเอามาบวกกัน (เครื่องหมายบวกในบูลีนแปลว่า OR) จะได้สมการแรกของเรา

แปลงทีละแถวที่ได้ 1

แถว 3 : A=0 B=1 C=1 → $\overline{A}BC$

แถว 5 : A=1 B=0 C=1 → $A\overline{B}C$

แถว 6 : A=1 B=1 C=0 → $AB\overline{C}$

แถว 7 : A=1 B=1 C=1 → ABC

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

สมการดิบจากตาราง — ถูกต้อง 100% แต่ยาว 12 ตัวอักษร

สมการนี้**ถูกแล้วครับ** ถ้าเอาไปต่อวงจรจริงก็ทำงานได้ แต่มันแพง — ต้องใช้ AND 3 ขาเข้าสี่ตัว แล้วค่อยเอา OR 4 ขาเข้ามารวบอีกตัวหนึ่ง

คำว่า “ย่อสมการ” ทั้งบทนี้หมายถึงการหาสมการที่**ให้ตารางค่าความจริงเดิมเป๊ะ ๆ** แต่ใช้ตัวอักษรน้อยกว่า ไม่ใช่การเปลี่ยนคำตอบ

ย่อด้วยพีชคณิต — ทำได้ แต่ต้องรู้ทางลัดก่อน

เครื่องมือที่ใช้ย่อมีอันเดียวครับ คือกฎ $Y + \overline{Y} = 1$ แปลเป็นภาษาคนว่า “ตัวแปรตัวหนึ่ง ถ้ามันทั้งเป็นและไม่เป็นก็ครบทุกกรณีแล้ว ตัดทิ้งได้”

กฎนี้ใช้ได้เมื่อ**สองพจน์ต่างกันแค่ตัวอักษรเดียว** เท่านั้น เช่น $AB\overline{C} + ABC$ ต่างกันแค่ตัว C จึงดึง AB ออกมาเป็น $AB(\overline{C} + C) = AB \cdot 1 = AB$

ปัญหาคือในสมการของเรา ABC มีตัวเดียว แต่มีถึงสามพจน์ที่อยากจับคู่กับมัน ทางแก้คือใช้กฎ $Y + Y = Y$ ก๊อป ABC เพิ่มอีกสองชุด — เพิ่มแล้วค่าไม่เปลี่ยน

ย่อด้วยพีชคณิตทีละบรรทัด

$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ ← สมการดิบ

$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + \mathbf{ABC + ABC + ABC}$ ← ก๊อป ABC เพิ่ม 2 ชุด (Y+Y=Y)

$X = (AB\overline{C} + ABC) + (A\overline{B}C + ABC) + (\overline{A}BC + ABC)$ ← จัดคู่ใหม่

$X = AB(\overline{C}+C) + AC(\overline{B}+B) + BC(\overline{A}+A)$ ← ดึงตัวร่วมของแต่ละคู่

$X = AB\cdot 1 + AC\cdot 1 + BC\cdot 1$ ← ใช้ $Y+\overline{Y}=1$

$X = AB + AC + BC$ ← คำตอบ

เช็กคำตอบกลับเข้าโจทย์เดิมครับ: AB แปลว่าคนที่ 1 กับ 2 ยกมือ · AC คือคนที่ 1 กับ 3 · BC คือคนที่ 2 กับ 3 — ตรงกับที่น้องพูดเองในกล่อง “ลองก่อน” เป๊ะ ๆ

จาก 12 ตัวอักษรเหลือ 6 ตัวอักษร จาก AND 3 ขาสี่ตัวเหลือ AND 2 ขาสามตัว วงจรถูกลงครึ่งหนึ่ง

ตรงนี้แหละที่คนพลาด

ขั้นที่ยากที่สุดคือขั้น “ก๊อป ABC เพิ่มสองชุด” ครับ มันเป็นขั้นที่**ต้องเดาล่วงหน้าว่าจะได้ใช้** ถ้าไม่ได้ก๊อป น้องจะจับคู่ได้แค่คู่เดียว

ลองเดินแบบไม่ก๊อปให้ดู: จับ $AB\overline{C} + ABC = AB$ ไปแล้ว ABC ก็หมดตัว เหลือ $\overline{A}BC$ กับ $A\overline{B}C$ ลอยอยู่ จับคู่กับใครไม่ได้เลย ได้คำตอบ $X = AB + \overline{A}BC + A\overline{B}C$ ซึ่ง**ถูก** แต่ยาว 8 ตัวอักษร ไม่ใช่ 6 — ข้อสอบที่ถามหา *minimum* จะตัดคะแนนตรงนี้

ของเดียวกัน แต่วางบนกระดาษตาราง

ทีนี้ลองทำโจทย์เดิมใหม่ครับ แต่แทนที่จะเขียนสี่พจน์เรียงเป็นบรรทัดเดียว เราจะเอาผลลัพธ์ทั้งแปดแถวไปวางลงในตารางแปดช่อง

ตารางนี้มี 4 แถว 2 คอลัมน์ แถวคุมค่าของ AB คอลัมน์คุมค่าของ C หนึ่งช่องเก็บผลลัพธ์ของหนึ่งแถวในตารางค่าความจริง — เลขสีส้มมุมขวาบนบอกว่าช่องนั้นคือแถวที่เท่าไร

AB \ C	0	1
00	0 (0)	0 (1)
01	0 (2)	1 (3)
11	1 (6)	1 (7)
10	0 (4)	1 (5)

ผลลัพธ์ทั้ง 8 แถวจากตารางหน้าที่แล้ว ย้ายมาอยู่ในตาราง 8 ช่อง — ตัวเลขชุดเดียวกันเป๊ะ

ยังไม่ต้องสนใจว่าทำไมหัวแถวถึงเรียง 00 01 11 10 ครับ เดี๋ยวหัวข้อหน้าอธิบาย ตอนนี้ขอแค่เชื่อว่าเลขในช่องคือเลขชุดเดิม

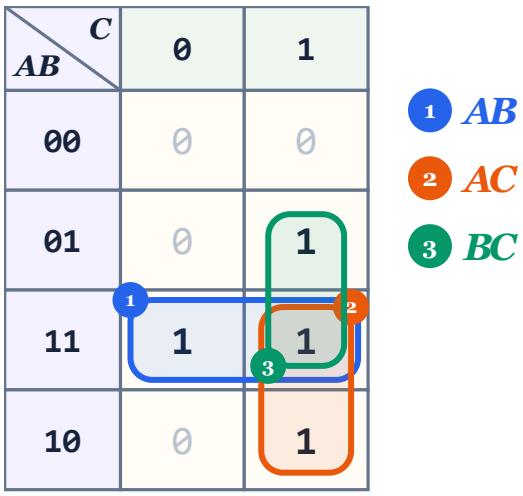
ลองก่อน ไม่ต้องเขียน

มองรูปข้างบน แล้วเอานิ้วลากวงรอบเลข 1 ที่**ติดกัน** ให้ได้วงที่ใหญ่ที่สุดเท่าที่จะทำได้ วงซ้อนทับกันได้ และวงหนึ่งจะมี 2 ช่องหรือ 4 ช่องก็ได้

คำถามคือ — น้องลากได้กี่วง

เฉลย — 3 วงครับ วงละ 2 ช่อง เลข 1 ที่แถว 11 สองช่องเป็นวงหนึ่ง · เลข 1 ที่คอลัมน์ขวาแถว 11 กับ 10 เป็นอีกวง · และเลข 1 ที่คอลัมน์ขวาแถว 01 กับ 11 เป็นวงที่สาม สังเกตว่าช่องมุมขวาแถว 11 ถูกใช้ทั้งสามวง — ใช้ซ้ำได้ครับ ไม่ผิด

วงที่น้องเพิ่งลากคือคำตอบแล้วครับ นี่คือสามวงเดียวกันระบายสีให้ดู



สามวงจากโจทย์เดิม — ป้าย ①②③ คู่กับพจน์ทางขวา

วงสีน้ำเงิน ① กินสองช่องของแถว $AB = 11$ ทั้งแถว อ่านได้เป็น AB · วงสีส้ม ② อ่านได้เป็น AC · วงสีเขียว ③ อ่านได้เป็น BC (วิธีอ่านค่อยสอนละเอียดในหัวข้อ 5 ครับ ตอนนี้ดูผลลัพธ์ก่อน)

$$X = AB + AC + BC$$

คำตอบเดียวกับที่พีชคณิตห้าบรรทัดเมื่อกี้ให้มา

สิ่งที่หายไปคือขั้น “ก๊อป ABC เพิ่มสองชุด” ครับ บนกระดาษตาราง น้องไม่ต้องรู้ล่วงหน้าเลยว่าต้องก๊อป — การที่สามวงซ้อนทับกันที่ช่องเดียวกัน **คือ** การก๊อปนั่นเอง แต่มันเกิดขึ้นเองจากการวงทับ

ทำไมภาพถึงช่วยได้จริง

กฎเดียวที่ใช้ย่อคือ “สองพจน์ที่ต่างกันแค่ตัวอักษรเดียว รวมกันได้” ใช่ไหมครับ ปัญหาของการเขียนเป็นบรรทัดคือน้องต้องเอาทุกพจน์ไปเทียบกับทุกพจน์เองเพื่อหาว่าคู่ไหนต่างกันตัวเดียว

สี่พจน์ก็ต้องเทียบ 6 คู่แล้วครับ ถ้าโจทย์มีสิบพจน์ต้องเทียบ 45 คู่ — มือไม่ทัน และพลาดง่าย

ตารางนี้ถูกออกแบบมาให้ **ช่องที่อยู่ติดกัน = พจน์ที่ต่างกันแค่ตัวอักษรเดียว** เสมอ พอเป็นอย่างนี้ น้องไม่ต้องเทียบอะไรเลย แค่มองว่าเลข 1 ตัวไหนอยู่ติดกัน

ตารางแบบนี้มีชื่อว่า `Karnaugh map` ครับ ภาษาไทยเรียกแผนที่คาร์นอห์ เรียกสั้น ๆ ว่า K-map และแต่ละช่องในนั้นเรียกว่า `cell`

ประโยคเดียวที่ต้องจำจากหัวข้อนี้

K-map **ไม่ได้ใช้กฎใหม่**เลยครับ มันคือกฎ $Y + \overline{Y} = 1$ อันเดิม เพียงแต่เอาตัวเลขมาวางใหม่ให้คู่ที่รวมกันได้**มาอยู่ติดกัน** งานของน้องจึงเปลี่ยนจาก “คิด” เป็น “มอง”

ลำดับที่จะทำทุกครั้ง

1. ได้โจทย์มาเป็นตารางค่าความจริง หรือนิพจน์ SOP หรือเลข minterm

2. วาดตาราง K-map ให้ถูกขนาด แล้วเติม 1 / 0 ลงไป

3. วงกลุ่มเลข 1 ที่ติดกัน ให้วงใหญ่ที่สุดและจำนวนวงน้อยที่สุด

4. อ่านแต่ละวงออกมาเป็นพจน์เดียว

5. เอาทุกพจน์มา OR กัน แล้วเช็กกลับเข้าตารางเดิม

ห้าขั้นนี้ใช้กับทุกข้อในบทนี้ครับ ที่เหลือทั้งบทคือการลงรายละเอียดทีละขั้น

2 หน้าตาของแผนที่ และทำไมหัวตารางเรียง 00 01 11 10

ขนาดของแผนที่ มาจากจำนวนแถวของตารางค่าความจริง

กฎข้อแรกง่ายมากครับ: **หนึ่งช่องในแผนที่ = หนึ่งแถวในตารางค่าความจริง** ห้ามขาดห้ามเกิน

ตัวแปร 3 ตัวมีตาราง 8 แถว แผนที่จึงต้องมี 8 ช่อง · ตัวแปร 4 ตัวมี 16 แถว แผนที่ต้องมี 16 ช่อง พูดอีกอย่างคือ แผนที่ของ n ตัวแปรมี 2^n ช่องเสมอ

หน้าตาที่วิชานี้ใช้คือแบบนี้ครับ — ตัวแปรสองตัวแรกอยู่ที่หัวแถวทางซ้าย ตัวแปรที่เหลืออยู่ที่หัวคอลัมน์ด้านบน

AB \ C	0	1
00	0	1
01	2	3
11	6	7
10	4	5

แผนที่ 3 ตัวแปร: AB คุมแถว (4 แถว) · C คุมคอลัมน์ (2 คอลัมน์) = 8 ช่อง

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

แผนที่ 4 ตัวแปร: AB คุมแถว · CD คุมคอลัมน์ = 16 ช่อง

เลขสีส้มเล็ก ๆ มุมขวาบนของแต่ละช่องคือ**เลขแถวของตารางค่าความจริง** หาได้จากการเอาค่าหัวแถวต่อหน้าค่าหัวคอลัมน์แล้วอ่านเป็นเลขฐานสอง

เช่นช่องที่หัวแถว $AB = 10$ ตัดกับหัวคอลัมน์ $CD = 11$ ต่อกันได้ 1011 แปลงเป็นเลขฐานสิบคือ 11 ช่องนั้นจึงคือแถวที่ 11 ของตาราง

ศัพท์ที่ต้องรู้ก่อนไปต่อ

เลขแถวนี้มีชื่อเรียกว่าเลขของ **minterm** ครับ เขียนย่อว่า m_{11} และเวลาโจทย์เขียนว่า $F(A,B,C,D) = \sum m(3,7,11)$ แปลว่า “ช่องที่ 3, 7, 11 เป็น 1 นอกนั้นเป็น 0 ทั้งหมด” — ไม่มีอะไรมากกว่านี้เลยครับ

ลองก่อน ไม่ต้องเขียน

ดูหัวคอลัมน์ของแผนที่ 4 ตัวแปรข้างบน: **00 01 11 10**

เดินจากซ้ายไปขวาทีละช่อง แล้วถามตัวเองว่า “จากช่องนี้ไปช่องถัดไป มีบิตเปลี่ยนกี่ตัว”

เฉลย — 1 ตัวเสมอครับ 00→01 เปลี่ยนบิตขวา · 01→11 เปลี่ยนบิตซ้าย · 11→10 เปลี่ยนบิตขวา และถ้าน้องลองวนจากช่องขวาสุด 10 กลับไปช่องซ้ายสุด 00 ก็ยังเปลี่ยนบิตเดียวอีก

ลำดับที่น้องเพิ่งไล่นั้นมีชื่อ

ลำดับ 00 01 11 10 ที่เพื่อนบ้านต่างกันบิตเดียวตลอด เรียกว่า **Gray code sequence** หรือ **รหัสเกรย์** ครับ

สังเกตว่ามันไม่ใช่การนับเลขปกติ — นับเลขปกติคือ 00 01 10 11 แต่รหัสเกรย์สลับสองตัวท้ายเป็น 11 ก่อนแล้วค่อย 10



บนคือรหัสเกรย์ที่ K-map ใช้ · ล่างคือการนับเลขปกติที่ใช้ไม่ได้

หัวแถวทางซ้ายก็เรียงแบบเดียวกันครับ 00 01 11 10 จากบนลงล่าง เพราะฉะนั้นในแผนที่นี้ **ทั้งแนวนอนและแนวตั้ง เพื่อนบ้านต่างกันบิตเดียวเสมอ**

ทำไมต้องบิตเดียว

ย้อนกลับไปดูกฎเดียวที่ใช้ย่อครับ: $Y + \overline{Y} = 1$ ใช้ได้ก็ต่อเมื่อ สองพจน์ต่างกันแค่ตัวอักษรเดียว

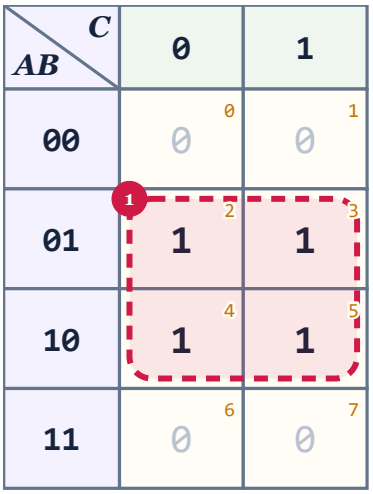
ถ้าสองช่องที่อยู่ติดกันต่างกันบิตเดียวจริง พจน์ของสองช่องนั้นก็ต่างกันตัวอักษรเดียวจริง วงมันเข้าด้วยกันแล้วตัดตัวที่เปลี่ยนทิ้งได้ทันที

เพราะฉะนั้นลำดับเกรย์ไม่ใช่ของสวยงามครับ มันคือ**สิ่งเดียว**ที่ทำให้การ “วงกลม” เท่ากับการใช้กฎพีชคณิตอย่างถูกต้อง ถ้าลำดับผิด การวงกลมก็ผิดตามทันที

เดินให้เห็นตอนพัง: ถ้าดันเรียงแบบนับเลข

สมมติว่าน้องเผลอวาดหัวแถวเป็น 00 01 10 11 แบบนับเลขปกติ แล้วโจทย์ให้ข้อมูลที่ช่องแถว $AB = 01$ ทั้งสองช่อง และแถว $AB = 10$ ทั้งสองช่องเป็น 1

บนกระดาษที่เรียงผิด สองแถวนี้จะมาอยู่ติดกัน หน้าตาน่าวงมาก น้องจะวงเป็นกลุ่ม 4 ช่อง



1 ?
วงนี้ไม่มีตัวแปรตัวไหนคงที่เลย อ่านเป็นพจน์ไม่ได้

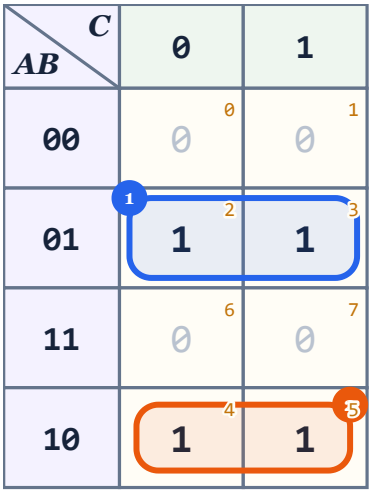
เรียงแถวแบบนับเลข 00 01 10 11 — แถว 01 กับ 10 มาอยู่ติดกันทั้งที่ต่างกัน 2 บิต

วงที่ดูดีมากแต่ผิด — เพราะแถว 01 กับ 10 ไม่ได้เป็นเพื่อนบ้านกันจริง

ลองอ่านพจน์จากวงนี้ดูครับ: ในวงมี A ทั้ง 0 และ 1 · มี B ทั้ง 0 และ 1 · มี C ทั้ง 0 และ 1 — **ไม่มีตัวแปรตัวไหนคงที่เลยสักตัว**

ตัดทุกตัวทิ้งแล้วเหลืออะไรครับ เหลือ 1 เฉย ๆ ซึ่งแปลว่า “ไฟติดทุกกรณี” ทั้งที่โจทย์บอกว่าติดแค่ 4 กรณีจาก 8 — ผิดไปครึ่งหนึ่งของตาราง

ทีนี้วาดใหม่ด้วยลำดับเกรย์ที่ถูก ข้อมูลชุดเดิมเป๊ะ ๆ



1 $\bar{A}B$

2 $A\bar{B}$

เรียงแบบเกรย์ 00 01 11 10 — สองแถวนี้ไม่ติดกัน จึงไม่มีทางวงผิด

ลำดับถูกแล้ว สองแถวนี้ถูกแถว 11 คั่นไว้ จึงวงรวมกันไม่ได้ตั้งแต่แรก

คำตอบที่ถูกคือ $X = \overline{A}B + A\overline{B}$ ครับ สองพจน์ ไม่ใช่พจน์เดียว และนี่คือคำตอบที่สั้นที่สุดแล้วจริง ๆ

กับดักที่เสียคะแนนบ่อยที่สุดในหัวข้อนี้

คนที่พลาดข้อนี้ในห้องสอบ**ไม่ได้พลาดตอนวงกลม**ครับ เขาพลาดตั้งแต่ตอนเขียนหัวตาราง แล้วทุกอย่างหลังจากนั้นก็ผิดตามหมด เพราะฉะนั้นเขียนหัวตารางเสร็จ ให้ไล่ปากเปล่าหนึ่งรอบว่า “ศูนย์ศูนย์ · ศูนย์หนึ่ง · **หนึ่งหนึ่ง** · หนึ่งศูนย์” ก่อนเติมเลขลงช่องเสมอ

ขั้นตอนวาดแผนที่เปล่า

1. นับตัวแปร → 3 ตัววาด 4×2 · 4 ตัววาด 4×4

2. เขียนชื่อตัวแปรที่มุมซ้ายบน: สองตัวแรกไว้ล่างเส้นทแยง ที่เหลือไว้บนเส้น

3. เขียนหัวคอลัมน์ 00 01 11 10 แล้ว**ไล่ปากเปล่าซ้ำหนึ่งรอบ**

4. เขียนหัวแถว 00 01 11 10 จากบนลงล่าง แล้วไล่ซ้ำอีกรอบ

5. ถ้าอยากกันพลาด เขียนเลข minterm ตัวเล็ก ๆ ไว้มุมช่องให้ครบก่อน แล้วค่อยเติม 1 กับ 0 — เสียเวลา 20 วินาที แต่กันพลาดได้ทั้งข้อ

3 เติมแผนที่ — จากตาราง และจากนิพจน์

แบบที่ 1: โจทย์ให้ตารางค่าความจริงมา

อันนี้ง่ายที่สุดครับ เพราะหนึ่งแถวไปหนึ่งช่องตรง ๆ ไม่ต้องคิดอะไรเลย

วิธีที่พลาดยากที่สุดคือ**ไล่ตามเลข minterm** ครับ: อ่านแถวที่ 0 ของตาราง แล้วไปหาช่องที่มีเลข 0 กำกับ เติมค่าลงไป · อ่านแถวที่ 1 ไปหาช่องเลข 1 · ทำไปจนครบทุกแถว

อย่าไล่ตามลำดับช่องซ้ายไปขวานะครับ เพราะลำดับช่องคือ 0, 1, 3, 2 ไม่ใช่ 0, 1, 2, 3 คนที่ไล่ตามช่องมักสลับเลขสองตัวสุดท้ายของทุกแถวโดยไม่รู้ตัว

ช่องที่โจทย์ไม่ได้พูดถึง

ถ้าโจทย์เขียนมาเป็น $\sum m(\dots)$ คือบอกมาแต่ช่องที่เป็น 1 ช่องที่เหลือ**เป็น 0 ทั้งหมด**ครับ ไม่ใช่ “ไม่รู้” — เติม 0 ลงไปให้ครบทุกช่องเลย อย่าปล่อยว่าง เพราะตอนหา POS ในหัวข้อ 6 น้องต้องใช้เลข 0 พวกนี้

แบบที่ 2: โจทย์ให้นิพจน์ SOP มา

ก่อนอื่นขอทวนคำว่า `SOP` ก่อนครับ ย่อมาจาก `sum of products` แปลว่า “ผลรวมของผลคูณ” คือหน้าตาแบบที่เราเจอมาตลอด — พจน์ที่เป็นตัวแปรคูณกัน แล้วเอามาบวก (OR) กัน

ลองของจริงเลยครับ โจทย์ให้มาว่า

$$X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + AB\overline{C} + A\overline{B}\,\overline{C}$$

หลักการมีข้อเดียว: **หนึ่งพจน์ = เติม 1 หนึ่งช่อง** (ถ้าพจน์นั้นมีตัวแปรครบทุกตัว) และช่องไหนที่หาพจน์มาลงไม่ได้ก็เป็น 0

วิธีหาว่าพจน์ไหนลงช่องไหน: อ่านพจน์ทีละตัวอักษร ตัวเปล่าแปลว่า 1 ตัวมีขีดบนแปลว่า 0 เขียนเรียงกันได้เลขฐานสอง แล้วเอาเลขนั้นไปหาช่อง

แปลงทีละพจน์

$\overline{A}\,\overline{B}\,\overline{C}$	A=0 B=0 C=0	000	ช่อง m0
$\overline{A}\,\overline{B}C$	A=0 B=0 C=1	001	ช่อง m1
$AB\overline{C}$	A=1 B=1 C=0	110	ช่อง m6
$A\overline{B}\,\overline{C}$	A=1 B=0 C=0	100	ช่อง m4

เติม 1 ลงสี่ช่องนั้น ที่เหลือเติม 0 ได้แผนที่แบบนี้ครับ

AB \ C	0	1
00	1 (0)	1 (1)
01	0 (2)	0 (3)
11	1 (6)	0 (7)
10	1 (4)	0 (5)

สี่พจน์กลายเป็นสี่ช่องที่เป็น 1 — ช่อง m0, m1, m4, m6

ลองก่อน ไม่ต้องเขียน

มองแผนที่ข้างบน แล้วตอบว่าเลข 1 คู่ไหนที่**อยู่ติดกัน**บ้าง (ติดกันแปลว่าแชร์ขอบกัน ไม่ใช่แตะกันแค่มุม)

เฉลย — คู่ที่แถว $AB = 00$ สองช่อง (m0 กับ m1) อยู่ติดกันซ้าย-ขวา · และ m4 กับ m6 ที่คอลัมน์ $C = 0$ แถว 11 กับ 10 อยู่ติดกันบน-ล่าง

ส่วน m0 กับ m4 อยู่คอลัมน์เดียวกัน แต่คนละหัวแถว (00 กับ 10) ดูเผิน ๆ เหมือนห่างกันสุดตาราง — จริง ๆ แล้วสองช่องนี้ก็**ติดกัน**ครับ เพราะแถวบนสุดกับแถวล่างสุดต่อกัน เดี๋ยวหัวข้อ 4 จะอธิบายเรื่องนี้เต็ม ๆ ตอนนี้ยังไม่ต้องใช้

วงสองวงที่น้องเจอ ให้คำตอบนี้ครับ

AB \ C	0	1
00	1 ①	1 ①
01	0	0
11	1 ②	0
10	1 ②	0

① $\overline{A}\,\overline{B}$

② $A\overline{C}$

① คลุมแถว $AB = 00$ ทั้งแถว · ② คลุมคอลัมน์ $C = 0$ ที่แถว 11 กับ 10

$$X = \overline{A}\,\overline{B} + A\overline{C}$$

จาก 12 ตัวอักษร เหลือ 4 ตัวอักษร

เช็กกลับเข้าโจทย์เดิมครับ: $\overline{A}\,\overline{B}$ เป็น 1 ที่ m_0 กับ $m_1 \cdot A\overline{C}$ เป็น 1 ที่ m_4 กับ m_6 — รวมกันได้ $\{0, 1, 4, 6\}$ ตรงกับโจทย์เป๊ะ และไม่เผลอไปแตะช่องอื่นเลย

กรณีที่พจน์มีตัวแปรไม่ครบ

บางทีโจทย์ให้พจน์แบบขาดตัวแปรมาครับ เช่น $\overline{A}B$ ในโจทย์ 3 ตัวแปร — พจน์นี้ไม่ได้บอกว่า C เป็นอะไร

ความหมายของมันคือ “ขอแค่ $A = 0$ และ $B = 1$ ก็พอ ส่วน C จะเป็นอะไรก็ได้” เพราะฉะนั้นมันเป็น 1 ได้**สองช่อง** คือ $C = 0$ และ $C = 1$

กฎนับง่าย ๆ คือ **ขาดตัวแปรไป k ตัว ก็เติม 1 ลง 2^k ช่อง** ครับ ขาด 1 ตัวเติม 2 ช่อง · ขาด 2 ตัวเติม 4 ช่อง · ขาด 3 ตัวเติม 8 ช่อง

กับดัก

คนที่รีบมักเห็น $\overline{A}B$ แล้วเติม 1 **ช่องเดียว** เพราะเคยชินว่าหนึ่งพจน์หนึ่งช่องครับ ผลคือแผนที่ขาดไป 1 ช่อง แล้ววงกลุ่มออกมาเล็กกว่าความจริง คำตอบจะยาวกว่าที่ควรโดยไม่รู้ตัว
วิธีกันพลาด: ก่อนเติม ให้นับก่อนว่าพจน์นี้มีตัวอักษรกี่ตัว ถ้าน้อยกว่าจำนวนตัวแปรทั้งหมด แปลว่ามันกินมากกว่า 1 ช่องแน่นอน

เช็กลิสต์การเติมแผนที่

1. วาดแผนที่เปล่าให้ถูกขนาด พร้อมเลข minterm ทุกช่อง
2. ถ้าได้ตารางมา → ไล่ทีละแถว เอาเลขแถวไปหาช่อง
3. ถ้าได้นิพจน์มา → ไล่ทีละพจน์ แปลงตัวอักษรเป็น 0/1 แล้วหาช่อง
4. พจน์ที่ตัวแปรไม่ครบ ให้เติมทุกช่องที่ตัวแปรที่มีอยู่ตรงกัน
5. เติม 0 ให้ครบทุกช่องที่เหลือ แล้ว**นับจำนวน 1 เทียบกับโจทย์** ก่อนไปวงกลุ่ม

หยุดตรงนี้ก่อนครับ อย่าเพิ่งวงกลุ่มจนกว่าจะนับเลข 1 ครบ

4 กฎการวงกลุ่ม

วงยิ่งใหญ่ พจน์ยิ่งสั้น — ดูของจริงก่อน

ก่อนจะท่องกฎ ขอให้เห็นก่อนครับว่าทำไมเราถึงอยากวงให้ใหญ่

เอาวง 4 ช่องวงหนึ่งมากางให้ดูทีละบรรทัด เป็นสี่ช่องตรงกลางแผนที่ 4 ตัวแปร คือช่อง m_5, m_7, m_{13}, m_{15}

กางวง 4 ช่องด้วยพีชคณิต

$m5 = 0101 \rightarrow \bar{A}B\bar{C}D$

$m7 = 0111 \rightarrow \bar{A}BCD$

$m13 = 1101 \rightarrow AB\bar{C}D$

$m15 = 1111 \rightarrow ABCD$

รวมคู่บน : $\bar{A}B\bar{C}D + \bar{A}BCD = \bar{A}BD(\bar{C}+C) = \bar{A}BD$

รวมคู่ล่าง : $AB\bar{C}D + ABCD = ABD(\bar{C}+C) = ABD$

รวมสองก้อน : $\bar{A}BD + ABD = BD(\bar{A}+A) = BD$

เห็นจังหวะไหมครับ — รวมรอบแรกตัด C ทิ้ง รวมรอบสองตัด A ทิ้ง สี่ช่องจาก 16 ตัวอักษร เหลือแค่ BD สองตัวอักษร

ทุกครั้งที่จำนวนช่องเพิ่มเป็นสองเท่า จะมีตัวอักษรหายไปหนึ่งตัวเสมอ สรุปเป็นตารางได้แบบนี้

ขนาดวง	ตัดทิ้งกี่ตัว	เหลือกี่ตัว (โจทย์ 4 ตัวแปร)
1 ช่อง	0	4
2 ช่อง	1	3
4 ช่อง	2	2
8 ช่อง	3	1
16 ช่อง	4	0 $(X = 1)$

ตารางนี้คือเหตุผลทั้งหมดของกฎ “วงให้ใหญ่ที่สุด” ครับ — วงใหญ่ขึ้นเท่าตัว แปลว่าพจน์สั้นลงหนึ่งตัวอักษร แปลว่าวงจรถูกลงหนึ่งขา

ลองก่อน ไม่ต้องเขียน

ถ้าน้องวงได้ 3 ช่อง จะตัดตัวอักษรทิ้งได้กี่ตัวครับ

เฉลย — ตอบ “ไม่ได้เลย” ครับ ลองคิดตาม: 3 ช่องจับคู่ได้ 1 คู่ แล้วเหลือช่องเศษอีก 1 ช่องที่หาคู่ไม่ได้ ช่องเศษนั้นก็ยังต้องเขียนเต็มตัวอักษรอยู่ดี สรุปคือ**ไม่มีวง 3 ช่อง**ในโลกของ K-map

กฎ 6 ข้อ พร้อมเหตุผลของแต่ละข้อ

1

จำนวนช่องในวงต้องเป็นกำลังของสอง — 1, 2, 4, 8, 16 เท่านั้น

เพราะ การรวมทำได้ทีละครึ่ง ถ้าหารสองแล้วไม่ลงตัวจะมีช่องเศษที่หาคู่ไม่ได้ เลขที่หารสองไปเรื่อย ๆ จนเหลือ 1 ได้พอดี มีแค่ 2, 4, 8, 16

2

วงได้เฉพาะช่องที่ติดกันด้านต่อด้าน ห้ามวงทแยง

เพราะ ช่องที่แตะกันแค่มุม ต่างกัน 2 บิต ไม่ใช่ 1 บิต กฎ $Y + \overline{Y} = 1$ จึงใช้ไม่ได้ (เหตุผลเดียวกับหัวข้อ 2 เป๊ะ)

3

วงต้องใหญ่ที่สุดเท่าที่ทำได้ และจำนวนวงต้องน้อยที่สุด

เพราะ วงใหญ่ = พจน์สั้น (ตารางข้างบน) · วงน้อย = จำนวนพจน์น้อย ข้อสอบที่เขียนว่า *minimum* ตรวจสองอย่างนี้

4

วงซ้อนทับกันได้ แต่ห้ามมีวงที่จมอยู่ในวงอื่นทั้งวง

เพราะ ซ้อนทับได้ด้วยกฎ $Y + Y = Y$ ใช้ช่องซ้ำไม่เสียอะไร แต่ถ้าวงเล็กจมอยู่ในวงใหญ่ทั้งวง ทุกช่องของมันถูกคลุมไปแล้ว พจน์นั้นจึงไม่ได้ทำอะไรเลย เขียนไปก็เปลืองเกตเปล่า ๆ

5

เลข 1 ทุกตัวต้องอยู่ในวงอย่างน้อยหนึ่งวง

เพราะ 1 ที่ไม่ถูกวง แปลว่าไม่มีพจน์ไหนทำให้ช่องนั้นเป็น 1 วงจรที่ได้จะให้ผลผิดที่กรณีนั้น · ถ้าเจอ 1 ที่ไม่มีเพื่อนบ้านจริง ๆ ก็วงมันตัวเดียว — น่าเกลียดแต่ถูก

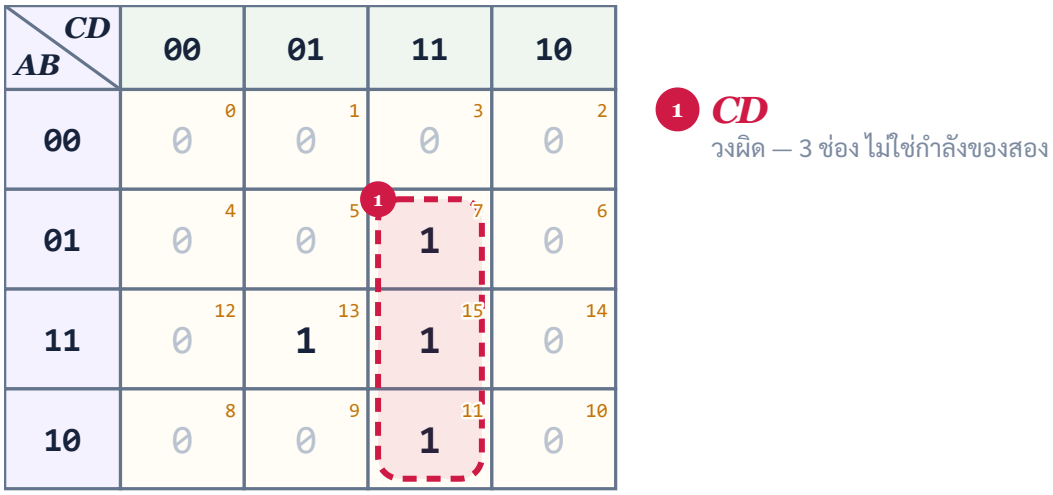
6

ขอบแผนที่ต่อกัน — บนต่อล่าง ซ้ายต่อขวา

เพราะ หัวแถวเรียง 00 01 11 10 ซึ่งวนกลับมาที่ 00 ได้โดยเปลี่ยนบิตเดียว (10 → 00 เปลี่ยนบิตซ้ายตัวเดียว) แถวบนสุดกับแถวล่างสุดจึงเป็นเพื่อนบ้านกันจริง ๆ คอลัมน์ซ้ายสุดกับขวาสุดก็เหมือนกัน

เดินให้เห็นตอนพัง ตอนที่ 1: วง 3 ช่อง

แผนที่นี้มีเลข 1 อยู่ 4 ตัว ที่ $m_7, m_{11}, m_{13}, m_{15}$ มีคนวงคอลัมน์ $CD = 11$ รวดเดียวสามช่องเพราะเห็นว่าเรียงกันสวย

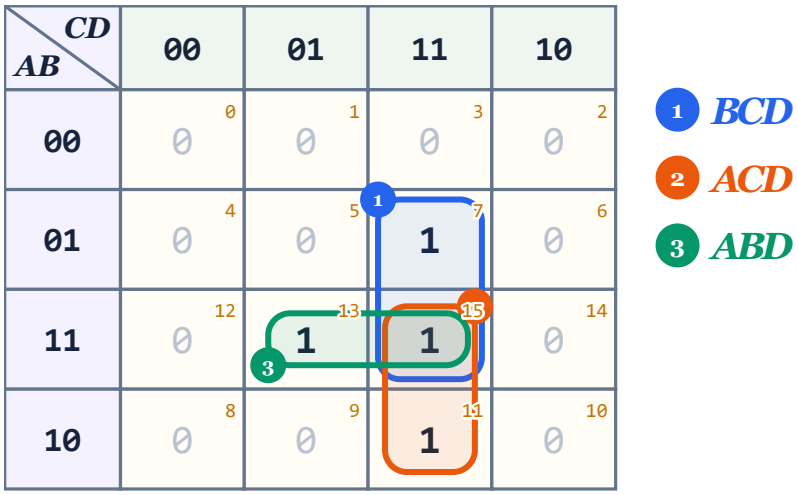


วง 3 ช่อง — ผิดกฎข้อ 1

ถ้าอ่านพจน์จากวงนี้ จะได้ว่าในวงมี $C = 1$ ตลอด และ $D = 1$ ตลอด ส่วน A กับ B เปลี่ยน จึงเขียนออกมาได้ CD

ปัญหาคือ CD เป็น 1 ที่ช่องไหนบ้างครับ — ทุกช่องในคอลัมน์ $CD = 11$ ซึ่งมี**สี่ช่อง** ไม่ใช่สาม ช่องที่เกินมาคือ m_3 ไปดูในแผนที่ ช่อง m_3 เป็น **0** ครับ แปลว่าคำตอบ CD สั่งให้ไฟติดในกรณีที่โจทย์บอกว่าต้องดับ — ผิดหนึ่งกรณีเต็ม ๆ

วงที่ถูกต้องคือวงละ 2 ช่อง สามวง แบบนี้ครับ



สามวง วงละ 2 ช่อง ทับกันที่ m_{15} ได้ไม่ผิดกฎ

$$X = BCD + ACD + ABD$$

สามพจน์ พจน์ละ 3 ตัวอักษร — นี่คือคำตอบที่สั้นที่สุดของแผนที่นี้จริง ๆ

เช็กกลับ: BCD ให้ m_7 กับ m_{15} · ACD ให้ m_{11} กับ m_{15} · ABD ให้ m_{13} กับ m_{15} รวมได้ $\{7, 11, 13, 15\}$ พอดี ไม่มีช่อง 0 หลุดเข้ามา

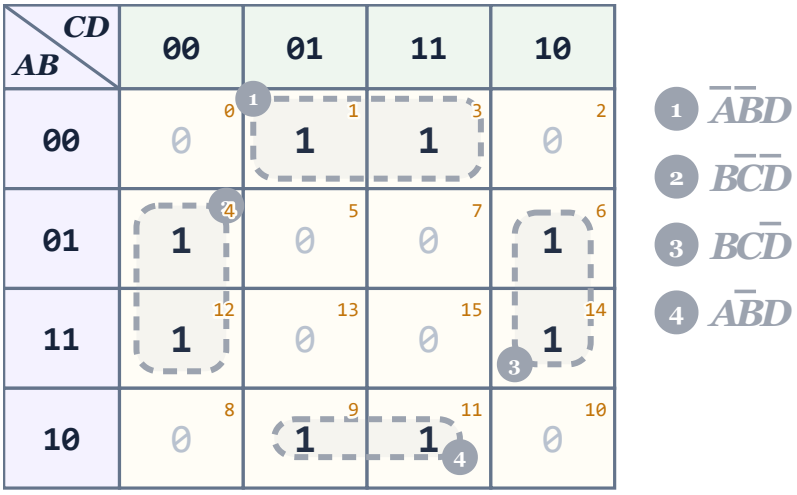
คำตอบที่ไม่สวยไม่ได้แปลว่าผิด

แผนที่นี้ไม่มีวง 4 ช่องเลยครับ ถึงจะดูเหมือนควรจะมี บางโจทย์ก็ให้คำตอบที่ยาวแบบนี้แหละ **อย่าฝืนวงให้ใหญ่**เพียงเพราะอยากได้พจน์สั้น — วงผิดกฎเสียคะแนนหนักกว่าคำตอบยาว

เดินให้เห็นตอนพัง ตอนที่ 2: ลืมว่าขอบต่อกัน

กฎข้อ 6 เป็นข้อที่ลืมกันมากที่สุดครับ เพราะกระดาษมันแบน แต่แผนที่นี้ต้องคิดเหมือนกระดาษที่ม้วนเป็นท่อ — ขอบซ้ายไปแปะกับขอบขวา แล้วม้วนอีกทีให้ขอบบนไปแปะขอบล่าง

ดูแผนที่นี้ครับ เลข 1 อยู่ 8 ตัว กระจายเป็นลายตาราง

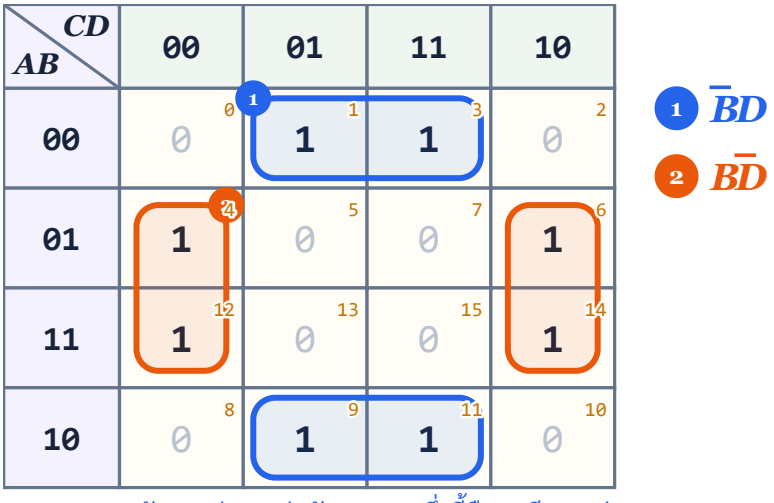


คำตอบของคนที่มองว่ากระดาษแบน — วงได้แค่วงละ 2 ช่อง สี่วง

คนที่ไม่คิดเรื่องขอบจะวงได้แค่นี้ครับ สี่วง วงละ 2 ช่อง อ่านออกมาได้ $X = \overline{A}\,\overline{B}D + B\overline{C}\,\overline{D} + BC\overline{D} + A\overline{B}D$

คำตอบนี้**ถูก**นะครับ ค่าตรงทุกช่อง แต่ยาว 12 ตัวอักษร ถ้าข้อสอบเขียนว่า *minimum SOP* ข้อนี้ได้ไม่เต็ม

ทีนี้มองใหม่โดยจำไว้ว่าแถวบนสุด (00) กับแถวล่างสุด (10) ติดกัน และคอลัมน์ซ้ายสุด (00) กับขวาสุด (10) ก็ติดกัน



① แถวบนสุดกับแถวล่างสุดต่อกัน — สองครึ่งนี้คือวงเดียว 4 ช่อง

② คอลัมน์ซ้ายสุดกับขวาสุดต่อกัน — สองครึ่งนี้คือวงเดียว 4 ช่อง

วงเดียวกันที่โผล่สองที่ — ① ต่อกันรอบขอบบน-ล่าง · ② ต่อกันรอบขอบซ้าย-ขวา

$$X = \overline{B}D + B\overline{D}$$

สองพจน์ 4 ตัวอักษร — จาก 12 เหลือ 4

เช็กกลับเข้าแผนที่ครับ: $\overline{B}D$ คือช่องที่ $B = 0$ และ $D = 1$ ได้แก่ m_1, m_3, m_9, m_{11} — ตรงกับสี่ช่องของวง ① พอดี ส่วน $B\overline{D}$ คือช่องที่ $B = 1$ และ $D = 0$ ได้แก่ m_4, m_6, m_{12}, m_{14} ตรงกับวง ② พอดี รวมแปดช่องครบ

วิธีกันลืมเรื่องขอบ

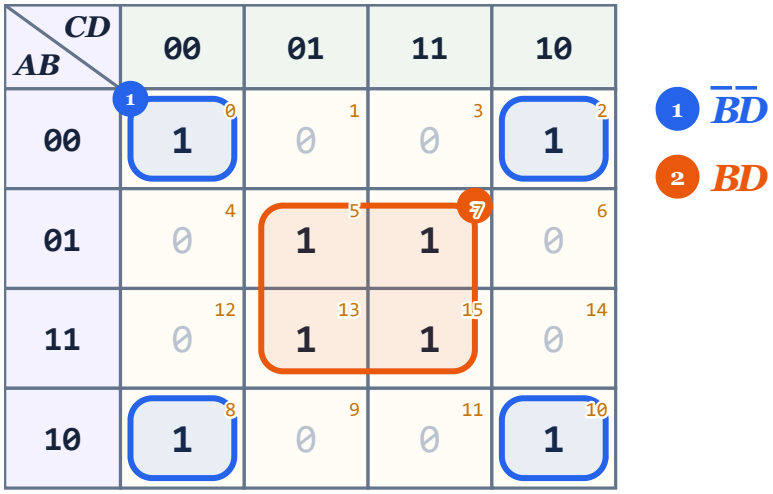
ทุกครั้งที่วงเสร็จ ให้**กวาดตาดูขอบทั้งสี่ด้าน**ก่อนสรุปครับ — มี 1 ตัวไหนติดขอบซ้ายไหม ถ้ามี ไปดูช่องตำแหน่งเดียวกันที่ขอบขวา · มี 1 ติดขอบบนไหม ไปดูขอบล่างแถวเดียวกัน
ใช้เวลา 10 วินาที แต่เป็นจุดที่คนเสียคะแนนกันมากที่สุดในเรื่องนี้

กรณีพิเศษที่ออกสอบบ่อย: สี่มุม

ถ้าขอบซ้ายต่อขวา และขอบบนต่อล่างจริง ลองคิดตามครับว่าช่องมุมซ้ายบนติดกับใครบ้าง

มันติดกับมุมขวาบน (ข้ามขอบซ้าย-ขวา) และติดกับมุมซ้ายล่าง (ข้ามขอบบน-ล่าง) ส่วนมุมขวาล่างก็ติดกับอีกสองมุมเช่นกัน

แปลว่า **สี่มุมของแผนที่ 4 ตัวแปร เป็นวง 4 ช่องที่ถูกกฎ** ครับ เป็นกรณีที่อาจารย์ชอบออกเพราะมันมองไม่เห็นด้วยตาเปล่า



① สี่มุมคือกลุ่มเดียวกัน — ม้วนซ้าย-ขวาชนกัน แล้วม้วนบน-ล่างชนกัน

① คือสี่มุม เป็นวงเดียวกัน 4 ช่อง · ② คือวง 4 ช่องตรงกลางแบบปกติ

อ่านวง ① ครับ: ทั้งสี่มุมมี $B = 0$ เหมือนกันหมด และ $D = 0$ เหมือนกันหมด ส่วน A กับ C เปลี่ยน จึงได้ $\overline{B}\,\overline{D}$

อ่านวง ② : สี่ช่องกลางมี $B = 1$ และ $D = 1$ เหมือนกัน ได้ BD

$$X = \overline{B}\,\overline{D} + BD$$

แปดช่อง เหลือสองพจน์ พจน์ละ 2 ตัวอักษร

ถ้าน้องไม่เห็นวงสี่มุม น้องจะวงมุมทีละคู่ได้สี่พจน์ คำตอบจะยาวขึ้นเท่าตัวทันที

ลำดับการวงกลุ่มที่ทำตามได้ในห้องสอบ

1. มองหาวง 8 ช่องก่อน (ครึ่งแผนที่) — มีไหม ถ้ามีวงเลย

2. มองหาวง 4 ช่อง — ทั้งแบบบล็อก 2×2 · แบบแถวเต็ม · แบบคอลัมน์เต็ม · **แบบข้ามขอบ** · และ**แบบสี่มุม**

3. เหลือ 1 ตัวไหนยังไม่ถูกวง ค่อยวงเป็นคู่ 2 ช่อง

4. ยังเหลืออีกไหม ถ้าเหลือจริง ๆ วงตัวเดียว

5. กวาดขอบทั้งสี่ด้านซ้ำหนึ่งรอบ

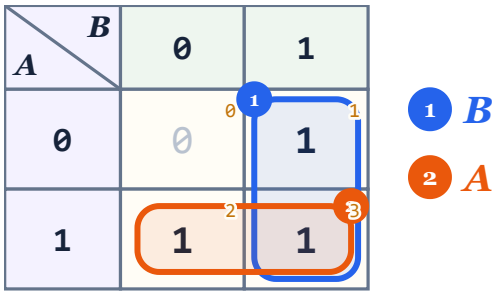
6. ตรวจว่ามีวงไหนจมอยู่ในวงอื่นทั้งวงหรือเปล่า ถ้ามีให้**ลบวงนั้นทิ้ง**

หยุดตรงนี้ — อย่าเพิ่งเขียนคำตอบจนกว่าจะนับได้ว่าเลข 1 ทุกตัวอยู่ในวงแล้ว

5 อ่านพจน์จากวง — “มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน”

ของจริงก่อน: แผนที่เล็กที่สุดที่มี

เริ่มจากแผนที่ 2 ตัวแปรครับ มีแค่ 4 ช่อง A คุมแถว B คุมคอลัมน์ โจทย์นี้มีเลข 1 อยู่สามตัว



วงแนวตั้ง ① กินคอลัมน์ $B = 1$ สองช่อง · วงแนวนอน ② กินแถว $A = 1$ สองช่อง

ลองก่อน ไม่ต้องเขียน

ดูวง ① (วงแนวตั้งสีน้ำเงิน) แล้วตอบสองข้อครับ

หนึ่ง — เลื่อนตาไป**ทางซ้าย**ของวง เห็นหัวแถวอะไรบ้าง

สอง — เลื่อนตา**ขึ้นบน**ของวง เห็นหัวคอลัมน์อะไรบ้าง

เฉลย — ทางซ้ายเห็น A เป็น 0 กับ 1 คือ**เปลี่ยน** · ข้างบนเห็น B เป็น 1 อย่างเดียว คือ**ไม่เปลี่ยน**

ที่น้องเพิ่งทำนั่นแหละคือทริกทั้งหมดของหัวข้อนี้ครับ

ทริกเต็ม ๆ สามท่อน

อาจารย์สรุปไว้เป็นประโยคภาษาอังกฤษว่า **“look left, look up, look for variable that do not change”** แปลตรงตัวคือ “มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน”

ทำจริงคือสามท่อนนี้ครับ

1 **มองซ้าย — อ่านหัวแถวทุกแถวที่วงนี้กินอยู่**

จดไว้ว่าตัวแปรแต่ละตัวที่หัวแถวเป็นค่าอะไรบ้าง

2 **มองบน — อ่านหัวคอลัมน์ทุกคอลัมน์ที่วงนี้กินอยู่**

จดเหมือนกัน ตอนนี้น้องจะมีค่าของตัวแปรครบทุกตัวแล้ว

3 **ตัวไหนมีทั้ง 0 และ 1 → ทิ้ง · ตัวไหนมีค่าเดียว → เก็บ**

ตัวที่เก็บ ถ้าค่านั้นคือ **1** เขียนตัวอักษรเปล่า · ถ้าค่านั้นคือ **0** เขียนตัวอักษรแบบมีขีดบนหัว · แล้วเอาตัวที่เก็บมาคูณกัน

ใช้กับวง ① ของแผนที่เมื่อกี้: A มีทั้ง 0 และ 1 → ทิ้ง · B มีแต่ 1 → เก็บเป็น B ได้พจน์ B ตัวเดียว

ใช้กับวง ② : A มีแต่ 1 → เก็บเป็น $A \cdot B$ มีทั้ง 0 และ 1 → ทิ้ง ได้พจน์ A

$$X = A + B$$

เช็กกลับ: $A + B$ เป็น 0 เฉพาะตอน $A = 0, B = 0$ ซึ่งคือช่องเดียวที่เป็น 0 พอดี

ใช้กับแผนที่ 4 ตัวแปร — ทำให้ดูทุกบรรทัด

แผนที่ 4 ตัวแปรใช้ทริกเดียวกันเป๊ะครับ ต่างแค่ “มองซ้าย” ได้สองตัวแปร และ “มองบน” ได้อีกสองตัวแปร

AB \ CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	0 (4)	1 (5)	1 (7)	0 (6)
11	0 (12)	1 (13)	1 (15)	0 (14)
10	1 (8)	0 (9)	0 (11)	1 (10)

1 ?

มองซ้ายเห็นหัวแถว 01 กับ 11 · มองบนเห็นหัวคอลัมน์ 01 กับ 11

วง 4 ช่องตรงกลาง — ยังไม่บอกคำตอบ อ่านไปพร้อมกัน

อ่านวงทีละบรรทัด

มองซ้าย : หัวแถวที่วงกิน = 01 และ 11

→ A เป็น 0 (จาก 01) และ 1 (จาก 11) **เปลี่ยน → ทิ้ง**

→ B เป็น 1 (จาก 01) และ 1 (จาก 11) **ไม่เปลี่ยน → เก็บ**

มองบน : หัวคอลัมน์ที่วงกิน = 01 และ 11

→ C เป็น 0 และ 1 **เปลี่ยน → ทิ้ง**

→ D เป็น 1 และ 1 **ไม่เปลี่ยน → เก็บ**

เก็บ B (ค่า 1 → เขียนตัวเปล่า) และ D (ค่า 1 → เขียนตัวเปล่า)

พจน์ = BD

เช็กกลับครับ: BD เป็น 1 ที่ช่องที่ $B = 1$ และ $D = 1$ คือ m_5, m_7, m_{13}, m_{15} — สี่ช่องที่วงคลุมอยู่พอดี ไม่ขาดไม่เกิน

ทางลัดที่ใช้ได้เสมอ

วงขนาด 2^k ช่อง จะ**ทิ้งตัวแปรไป k ตัวพอดี** ครับ เพราะฉะนั้นถ้าวง 4 ช่องแล้วอ่านได้ 3 ตัวอักษร แปลว่าอ่านผิดแน่นอน — กลับไปดูใหม่ นี่เป็นวิธีตรวจคำตอบที่เร็วที่สุด

เดินให้เห็นตอนพัง: เขียนขีดบนกลับข้าง

ความผิดพลาดที่เจอบ่อยที่สุดในหัวข้อนี้คือเขียนขีดบนสลับกันครับ เพราะมันฝืนสัญชาตญาณ

สมมติวงหนึ่งอยู่ในแถวที่ $A = 0$ ตลอด คนจำนวนมากเขียนเป็น A เพราะ “ก็แถว A นี่” แต่ที่ถูกคือ $\overline{A}$ เพราะค่าที่คงที่คือ **0**

ผลของการเขียนกลับข้าง: $\overline{A}$ กับ A คลุมคนละครึ่งของแผนที่กันเลย คำตอบจะผิดทุกช่องที่วงนั้นดูแล

ประโยคกันพลาด

ท่องว่า “**เห็นศูนย์ ใส่ขีด**” ครับ — ค่าคงที่เป็น 0 ใส่ขีดบน · ค่าคงที่เป็น 1 ไม่ใส่
และก่อนส่ง ให้สุ่มหยิบพจน์มาหนึ่งพจน์ แล้วไล่ดูว่ามันเป็น 1 ที่ช่องไหนบ้าง ต้องตรงกับวงที่น้องวาดไว้เป๊ะ

6 minimum POS — วงเลข 0 แทน

ของจริงก่อน: คำตอบอีกหน้าตาหนึ่ง

ที่ผ่านมาคำตอบของเราหน้าตาแบบนี้ครับ $AB + AC + BC$ คือ**คูณก่อน แล้วค่อยบวก** ชื่อของหน้าตานี้คือ SOP

แต่บางข้อ อาจารย์จะสั่งว่า *“determine the minimum POS expression”* ซึ่งต้องการคำตอบหน้าตาอีกแบบหนึ่ง คือ **บวกก่อน แล้วค่อยคูณ** เช่น $(A + B)(A + C)$

หน้าตาแบบนี้เรียกว่า `POS` ย่อมาจาก `product of sums` แปลว่า “ผลคูณของผลบวก” ครับ แต่ละวงเล็บเรียกว่า `sum term` หรือ**พจน์ผลบวก**

เหตุผลที่โลกต้องมีสองหน้าตา: SOP ต่อวงจรได้ด้วย AND ตามด้วย OR · POS ต่อด้วย OR ตามด้วย AND ซึ่งบางทีก็ถูกกว่า ขึ้นอยู่กับว่าแผนที่มี 1 เยอะหรือ 0 เยอะ

โจทย์ตัวอย่าง

ใช้ตารางนี้ครับ ตัวแปร 3 ตัว

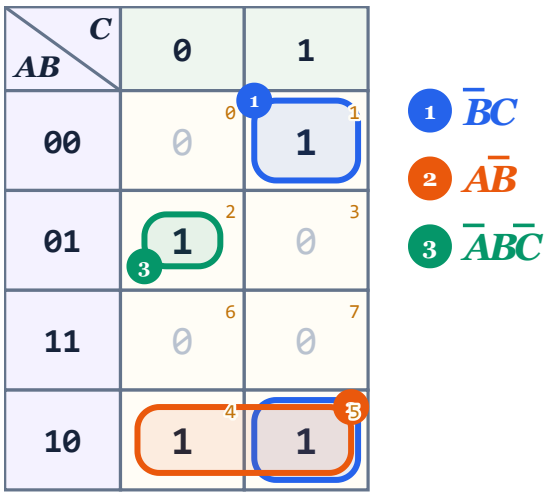
A	B	C	X	แถวที่
0	0	0	0	0
0	0	1	**1**	1
0	1	0	**1**	2
0	1	1	0	3
1	0	0	**1**	4
1	0	1	**1**	5
1	1	0	0	6
1	1	1	0	7

เติมลงแผนที่ตามวิธีหัวข้อ 3 ได้แบบนี้ครับ

AB \ C	0	1
00	0 (0)	1 (1)
01	1 (2)	0 (3)
11	0 (6)	0 (7)
10	1 (4)	1 (5)

แผนที่ของ X — มี 1 อยู่ 4 ช่อง และ 0 อยู่ 4 ช่อง

ถ้าทำแบบ SOP ปกติ จะวงได้สามวงและได้คำตอบนี้



① วงเดียวกัน ต่อกันรอบขอบบน-ล่าง

SOP ปกติ — ① ต้องข้ามขอบบน-ล่าง · ③ เป็น 1 โดดเดี่ยวจึงวงตัวเดียว

$$X = \overline{B}C + A\overline{B} + \overline{A}B\overline{C}$$

minimum SOP: 3 พจน์ 7 ตัวอักษร

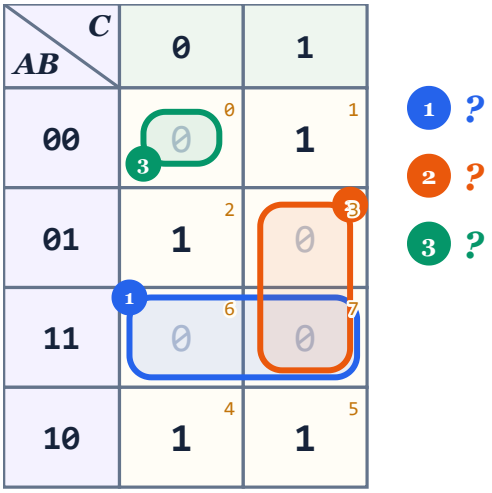
ลองก่อน ไม่ต้องเขียน

กลับไปดูแผนที่ของ X อีกครั้ง แต่คราวนี้ให้มอง**เลข 0** แทนเลข 1

ถ้าใช้กฎวงกลุ่มชุดเดิมทุกข้อ แต่วงเลข 0 — น้องจะวงได้กี่วง วงละกี่ช่อง

เฉลย — 3 วงครับ: m_6 กับ m_7 เป็นคู่ (แถว $AB = 11$) · m_3 กับ m_7 เป็นคู่ (คอลัมน์ $C = 1$ แถว 01 กับ 11) · ส่วน m_0 ไม่มีเพื่อนบ้านที่เป็น 0 เลย จึงวงตัวเดียว

นั่นคือทั้งหมดของหัวข้อนี้แล้วครับ POS ไม่มีกฎใหม่เลย ใช้กฎวงกลุ่มเดิมกับเลข 0

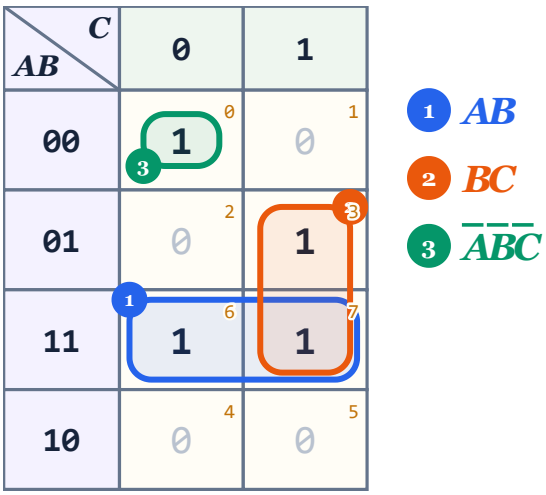


สามวงบนเลข 0 — ยังไม่ต้องอ่านเป็นพจน์ อ่านยังไงอยู่ย่อหน้าถัดไป

ทำไมวงเลข 0 แล้วได้ POS

ขอเรียกของใหม่ว่า $\overline{X}$ ครับ อ่านว่า “เอ็กซ์บาร์” แปลว่ากลับค่าของ X — ช่องไหน X เป็น 1 $\overline{X}$ เป็น 0 และช่องไหน X เป็น 0 $\overline{X}$ เป็น 1

เพราะฉะนั้น **การวงเลข 0 บนแผนที่ของ X ก็คือการวงเลข 1 บนแผนที่ของ $\overline{X}$** เป๊ะ ๆ ครับ เป็นงานเดียวกัน แค่มองคนละมุม



แผนที่ของ $\overline{X}$ — เลข 1 ในนี้คือเลข 0 ของ X · สามวงเดิม อ่านเป็นพจน์ SOP ได้ตามปกติ

อ่านสามวงนี้ด้วยทริก “มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน” แบบเดิมเป๊ะ

อ่านวงบนแผนที่ของ X บาร์

① แถว 11 ทั้งแถว : A=1 คงที่ · B=1 คงที่ · C เปลี่ยน → AB

② คอลัมน์ C=1 แถว 01,11 : A เปลี่ยน · B=1 คงที่ · C=1 คงที่ → BC

③ ช่อง m0 ตัวเดียว : A=0 · B=0 · C=0 → $\overline{A}\,\overline{B}\,\overline{C}$

$$\overline{X} = \overline{A}\,\overline{B}\,\overline{C} + BC + AB$$

นี่คือ minimum SOP ของ $\overline{X}$ ยังไม่ใช่คำตอบ

ตอนนี้เรามี $\overline{X}$ แล้ว แต่โจทย์ถามหา X ครับ วิธีกลับคืนคือใส่ขีดบนทั้งก้อนอีกที เพราะขีดสองชั้นหักล้างกัน ($\overline{\overline{X}} = X$)

ใช้เดอมอร์แกนกลับคืน ทีละบรรทัด

X = ขีดบนของ ($\overline{A}\,\overline{B}\,\overline{C}$ + BC + AB) ← ขีดสองชั้นหักล้างกัน

ขีดคลุมเครื่องหมาย + → เปลี่ยนเป็น × แล้วขีดแตกลงไปทีละก้อน

X = (ขีดของ $\overline{A}\,\overline{B}\,\overline{C}$) × (ขีดของ BC) × (ขีดของ AB) ← เดอมอร์แกนรอบที่ 1

ในแต่ละวงเล็บ ขีดคลุมการคูณ → เปลี่ยนเป็น + แล้วขีดแตกลงตัวอักษร

X = (A + B + C)($\overline{B}$ + $\overline{C}$)($\overline{A}$ + $\overline{B}$) ← เดอมอร์แกนรอบที่ 2 = คำตอบ

$$X = (A + B + C)(\overline{B} + \overline{C})(\overline{A} + \overline{B})$$

minimum POS: 3 วงเล็บ 7 ตัวอักษร

เช็กคำตอบกลับเข้าตารางเดิมครับ ลองแถวที่ 7 ($A = 1, B = 1, C = 1$): วงเล็บที่สาม $(\overline{A} + \overline{B}) = (0 + 0) = 0$ พอมีวงเล็บไหนเป็น 0 ผลคูณทั้งหมดก็เป็น 0 — ตรงกับตารางที่บอกว่าแถว 7 เป็น 0

ลองแถวที่ 1 ($A = 0, B = 0, C = 1$): วงเล็บแรก $(0 + 0 + 1) = 1$ · วงเล็บสอง $(1 + 0) = 1$ · วงเล็บสาม $(1 + 1) = 1$ คูณกันได้ 1 — ตรงกับตารางที่บอกว่าแถว 1 เป็น 1

ทำไมเครื่องหมายถึงกลับข้าง

สังเกตไหมครับว่าวง ① บนแผนที่อ่านได้ AB แต่ในคำตอบกลายเป็น $(\overline{A} + \overline{B})$ — ตัวอักษรกลับข้างหมด และเครื่องหมายคูณกลายเป็นบวก

เพราะมันผ่านเดอมอร์แกนหนึ่งรอบไงครับ กฎเดอมอร์แกนบอกว่า $\overline{AB} = \overline{A} + \overline{B}$ — ขีดคลุมก้อนไหน เวลากระจายลงไป ต้องกลับทั้งเครื่องหมายและตัวอักษรพร้อมกัน

ทางลัด — ข้ามเดอมอร์แกนไปเลย

พอทำหลายข้อจะเห็นว่าผลลัพธ์มีแบบแผนตายตัวครับ อ่านวงเลข 0 ให้เป็นวงเล็บได้เลยโดยไม่ต้องเขียน $\overline{X}$

กฎของวงเลข 0: มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน เหมือนเดิมทุกอย่าง แต่ **ค่าคงที่เป็น 0 → เขียนตัวเปล่า** · **ค่าคงที่เป็น 1 → เขียนแบบมีขีด** แล้วเชื่อมด้วย + ใส่วงเล็บ
จำสั้น ๆ ว่า SOP “เห็นศูนย์ ใส่ขีด” · POS “**เห็นหนึ่ง ใส่ขีด**” — กลับกันพอดี

ลองใช้ทางลัดกับวง ③ ครับ: ช่อง m_0 มี $A=0, B=0, C=0$ ทั้งสามตัวคงที่และเป็น 0 จึงเขียนตัวเปล่าทั้งหมด ได้ $(A+B+C)$ — ตรงกับที่คำนวณยาว ๆ เมื่อกี้

เดินให้เห็นตอนพัง: ลืมกลับเครื่องหมาย

คนที่รีบมักวงเลข 0 ถูกหมด แล้วเขียนวงเล็บโดยใช้กฎของ SOP ติดมือ คือ ① เขียนเป็น $(A+B)$ แทนที่จะเป็น $(\overline{A}+\overline{B})$

คำตอบที่ได้คือ $(\overline{A}+\overline{B}+\overline{C})(B+C)(A+B)$ ซึ่งเป็น 1 ที่แถว 2, 3, 5, 6

แต่คำตอบที่ถูกเป็น 1 ที่แถว 1, 2, 4, 5 — **ผิดไป 4 แถวจาก 8 แถว** ครึ่งตารางเลยครับ ทั้งที่วงกลุ่มถูกหมด

เช็กลิสต์หา minimum POS

1. เติมแผนที่ให้ครบทั้ง 1 และ 0 (ต้องมีเลข 0 ครบทุกช่อง)
2. วงกลุ่ม**เลข 0** ด้วยกฎ 6 ข้อเดิมทุกข้อ — ใหญ่ที่สุด น้อยวงที่สุด ขอบต่อกัน
3. อ่านแต่ละวง: มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน
4. เขียนพจน์แบบ**กลับเครื่องหมาย** — เห็นหนึ่งใส่ขีด เชื่อมด้วย + ใส่วงเล็บ
5. เอาทุกวงเล็บมาคูณกัน (เขียนติดกัน)
6. เช็กกลับ: หยิบแถวที่เป็น 0 มาหนึ่งแถว แทนค่าแล้วต้องมีวงเล็บหนึ่งอันเป็น 0 พอดี **หยุดตรงนี้**

7 ช่องที่เขียนว่า X — don’t care

ของจริงก่อน: รหัสที่ไม่มีวันเกิดขึ้น

นาฬิกาดิจิทัลกับเครื่องคิดเลขแสดงตัวเลข 0 ถึง 9 ใช่ไหมครับ ข้างในมันเก็บเลขแต่ละหลักด้วยบิต 4 ตัว

บิต 4 ตัวนับได้ถึง 16 ค่า คือ 0000 ถึง 1111 แต่เลขหลักหนึ่งหลักมีแค่ 0–9 เพราะฉะนั้นรหัส 1010 · 1011 · 1100 · 1101 · 1110 · 1111 (คือค่า 10 ถึง 15) **ไม่มีวันปรากฏ**ที่ขาเข้าของวงจรนี้เลย

การเก็บเลขหลักแบบนี้เรียกว่า BCD code ครับ และหกรหัสที่ใช้ไม่ได้นั้นคือใจความทั้งหมดของหัวข้อนี้

คำถามคือ ถ้าอินพุตหนึ่งไม่มีวันเกิด เราจะออกแบบให้เอาต์พุตของมันเป็นอะไร — คำตอบคือ **อะไรก็ได้** เพราะไม่มีใครมาเห็น

ในตารางค่าความจริง แถวพวกนี้เขียนว่า **X** แทนที่จะเขียน 0 หรือ 1 เรียกว่าเงื่อนไข don’t care ครับ ภาษาไทยคือ “ไม่สน”

โจทย์จะเขียนมายังไง

$F(A,B,C,D) = \sum m(7,8,9) + \sum d(10,11,12,13,14,15)$

ก้อน $\sum m$ คือช่องที่**ต้องเป็น 1** · ก้อน $\sum d$ คือช่องที่**เป็น X** · ที่เหลือเป็น 0 ทั้งหมดครับ

โจทย์ตัวอย่าง: วงจรเช็กว่าเลขหลักนี้มากกว่าหรือเท่ากับ 7 ไหม

ใช้โจทย์ข้างบนเลยครับ เอาต์พุตเป็น 1 เมื่อเลขที่เข้ามาคือ 7, 8 หรือ 9 ส่วนรหัส 10–15 เป็น X เพราะไม่มีวันเกิด

AB \ CD	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (4)	0 (5)	1 (7)	0 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

1 อยู่สามช่อง · X อยู่หกช่อง · ที่เหลือเป็น 0

ลองก่อน ไม่ต้องเขียน

มองแผนที่ข้างบนสองรอบครับ

รอบแรก — ถ้าถือว่า X ทุกตัวเป็น **0** วงใหญ่ที่สุดที่ได้คือกี่ช่อง

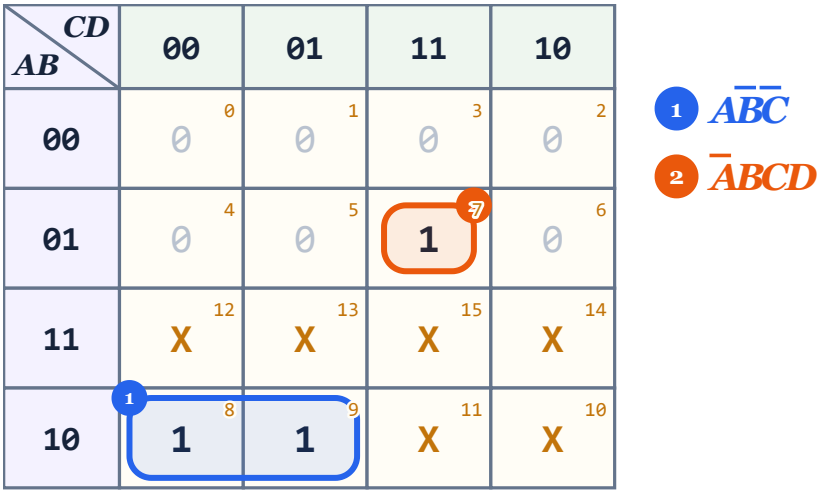
รอบสอง — ถ้าถือว่า X ทุกตัวเป็น **1** วงใหญ่ที่สุดที่ได้คือกี่ช่อง

เฉลย — รอบแรกได้แค่ 2 ช่อง (คู่ m_8, m_9) · รอบสองได้ถึง **8 ช่อง** เพราะแถว $AB = 11$ กับ $AB = 10$ กลายเป็น 1 หมดทั้งสองแถว

ส่วนต่าง 2 ช่อง กับ 8 ช่อง คือเหตุผลว่าทำไมเราถึงสนใจ X ครับ

ทำสองแบบเทียบกัน — แบบที่ไม่ใช้ X เลย

ถ้าถือว่า X เป็น 0 ทุกตัว เราจะมีเลข 1 แค่สามตัว คือ m_7, m_8, m_9



ถือว่า X เป็น 0 ทุกตัว — วงได้เล็กนิดเดียว

① คือคู่ m_8, m_9 · ② คือ m_7 ที่วงตัวเดียวเพราะไม่มีเพื่อนบ้านเป็น 1

อ่านสองวง

① แถว AB=10 คอลัมน์ 00,01 : A=1 · B=0 · C=0 · D เปลี่ยน → $A\overline{B}\,\overline{C}$

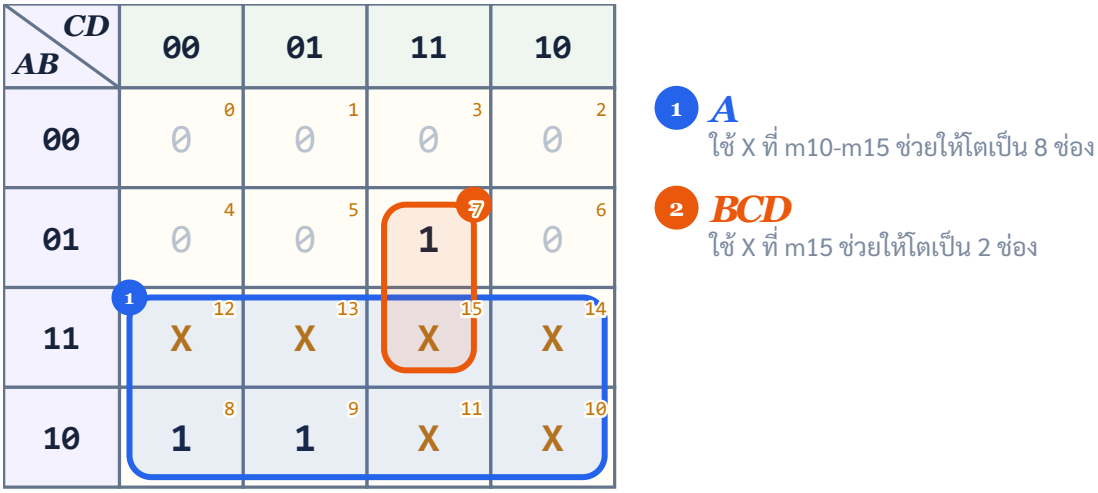
② ช่อง m7 ตัวเดียว : A=0 · B=1 · C=1 · D=1 → $\overline{A}BCD$

$$F = A\overline{B}\,\overline{C} + \overline{A}BCD$$

2 พจน์ 7 ตัวอักษร

แบบที่ใช้ X ช่วย

คราวนี้ให้สิทธิ์ตัวเองว่า X แต่ละตัวจะนับเป็น 1 ก็ได้ นับเป็น 0 ก็ได้ **เลือกทีละตัว** ให้วงโตที่สุด



ใช้ X เป็น 1 เฉพาะตัวที่ช่วยให้วงโตขึ้น

① กินครึ่งล่างของแผนที่ทั้ง 8 ช่อง · ② กินสองช่องโดยยืม X ที่ m_{15}

อ่านสองวงใหม่

① แถว AB=11 และ 10 ทั้งสองแถว : A=1 คงที่ · B,C,D เปลี่ยนหมด → A

② ช่อง m7 กับ m15 : A เปลี่ยน · B=1 · C=1 · D=1 → BCD

$$F = A + BCD$$

2 พจน์ 4 ตัวอักษร — จาก 7 เหลือ 4

เช็กว่าไม่พังครับ: พจน์ A ทำให้ช่อง m_8 ถึง m_{15} เป็น 1 ทั้งหมด ในนั้นมี 1 ที่ต้องการอยู่สองตัว (m_8, m_9) และที่เหลือเป็น X ซึ่งเป็นอะไรก็ได้ — ไม่ผิดกติกา

พจน์ BCD ทำให้ m_7 กับ m_{15} เป็น 1 · m_7 คือ 1 ที่ต้องการ · m_{15} เป็น X — ไม่ผิดเช่นกัน

และสำคัญที่สุด: ไม่มีพจน์ไหนไปแตะช่องที่เป็น **0** จริง ๆ เลย (m_0 ถึง m_6) นี่คือเงื่อนไขเดียวที่ห้ามละเมิด

เมื่อไรใช้ X เป็น 1 เมื่อไรใช้เป็น 0

กฎตัดสินมีข้อเดียวครับ: **ใช้ X เป็น 1 ถ้ามันทำให้วงโตขึ้น · นอกนั้นปล่อยเป็น 0**

ย้ำว่าเลือกได้**ทีละตัว** ไม่ใช่ต้องเลือกเหมือนกันทั้งแผนที่ — ในตัวอย่างนี้เราใช้ X ที่ m_{10} ถึง m_{15} เป็น 1 แต่ถ้ามี X ตัวไหนอยู่โดดเดี่ยว ก็ปล่อยให้เป็น 0 ไปเฉย ๆ

ตอนหา **POS** ก็หลักเดียวกันครับ แต่กลับด้าน — ตอนวงเลข 0 ถ้า X ตัวไหนช่วยให้วง 0 โตขึ้น ก็นับมันเป็น 0 ได้

เดินให้เห็นตอนพัง: วงที่มีแต่ X

คนที่เพิ่งรู้ว่า X ใช้เป็น 1 ได้ มักเผลอวงกลุ่มที่**ไม่มีเลข 1 จริงอยู่ในวงเลย**

เช่นวงคู่ m_{12}, m_{13} ซึ่งเป็น X ทั้งคู่ อ่านได้พจน์ $AB\overline{C}$ แล้วเอาไปบวกในคำตอบ

คำตอบจะยัง**ไม่ผิด**ครับ เพราะพจน์นี้ไปแตะแต่ช่อง X แต่มัน**เปลืองเปล่า ๆ** — เพิ่มเกต AND อีกตัวโดยไม่ได้ครอบ 1 สักตัว ข้อสอบที่ถามหา *minimum* จะหักคะแนนตรงนี้

AB \ CD	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (4)	0 (5)	1 (7)	0 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

1 $AB\overline{C}$
วงผิด — ในวงมีแต่ X ไม่มี 1 สักตัว

วงที่มีแต่ X — ถูกกฎรูปทรง แต่ไม่มีประโยชน์ ห้ามเขียนลงคำตอบ

เช็กลิสต์ don’t care

1. เติมแผนที่ให้ครบสามแบบ: 1 · 0 · X (อย่าเขียน X เป็น 0)
2. วงเลข 1 ตามปกติ แต่ตอนจะขยายวง ให้ถาม “มี X ติดอยู่ข้าง ๆ ไหม”
3. ถ้า X ทำให้วงโตขึ้นเป็นกำลังของสองถัดไป → ยืมมาใช้
4. X ที่ไม่ได้ช่วยอะไร → ปล่อยไว้ ไม่ต้องวง
5. ตรวจว่า**ทุกวงมีเลข 1 จริงอย่างน้อยหนึ่งตัว** ถ้าวงไหนมีแต่ X ให้ลบทิ้ง
6. ตรวจว่าไม่มีพจน์ไหนแตะช่องที่เป็น 0 จริง **หยุดตรงนี้**

8 โจทย์ทำให้ดู ทุกบรรทัด

สี่ข้อต่อจากนี้ทำแบบไม่ข้ามขั้นเลยครับ ตั้งแต่ตารางค่าความจริงจนถึงคำตอบ รวมถึงขั้นที่คิดในใจได้ก็ยังเขียนให้ดู

ข้อ ก · 4 ตัวแปร หา minimum SOP

โจทย์: $X(A,B,C,D) = \sum m(3,4,5,8,10,11,12,14,15)$ จงหา minimum SOP

ขั้นที่ 1 — กางเป็นตารางค่าความจริง

สัญลักษณ์ $\sum m$ บอกว่าช่องที่ระบุเป็น 1 ที่เหลือเป็น 0 กางออกมาได้ 16 แถวแบบนี้

A	B	C	D	X	แถวที่
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	2
0	0	1	1	**1**	3
0	1	0	0	**1**	4
0	1	0	1	**1**	5
0	1	1	0	0	6
0	1	1	1	0	7

A	B	C	D	X	แถวที่
1	0	0	0	**1**	8
1	0	0	1	0	9
1	0	1	0	**1**	10
1	0	1	1	**1**	11
1	1	0	0	**1**	12
1	1	0	1	0	13
1	1	1	0	**1**	14
1	1	1	1	**1**	15

นับเลข 1 ได้ 9 ตัว ตรงกับจำนวนตัวเลขในวงเล็บของโจทย์ — ผ่านขั้นแรก

ขั้นที่ 2 — วาดแผนที่แล้วเติมค่า

4 ตัวแปร ใช้ตาราง 4×4 หัวแถว AB หัวคอลัมน์ CD เรียง 00 01 11 10 ทั้งคู่

AB \ CD	00	01	11	10
00	0 (0)	0 (1)	**1** (3)	0 (2)
01	**1** (4)	**1** (5)	0 (7)	0 (6)
11	**1** (12)	0 (13)	**1** (15)	**1** (14)
10	**1** (8)	0 (9)	**1** (11)	**1** (10)

เติมครบ 16 ช่องแล้ว — เลข 1 เก้าตัว ตรงกับตาราง

ลองก่อน ไม่ต้องเขียน

ก่อนดูเฉลย ลองไล่สายตาหาว่ามีวง 4 ช่องกี่วง และอย่าลืมมองขอบซ้าย-ขวา กับ ขอบบน-ล่างด้วย

เฉลย — มีวง 4 ช่องสองวงครับ วงหนึ่งเป็นบล็อก 2×2 ที่มุมขวาล่าง อีกวงหนึ่งต้องข้ามขอบซ้าย-ขวา (คอลัมน์ 00 กับ 10 ของสองแถวล่าง) ที่เหลือเป็นวง 2 ช่องอีกสองวง

ขั้นที่ 3 — วงกลุ่มตามลำดับในเช็กลิสต์

หาวง 8 ช่องก่อน — ไม่มีครับ ไม่มีครึ่งแผนที่ไหนเป็น 1 หมด (แถว $AB = 00$ กับ 01 มี 0 เยอะ)

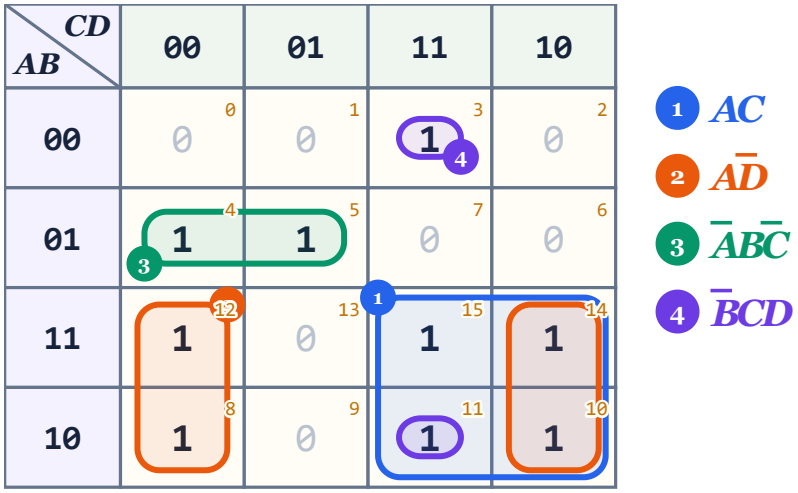
หาวง 4 ช่อง — เจอสองวง

วงแรกคือบล็อก 2×2 ที่แถว $AB = 11$ กับ 10 ตัดกับคอลัมน์ $CD = 11$ กับ 10 คือช่อง $m_{15}, m_{14}, m_{11}, m_{10}$ เป็น 1 ครบทั้งสี่

วงที่สองคือแถว $AB = 11$ กับ 10 ตัดกับคอลัมน์ $CD = 00$ กับ 10 — คอลัมน์สองอันนี้อยู่คนละฝั่งของแผนที่ ต้องใช้กฎขอบต่อกัน ช่องคือ $m_{12}, m_{14}, m_{8}, m_{10}$

เหลือ 1 ตัวไหนยังไม่ถูกวง — เหลือ m_4, m_5 และ m_3

m_4 กับ m_5 อยู่ติดกันที่แถว $AB = 01$ วงเป็นคู่ได้เลย · ส่วน m_3 อยู่แถวบนสุดคอลัมน์ $CD = 11$ เพื่อนบ้านที่เป็น 1 มีตัวเดียวคือ m_{11} ซึ่งอยู่แถวล่างสุด — ใช้กฎขอบบน-ล่างต่อกัน



② วงเดียวกัน ต่อกันรอบขอบซ้าย-ขวา

④ วงเดียวกัน ต่อกันรอบขอบบน-ล่าง

สี่วงครบแล้ว — ② และ ④ เป็นวงที่ต้องข้ามขอบ

ขั้นที่ 4 — อ่านทีละวง

มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน

① แถว 11,10 × คอลัมน์ 11,10

A: 1,1 คงที่ → A | B: 1,0 เปลี่ยน → ทิ้ง

C: 1,1 คงที่ → C | D: 1,0 เปลี่ยน → ทิ้ง = **AC**

② แถว 11,10 × คอลัมน์ 00,10

A: 1,1 คงที่ → A | B: 1,0 เปลี่ยน → ทิ้ง

C: 0,1 เปลี่ยน → ทิ้ง | D: 0,0 คงที่เป็น 0 → $\overline{D}$ = **A$\overline{D}$**

③ แถว 01 × คอลัมน์ 00,01

A: 0 → $\overline{A}$ | B: 1 → B | C: 0,0 → $\overline{C}$ | D: 0,1 เปลี่ยน → ทิ้ง = **$\overline{A}$B$\overline{C}$**

④ แถว 00,10 × คอลัมน์ 11

A: 0,1 เปลี่ยน → ทิ้ง | B: 0,0 → $\overline{B}$

C: 1 → C | D: 1 → D = **$\overline{B}$CD**

ขั้นที่ 5 — รวมคำตอบแล้วเช็กกลับ

$$X = AC + A\overline{D} + \overline{A}B\overline{C} + \overline{B}CD$$

4 พจน์ 10 ตัวอักษร

เช็กว่าครอบ 1 ครบครับ: $AC \to \{10, 11, 14, 15\}$ · $A\overline{D} \to \{8, 10, 12, 14\}$ · $\overline{A}B\overline{C} \to \{4, 5\}$ · $\overline{B}CD \to \{3, 11\}$

รวมทั้งหมดได้ $\{3, 4, 5, 8, 10, 11, 12, 14, 15\}$ — ตรงกับโจทย์เป๊ะ 9 ตัว และไม่มีตัวไหนหลุดไปโดนช่อง 0

ตรวจด้วยกฎขนาดวง

วง ① 4 ช่อง → พจน์ต้องมี 2 ตัวอักษร (AC ✓) · วง ② 4 ช่อง → 2 ตัวอักษร ($A\overline{D}$ ✓) · วง ③ 2 ช่อง → 3 ตัวอักษร ($\overline{A}B\overline{C}$ ✓) · วง ④ 2 ช่อง → 3 ตัวอักษร ($\overline{B}CD$ ✓) — ผ่านหมด

ข้อ ข · 4 ตัวแปร หาทั้ง minimum SOP และ minimum POS

โจทย์: $X(A,B,C,D) = \sum m(0,1,2,3,5,8,9,10,11,14,15)$ จงหา minimum SOP และ minimum POS

ขั้นที่ 1 — กางเป็นตารางค่าความจริง

A	B	C	D	X	แถวที่
0	0	0	0	**1**	0
0	0	0	1	**1**	1
0	0	1	0	**1**	2
0	0	1	1	**1**	3
0	1	0	0	0	4
0	1	0	1	**1**	5
0	1	1	0	0	6
0	1	1	1	0	7

A	B	C	D	X	แถวที่
1	0	0	0	**1**	8
1	0	0	1	**1**	9
1	0	1	0	**1**	10
1	0	1	1	**1**	11
1	1	0	0	0	12
1	1	0	1	0	13
1	1	1	0	**1**	14
1	1	1	1	**1**	15

นับ 1 ได้ 11 ตัว และ 0 ได้ 5 ตัว รวม 16 แถวครบ

สังเกตก่อนลงมือ

ข้อนี้มี 1 เยอะ (11 ตัว) และ 0 น้อย (5 ตัว) ครับ เดาได้เลยว่าตอนหา POS จะง่ายกว่า เพราะวง 0 แค่ 5 ตัว ส่วนตอนหา SOP ต้องวงให้ครบ 11 ตัว

ขั้นที่ 2 — วาดแผนที่แล้วเติมค่า

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	1 (2)
01	0 (4)	1 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

แถว $AB = 00$ กับ $AB = 10$ เป็น 1 หมดทั้งแถว — จุดนี้น่าสนใจ

ขั้นที่ 3 — วงกลุ่มหา SOP

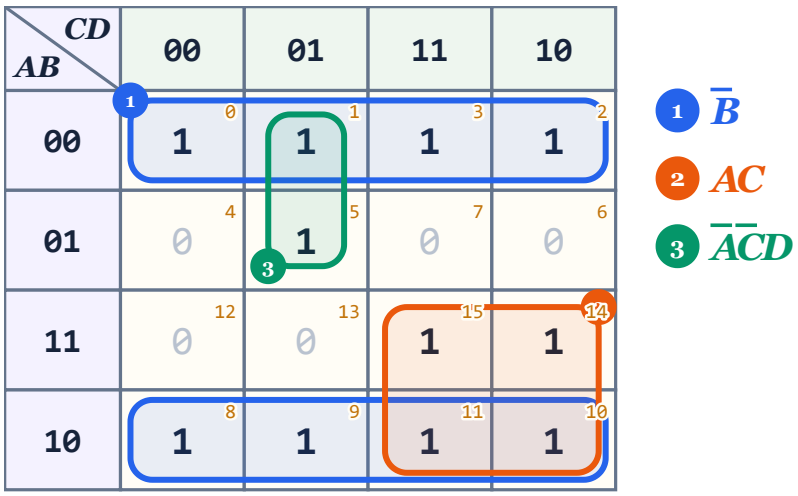
หาวง 8 ช่องก่อน — คราวนี้มีครับ แถวบนสุด ($AB = 00$) เป็น 1 ครบ 4 ช่อง และแถวล่างสุด ($AB = 10$) ก็เป็น 1 ครบ 4 ช่องเหมือนกัน

สองแถวนี้ติดกันด้วยกฎขอบบน-ล่าง เพราะฉะนั้นรวมกันเป็นวงเดียว 8 ช่องได้เลย

หาวง 4 ช่อง — เหลือ 1 ที่ยังไม่ถูกวงคือ m_5, m_{14}, m_{15}

m_{14} กับ m_{15} อยู่ที่แถว $AB = 11$ คอลัมน์ $CD = 11$ และ 10 ข้างล่างมันคือ m_{11} กับ m_{10} ซึ่งเป็น 1 อยู่แล้ว จึงวงเป็นบล็อก 2×2 ได้ 4 ช่อง (ทับกับวง 8 ช่องก็ไม่ผิด)

เหลือ m_5 ตัวเดียว — เพื่อนบ้านของมันคือ m_4 (เป็น 0) · m_7 (เป็น 0) · m_{13} (เป็น 0) · และ m_1 (เป็น 1) จึงวงได้แค่คู่เดียวคือ m_1 กับ m_5



① วงเดียวกัน ต่อกันรอบขอบบน-ล่าง (8 ช่อง)

สามวง — ① 8 ช่องข้ามขอบบน-ล่าง · ② บล็อก 2×2 · ③ คู่เดียวที่เป็นไปได้ของ m_5

อ่านทีละวง

① แถว 00,10 × ทุกคอลัมน์

A: 0,1 เปลี่ยน → ทิ้ง | B: 0,0 คงที่เป็น 0 → $\overline{B}$

C,D: กินครบทุกคอลัมน์ → เปลี่ยนทั้งคู่ → ทิ้ง = $\overline{B}$

② แถว 11,10 × คอลัมน์ 11,10

A: 1,1 → A | B: 1,0 → ทิ้ง | C: 1,1 → C | D: 1,0 → ทิ้ง = **AC**

③ แถว 00,01 × คอลัมน์ 01

A: 0,0 → $\overline{A}$ | B: 0,1 → ทิ้ง | C: 0 → $\overline{C}$ | D: 1 → D = $\overline{A}\,\overline{C}D$

$$X = \overline{B} + AC + \overline{A}\,\overline{C}D$$

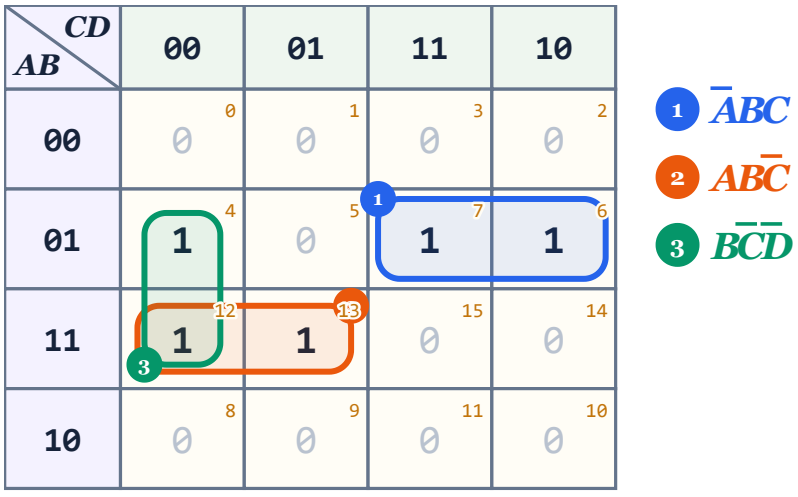
minimum SOP: 3 พจน์ 6 ตัวอักษร

เช็กกลับ: $\overline{B} \to \{0,1,2,3,8,9,10,11\} \cdot AC \to \{10,11,14,15\} \cdot \overline{A}\,\overline{C}D \to \{1,5\}$ รวมได้ $\{0,1,2,3,5,8,9,10,11,14,15\}$ ครบ 11 ตัวตามโจทย์ ไม่แตะช่อง 0 เลย

ขั้นที่ 4 — คราวนี้หา POS จากแผนที่เดิม

กลับไปที่แผนที่เดิม แล้วสลับสายตาไปมองเลข **0** ครับ มีอยู่ 5 ช่อง คือ $m_4, m_6, m_7, m_{12}, m_{13}$

วาดแผนที่ของ $\overline{X}$ ออกมาเลยจะเห็นชัดกว่า — เอา 0 กับ 1 สลับกันทั้งแผนที่



แผนที่ของ $\overline{X}$ — เลข 1 ห้าตัวนี้คือเลข 0 ห้าตัวของ X

วง 4 ช่อง — ไม่มีครับ ไล่ดูให้ครบทุกแบบแล้ว ทั้งบล็อก 2×2 · แถวเต็ม · คอลัมน์เต็ม · และแบบข้ามขอบ ไม่มีอันไหนที่เป็น 1 ครบสี่ช่อง

วง 2 ช่อง — m_6 กับ m_7 ติดกันที่แถว $AB = 01$ · m_{12} กับ m_{13} ติดกันที่แถว $AB = 11$ · เหลือ m_4 ที่ยังไม่ถูกวง

m_4 จับคู่ได้สองทางครับ ทางแรกคู่กับ m_6 — อยู่แถวเดียวกัน คอลัมน์ 00 กับ 10 ซึ่งต่อกันที่ขอบซ้าย-ขวา · ทางที่สองคู่กับ m_{12} — คอลัมน์เดียวกัน แถว 01 กับ 11 ที่ติดกันตรง ๆ

สองทางนี้ยาวเท่ากันเป๊ะ เลือกทางไหนก็ได้ครับ ในที่นี้ขอเลือกคู่ m_4 กับ m_{12} เพราะวาดง่ายกว่า (ไม่ต้องข้ามขอบ)

อ่านวงบนแผนที่ของ X บาร์ แล้วกลับเครื่องหมายทันที

① แถว 01 × คอลัมน์ 11,10 : A=0 · B=1 · C=1 · D เปลี่ยน

SOP อ่านได้ $\overline{A}BC$ → POS กลับเครื่องหมายเป็น **(A + $\overline{B}$ + $\overline{C}$)**

② แถว 11 × คอลัมน์ 00,01 : A=1 · B=1 · C=0 · D เปลี่ยน

SOP อ่านได้ $AB\overline{C}$ → POS เป็น **($\overline{A}$ + $\overline{B}$ + C)**

③ แถว 01,11 × คอลัมน์ 00 : A เปลี่ยน · B=1 · C=0 · D=0

SOP อ่านได้ $B\overline{C}\,\overline{D}$ → POS เป็น **($\overline{B}$ + C + D)**

$$X = (A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(\overline{B} + C + D)$$

minimum POS: 3 วงเล็บ 9 ตัวอักษร

เช็กกลับเข้าตารางครับ ลองแถวที่ 7 ($A = 0, B = 1, C = 1, D = 1$) ซึ่งตารางบอกว่าเป็น 0: วงเล็บแรก $(0 + 0 + 0) = 0$ — ผลคูณเป็น 0 ทันที ✓

ลองแถวที่ 9 ($A = 1, B = 0, C = 0, D = 1$) ซึ่งตารางบอกว่าเป็น 1: วงเล็บแรก $(1 + 1 + 1) = 1$ · วงเล็บสอง $(0 + 1 + 0) = 1$ · วงเล็บสาม $(1 + 0 + 1) = 1$ คูณกันได้ 1 ✓

คำตอบที่สั้นเท่ากันมีได้หลายแบบ

ถ้าน้องเลือกจับ m_4 คู่กับ m_6 แทน จะได้วงเล็บที่สามเป็น $(A + \overline{B} + D)$ แทน $(\overline{B} + C + D)$ — **ยาวเท่ากัน 3 ตัวอักษร และถูกทั้งคู่**

ข้อสอบแบบนี้มีคำตอบถูกมากกว่าหนึ่งแบบครับ ขอแค่จำนวนวงเล็บกับจำนวนตัวอักษรน้อยที่สุด ไม่ต้องกังวลว่าไม่ตรงกับเฉลย

ข้อ ค · 4 ตัวแปร มี don’t care

โจทย์: $F(A,B,C,D) = \sum m(0,1,2,3,6,8,9,10,11) + \sum d(5,7,12,14,15)$ จงหา minimum SOP

ขั้นที่ 1 — กางเป็นตารางค่าความจริง

ก้อน $\sum m$ เป็น 1 · ก้อน $\sum d$ เป็น X · ที่เหลือเป็น 0 — ข้อนี้จะเหลือ 0 จริงแค่สองแถว

A	B	C	D	F	แถวที่
0	0	0	0	1	0
0	0	0	1	1	1
0	0	1	0	1	2
0	0	1	1	1	3
0	1	0	0	0	4
0	1	0	1	X	5
0	1	1	0	1	6
0	1	1	1	X	7

A	B	C	D	F	แถวที่
1	0	0	0	1	8
1	0	0	1	1	9
1	0	1	0	1	10
1	0	1	1	1	11
1	1	0	0	X	12
1	1	0	1	0	13
1	1	1	0	X	14
1	1	1	1	X	15

นับดู: 1 มี 9 ตัว · X มี 5 ตัว · 0 มี 2 ตัว (แถวที่ 4 กับ 13) รวม 16 แถว ✓

ขั้นที่ 2 — วาดแผนที่

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	1 (2)
01	0 (4)	X (5)	X (7)	1 (6)
11	X (12)	0 (13)	X (15)	X (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

เขียน X ให้เป็น X อย่าเขียนเป็น 0 — ไม่งั้นจะเสียโอกาสขยายวง

ลองก่อน ไม่ต้องเขียน

มองคอลัมน์ $CD = 11$ กับ $CD = 10$ สองคอลัมน์ขวาของแผนที่ แล้วไล่ดูทีละช่องว่ามี 0 จริงอยู่ในนั้นไหม

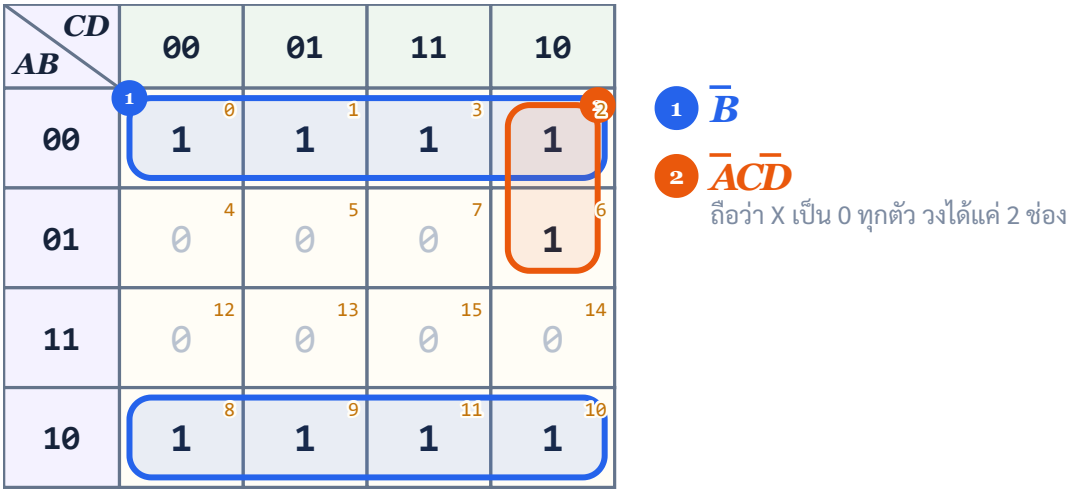
เฉลย — ไม่มีเลยครับ แปดช่องนั้นเป็น 1 หรือไม่ก็ X ทั้งหมด ($m_3, m_2, m_7, m_6, m_{15}, m_{14}, m_{11}, m_{10}$) แปลว่าเราวงรวดเดียว 8 ช่องได้ ซึ่งจะให้พจน์ที่มีตัวอักษรตัวเดียว

ขั้นที่ 3 — ทำแบบไม่ใช้ X ก่อน เพื่อให้เห็นส่วนต่าง

ถ้าถือว่า X เป็น 0 ทุกตัว เลข 1 จะมีเก้าตัวคือ $m_0, m_1, m_2, m_3, m_6, m_8, m_9, m_{10}, m_{11}$

แถวบนสุด ($AB = 00$) เป็น 1 ครบสี่ช่อง และแถวล่างสุด ($AB = 10$) ก็ครบสี่ช่อง รวมกันด้วยกฎขอบบน-ล่างได้วง 8 ช่อง

เหลือ m_6 ตัวเดียวที่ยังไม่ถูกวง เพื่อนบ้านของมันคือ m_2 (เป็น 1) · m_4 (เป็น 0) · m_7 (ตอนนี้ถือเป็น 0) · m_{14} (ตอนนี้ถือเป็น 0) จึงวงได้แค่คู่ m_2 กับ m_6



① วงเดียวกัน ต่อกันรอบขอบบน-ล่าง (8 ช่อง)

ไม่ใช้ X เลย — วง ② ได้แค่ 2 ช่อง จึงต้องเขียนถึง 3 ตัวอักษร

$$F = \overline{B} + \overline{A}C\overline{D}$$

2 พจน์ 4 ตัวอักษร

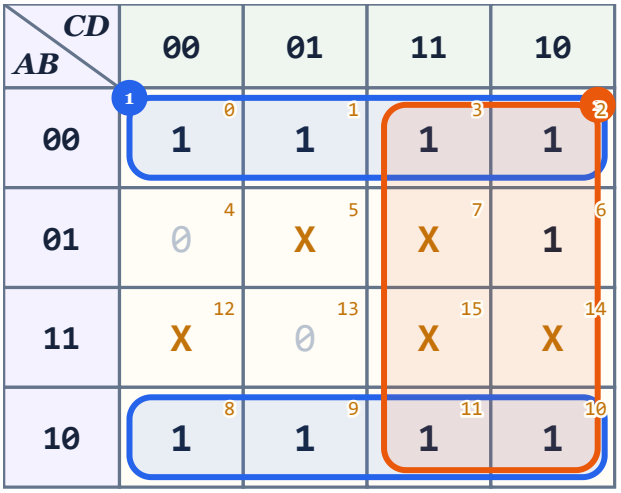
ขั้นที่ 4 — ทำใหม่โดยใช้ X ช่วย

คราวนี้ให้ X แต่ละตัวเลือกเองว่าจะเป็น 1 หรือ 0 ตามที่ทำให้วงโตที่สุด

วง ① เหมือนเดิมครับ — แถวบนสุดกับแถวล่างสุด 8 ช่อง ให้พจน์ $\overline{B}$ (วงนี้ไม่ต้องพึ่ง X เลยเพราะทั้งแปดช่องเป็น 1 อยู่แล้ว)

วง ② คือจุดที่เปลี่ยนครับ จากกล่อง “ลองก่อน” เราเห็นแล้วว่าสองคอลัมน์ขวา ($CD = 11$ และ $CD = 10$) ไม่มี 0 จริงสักช่อง

ในแปดช่องนั้นมี 1 จริงห้าตัว ($m_3, m_2, m_6, m_{11}, m_{10}$) และ X สามตัว ($m_7, m_{15}, m_{14}$) — ยืม X สามตัวนั้นมาเป็น 1 แล้ววงรวดเดียว 8 ช่องได้



① วงเดียวกัน ต่อกันรอบขอบบน-ล่าง (8 ช่อง)

1 $\overline{B}$

2 C

ใช้ X ที่ m7 m14 m15 ช่วยให้วงโตเป็น 8 ช่อง

ยืม X สามตัว ทำให้วง ② โตจาก 2 ช่องเป็น 8 ช่อง

อ่านสองวง

① แถว 00,10 × ทุกคอลัมน์ : B คงที่เป็น 0 ตัวเดียว = $\overline{B}$

② ทุกแถว × คอลัมน์ 11,10

A: กินครบทุกแถว → ทิ้ง | B: กินครบทุกแถว → ทิ้ง

C: 1,1 คงที่ → C | D: 1,0 เปลี่ยน → ทิ้ง = C

$$F = \overline{B} + C$$

2 พจน์ 2 ตัวอักษร — จาก 4 ตัวอักษรเหลือ 2

ขั้นที่ 5 — เช็กว่าไม่ละเมิดช่อง 0 จริง

ข้อนี้ต้องเช็กให้ดีครับ เพราะเรายืม X มาใช้ ต้องมั่นใจว่าไม่เผลอไปคลุมช่อง 0 จริง

ช่อง 0 จริงมีสองช่อง คือ m_4 $(A{=}0, B{=}1, C{=}0, D{=}0)$ และ m_{13} $(A{=}1, B{=}1, C{=}0, D{=}1)$

แทนค่าช่อง 0 จริงลงในคำตอบ

ที่ m4 : $\overline{B} = \overline{1} = 0$ | C = 0 → F = 0 + 0 = **0** ✓

ที่ m13 : $\overline{B} = \overline{1} = 0$ | C = 0 → F = 0 + 0 = **0** ✓

และเช็กว่าครอบ 1 ครบ: $\overline{B} \to \{0,1,2,3,8,9,10,11\} \cdot C \to \{2,3,6,7,10,11,14,15\}$

รวมสองก้อนแล้วมี 1 ที่ต้องการครบทั้งเก้าตัว $\{0,1,2,3,6,8,9,10,11\}$ ✓ ส่วนที่เกินมาคือ m_7, m_{14}, m_{15} ซึ่งเป็น X ทั้งหมด — เป็นอะไรก็ได้ ไม่ผิด

สรุปส่วนต่างของข้อนี้

ไม่ใช้ X → $F = \overline{B} + \overline{A}C\overline{D}$ ต้องใช้ AND 3 ขาหนึ่งตัว
ใช้ X → $F = \overline{B} + C$ ไม่ต้องใช้ AND เลย เหลือแค่ NOT กับ OR
นี่คือเหตุผลว่าทำไมต้องเขียน X ให้เป็น X ตั้งแต่ตอนเติมแผนที่ครับ

ข้อ ง · 3 ตัวแปร — จากนิพจน์ ไป K-map แล้วกลับมาเป็นนิพจน์ที่สั้นลง

โจทย์: จงย่อ $X = \overline{A} + A\overline{B} + AB\overline{C}$ ให้สั้นที่สุด

โจทย์แบบนี้ไม่ได้ให้ตารางมาครับ ให้นิพจน์มาเลย และนิพจน์นี้มีพจน์ที่**ตัวแปรไม่ครบ** (พจน์แรกมีตัวเดียว พจน์ที่สองมีสองตัว) — ต้องกางให้ครบก่อน

ขั้นที่ 1 — กางแต่ละพจน์ว่ากินช่องไหนบ้าง

ใช้กฎจากหัวข้อ 3: ขาดตัวแปรไป k ตัว กิน 2^k ช่อง

กางทีละพจน์

```
Ā  : ขาด B และ C ไป 2 ตัว → กิน 2² = 4 ช่อง
   ทุกช่องที่ A=0 คือ 000 001 010 011 → m0 m1 m2 m3

AB̄ : ขาด C ไป 1 ตัว → กิน 2¹ = 2 ช่อง
   ทุกช่องที่ A=1 B=0 คือ 100 101 → m4 m5

ABC̄ : ครบทุกตัวแปร → กิน 1 ช่อง
   A=1 B=1 C=0 คือ 110 → m6
```

รวมช่องที่เป็น 1 ได้ $\{0,1,2,3,4,5,6\}$ ครับ เจ็ดช่อง เหลือ m_7 ช่องเดียวที่เป็น 0

ขั้นที่ 2 — เขียนเป็นตารางค่าความจริง

A	B	C	X	แถวที่
0	0	0	**1**	0
0	0	1	**1**	1
0	1	0	**1**	2
0	1	1	**1**	3
1	0	0	**1**	4
1	0	1	**1**	5
1	1	0	**1**	6
1	1	1	0	7

ขั้นที่ 3 — วาดแผนที่

AB \ C	0	1
00	1 (0)	1 (1)
01	1 (2)	1 (3)
11	1 (6)	0 (7)
10	1 (4)	1 (5)

1 เจ็ดช่อง 0 หนึ่งช่อง — ช่องที่เป็น 0 คือ m_7 ที่แถว $AB = 11$ คอลัมน์ $C = 1$

ลองก่อน ไม่ต้องเขียน

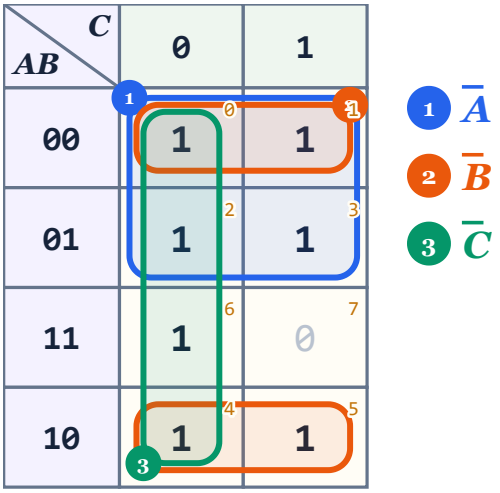
แผนที่นี้มีวง 4 ช่องได้กี่วงครับ (อย่าลืมว่าแถวบนสุดกับล่างสุดติดกัน)

เฉลย — สามวงครับ: สองแถวบน (00, 01) เป็นวงหนึ่ง · แถวบนสุดกับแถวล่างสุด (00, 10) เป็นอีกวง · และคอลัมน์ $C = 0$ ทั้งสี่แถวเป็นวงที่สาม
สามวงนี้ทับกันมั่วไปหมด แต่ไม่ผิดครับ ทับได้

ขั้นที่ 4 — วงกลุ่มแล้วอ่าน

หาวง 8 ช่องก่อน — ไม่ได้ครับ เพราะแผนที่มีแค่ 8 ช่องและมีหนึ่งช่องเป็น 0

หาวง 4 ช่อง — เจอสามวงตามที่ลองไว้ และสามวงนี้คลุม 1 ได้ครบทั้งเจ็ดตัวพอดี



สามวง วงละ 4 ช่อง — ② ต้องข้ามขอบบน-ล่าง

อ่านทีละวง

① แถว 00,01 × ทุกคอลัมน์ : A คงที่เป็น 0 · B เปลี่ยน · C เปลี่ยน = $\overline{A}$

② แถว 00,10 × ทุกคอลัมน์ : A เปลี่ยน · B คงที่เป็น 0 · C เปลี่ยน = $\overline{B}$

③ ทุกแถว × คอลัมน์ 0 : A เปลี่ยน · B เปลี่ยน · C คงที่เป็น 0 = $\overline{C}$

$$X = \overline{A} + \overline{B} + \overline{C}$$

จาก 6 ตัวอักษร เหลือ 3 ตัวอักษร

ขั้นที่ 5 — เช็กกลับ

คำตอบนี้เป็น 0 ก็ต่อเมื่อ $\overline{A}, \overline{B}, \overline{C}$ เป็น 0 พร้อมกันทั้งสามตัว คือตอน $A=1, B=1, C=1$ ซึ่งคือแถวที่ 7 เท่านั้น ตรงกับแผนที่ที่มี 0 อยู่ช่องเดียวคือ m_7 ✓ และเช็กกับนิพจน์โจทย์เดิมก็ได้ครับ: ที่ $A=1, B=1, C=1$ จะได้ $\overline{A} = 0 \cdot A\overline{B} = 1{\cdot}0 = 0 \cdot AB\overline{C} = 1{\cdot}1{\cdot}0 = 0$ รวมเป็น 0 เหมือนกัน ✓

สังเกตเพิ่มอีกนิด

$\overline{A} + \overline{B} + \overline{C}$ คือหน้าตาของ $\overline{ABC}$ ตามกฎเดอมอร์แกนพอดีครับ — แปลว่าวงจรทั้งหมดนี้คือ **NAND 3 ขา เข้าตัวเดียว** จากนิพจน์ตั้งต้นที่หน้าตาเหมือนต้องใช้เกตสี่ห้าตัว

9 เช็กลิสต์ในห้องสอบ

หน้านี้คือสิ่งที่ต้องทำจริงเมื่อเจอข้อสอบ K-map ครับ ทำตามลำดับนี้ทุกข้อ ไม่ต้องคิดใหม่

ลำดับที่ทำทุกข้อ

1 อ่านโจทย์ให้ชัดว่าขออะไร

SOP หรือ POS · มี don't care ไหม · กี่ตัวแปร

ขีดเส้นใต้คำว่า SOP หรือ POS ในข้อสอบ ก่อนลงมือ

2 วาดแผนที่เปล่าให้ถูกขนาด แล้วไล่หัวตารางปากเปล่า

3 ตัวแปร → 4×2 · 4 ตัวแปร → 4×4

ไล่ว่า "ศูนย์ศูนย์ · ศูนย์หนึ่ง · หนึ่งหนึ่ง · หนึ่งศูนย์" ทั้งหัวแถวและหัวคอลัมน์

3 เขียนเลข minterm ตัวเล็กไว้มุมช่องให้ครบทุกช่อง

เสียเวลา 20 วินาที แต่ตัดปัญหาการเติมผิดช่องออกไปทั้งหมด

4 เติมค่าลงทุกช่อง แล้วนับก่อนไปต่อ

เติมให้ครบทั้ง 1 · 0 · X (X ต้องเขียนเป็น X ห้ามเขียนเป็น 0)

นับจำนวน 1 เทียบกับโจทย์ — ตรงแล้วค่อยไปขั้นต่อไป

5 ถ้าโจทย์ขอ SOP วงเลข 1 · ถ้าขอ POS วงเลข 0

ไล่จากวงใหญ่ไปเล็ก: 8 ช่อง → 4 ช่อง → 2 ช่อง → 1 ช่อง

ตอนหาวง 4 ช่อง ให้ไล่ครบห้าแบบ: บล็อก 2×2 · แถวเต็ม · คอลัมน์เต็ม · ข้ามขอบ · สี่มุม

6 กวาดขอบทั้งสี่ด้าน แล้วลบวงที่ไม่จำเป็น

มี 1 ติดขอบไหม → ไปดูขอบตรงข้าม

มีวงไหนจมอยู่ในวงอื่นทั้งวงไหม → ลบทิ้ง

มีวงไหนที่ไม่มี 1 จริงเลย (มีแต่ X) ไหม → ลบทิ้ง

7 อ่านทีละวง: มองซ้าย มองบน หาตัวที่ไม่เปลี่ยน

SOP: เห็นศูนย์ใส่ขีด · คูณกัน · เชื่อมวงด้วย +

POS: **เห็นหนึ่งใส่ขีด** · บวกกันในวงเล็บ · เชื่อมวงด้วยการคูณ (เขียนติดกัน)

8

ตรวจสองชั้น แล้วหยุด

ชั้นแรก — นับตัวอักษร: วง 2^k ช่อง ต้องได้พจน์ที่เหลือ (จำนวนตัวแปร $-\ k$) ตัวอักษร

ชั้นสอง — สุ่มหยิบแถวที่เป็น 0 มาหนึ่งแถว แทนค่าลงคำตอบ ต้องได้ 0

ผ่านสองชั้นแล้วหยุดเลยครับ อย่าไปนั่งจัดพีชคณิตต่อ คำตอบจาก K-map สั้นที่สุดอยู่แล้ว

สามข้อที่เสียคะแนนมากที่สุด

ข้อ 1 · หัวตารางเรียงผิด

เขียน 00 01 **10 11** แบบนับเลขปกติ แทนที่จะเป็น 00 01 **11 10**

ทำไมแพง — ผิดตั้งแต่ก่อนเริ่ม ทุกอย่างหลังจากนั้นผิดตามหมด และมันมองไม่ออกตอนตรวจเพราะวงกลมดูสวยงามปกติ

กัน — ไล่ปากเปล่าหนึ่งรอบหลังเขียนหัวตาราง ก่อนเติมเลขลงช่อง

ข้อ 2 · ลืมว่าขอบแผนที่ต่อกัน

วงได้แค่ในกรอบสี่เหลี่ยมที่ตาเห็น ไม่ได้มองว่าซ้ายต่อขวา บนต่อล่าง และสี่มุมเป็นวงเดียวกัน

ทำไมแพง — คำตอบยัง**ถูก** แต่ยาวกว่าที่ควร ข้อที่เขียนว่า *minimum* จะได้ไม่เต็ม ในบทนี้มีตัวอย่างที่ยาวจาก 4 ตัวอักษรเป็น 12

กัน — หลังวงเสร็จ กวาดตาดูขอบทั้งสี่ด้านและสี่มุมเสมอ ใช้เวลา 10 วินาที

ข้อ 3 · ทำ POS แต่ลืมกลับเครื่องหมาย

วงเลข 0 ถูกหมด แต่เขียนวงเล็บด้วยกฎของ SOP ติดมือ เช่นวงที่ $A{=}1, B{=}1$ เขียนเป็น $(A + B)$ แทนที่จะเป็น $(\overline{A} + \overline{B})$

ทำไมแพง — คำตอบผิดทั้งข้อ ในตัวอย่างหัวข้อ 6 ผิดไป 4 แถวจาก 8 แถว ทั้งที่วงกลุ่มถูกทุกวง

กัน — ท่องว่า “SOP เห็นศูนย์ใส่ขีด · POS เห็นหนึ่งใส่ขีด” แล้วเช็กด้วยการแทนค่าแถวที่เป็น 0 หนึ่งแถว

การ์ดทวนก่อนเข้าห้องสอบ

ของ	จำแบบนี้
หัวตาราง	00 · 01 · 11 · 10
จำนวนช่อง	2^n ช่อง — 3 ตัวแปร 8 ช่อง · 4 ตัวแปร 16 ช่อง
ขนาดวง	1 · 2 · 4 · 8 · 16 เท่านั้น (ไม่มีวง 3)
วง 2^k ช่อง	ตัดตัวอักษรทิ้ง k ตัว
ขอบ	ซ้ายต่อขวา · บนต่อล่าง · สี่มุมเป็นวงเดียว
วงทับกัน	ทับได้ แต่ห้ามมีวงที่จมในวงอื่นทั้งวง
SOP	วงเลข 1 · เห็นศูนย์ใส่ขีด · คูณในพจน์ บวกระหว่างพจน์
POS	วงเลข 0 · เห็นหนึ่งใส่ขีด · บวกในวงเล็บ คูณระหว่างวงเล็บ
X	ใช้เป็น 1 ถ้าทำให้วงโตขึ้น · วงที่มีแต่ X ห้ามเขียน
ตรวจสุดท้าย	นับตัวอักษรตามขนาดวง + แทนค่าแถวที่เป็น 0 หนึ่งแถว

บทที่ 5 · วงจรผสมที่เอาไปใช้งานจริง

บทที่แล้วน้องรู้จักเกตทีละตัว รู้ว่า AND ให้ 1 เมื่อขาเข้าเป็น 1 ทั้งคู่ รู้ว่า XOR ให้ 1 เมื่อขาเข้าต่างกัน บทนี้เราจะเอาเกตพวกนั้นมา**ต่อกันเป็นเครื่องมือที่ทำงานจริง**ครับ

เครื่องมือที่ว่ามี 8 ชนิด และข้อสอบกลางภาคออกครบทั้ง 8 ชนิด พี่จะไล่ให้ตามลำดับที่มัน**ต่อยอดกันจริง ๆ** ไม่ใช่ตามลำดับในสไลด์ เพราะของบางอย่างต้องรู้ก่อนถึงจะเข้าใจของถัดไป

บทนี้มีอะไรบ้าง · และทำไมเรียงแบบนี้

หัวข้อ 1–4 เป็นสายเดียวกันทั้งหมด คือ**เครื่องบวกเลข** ที่ค่อย ๆ โตขึ้นทีละขั้น ใครข้ามหัวข้อ 1 ไปอ่านหัวข้อ 4 เลยจะไม่รอดครับ

1. **ตัวบวกครึ่ง (half-adder)** — บวกเลขหนึ่งบิตสองตัว
2. **ตัวบวกเต็ม (full-adder)** — เพิ่มขาให้รับตัวทดจากหลักก่อนหน้า
3. **ตัวบวกขนาน + ตัวทดไล่ (ripple carry)** — เอาตัวบวกเต็มมาเรียงกันหลายหลัก และมันช้าตรงไหน
4. **ตัวทดมองล่วงหน้า (look-ahead carry)** — แก้ความช้านั้น · **ยากที่สุดในบท ออกสอบแน่**
5. **ตัวลบครึ่ง / ตัวลบเต็ม** — เทียบกับตัวบวกทีละบรรทัด
6. **ตัวเปรียบเทียบ (comparator)** — เท่ากันไหม ใครมากกว่า
7. **ตัวเข้ารหัส (encoder)** — 10 ปุ่ม เหลือ 4 เส้น
8. **ตัวถอดรหัส (decoder)** — 3 เส้น แตกเป็น 8 เส้น
9. **ตัวถอดรหัสจอ 7 ขีด** — ทำให้เลขโผล่บนจอจริง
10. **ตัวแปลงรหัส binary ↔ Gray** — โซ่ XOR สองทิศ
11. **MUX / DEMUX** — สวิตช์เลือกทาง และ**เอา MUX มาสร้างฟังก์ชัน**
12. **เช็กลิสต์ปิดท้าย** — เห็นโจทย์แล้วรู้ว่าเป็นวงจรอะไร ต้องเขียนอะไรลงไป

ป้ายสีที่ใช้ในรูปทั้งบท

ฟ้า = ของที่ป้อนเข้าไป (ตัวตั้ง) · **ส้ม** = สิ่งที่กำลังทำอยู่ตอนนี้ · **เขียว** = ผลลัพธ์ที่ออกมา · **เทา** = ของที่พูดถึงไปแล้ว

ตัวเลขบนสายไฟ `1` คือไฟติด (HIGH) และ `0` คือไฟดับ (LOW)

ป้าย ①②③④ ในรูป บอกว่า**อะไรเกิดก่อนเกิดหลัง** ให้อ่านตามเลขเสมอครับ

1 บวกเลขหนึ่งบิตสองตัว ได้อะไรบ้าง

เริ่มจากเลขที่น้องบวกมาตั้งแต่ประถม

ลองบวก 7 + 5 ในใจครับ ได้ 12

แต่ตอนเขียนลงกระดาษ น้องเขียน 12 ลงในช่องเดียวไม่ได้ เพราะหนึ่งช่องใส่ได้แค่เลขหลักเดียว (0 ถึง 9) น้องเลยเขียน **2** ลงในช่อง แล้วเอา **1** ไป**ทด**ไว้บนหลักถัดไปทางซ้าย

สังเกตให้ดีนะครับว่าการบวกเลขหลักเดียว มันให้ผลออกมา**สองอย่างเสมอ** — อย่างแรกคือเลขที่เขียนลงช่อง อย่างที่สองคือเลขที่ทดไป

ในฐานสองก็เหมือนกันเป๊ะ ต่างแค่ช่องหนึ่งช่องใส่ได้แค่ 0 กับ 1 เท่านั้น พอเจอ 1 + 1 ซึ่งได้ 2 น้องก็เขียน 2 ลงช่องไม่ได้ ต้องเขียน 0 แล้วทด 1

ลองก่อน (คิดในหัวพอ ไม่ต้องเขียน)

เลขหนึ่งบิตมีได้แค่ 0 กับ 1 ใช่ไหมครับ ถ้าเอาสองตัวมาบวกกัน มันจะมีได้ทั้งหมด**กี่แบบ** และแต่ละแบบ **เขียนอะไรลงช่อง** กับ **ทดอะไรไป**

เฉลย: มี 4 แบบครับ — 0+0 ได้ 0 ไม่ทด · 0+1 ได้ 1 ไม่ทด · 1+0 ได้ 1 ไม่ทด · 1+1 ได้ 0 **ทด 1**
มีแบบเดียวเท่านั้นที่ต้องทด คือแบบสุดท้าย

ที่น้องเพิ่งไล่ในหัวครบ 4 แบบนั่นแหละครับ คือ**ตารางค่าความจริง**ของวงจรบวกตัวแรกของเรา ไม่ต้องไปท่องจากที่ไหน น้องสร้างมันขึ้นมาเองเมื่อกี้แล้ว

ตั้งชื่อของที่น้องเพิ่งหาได้

เลขสองตัวที่เอามาบวก เราเรียกว่า **A** กับ **B** ครับ

เลขที่เขียนลงช่อง เราเรียกว่า **ผลบวก** และเขียนแทนด้วยสัญลักษณ์ **Σ** (อ่านว่า ซิกมา) สไลด์ของอาจารย์ใช้ตัวนี้ตลอดทั้งบท น้องจะเห็นมันบ่อยมาก

เลขที่ทดไปหลักถัดไป เราเรียกว่า **ตัวทดออก** เขียนว่า $\mathbf{C_{out}}$ (ย่อจาก carry out) คำว่า carry แปลตรงตัวว่า "หิ้วไป" ซึ่งก็คือการทดนั่นเอง

ส่วนวงจรที่รับ A กับ B แล้วคาย Σ กับ C_{out} ออกมา มีชื่อว่า **ตัวบวกครึ่ง** หรือ **half-adder** ครับ

ทำไมต้อง "ครึ่ง" ด้วย

เพราะมันบวกได้แค่**ครึ่งเดียวของงานจริง**ครับ — มันมีขาเข้าแค่ 2 ขา คือ A กับ B จึงรับ**ตัวทดที่วิ่งมาจากหลักทางขวา**ไม่ได้เลย
ตอนบวก 7+5 น้องได้ทด 1 ไปหลักถัดไปใช่ไหมครับ หลักถัดไปนั้นต้องบวกเลข**สามตัว** (เลขบน + เลขล่าง + ตัวที่ทดมา) แต่ตัวบวกครึ่งมีขาแค่สองขา มันรับไม่ไหว
เพราะฉะนั้นตัวบวกครึ่งใช้ได้กับ**หลักขวาสุดหลักเดียว**เท่านั้น เพราะหลักขวาสุดไม่มีใครทดมาให้ หัวข้อ 2 เราจะซ่อมข้อจำกัดนี้ครับ

ตารางค่าความจริง แล้วมองหาหน้าตาที่คุ้น

A	B	Σ	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

แถวเขียวคือแถว 1+1 ซึ่งเป็นแถวเดียวที่ต้องทด

ทีนี้ปิดคอลัมน์ C_{out} ไว้ก่อน แล้วดูแค่คอลัมน์ Σ ครับ — อ่านจากบนลงล่างได้ **0 1 1 0**

น้องเคยเห็นลำดับ 0 1 1 0 ที่ไหนมาก่อนไหมครับ นั่นคือ**ตารางของเกต XOR** เป๊ะ ๆ เลย (ออก 1 เมื่อขาเข้าต่างกัน)

ทีนี้ปิดคอลัมน์ Σ แล้วดูคอลัมน์ C_{out} บ้าง ได้ **0 0 0 1** ซึ่งคือ**ตารางของเกต AND** เป๊ะ ๆ อีกเหมือนกัน (ออก 1 เมื่อขาเข้าเป็น 1 ทั้งคู่)

สองบรรทัดนี้ต้องจำให้ขึ้นใจ

$$\Sigma = A \oplus B$$

ผลบวก = XOR

$$C_{out} = A \cdot B$$

ตัวทด = AND

เครื่องหมาย ⊕ คือ XOR ครับ ส่วนจุด · คือ AND (หลายที่เขียนติดกันเฉย ๆ ว่า AB ก็คืออันเดียวกัน)

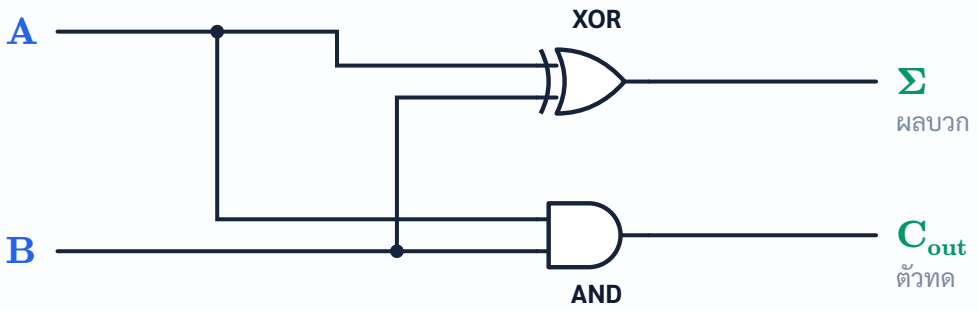
ทำไมมันถึงเป็น XOR กับ AND — ไม่ใช่เรื่องบังเอิญ

ผลบวกเป็น XOR เพราะ**ผลบวกจะเป็น 1 ก็ต่อเมื่อมีเลข 1 อยู่ตัวเดียว** — ถ้าไม่มี 1 เลยก็ได้ 0 ถ้ามี 1 ทั้งสองตัวก็ล้นเป็น 10 ซึ่งช่องนี้เหลือ 0 เหลือแค่กรณี "มี 1 ตัวเดียว" เท่านั้นที่ช่องนี้เป็น 1 และนั่นคือนิยามของ XOR พอดี

ตัวทดเป็น AND เพราะ**เราจะทดก็ต่อเมื่อมันล้น** และมันล้นได้กรณีเดียวคือ 1+1 ซึ่งแปลว่าต้องเป็น 1 ทั้งสองขา นั่นคือนิยามของ AND พอดี

พูดอีกแบบคือ เราไม่ได้ไปหยิบ XOR กับ AND มาลองต่อดูแล้วบังเอิญได้ แต่**ตารางมันบังคับ**ว่าต้องเป็นสองตัวนี้เท่านั้น

วงจรจริง



ตัวบวกครึ่ง — สาย A กับ B แต่ละเส้นแยกออกเป็นสองทาง (จุดดำคือจุดที่สายต่อถึงกันจริง) ทางหนึ่งเข้า XOR ได้ผลบวก อีกทางเข้า AND ได้ตัวทด · เกตสองตัวนี้ทำงาน**พร้อมกัน** ไม่ได้รอกัน

เดินตัวอย่างให้ดูทุกบรรทัด — ป้อน A = 1, B = 1

ขั้น	ทำอะไร	ได้อะไร
①	ป้อน A = 1 เข้าที่สายบน	สายบนเป็น `1`
②	ป้อน B = 1 เข้าที่สายล่าง	สายล่างเป็น `1`
③	เกต XOR เห็นขาเข้า 1 กับ 1 — **เหมือนกัน**	Σ = `0`
④	เกต AND เห็นขาเข้า 1 กับ 1 — **เป็น 1 ทั้งคู่**	C_{out} = `1`

เช็กคำตอบกลับเข้าโจทย์: วงจรตอบว่า Σ = 0 และ C_{out} = 1 อ่านรวมกันเป็นเลขสองบิตได้ **10** ซึ่งในฐานสองคือเลข 2

แล้วโจทย์เดิมคือ 1 + 1 ซึ่งก็เท่ากับ 2 พอดีครับ ตรงกัน

กับดักข้อสอบ · เอาตัวบวกครึ่งไปใช้ผิดที่

สมมติน้องเอาตัวบวกครึ่งสองตัวมาเรียงกัน เพื่อบวกเลขสองบิต **11 + 01** (คือ 3 + 1 ซึ่งต้องได้ 4 หรือ 100) มาดูกันว่าพังตรงไหน

หลัก	วงจรทำอะไร	ผลที่ได้
บิต 0 (ขวา)	บวก 1 + 1 — ตัวบวกครึ่งรับได้พอดี	Σ = `0` · ทด `1` ออกมา
บิต 1 (ซ้าย)	ต้องบวก 1 + 0 + **ตัวที่ทดมา 1** แต่มีขาแค่ 2 ขา — **ตัวที่ทดมาไม่มีรูให้เสียบ** มันเลยคิดแค่ 1 + 0	Σ = `1` · ทด `0`

```
  1 1
+ 0 1
-------
  0 1 0   ← ผิด
```

วงจรตอบ 010 ซึ่งคือเลข 2 แต่คำตอบจริงคือ 100 ซึ่งคือเลข 4

พังเพราะ**ไม่มีขาให้ตัวทดเข้า** ไม่ใช่เพราะเกตทำงานผิดครับ นี่คือเหตุผลทั้งหมดที่หัวข้อ 2 ต้องมีอยู่

เช็กลิสต์ตัวบวกครึ่ง

1. **นับขาเข้าก่อน**
   เห็นโจทย์ให้มา 2 ขาเข้า (A, B) กับ 2 ขาออก (Σ, C_{out}) → เป็นตัวบวกครึ่งแน่นอน
2. **เขียนตาราง 4 แถวจากการบวกจริง**
   ไม่ต้องท่อง ให้บวก 0+0, 0+1, 1+0, 1+1 แล้วแยกเป็น "เขียนลงช่อง" กับ "ทด"
3. **อ่านคอลัมน์ออกมาเป็นนิพจน์**
   คอลัมน์ Σ ได้ 0110 = XOR · คอลัมน์ C_{out} ได้ 0001 = AND
4. **วาดเกตสองตัวขนานกัน แล้วหยุด**
   XOR หนึ่งตัวกับ AND หนึ่งตัว ใช้สาย A กับ B ร่วมกัน · **หยุดตรงนี้** อย่าเผลอเติม OR หรือเติมขาที่สาม เพราะนั่นเป็นของหัวข้อ 2 แล้ว

2 ตัวบวกเต็ม — เอาตัวบวกครึ่งสองตัวมาต่อกัน

กลับไปที่จุดที่เมื่อกี้พัง

ตอนท้ายหัวข้อที่แล้ว เราบวก 11 + 01 แล้ววงจรตอบผิด เพราะหลักซ้ายต้องบวก**เลขสามตัว** แต่วงจรมีขาให้เสียบแค่สองขา

ลองนึกถึงการบวกเลขยาว ๆ ด้วยมือดูครับ เช่น 178 + 265 — หลักขวาสุดน้องบวกเลข**สองตัว** (8 กับ 5) แต่หลักกลางน้องบวกเลข**สามตัว** คือ 7 กับ 6 แล้วบวกตัวที่ทดมาจากหลักขวาอีก 1

เพราะฉะนั้น**ทุกหลักยกเว้นหลักขวาสุด ต้องการเครื่องที่บวกเลขสามตัวได้**ครับ ไม่ใช่สองตัว

ลองก่อน (คิดในหัวพอ ไม่ต้องเขียน)

เลขหนึ่งบิตสามตัว แต่ละตัวเป็นได้แค่ 0 หรือ 1 ถ้าเอามาบวกกันทั้งสามตัว **ผลมากที่สุด**ที่เป็นไปได้คือเท่าไหร่ และผลนั้นต้องใช้**กี่บิต**ถึงจะเขียนได้

เฉลย: มากที่สุดคือ 1 + 1 + 1 = 3 ครับ
เลข 3 ในฐานสองคือ **11** ซึ่งใช้ 2 บิตพอดี
แปลว่าถึงจะเพิ่มขาเข้าเป็นสามขา แต่**ขาออกยังใช้แค่สองขาเหมือนเดิม** คือช่องผลบวกหนึ่งช่อง กับตัวทดอีกหนึ่ง — ไม่ต้องเพิ่มขาออกเลย

ที่น้องเพิ่งคิดออกนั่นแหละครับ คือเหตุผลว่าทำไมวงจรถัดไปถึงมี**3 ขาเข้า 2 ขาออก**

ตั้งชื่อของใหม่

ขาที่สามซึ่งรับตัวทดมาจากหลักทางขวา เรียกว่า **ตัวทดเข้า** เขียนว่า **C_{in}** (carry in) — คู่กับ C_{out} หรือตัวทดออก ที่เราตั้งชื่อไว้แล้วในหัวข้อ 1

วงจรที่รับ A, B, C_{in} แล้วคาย Σ กับ C_{out} ออกมา ชื่อว่า **ตัวบวกเต็ม** หรือ **full-adder** ครับ

ที่เรียกว่า "เต็ม" เพราะมันบวกได้ครบทุกอย่างที่หนึ่งหลักต้องการจริง ๆ ไม่เหลืออะไรค้างเหมือนตัวบวกครึ่ง

ตารางค่าความจริง 8 แถว

สามขาเข้า แต่ละขาเป็นได้ 2 ค่า จึงมี 2 × 2 × 2 = **8 แถว** พี่เรียงคอลัมน์แบบเดียวกับสไลด์ของอาจารย์ คือ C_{in} อยู่ซ้ายสุดครับ

C_{in}	A	B	Σ	C_{out}	เช็ก
0	0	0	0	0	0+0+0=0
0	0	1	1	0	0+0+1=1
0	1	0	1	0	0+1+0=1
0	1	1	0	1	0+1+1=2
1	0	0	1	0	1+0+0=1
1	0	1	0	1	1+0+1=2
1	1	0	0	1	1+1+0=2
1	1	1	1	1	1+1+1=3

คอลัมน์ขวาสุดพี่ใส่ไว้ให้น้อง**ตรวจเองได้ทุกแถว**ครับ ไม่มีแถวไหนที่ต้องเชื่อพี่ — บวกเลขตามที่เขียนไว้ แล้วเทียบกับ Σ กับ C_{out} ได้เลย

วิธีอ่านคอลัมน์เช็ก: ผลบวกที่ได้ ถ้าเป็น 0 หรือ 1 ก็ลงช่อง Σ ตรง ๆ ไม่ต้องทด ถ้าเป็น 2 คือ "10" จึงลง Σ = 0 แล้วทด 1 · ถ้าเป็น 3 คือ "11" จึงลง Σ = 1 แล้วทด 1

กฎง่าย ๆ ของคอลัมน์ C_{out}: มันเป็น 1 ทุกแถวที่มีเลข 1 **ตั้งแต่สองตัวขึ้นไป** ซึ่งมี 4 แถวพอดี (แถวที่ระบายเขียว)

ทำไมมันถึงเป็น "ตัวบวกครึ่งสองตัวต่อกัน"

เราจะบวกเลขสามตัว คือ A + B + C_{in} ใช้ไหมครับ แต่เครื่องที่เรามีอยู่แล้วบวกได้ทีละสองตัว

งั้นก็บวกทีละคู่สิครับ — เอา **A + B ก่อน** ด้วยตัวบวกครึ่งตัวที่หนึ่ง ได้ผลบวกออกมาหนึ่งค่า สมมติเรียกมันว่า s_1

แล้วเอา **s_1 + C_{in}** ต่อ ด้วยตัวบวกครึ่งอีกตัวหนึ่ง ผลบวกที่ได้จากตัวที่สองนี้แหละครับ คือ Σ ตัวจริงของเรา

แต่ตัวบวกครึ่งแต่ละตัวมันคาย**ตัวทด**ออกมาด้วยคนละหนึ่งตัวใช่ไหมครับ เราเลยมีตัวทดค้างอยู่สองตัว เรียกว่า c_1 (จากตัวแรก) กับ c_2 (จากตัวที่สอง)

คำถามคือ แล้ว C_{out} ตัวจริงควรเป็นอะไร ในเมื่อมีตัวทดโผล่มาสองตัว

คำตอบคือ OR — และนี่คือเหตุผลที่ใช้ OR ได้

C_{out} = c_1 **OR** c_2 ครับ แปลว่า "ทดออกไปหนึ่ง ถ้าตัวใดตัวหนึ่งบอกให้ทด"

แต่เดี๋ยวก่อน — ถ้าทั้ง c_1 และ c_2 เป็น 1 **พร้อมกัน** เราต้องทดเป็น 2 ไม่ใช่หรือ แล้ว OR ที่ให้แค่ 1 จะไม่ผิดเหรอ

ตรงนี้คือจุดสวยของมันครับ — **c_1 กับ c_2 ไม่มีทางเป็น 1 พร้อมกันได้เลย**

เพราะถ้า c_1 = 1 แปลว่า A กับ B เป็น 1 ทั้งคู่ ซึ่งทำให้ s_1 = 0

และถ้า s_1 = 0 ตัวบวกครึ่งตัวที่สองก็ทดไม่ได้เลย (0 บวกอะไรก็ไม่ล้น) จึงได้ c_2 = 0 เสมอ

พอไม่มีทางเป็น 1 พร้อมกัน OR จึงให้คำตอบถูกทุกกรณี ไม่ต้องมีวงจรบวกซ้อนอีกชั้น

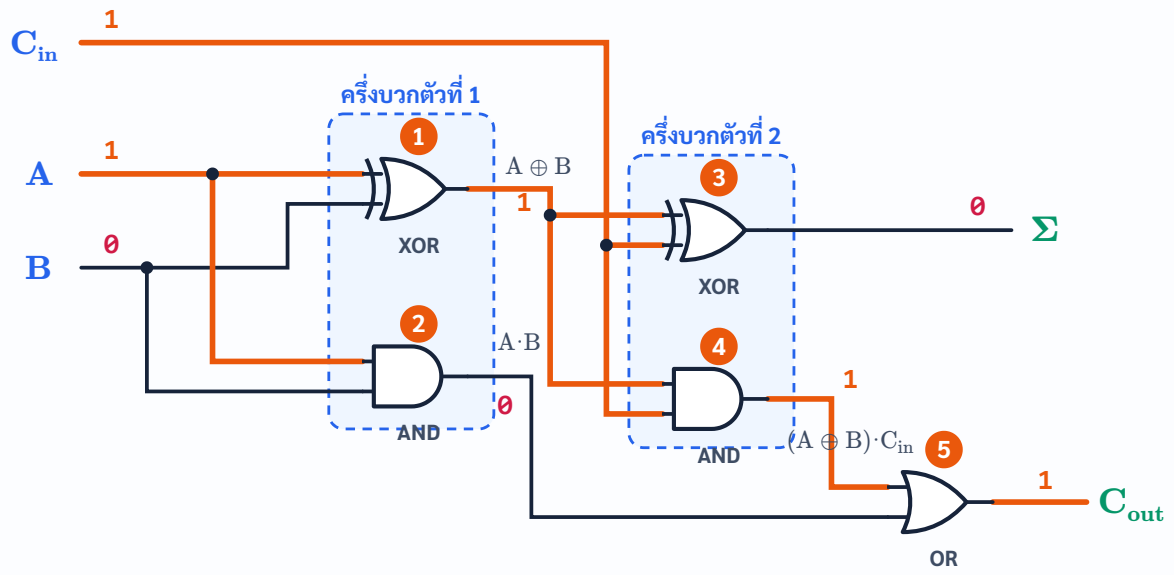
สรุปออกมาเป็นนิพจน์ที่อาจารย์เขียนไว้ในสไลด์ ซึ่งก็คือสิ่งเดียวกับที่เราเพิ่งไล่มาทั้งหมดครับ

$$\Sigma = (A \oplus B) \oplus C_{in}$$

ผลบวกของครึ่งบวกตัวที่สอง

$$C_{out} = A{\cdot}B + C_{in}{\cdot}(A \oplus B)$$

วงจรจริง — พร้อมเดินตัวอย่าง $C_{in} = 1, A = 1, B = 0$



ตัวบวกเต็ม = ครึ่งบวก 2 ตัว (กรอบฟ้า) + OR 1 ตัว · **เส้นส้ม**คือสายที่มีค่าเป็น 1 ในตัวอย่าง C_{in}=1, A=1, B=0 · ตัวเลขในวงกลมส้มคือลำดับที่เกตแต่ละตัวให้คำตอบ

ขั้น	ทำอะไร	ได้อะไร
①	XOR ตัวแรกเห็น A = 1 กับ B = 0 — **ต่างกัน**	A ⊕ B = `1`
②	AND ตัวแรกเห็น A = 1 กับ B = 0 — **ไม่ได้เป็น 1 ทั้งคู่**	A·B = `0`
③	XOR ตัวที่สองเห็น 1 (จากขั้น ①) กับ C_{in} = 1 — **เหมือนกัน**	Σ = `0`
④	AND ตัวที่สองเห็น 1 (จากขั้น ①) กับ C_{in} = 1 — **เป็น 1 ทั้งคู่**	(A⊕B)·C_{in} = `1`
⑤	OR เห็น 0 (จากขั้น ②) กับ 1 (จากขั้น ④) — **มีตัวหนึ่งเป็น 1**	C_{out} = `1`

เช็กคำตอบกลับเข้าโจทย์: วงจรตอบ Σ = 0 กับ C_{out} = 1 อ่านรวมกันได้ **10** ซึ่งคือเลข 2

โจทย์คือ C_{in} + A + B = 1 + 1 + 0 ซึ่งเท่ากับ 2 พอดีครับ ตรงกัน และถ้าน้องเปิดตาราง 8 แถวข้างบนไปดูแถว C_{in}=1, A=1, B=0 ก็เขียนไว้ว่า Σ=0, C_{out}=1 เหมือนกันเป๊ะ

กับดักข้อสอบ · เผลอใส่ AND แทน OR ตรงเกตปิดท้าย

ตอนวาดวงจรจากความจำ หลายคนจำได้ว่า "ตัวทดเกิดจาก AND" มาตลอดตั้งแต่หัวข้อ 1 พอถึงเกตปิดท้ายเลยเผลอวาด AND อีกตัว ลองดูว่าเกิดอะไรขึ้นครับ

เราพิสูจน์ไปแล้วว่า c_1 กับ c_2 **ไม่มีทางเป็น 1 พร้อมกัน** แปลว่าอย่างน้อยหนึ่งตัวต้องเป็น 0 เสมอ

ถ้าใช้ AND ปิดท้าย ผลคือ $c_1 \cdot c_2$ ซึ่ง**เป็น 0 ตลอดกาล** — วงจรจะไม่ทดอะไรเลยสักครั้ง ในตัวอย่างเมื่อกี้จะได้ $C_{out} = 0$ ทั้งที่คำตอบจริงคือ 1

วิธีจำ: เกตปิดท้ายของตัวบวกเต็มคือ **OR เสมอ** และตัวบวกเต็มหนึ่งตัวใช้เกตทั้งหมด 5 ตัว คือ XOR 2 · AND 2 · OR 1

เช็กลิสต์ตัวบวกเต็ม

1. **นับขาเข้าที่โจทย์ให้มาก่อน**
   3 ขา (A, B, C_{in}) = ตัวบวกเต็ม · 2 ขา = ตัวบวกครึ่ง · ขาออกเป็น 2 ขาเท่ากันทั้งคู่

2. **ถ้าโจทย์ให้เขียนตาราง ให้ไล่ 8 แถวแล้วบวกจริง**
   เขียนคอลัมน์ "ผลบวกเป็นเลขอะไร" ไว้ข้าง ๆ ก่อน แล้วค่อยแปลงเป็น Σ กับ C_{out} ทีหลัง

3. **ถ้าโจทย์ให้วาดวงจร ให้วาดครึ่งบวกทีละตัว**
   ตัวที่ 1 รับ A กับ B · ตัวที่ 2 รับผลบวกของตัวที่ 1 กับ C_{in} · แล้วค่อยเอา OR มาเก็บตัวทดสองตัวปิดท้าย

4. **ตรวจก่อนส่ง แล้วหยุด**
   นับเกตให้ได้ 5 ตัว · เกตสุดท้ายต้องเป็น OR · **หยุดตรงนี้** อย่าเพิ่งลากสายต่อไปหลักถัดไป นั่นเป็นเรื่องของหัวข้อ 3 ครับ

3 เอาตัวบวกเต็มมาเรียงกัน — และมันช้าตรงไหน

กลับไปดูมือตัวเองตอนบวกเลข

เวลาน้องบวก 178 + 265 ด้วยมือ น้องเริ่มจากหลักไหนครับ — หลัก**ขวาสุด**เสมอ

แล้วทำไมต้องเริ่มขวาสุด ทำไมไม่เริ่มซ้ายเลย เพราะหลักซ้าย**ยังตอบไม่ได้**จนกว่าจะรู้ว่าหลักขวาทดมาให้หรือเปล่าครับ

ลองสังเกตอีกนิด — ตอนน้องบวกหลักกลาง น้อง**รอ**ผลจากหลักขวาอยู่จริง ๆ ไม่ใช่ทำพร้อมกัน นี่คือหัวใจของหัวข้อนี้ทั้งหัวข้อเลยครับ

ลองก่อน (คิดในหัวพอ)

สมมติน้องมีตัวบวกเต็มอยู่ 4 ตัว และอยากบวกเลข 4 บิตสองจำนวน
คำถามที่ 1: ขา C_{out} ของตัวที่ดูแลหลักขวาสุด ควรต่อไปที่ขาไหนของตัวถัดไป
คำถามที่ 2: ขา C_{in} ของตัวที่ดูแล**หลักขวาสุด** ควรต่อกับอะไร ในเมื่อไม่มีหลักไหนอยู่ขวากว่ามันแล้ว

เฉลย:
ข้อ 1 — ต่อไปที่ขา C_{in} ของตัวถัดไปทางซ้ายครับ เพราะ "ทดไปหลักถัดไป" แปลตรงตัวว่าอย่างนี้เลย
ข้อ 2 — ต่อกับ **0** ครับ (ในวงจรจริงคือต่อลงกราวด์) เพราะหลักขวาสุดไม่มีใครทดมาให้
ที่น้องเพิ่งตอบไปสองข้อนี้ คือวิธีต่อวงจรบวกหลายบิตทั้งหมดแล้ว ไม่มีอะไรมากกว่านี้

ตั้งชื่อของที่น้องเพิ่งต่อ

วงจรที่เอาตัวบวกเต็มมาเรียงกันหลายตัว ตัวละหนึ่งหลัก เรียกว่า **ตัวบวกขนาน** หรือ **parallel binary adder** ครับ

คำว่า "ขนาน" หมายถึงบิตทุกบิตของ A และ B **ถูกป้อนเข้าไปพร้อมกันทีเดียว** ไม่ต้องป้อนทีละบิตเหมือนการพิมพ์ทีละตัว

ส่วนวิธีต่อตัวทดแบบที่น้องเพิ่งตอบ — คือให้ตัวทดไหลจากตัวขวาไปตัวซ้ายไปเรื่อย ๆ — มีชื่อเฉพาะว่า **ตัวทดแบบไล่** หรือ **ripple carry**

คำว่า ripple แปลว่า "ระลอกคลื่น" ครับ เหมือนโยนก้อนหินลงน้ำแล้วคลื่นค่อย ๆ แผ่ออกไปทีละวง ตัวทดในวงจรนี้ก็เดินทีละชั้นแบบนั้นเป๊ะ

จำนวนตัวบวกที่ต้องใช้

บวกเลข **n บิต** ใช้ตัวบวกเต็ม **n ตัว** ครับ — เลข 2 บิตใช้ 2 ตัว · เลข 4 บิตใช้ 4 ตัว
ผลลัพธ์ที่ได้จะมี **n + 1 บิต** เสมอ เพราะตัวทดที่ออกจากตัวซ้ายสุดคือบิตบนสุดของคำตอบ
(ถ้าใช้ตัวบวกครึ่งตรงหลักขวาสุดแทนก็ได้ เพราะ C_{in} ของหลักนั้นเป็น 0 อยู่แล้ว แต่ในสไลด์และข้อสอบจะใช้ตัวบวกเต็มทั้งแถวแล้วมัด C_{in} ตัวขวาสุดลงกราวด์ ซึ่งง่ายกว่า)

ทำโจทย์ให้ดู — 101 + 011 โดยให้ $C_{in} = 0$

นี่คือโจทย์เดียวกับในสไลด์ของอาจารย์ครับ เราจะเดินให้ครบทุกหลัก แล้วจะได้เอาไปเทียบกับวิธีเร็วในหัวข้อถัดไป

เลข 101 คือ 5 และ 011 คือ 3 ดังนั้นคำตอบที่ควรได้คือ 8 — เราจะเช็กตอนจบว่าตรงไหม

หลัก	ตัวบวกเต็มตัวไหน	รับอะไรเข้าไป	คายอะไรออกมา
บิต 0	FA0 (ขวาสุด)	$A_0=1$, $B_0=1$, $C_{in}=0$ → บวกได้ 2	Σ_0 = `0` , ทด C_0 = `1`
บิต 1	FA1	$A_1=0$, $B_1=1$, **$C_0=1$ ที่เพิ่งได้** → บวกได้ 2	Σ_1 = `0` , ทด C_1 = `1`
บิต 2	FA2 (ซ้ายสุด)	$A_2=1$, $B_2=0$, **$C_1=1$ ที่เพิ่งได้** → บวกได้ 2	Σ_2 = `0` , ทด C_2 = `1`
บิต 3	ไม่มีวงจรแล้ว	ตัวทดที่ออกจาก FA2 กลายเป็นบิตบนสุดของคำตอบเลย	Σ_3 = `1`

เขียนเป็นการบวกตั้งหลักแบบที่น้องคุ้นจะได้แบบนี้ครับ

ตัวเลขแถวบนสุดคือตัวทดที่วิ่งมาจากหลักขวา

```
  1 1 1
    1 0 1
+   0 1 1
-------------
  1 0 0 0
```

เช็กคำตอบกลับเข้าโจทย์: ได้ 1000 ซึ่งในฐานสองคือเลข 8

โจทย์คือ 5 + 3 ซึ่งเท่ากับ 8 พอดีครับ ตรงกัน

จุดที่อยากให้จำไว้ก่อนไปหัวข้อ 4

สังเกตคอลัมน์ "รับอะไรเข้าไป" ในตารางข้างบนนะครับ — FA1 ต้องใช้ค่า C_0 ซึ่ง**ยังไม่มีอยู่ตอนเริ่มต้น** มันต้องรอ FA0 คิดเสร็จก่อน

แล้ว FA2 ก็ต้องรอ C_1 ซึ่งต้องรอ FA1 อีกที

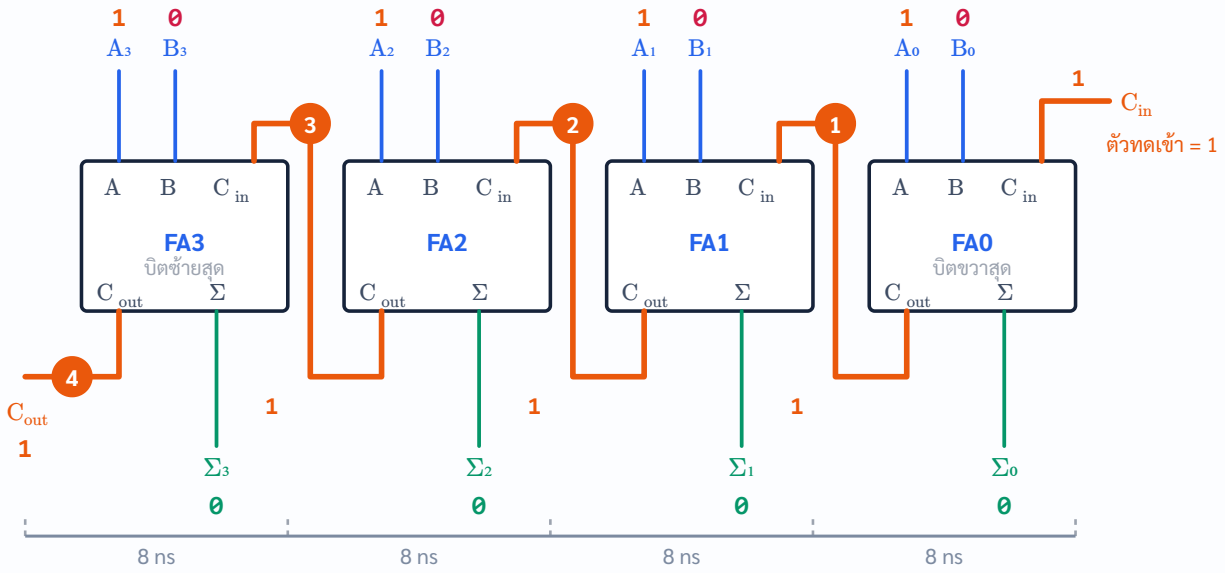
ทุกตัวรอตัวก่อนหน้าเป็นทอด ๆ — นั่นคือปัญหาที่หัวข้อ 4 จะมาแก้ครับ

วงจรจริง 4 บิต — และมาจับเวลากัน

ที่นี้มาดูกรณีที่แย่ที่สุด คือ **1111 + 0000** แล้วให้ C_{in} เป็น 1 (เป็นตัวอย่างในสไลด์ของอาจารย์)

กรณีนี้แย่ที่สุดเพราะตัวทด 1 ที่ป้อนเข้าทางขวา ต้อง**วิ่งทะลุทุกชั้นจนถึงซ้ายสุด** ไม่มีชั้นไหนหยุดมันได้เลย

สมมติให้ตัวบวกเต็มหนึ่งตัวใช้เวลา **8 นาโนวินาที (ns)** กว่าจะคายตัวทดออกมา ตัวเลขนี้พี่หยิบมาจากสไลด์เพื่อให้คิดง่ายครับ ของจริงแล้วแต่ชิป



ตัวบวกขนาน 4 บิตแบบตัวทดไล่ · **เส้นส้ม**คือตัวทดที่มีค่า 1 วิ่งจากขวาไปซ้าย ①②③④ คือลำดับเวลาที่มันไปถึงแต่ละชั้น · แถบเทาข้างล่างคือเวลาที่เสียไปต่อชั้น

ทีนี้มานับ "เวลารอ" กัน

ตอนเริ่มต้น เราป้อน A, B ครบทุกบิตพร้อมกัน และป้อน $C_{in} = 1$ เข้าทางขวาสุด ทุกอย่างมาถึงพร้อมกันหมด

แต่ FA0 ต้องใช้เวลา 8 ns กว่าจะคายตัวทดออกมา ระหว่างนั้น FA1 **นั่งรออยู่เฉย ๆ** ถึงมันจะมี A_1 กับ B_1 อยู่แล้วก็ตาม เพราะยังไม่รู้ว่า C_{in} ของตัวเองจะเป็นอะไร

พอ FA1 ได้ตัวทดตอน 8 ns มันก็ใช้เวลาอีก 8 ns ของตัวเอง ตัวทดจึงไปถึง FA2 ตอน **16 ns**

แล้ว FA2 ก็ใช้อีก 8 ns ตัวทดถึง FA3 ตอน **24 ns** · สุดท้าย FA3 ใช้อีก 8 ns คำตอบจึงนิ่งจริง ๆ ตอน **32 ns**

เวลา	ใครทำงานเสร็จ	ตัวทดไปถึงไหนแล้ว
0 ns	ป้อน A, B, C_{in} เข้าไปพร้อมกันหมด	ยังไม่ไปไหน
8 ns ①	FA0 เสร็จ	ถึง C_{in} ของ FA1
16 ns ②	FA1 เสร็จ	ถึง C_{in} ของ FA2
24 ns ③	FA2 เสร็จ	ถึง C_{in} ของ FA3
32 ns ④	FA3 เสร็จ	ออกมาเป็น C_{out} ตัวสุดท้าย

เช็กคำตอบกลับเข้าโจทย์: 1111 + 0000 + ตัวทดเข้า 1 = 15 + 0 + 1 = 16 และ 16 ในฐานสองคือ **10000** ซึ่งตรงกับ $\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = 0000$ บวกกับ $C_{out} = 1$ ข้างหน้าพอดีครับ

กับดักข้อสอบ · อ่านคำตอบเร็วเกินไป

ถ้ามีใครไปอ่านค่าที่ขา Σ_3 ตอน 10 ns จะเห็นอะไรครับ

ตอน 10 ns ตัวทดยังไปไม่ถึง FA3 เลย (มันเพิ่งถึง FA1 ตอน 8 ns) FA3 จึงยังคิดอยู่บนสมมติฐานว่า C_{in} ของมันเป็น 0 และตอบ $\Sigma_3 = 1$

แต่คำตอบที่ถูกคือ $\Sigma_3 = 0$ — **ค่าที่อ่านได้ตอนนั้นเป็นขยะ** ไม่ใช่เพราะวงจรผิด แต่เพราะยังคิดไม่เสร็จ

เพราะฉะนั้นเวลาข้อสอบถามว่า "วงจรนี้ใช้เวลาเท่าไหร่" ให้ตอบด้วย**กรณีแย่ที่สุด**เสมอ ไม่ใช่กรณีทั่วไป เพราะเราต้องรอให้มันนิ่งก่อนถึงจะอ่านได้

ตัวเลขที่ควรคิดได้ทันที

เวลารวมของตัวทดแบบไล่ = **(เวลาต่อชั้น) × (จำนวนบิต)**

4 บิต : 8 × 4 = **32 ns** · 8 บิต : 8 × 8 = **64 ns** · 32 บิต : 8 × 32 = **256 ns**

เห็นไหมครับว่ามัน**โตเป็นเส้นตรงตามจำนวนบิต** — ยิ่งเลขยาว ยิ่งช้าเป็นสัดส่วนกันเป๊ะ

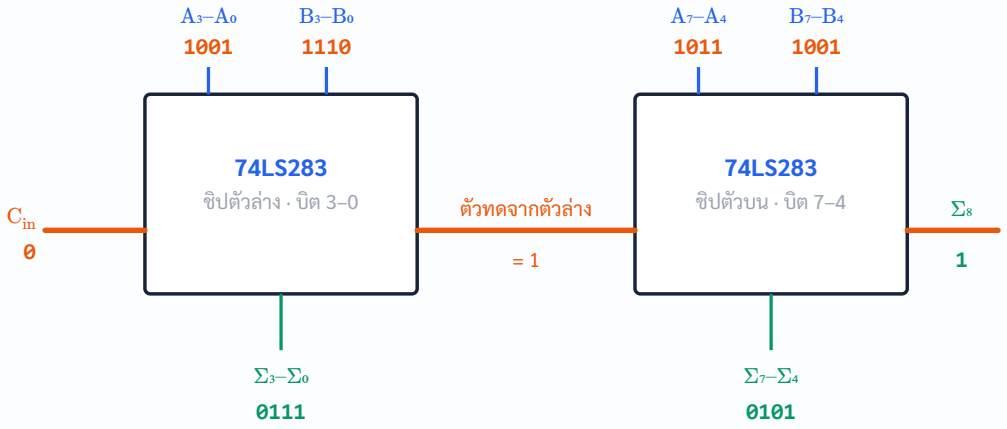
คอมพิวเตอร์จริงบวกเลข 64 บิต ถ้าใช้วิธีนี้จะช้าจนใช้งานไม่ได้ครับ นี่คือเหตุผลทางวิศวกรรมที่ทำให้ต้องมีหัวข้อ 4

ของจริงในห้องแลบ — ชิป 74LS283 และการต่อพ่วง

ตัวบวกขนาน 4 บิตมีขายเป็นชิปสำเร็จรูปชื่อ **74LS283** ครับ ข้างในคือสิ่งที่เราเพิ่งวาดกันทั้งหมด แต่ยัดมาในตัวถังเดียว

มันมีขา A_3–A_0 , B_3–B_0 , C_{in} เป็นขาเข้า และ Σ_3–Σ_0 กับ C_{out} เป็นขาออก

ถ้าอยากบวกเลข 8 บิต ก็ใช้สองตัว โดย**ต่อสายเส้นเดียว** คือเอา C_{out} ของตัวที่ดูแลบิตล่าง ไปเสียบขา C_{in} ของตัวที่ดูแลบิตบน — หลักการเดียวกับที่เราต่อตัวบวกเต็มเป๊ะ ๆ



ต่อพ่วงเป็นตัวบวก 8 บิต — สายส้มเส้นเดียวคือทั้งหมดที่เชื่อมสองชิปนี้ · ค่าที่แสดงคือตัวอย่าง 10111001 + 10011110 ข้างล่าง

ลองแทนตัวเลขจริงดูครับ โจทย์ในสไลด์คือ A = **10111001** และ B = **10011110**

ทำอะไร	รายละเอียด	ผล
แปลงเป็นเลขฐานสิบก่อน (ไว้เช็กตอนท้าย)	10111001 = 185 · 10011110 = 158	คำตอบควรได้ 343
ชิปตัวล่าง	บิต 3–0 ของทั้งสองจำนวน คือ 1001 + 1110 โดย C_{in} = 0 → 9 + 14 = 23	Σ_3–Σ_0 = **0111** · ทด **1**
สายเชื่อม	ตัวทด 1 วิ่งจาก C_{out} ของตัวล่าง ไปเข้า C_{in} ของตัวบน	ตัวบนได้ C_{in} = **1**
ชิปตัวบน	บิต 7–4 คือ 1011 + 1001 บวกตัวทดอีก 1 → 11 + 9 + 1 = 21	Σ_7–Σ_4 = **0101** · ทด **1** = Σ_8

เช็กคำตอบกลับเข้าโจทย์: ต่อผลทั้งหมดเข้าด้วยกันได้ Σ_8...Σ_0 = **1 0101 0111**

แปลงกลับเป็นฐานสิบได้ 256 + 64 + 16 + 4 + 2 + 1 = **343** และ 185 + 158 ก็ได้ 343 พอดีครับ

เช็กลิสต์ตัวบวกขนาน / ตัวทดไล่

1. **ดูว่าบวกกี่บิต แล้ววางกล่องให้ครบ**
   n บิต = ตัวบวกเต็ม n ตัว · เรียงให้บิตขวาสุดอยู่**ขวาสุด** ตามที่สไลด์วาด
2. **ต่อโซ่ตัวทด แล้วมัดปลายทางขวา**
   C_{out} ของตัวขวา → C_{in} ของตัวซ้ายถัดไป ทุกคู่ · C_{in} ของตัวขวาสุดต่อลง**กราวด์ (0)** ถ้าโจทย์ไม่ได้บอกอย่างอื่น
3. **ถ้าโจทย์ให้หาผลบวก ให้ไล่จากขวาไปซ้าย**
   ทีละหลัก เขียนตัวทดที่ได้ไว้ก่อนแล้วค่อยขยับไปหลักถัดไป · บิตบนสุดของคำตอบคือ C_{out} ของตัวซ้ายสุด
4. **ถ้าโจทย์ถามเวลา ให้คูณ แล้วหยุด**
   เวลารวม = เวลาต่อชั้น × จำนวนบิต · **หยุดตรงนี้** อย่าเอาเวลาของ XOR หรือเกตย่อย ๆ ในนั้นมาบวกซ้ำ เพราะ 8 ns ที่โจทย์ให้รวมทั้งกล่องอยู่แล้ว

4 ตัวทดมองล่วงหน้า — คิดตัวทดทุกตัวพร้อมกัน

หัวข้อนี้ยาวที่สุดในบท และคุ้มที่สุด

ข้อสอบกลางภาคระบุไว้ตรง ๆ ว่าออก 3 เรื่องนี้ — **carry generation · carry propagation · output carry** ทั้งสามคำมาจากความคิดเดียวกันเรื่องเดียว ถ้าเข้าใจความคิดนั้น น้องเขียนสูตรเองได้ทั้งหมด ไม่ต้องท่องสักตัว พี่จะค่อย ๆ พาไปครับ

เริ่มจากแถวคนส่งของ ไม่ต้องมีวงจรเลย

ลองนึกภาพนักเรียน 4 คนยืนเรียงแถวกัน คนที่ 0 อยู่ขวาสุด คนที่ 3 อยู่ซ้ายสุด กติกาคือ**ส่งของไปทางซ้ายเท่านั้น**

แต่ละคนเป็นได้ 3 แบบครับ

- **แบบที่ 1 — มีของเป็นของตัวเอง** คนแบบนี้**ส่งของไปทางซ้ายแน่นอน** ไม่ว่าคนทางขวาจะส่งอะไรมาให้หรือไม่ก็ตาม
- **แบบที่ 2 — ไม่มีของเป็นของตัวเอง แต่เป็นคนช่วยส่ง** ถ้ามีคนส่งของมาให้จากทางขวา เขาจะส่งต่อไปทางซ้าย ถ้าไม่มีใครส่งมา เขาก็ไม่ส่งอะไร
- **แบบที่ 3 — ไม่มีของ และไม่ยอมส่งต่อ** ของที่มาถึงเขาจะ**ตายอยู่ตรงนั้น** ไปต่อไม่ได้

จุดสำคัญคือ น้อง**รู้ว่าใครเป็นแบบไหนตั้งแต่ก่อนเริ่มส่ง**ครับ มันเป็นนิสัยของแต่ละคน ไม่ได้ขึ้นกับว่าใครส่งอะไรมา

ลองก่อน (คิดในหัวพอ)

สมมติแถวเป็นแบบนี้ — คนที่ 0 = **มีของเอง** · คนที่ 1 = **ช่วยส่ง** · คนที่ 2 = **ช่วยส่ง** · คนที่ 3 = **ไม่ส่งต่อ**
คำถาม: **คนที่ 2 จะส่งของไปให้คนที่ 3 หรือเปล่า**
และน้องต้อง**รอ**ให้การส่งเกิดขึ้นจริงก่อนถึงจะตอบได้ไหม

เฉลย: ส่งครับ
เหตุผล: คนที่ 0 มีของเอง จึงส่งแน่ → คนที่ 1 เป็นคนช่วยส่ง จึงส่งต่อ → คนที่ 2 ก็ช่วยส่ง จึงส่งต่อ
และ**ไม่ต้องรอเลย**ครับ — น้องตอบได้ทันทีที่รู้ว่าใครเป็นแบบไหน เพราะน้องเห็น "นิสัย" ของทุกคนพร้อมกันหมดตั้งแต่แรก

ที่น้องเพิ่งทำไปนั่นแหละครับ คือ **look-ahead carry** ทั้งดุ้น — มองนิสัยของทุกชั้นพร้อมกัน แล้ว**ทำนาย**ว่าตัวทดจะไปโผล่ที่ไหนบ้าง แทนที่จะนั่งรอมันเดิน

ทีนี้เอาคำว่า "นิสัย" มาแปลเป็นวงจร

คนที่ **มีของเป็นของตัวเอง** ในวงจรบวกคือชั้นที่**สร้างตัวทดขึ้นมาเอง** โดยไม่สนใจว่า C_{in} จะเป็นอะไร

ชั้นแบบนั้นเกิดตอนไหนครับ — ตอนที่ A = 1 และ B = 1 เท่านั้น เพราะ 1 + 1 ล้นอยู่แล้ว ถึง C_{in} จะเป็น 0 มันก็ยังทด

เราตั้งชื่อให้มันว่า **ตัวทดที่ถูกสร้าง** หรือ **carry generation** เขียนย่อว่า C_g

คนที่ **ช่วยส่งต่อ** ในวงจรบวกคือชั้นที่ตัวเองไม่ได้สร้างตัวทด แต่ถ้ามีตัวทดวิ่งเข้ามา มันจะ**ผลักออกไปให้ชั้นถัดไป**

ชั้นแบบนั้นเกิดตอนไหน — ตอนที่**มีเลข 1 อยู่อย่างน้อยหนึ่งตัว** ในบรรดา A กับ B เพราะ 1 + 0 + ตัวทด 1 = 2 ซึ่งล้นพอดี

เราตั้งชื่อให้มันว่า **ตัวทดที่ถูกส่งต่อ** หรือ **carry propagation** เขียนย่อว่า C_p

สองบรรทัดนี้คือทั้งหมดของหัวข้อนี้

$$C_g = A \cdot B$$

สร้างเอง = AND (ต้องเป็น 1 ทั้งคู่)

$$C_p = A + B$$

ส่งต่อ = OR (มี 1 สักตัวก็พอ)

สังเกตว่าทั้งสองตัวนี้**ใช้แค่ A กับ B เท่านั้น ไม่ใช้ C_{in} เลย** — จุดนี้แหละครับคือกุญแจของทั้งหัวข้อ ค่อยขยายต่อในหน้าถัดไป

เช็กกับตารางค่าความจริงว่าจริงไหม

ตารางข้างล่างนี้คือ 4 แถวของ A กับ B เท่านั้น (ยังไม่มี C_{in}) พี่ใส่คอลัมน์ "เป็นคนแบบไหน" ไว้ให้เทียบครับ

A	B	C_g	C_p	เป็นคนแบบไหน
0	0	0	0	ไม่ส่งต่อ (ตัวทดตายตรงนี้)
0	1	0	1	ช่วยส่ง
1	0	0	1	ช่วยส่ง
1	1	1	1	มีของเอง (และส่งต่อได้ด้วย)

แถวล่างสุดน่าสนใจครับ — A=1, B=1 เป็นทั้ง "มีของเอง" และ "ช่วยส่ง" พร้อมกัน คือ $C_g = 1$ และ $C_p = 1$

ฟังดูขัดกัน แต่ไม่ขัดเลยครับ เพราะ "มีของเอง" ทำให้มันส่งแน่นอนอยู่แล้ว ส่วน "ช่วยส่ง" เป็นแค่เงื่อนไขเพิ่ม ซึ่งไม่ได้ทำให้คำตอบเปลี่ยน — ส่งก็คือส่ง

สรุปคือ **แถวที่ $C_g = 1$ มีแถวเดียว** ส่วน **แถวที่ $C_p = 1$ มีสามแถว** ตรงกับนิยาม AND และ OR พอดี

ตำราบางเล่มเขียน $C_p = A \oplus B$ — ผิดไหม

ไม่ผิดครับ และใช้แล้วได้ตัวทดออกมา**เท่ากันเป๊ะทุกกรณี**

เหตุผล: กรณีเดียวที่ A + B กับ A ⊕ B ต่างกัน คือแถว A=1, B=1 (OR ให้ 1 ส่วน XOR ให้ 0) แต่แถวนั้น C_g เป็น 1 อยู่แล้ว ซึ่งบังคับให้ตัวทดออกเป็น 1 ไปแล้ว ค่าของ C_p จึงไม่มีผลต่อคำตอบในแถวนั้นเลย

ในข้อสอบวิชานี้ให้ใช้ $C_p = A + B$ ตามสไลด์ จะปลอดภัยที่สุดครับ ส่วน A ⊕ B ให้รู้ไว้เฉย ๆ เผื่อไปเจอในตำราเล่มอื่น

ตัวทดออก — เขียนใหม่โดยใช้สองตัวนี้

ตัวทดที่ออกจากชั้นหนึ่ง ๆ จะเป็น 1 ได้ด้วย**เหตุผลสองอย่างเท่านั้น**ครับ

เหตุผลที่ 1: ชั้นนี้สร้างตัวทดขึ้นมาเอง — นั่นคือ $C_g = 1$

เหตุผลที่ 2: ชั้นนี้เป็นคนช่วยส่ง **และ** มีตัวทดวิ่งเข้ามาจริง ๆ — นั่นคือ $C_p = 1$ และ $C_{in} = 1$ พร้อมกัน

เหตุผลสองอย่างนี้ต่อกันด้วยคำว่า "**หรือ**" เพราะเป็น 1 ด้วยเหตุผลไหนก็ได้ ส่วนภายในเหตุผลที่ 2 ต่อกันด้วยคำว่า "**และ**" เพราะต้องครบทั้งสองเงื่อนไข

$$C_{out} = C_g + C_p \cdot C_{in}$$

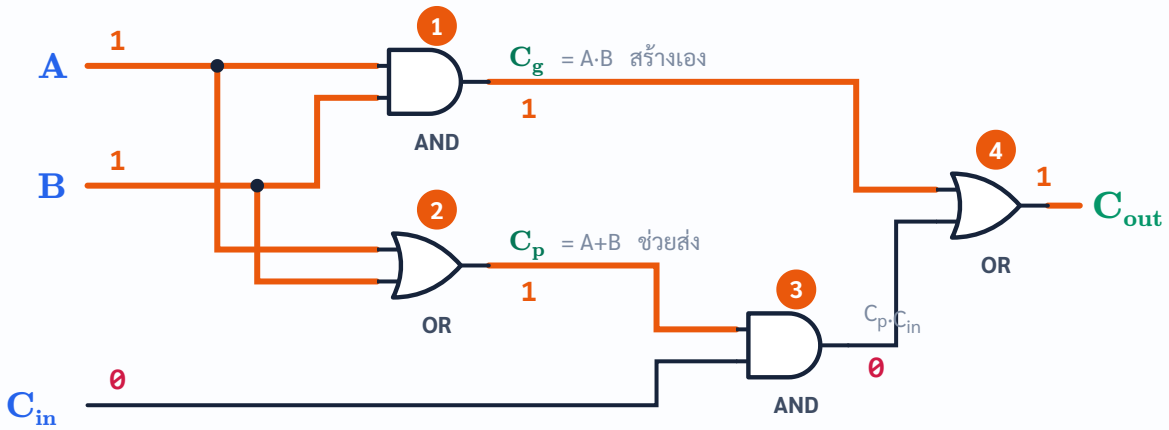
สร้างเอง หรือ (ช่วยส่ง และ มีตัวทดเข้ามา)

ถ้าแทน $C_g = A{\cdot}B$ และ $C_p = A+B$ ลงไป จะได้หน้าตาที่อาจารย์เขียนไว้ในสไลด์พอดีครับ

$$C_{out} = A{\cdot}B + (A + B){\cdot}C_{in}$$

และถ้าน้องเทียบกับนิพจน์ของตัวบวกเต็มในหัวข้อ 2 ซึ่งคือ $C_{out} = A{\cdot}B + C_{in}{\cdot}(A \oplus B)$ จะเห็นว่ามันคือ**ของสิ่งเดียวกัน** แค่เขียนคนละมุมมอง — เราไม่ได้เปลี่ยนวงจร เราแค่เปลี่ยนวิธีมองมันครับ

วงจรของหนึ่งชั้น — พร้อมเดินตัวอย่าง A = 1, B = 1, C in = 0



หนึ่งชั้นของตัวทดมองล่วงหน้า · **สายส้ม**คือสายที่มีค่า 1 ในตัวอย่าง A=1, B=1, C_{in}=0
กล่องซ้ายสองตัว (①②) ใช้แค่ A กับ B — คิดได้ทันทีตั้งแต่วินาทีแรก

ขั้น	ทำอะไร	ได้อะไร
①	AND ตัวบนเห็น A=1, B=1 — เป็น 1 ทั้งคู่ แปลว่าชั้นนี้**สร้างตัวทดเอง**	C_g = 1
②	OR ตัวล่างเห็น A=1, B=1 — มี 1 อยู่ แปลว่าชั้นนี้**ช่วยส่งได้ด้วย**	C_p = 1
③	AND ตัวขวาล่างเห็น C_p=1 กับ C_{in}=0 — ช่วยส่งได้ แต่**ไม่มีอะไรให้ส่ง**	$C_p{\cdot}C_{in}$ = 0

④	OR ตัวสุดท้ายเห็น 1 (จาก ①) กับ 0 (จาก ③) — **มีเหตุผลหนึ่งข้อพอ**	C_{out} = **1**

เช็กคำตอบกลับเข้าโจทย์: $A + B + C_{in} = 1 + 1 + 0 = 2$ ซึ่งคือ "10" จึงต้องทด 1

วงจรตอบ $C_{out} = 1$ ตรงกันครับ และเราได้คำตอบมาโดย**ไม่ต้องรออะไรเลย** เพราะ ① กับ ② คิดจาก A กับ B ที่มีอยู่แล้ว

ทีนี้ถึงท่าไม้ตาย — ไล่ตัวทดออกทุกตัวให้หมดค้าง

สูตร $C_{out} = C_g + C_p \cdot C_{in}$ ยัง**ไม่ได้แก้ปัญหาความช้า**เลยนะครับ เพราะชั้นที่ 1 ยังต้องใช้ C_{in} ของตัวเอง ซึ่งก็คือตัวทดที่ออกจากชั้น 0 อยู่ดี

แต่คราวนี้เรามีของดีอยู่ในมือแล้ว — เรารู้ว่าตัวทดที่ออกจากชั้น 0 **เท่ากับอะไร** เราก็**แทนค่ามันลงไปตรง ๆ** แทนที่จะรอมันสิครับ

เริ่มจากชั้น 0 ก่อน ชั้นนี้ง่ายที่สุดเพราะ C_{in} ของมันคือตัวทดที่ป้อนจากนอกวงจร ซึ่งเรารู้ค่าตั้งแต่แรกอยู่แล้ว

ชั้น 0

$$C_{out0} = C_{g0} + C_{p0} \cdot C_{in}$$

จบเลย ไม่มีอะไรค้าง เพราะทุกตัวในบรรทัดนี้รู้ค่าตั้งแต่วินาทีแรก

ชั้น 1 — เขียนตามกฎเดิมก่อน แล้วค่อยแทนค่า

①	เขียนตามกฎเดิม โดย C_{in} ของชั้น 1 คือ C_{out0}	$\mathbf{C_{out1} = C_{g1} + C_{p1} \cdot C_{out0}}$
②	แทนค่า C_{out0} ที่เพิ่งได้จากชั้น 0 ลงไป	$= C_{g1} + C_{p1} \cdot (C_{g0} + C_{p0} C_{in})$
③	กระจาย C_{p1} เข้าไปในวงเล็บ	$= C_{g1} + C_{p1} C_{g0} + C_{p1} C_{p0} C_{in}$

ชั้น 2 — ทำแบบเดียวกันอีกรอบ

①	เขียนตามกฎเดิม	$\mathbf{C_{out2} = C_{g2} + C_{p2} \cdot C_{out1}}$
②	แทนค่า C_{out1} ที่เพิ่งได้ แล้วกระจาย C_{p2} เข้าไปทุกพจน์	$= C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{g0} + C_{p2} C_{p1} C_{p0} C_{in}$

ชั้น 3 — รอบสุดท้าย

$$C_{out3} = C_{g3} + C_{p3} C_{g2} + C_{p3} C_{p2} C_{g1} + C_{p3} C_{p2} C_{p1} C_{g0} + C_{p3} C_{p2} C_{p1} C_{p0} C_{in}$$

สี่บรรทัดนี้คือสิ่งที่อยู่ในสไลด์ของอาจารย์เป๊ะ ๆ ครับ แต่ตอนนี้น้องไม่ได้ท่องมัน น้อง**สร้างมันขึ้นมาเองทีละขั้น**แล้ว

อ่านแต่ละพจน์ออกมาเป็นภาษาคน แล้วจะไม่มีวันลืม

เอาบรรทัดของชั้น 2 มาแยกดูทีละพจน์ครับ แต่ละพจน์คือ "เส้นทางหนึ่งเส้นทาง" ที่ทำให้ตัวทดโผล่ออกจากชั้น 2 ได้

พจน์	แปลว่า
$\mathbf{C_{g2}}$	ชั้น 2 **สร้างตัวทดเอง** — ไม่เกี่ยวกับใครเลย
$\mathbf{C_{p2} C_{g1}}$	ชั้น 1 สร้าง **แล้ว** ชั้น 2 ช่วยส่งต่อ

$C_{p2}C_{p1}C_{g0}$	ชั้น 0 สร้าง **แล้ว** ชั้น 1 ส่งต่อ **แล้ว** ชั้น 2 ส่งต่ออีกที
$C_{p2}C_{p1}C_{p0}C_{in}$	ตัวทดมาจาก**นอกวงจร** แล้วถูกส่งต่อผ่านทั้งสามชั้นจนทะลุออกมา

เห็นแบบแผนไหมครับ — ไล่จากพจน์บนลงล่าง **จุดกำเนิดของตัวทดถอยห่างออกไปทีละชั้น** และทุกชั้นที่มันต้องผ่าน ต้องเป็นคนช่วยส่งหมด

วิธีเขียนสูตรชั้นไหนก็ได้ ภายใน 20 วินาที

อยากได้ C_{out} ของชั้นที่ k ให้เขียนแบบนี้ครับ
① พจน์แรกคือ **C_g ของชั้น k** เฉย ๆ
② พจน์ถัดไป ถอย C_g ลงมาหนึ่งชั้น แล้วคูณ C_p ของชั้นที่อยู่เหนือมันทั้งหมด
③ ทำซ้ำจนถึง C_{g0}
④ พจน์สุดท้ายคือ **C_p ทุกชั้นคูณกัน แล้วคูณ C_{in}**
ผลคือชั้นที่ k จะมี **k + 2 พจน์** เสมอ — ชั้น 0 มี 2 พจน์ · ชั้น 1 มี 3 · ชั้น 2 มี 4 · ชั้น 3 มี 5

ทำไมมันถึงเร็ว — จุดที่ต้องตอบให้ได้ในข้อสอบ

มองบรรทัดของชั้น 3 อีกที แล้วถามตัวเองว่า **มีตัวไหนในบรรทัดนั้นที่เราต้องรอบ้าง**

คำตอบคือ**ไม่มีเลยครับ** — C_g ทุกตัวคำนวณจาก A กับ B, C_p ทุกตัวก็คำนวณจาก A กับ B, ส่วน C_{in} ก็ป้อนมาจากนอกวงจรตั้งแต่แรก

แปลว่าตัวทดของ**ทุกชั้นถูกคำนวณพร้อมกันหมด** ไม่มีใครรอใคร ต่างจากตัวทดแบบไล่ในหัวข้อ 3 ที่ชั้น 3 ต้องรอชั้น 2 ซึ่งรอชั้น 1 ซึ่งรอชั้น 0

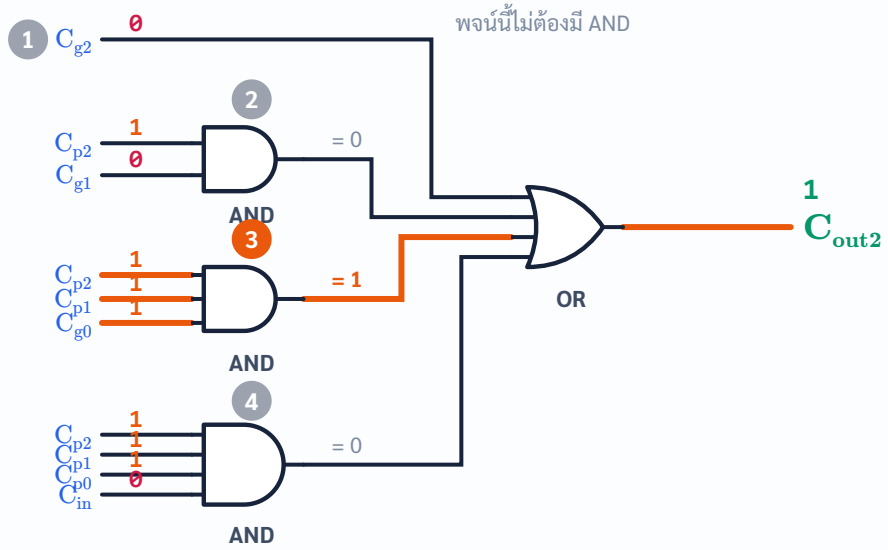
และสังเกตรูปร่างของสูตรครับ — มันเป็น "**AND หลายตัว แล้ว OR รวมทีเดียว**" ซึ่งลึกแค่ **2 ชั้นเกต** เท่านั้น ไม่ว่าจะบวกกี่บิต

เทียบกัน	ตัวทดไล่ (ripple)	ตัวทดมองล่วงหน้า (look-ahead)
ตัวทดต้องผ่านกี่ชั้น	เท่ากับจำนวนบิต — 4 บิตก็ 4 ชั้น 32 บิตก็ 32 ชั้น	**2 ชั้นเกตเสมอ** ไม่ว่าจะกี่บิต
เวลาที่ใช้	โตขึ้นเรื่อย ๆ ตามจำนวนบิต (4 บิต = 32 ns ที่ 8 ns ต่อชั้น)	เกือบคงที่ ไม่โตตามจำนวนบิต
จำนวนเกตที่ใช้	น้อย — ตัวบวกเต็ม 5 เกตต่อบิต	มากกว่ามาก และยิ่งบิตเยอะ พจน์ยิ่งยาว (ชั้น k มี k+2 พจน์)
ขาเข้า/ขาออกภายนอก	เหมือนกันทุกประการ — มองจากข้างนอกแยกไม่ออกเลย ต่างกันแค่ข้างในและความเร็ว	

นี่คือการ**แลก**ครับ — จ่ายเกตเพิ่มเพื่อซื้อความเร็ว ข้อสอบชอบถามว่าทั้งสองแบบต่างกันตรงไหน คำตอบคือ **ความเร็ว** ไม่ใช่ขาเข้าหรือขาออก

วงจรของ C out2 — วาดตามสูตรตรง ๆ ทีละพจน์

แต่ละพจน์คือ AND หนึ่งตัว แล้วเอาทุกพจน์มารวมกันด้วย OR ตัวเดียว พจน์แรก (C_{g2}) ไม่ต้องมี AND เพราะมันมีตัวเดียวอยู่แล้ว



วงจรของ C_{out2} · AND หนึ่งตัวต่อหนึ่งพจน์ แล้ว OR รวมทีเดียว = ลึกแค่ 2 ชั้น

ค่าที่ใส่คือตัวอย่าง A = 101, B = 011, C_{in} = 0 ข้างล่าง · **พจน์ที่ 3 (ส้ม) เป็นพจน์เดียวที่ติด** จึงทำให้ C_{out2} = 1

ทำโจทย์ให้ดูทั้งข้อ — A = 101 , B = 011 , C in = 0 แบบ 3 ชั้น

โจทย์ข้อนี้คือข้อเดียวกับที่เราบวกแบบไล่ทีละหลักในหัวข้อ 3 ครับ คราวนี้เราจะทำแบบมองล่วงหน้า แล้วดูว่าได้คำตอบเดียวกันไหม

ขั้นที่ 1 — แยกบิตออกมาก่อน อย่าเพิ่งคิดอะไรทั้งนั้น เขียนแค่ว่าแต่ละหลักมี A กับ B เป็นอะไร

ชั้นที่	A	B
ชั้น 0 (ขวาสุด)	$A_0 = 1$	$B_0 = 1$
ชั้น 1	$A_1 = 0$	$B_1 = 1$
ชั้น 2 (ซ้ายสุด)	$A_2 = 1$	$B_2 = 0$

ขั้นที่ 2 — หา C_g และ C_p ของทุกชั้น ขั้นนี้ทำ**พร้อมกันได้ทั้งสามชั้น** เพราะใช้แค่ A กับ B

ชั้นที่	$C_g = A \cdot B$	$C_p = A+B$	ชั้นนี้เป็นคนแบบไหน
ชั้น 0	1 · 1 = 1	1 + 1 = 1	มีของเอง
ชั้น 1	0 · 1 = 0	0 + 1 = 1	ช่วยส่ง

ชั้น 2	$1 \cdot 0 =$ `0`	$1 + 0 =$ `1`	ช่วยส่ง

อ่านคอลัมน์ขวาสุดดูสิครับ — **มีของเอง → ช่วยส่ง → ช่วยส่ง** นี่คือแถวคนส่งของในกล่อง "ลองก่อน" ตอนต้นหัวข้อเป๊ะ ๆ เลย เพราะฉะนั้นเราเดาได้เลยว่าตัวทดจะทะลุออกทางซ้าย

ขั้นที่ 3 — แทนค่าลงสูตรทั้งสามชั้น ทีละบรรทัด ไม่ข้ามขั้น

ชั้น 0

$$\begin{aligned} C_{out0} &= C_{g0} + C_{p0} \cdot C_{in} \\ &= 1 + (1 \cdot 0) \\ &= 1 + 0 \\ &= \mathbf{1} \end{aligned}$$

ชั้น 1

$$\begin{aligned} C_{out1} &= C_{g1} + C_{p1}C_{g0} + C_{p1}C_{p0}C_{in} \\ &= 0 + (1 \cdot 1) + (1 \cdot 1 \cdot 0) \\ &= 0 + 1 + 0 \\ &= \mathbf{1} \end{aligned}$$

ชั้น 2

$$\begin{aligned} C_{out2} &= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{g0} + C_{p2}C_{p1}C_{p0}C_{in} \\ &= 0 + (1 \cdot 0) + (1 \cdot 1 \cdot 1) + (1 \cdot 1 \cdot 1 \cdot 0) \\ &= 0 + 0 + 1 + 0 \\ &= \mathbf{1} \end{aligned}$$

ขั้นที่ 4 — หาผลบวกแต่ละหลัก ใช้สูตรเดิมจากหัวข้อ 2 คือ $\Sigma = A \oplus B \oplus C_{in}$ โดย C_{in} ของแต่ละชั้นคือตัวทดออกของชั้นก่อนหน้า ซึ่งเราเพิ่งได้มาครบแล้ว

หลัก	คิดยังไง	ได้
Σ_0	$A_0 \oplus B_0 \oplus C_{in} = 1 \oplus 1 \oplus 0$ → $1 \oplus 1$ ได้ 0 แล้ว $0 \oplus 0$ ได้ 0	`0`
Σ_1	$A_1 \oplus B_1 \oplus C_{out0} = 0 \oplus 1 \oplus 1$ → $0 \oplus 1$ ได้ 1 แล้ว $1 \oplus 1$ ได้ 0	`0`
Σ_2	$A_2 \oplus B_2 \oplus C_{out1} = 1 \oplus 0 \oplus 1$ → $1 \oplus 0$ ได้ 1 แล้ว $1 \oplus 1$ ได้ 0	`0`
Σ_3	ไม่มีชั้นที่ 3 แล้ว บิตบนสุดของคำตอบคือ C_{out2} ตรง ๆ	`1`

เช็กคำตอบกลับเข้าโจทย์: ได้ $\Sigma_3\Sigma_2\Sigma_1\Sigma_0 =$ **1000** ซึ่งคือเลข 8

โจทย์คือ 101 + 011 = 5 + 3 = 8 ครับ ตรงกัน

และที่สำคัญกว่านั้น — **ตรงกับคำตอบที่เราได้จากวิธีไล่ทีละหลักในหัวข้อ 3 ทุกตัวเลข** รวมถึงตัวทดระหว่างทางที่ได้ 1, 1, 1 เหมือนกัน · วงจรสองแบบให้คำตอบเดียวกันเสมอ ต่างกันแค่ความเร็ว

กับดักข้อสอบ · ลืมพจน์สุดท้ายที่มี C_{in}

พจน์ที่คนลืมบ่อยที่สุดคือพจน์ยาวที่สุด ซึ่งคือ C_p ทุกชั้นคูณกันแล้วคูณ C_{in}

หลายคนเขียนชั้น 1 แค่ว่า $C_{out1} = C_{g1} + C_{p1}C_{g0}$ แล้วจบ

ในตัวอย่างเมื่อกี้ $C_{in} = 0$ อยู่แล้ว พจน์นั้นเลยเป็น 0 และ**ลืมไปก็ยังได้คำตอบถูก** — นี่แหละคือความร้ายของมัน เพราะน้องจะไม่รู้ตัวว่าจำผิด

ลองเปลี่ยนโจทย์เป็น **A = 11 , B = 00 , C_{in} = 1** ดูครับ (คือ 3 + 0 + 1 ซึ่งต้องได้ 4 หรือ 100)

คิดยังไง	สูตรที่ลืมพจน์สุดท้าย	สูตรเต็ม
C_g , C_p	C_{g0}=0 , C_{p0}=1 , C_{g1}=0 , C_{p1}=1 (ทั้งสองชั้นเป็นคน "ช่วยส่ง")	
C_{out1}	= 0 + (1·0) = `0`	= 0 + (1·0) + (1·1·1) = `1`
คำตอบที่อ่านได้	Σ = 00 ไม่มีตัวทดออก → ตอบ **0**	Σ = 00 มีตัวทดออก 1 → ตอบ **100 = 4**

เห็นไหมครับว่าผิดไปคนละโลกเลย — **ตัวทดที่ป้อนเข้ามาจากนอกวงจร ถูกส่งต่อทะลุทุกชั้นได้จริง ๆ** ถ้าทุกชั้นเป็นคนช่วยส่ง

วิธีกันลืม: นับพจน์ก่อนส่ง — ชั้นที่ k ต้องมี **k + 2 พจน์** เสมอ ถ้านับได้น้อยกว่านั้น แปลว่าหายไปหนึ่งพจน์แน่นอน

เช็กลิสต์ตัวทดมองล่วงหน้า

1

แยกบิตแล้วหา C_g กับ C_p ให้ครบทุกชั้นก่อน

$C_g = A{\cdot}B$ · $C_p = A{+}B$ · **ห้ามแตะ C_{in} ในขั้นนี้** เพราะสองตัวนี้ไม่เกี่ยวกับ C_{in} เลย

2

เขียนสูตรตัวทดออกทีละชั้น จากชั้น 0 ขึ้นไป

ชั้น 0 เขียนตรง ๆ · ชั้นถัด ๆ ไปใช้แบบแผน "ถอย C_g ทีละชั้น คูณ C_p ของชั้นที่อยู่เหนือ" · จบด้วยพจน์ C_p ทุกชั้นคูณ C_{in}

3

นับพจน์ก่อนแทนค่า

ชั้น k ต้องมี k+2 พจน์ · ถ้าครบแล้วค่อยแทนตัวเลขลงไปทีละพจน์ แล้วรวมด้วย OR

4

ถ้าโจทย์ให้หาผลบวกด้วย ค่อยใช้ $\Sigma = A \oplus B \oplus C_{in}$

โดย C_{in} ของชั้น k คือ C_{out} ของชั้น $k-1$ ที่เพิ่งหาได้ · บิตบนสุดของคำตอบคือ C_{out} ของชั้นซ้ายสุด

5

ถ้าโจทย์ถามว่าเร็วกว่ายังไง ให้ตอบสั้น ๆ แล้วหยุด

"เพราะ C_g กับ C_p ทุกชั้นคิดได้พร้อมกันจาก A กับ B ตัวทดทุกตัวจึงคำนวณพร้อมกัน ไม่ต้องรอกันเป็นทอด ๆ และลึกแค่ 2 ชั้นเกตเสมอ" · **หยุดตรงนี้** ไม่ต้องไปคำนวณ ns ถ้าโจทย์ไม่ได้ให้ตัวเลขมา

5 ตัวลบครึ่งและตัวลบเต็ม — เหมือนตัวบวกจนน่าตกใจ

เริ่มจากการลบเลขด้วยมือ

ลองลบ 53 − 27 ในกระดาษครับ น้องเริ่มที่หลักขวา คือ 3 − 7

แต่ 3 น้อยกว่า 7 ลบตรง ๆ ไม่ได้ น้องเลยไป**ยืม**จากหลักถัดไปทางซ้ายมา 1 กลายเป็น 13 − 7 = 6

แล้วหลักซ้ายที่โดนยืมไป ก็ต้องจำไว้ว่า**ตัวเองหายไป 1** จึงเหลือ 5 − 1 − 2 = 2

เห็นไหมครับว่าการลบก็ให้ผลออกมา**สองอย่าง**เหมือนการบวกเป๊ะ — อย่างแรกคือเลขที่เขียนลงช่อง อย่างที่สองคือ "ยืมหรือไม่ยืม" ต่างกันแค่ตัวบวก**ทด**ไปข้างหน้า ส่วนตัวลบ**ยืม**มาจากข้างหน้า

ลองก่อน (คิดในหัวพอ)

เลขหนึ่งบิตสองตัว เอามาลบกันแบบ A − B มีทั้งหมด 4 แบบ
คำถาม: **แบบไหนที่ต้องยืม** และมีกี่แบบ

เฉลย: มีแบบเดียวครับ คือ **0 − 1**
เพราะ 0 น้อยกว่า 1 จึงต้องยืมมา 1 กลายเป็น 10 − 1 = 1 (ในฐานสอง) — ได้ผลต่าง 1 และยืม 1
ส่วนอีกสามแบบ (0−0, 1−0, 1−1) ลบได้ตรง ๆ ไม่ต้องยืมเลย

ตั้งชื่อ แล้วเทียบกับตัวบวกทันที

ผลต่างที่เขียนลงช่อง เรียกว่า **ผลต่าง** เขียนแทนด้วย **Δ** (อ่านว่า เดลต้า)

ส่วนการยืมที่ส่งไปให้หลักถัดไป เรียกว่า **ตัวยืมออก** เขียนว่า $\mathbf{B_{out}}$ (borrow out)

ระวังตัวอักษร B ซ้ำกัน

ในหัวข้อนี้ **B** ตัวเดี่ยว ๆ คือ**เลขที่เอามาลบ** (ตัวลบ) ส่วน $\mathbf{B_{out}}$ กับ $\mathbf{B_{in}}$ คือ**การยืม** ไม่ใช่ตัวเลขนั้น
สไลด์ใช้ตัวอักษรซ้ำกันแบบนี้ ต้องดูตัวห้อยให้ดีครับ

วงจรที่ทำสองอย่างนี้จาก A กับ B เรียกว่า **ตัวลบครึ่ง** หรือ **half-subtractor** และ "ครึ่ง" ก็มาจากเหตุผลเดียวกับตัวบวกครึ่ง — มันรับการยืมจากหลักก่อนหน้าไม่ได้

A	B	Δ	B_{out}	เช็ก
0	0	0	0	0−0 = 0
0	1	1	1	ยืมมา: 10−1 = 1
1	0	1	0	1−0 = 1
1	1	0	0	1−1 = 0

อ่านคอลัมน์ Δ จากบนลงล่างได้ **0 1 1 0** — **เหมือนคอลัมน์ Σ ของตัวบวกครึ่งเป๊ะ** เพราะฉะนั้น Δ = A ⊕ B ซึ่งคือ XOR ตัวเดิม

อ่านคอลัมน์ B_{out} ได้ **0 1 0 0** — อันนี้ไม่เหมือนของตัวบวก (ของตัวบวกคือ 0 0 0 1)

แถวที่ B_{out} เป็น 1 คือแถว A=0, B=1 · แปลว่าเงื่อนไขคือ "**A เป็น 0 และ B เป็น 1**" ซึ่งเขียนเป็น $\bar{A} \cdot B$

สองบรรทัดของตัวลบครึ่ง

$$\Delta = A \oplus B$$

ผลต่าง = XOR (เหมือนผลบวกทุกประการ)

$$B_{out} = \bar{A} \cdot B$$

ตัวยืม = AND ที่กลับค่าขา A ก่อน

ขีดบนหัว A (เขียน Ā อ่านว่า "เอ บาร์") แปลว่า**กลับค่า** — A เป็น 0 ก็อ่านเป็น 1
ถ้าเทียบกับตัวบวกครึ่งที่ $C_{out} = A \cdot B$ จะเห็นว่า**ต่างกันแค่ขีดเดียวนี้เท่านั้น**

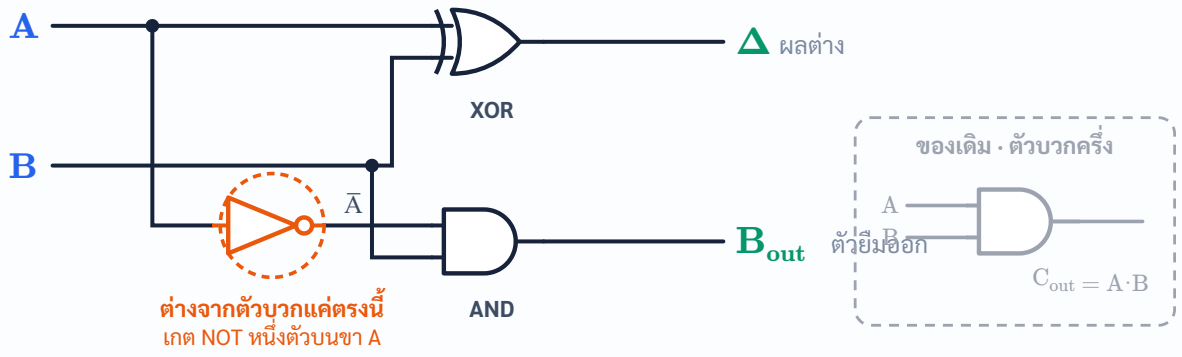
ทำไมตัวยืมถึงเป็น Ā·B — เหตุผลในหนึ่งบรรทัด

เราจะยืมก็ต่อเมื่อ**ตัวตั้งไม่พอลบ** ใช่ไหมครับ

ในเลขหนึ่งบิต "ไม่พอลบ" เกิดขึ้นกรณีเดียวคือ ตัวตั้ง A เป็น 0 แต่ตัวลบ B เป็น 1

เงื่อนไขนั้นเขียนเป็นวงจรคือ "A ต้องเป็น 0" (ซึ่งคือ Ā เป็น 1) **และ** "B ต้องเป็น 1" — คำว่า "และ" แปลเป็น AND ครับ จึงได้ Ā·B พอดี

วงจรจริง — และจุดเดียวที่ต่างจากตัวบวก



ตัวลบครึ่ง · XOR ให้ผลต่าง (เหมือนตัวบวกเป๊ะ) ส่วน AND ให้ตัวยืม แต่**ขา A ต้องผ่าน NOT ก่อน**
กรอบเทาขวามือคือตัวบวกครึ่งของเดิมไว้เทียบ · เส้นที่ตัดกันโดยไม่มีจุดดำ แปลว่าไม่ได้ต่อถึงกัน

ตัวลบเต็ม — เพิ่มขารับการยืมจากหลักก่อนหน้า

เหตุผลที่ต้องมีตัวลบเต็ม เหมือนกับเหตุผลของตัวบวกเต็มทุกประการครับ — ทุกหลักยกเว้นหลักขวาสุด อาจโดนหลักทางขวายืมไปแล้ว มันต้องรู้เรื่องนั้นด้วย

ขาที่เพิ่มเข้ามาเรียกว่า **ตัวยืมเข้า** หรือ B_{in} (borrow in) แปลว่า "หลักทางขวายืมของฉันไปแล้วหนึ่ง"

วงจรที่รับ A, B, B_{in} แล้วคาย Δ กับ B_{out} ชื่อว่า **ตัวลบเต็ม** หรือ **full-subtractor** ครับ

B_{in}	A	B	Δ	B_{out}	เช็ก (A − B − B_{in})
0	0	0	0	0	0 = 0
0	0	1	1	1	−1 → ยืม เหลือ 1
0	1	0	1	0	1 = 1
0	1	1	0	0	0 = 0
1	0	0	1	1	−1 → ยืม เหลือ 1
1	0	1	0	1	−2 → ยืม เหลือ 0
1	1	0	0	0	0 = 0
1	1	1	1	1	−1 → ยืม เหลือ 1

คอลัมน์ "เช็ก" คือผลของ A − B − B_{in} จริง ๆ ครับ ถ้าออกมาติดลบ แปลว่าต้องยืม 1 (ซึ่งมีค่า 2 ในหลักนี้) มาบวกก่อน แล้วค่อยอ่าน Δ

ตัวอย่างแถวที่ 6: 0 − 1 − 1 = −2 · ยืมมา 2 ได้ −2 + 2 = 0 จึง Δ = 0 และ B_{out} = 1

ตัวอย่างแถวสุดท้าย: 1 − 1 − 1 = −1 · ยืมมา 2 ได้ −1 + 2 = 1 จึง Δ = 1 และ B_{out} = 1

$$\Delta = (A \oplus B) \oplus B_{in}$$

เหมือนผลบวกของตัวบวกเต็มทุกตัวอักษร

$$B_{out} = \bar{A}\cdot B + B_{in}\cdot\overline{(A \oplus B)}$$

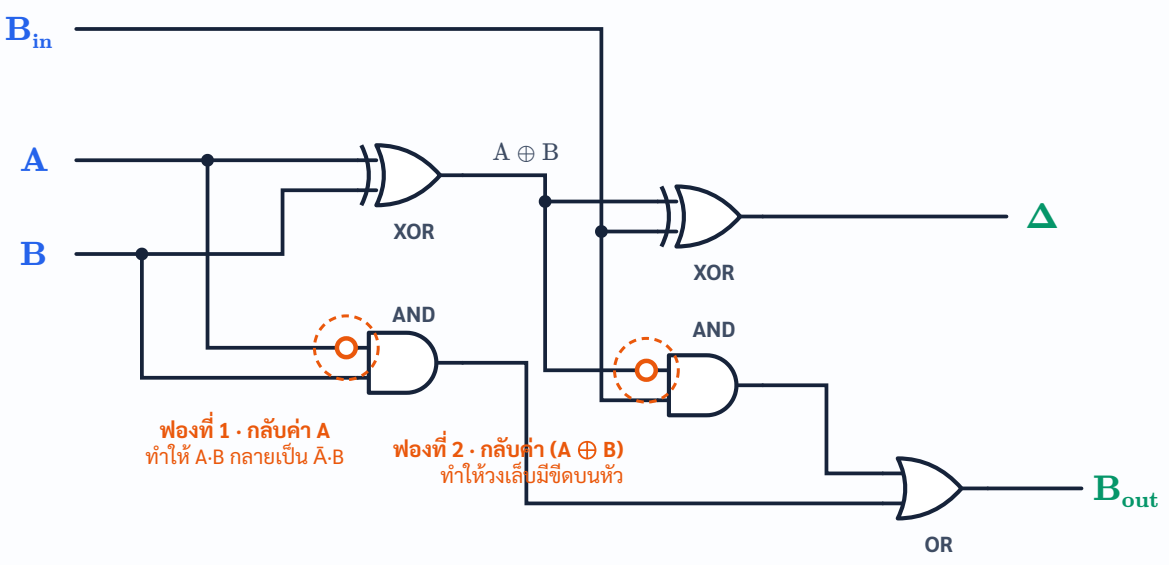
ขีดบนหัว (A ⊕ B) แปลว่ากลับค่าทั้งวงเล็บ — นั่นคือ XNOR

วงจรตัวลบเต็ม = วงจรตัวบวกเต็ม + ฟองสองฟอง

เอานิพจน์ของตัวบวกเต็มมาวางข้างกันแล้วมองทีละตัวอักษรครับ

ของ	ตัวบวกเต็ม	ตัวลบเต็ม	ต่างกันตรงไหน
ผลที่ลงช่อง	$\Sigma = (A \oplus B) \oplus C_{in}$	$\Delta = (A \oplus B) \oplus B_{in}$	ไม่ต่างเลย
พจน์แรกของตัวทด/ตัวยืม	$A \cdot B$	$\bar{\mathbf{A}} \cdot B$	กลับค่าขา A
พจน์ที่สอง	$C_{in} \cdot (A \oplus B)$	$B_{in} \cdot \overline{\mathbf{(A \oplus B)}}$	กลับค่าทั้งวงเล็บ
เกตปิดท้าย	OR	OR	ไม่ต่างเลย

สรุปว่าน้อง**ไม่ต้องจำวงจรตัวลบเต็มแยกอีกวงจรหนึ่ง**ครับ — จำวงจรตัวบวกเต็มให้แม่น แล้วเติมฟองกลับค่า 2 จุด ตามรูปข้างล่าง



ตัวลบเต็ม · โครงเหมือนตัวบวกเต็มในหัวข้อ 2 ทุกเส้น เปลี่ยนแค่ **ฟองสองฟองบนขาเข้าของ AND ทั้งสองตัว** และเปลี่ยนชื่อขาเข้า C_{in} เป็น B_{in}

ทำโจทย์ให้ดู — 1001 − 0110

เลข 1001 คือ 9 และ 0110 คือ 6 คำตอบที่ควรได้คือ 3 ครับ

หลัก	A − B − ตัวยืมเข้า	คิดยังไง	Δ	ยืมออก
บิต 0	1 − 0 − 0	ลบได้ตรง ๆ ไม่ต้องยืม	1	0
บิต 1	0 − 1 − 0	ไม่พอลบ ต้องยืม · 10 − 1 = 1	1	1
บิต 2	0 − 1 − **1**	โดนยืมไปแล้ว ยิ่งไม่พอ ต้องยืมอีก · 10 − 1 − 1 = 0	0	1
บิต 3	1 − 0 − **1**	โดนยืมไป 1 เหลือ 0 พอดี	0	0

เช็กคำตอบกลับเข้าโจทย์: อ่าน Δ จากบิต 3 ลงมาบิต 0 ได้ **0011** ซึ่งคือเลข 3

โจทย์คือ 9 − 6 = 3 ครับ ตรงกัน · และตัวยืมออกของบิตซ้ายสุดเป็น 0 แปลว่าไม่ได้ติดลบ ซึ่งถูกต้องเพราะ 9 มากกว่า 6

กับดักข้อสอบ · ลืมขีดบนหัว (A ⊕ B)

คนส่วนใหญ่จำสูตรตัวบวกเต็มได้ แล้วเติมขีดแค่ที่ A ตัวเดียว กลายเป็น $B_{out} = \bar{A}\cdot B + B_{in}\cdot(A \oplus B)$ — **ลืมขีดบนวงเล็บ**

สูตรที่ลืมขีดนี้ **ผิด 3 แถวจาก 8 แถว** ครับ ลองดูแถวสุดท้ายของตาราง คือ A=1, B=1, B_{in}=1

	สูตรที่ลืมขีด	สูตรเต็ม
พจน์แรก $\bar{A}\cdot B$	$0 \cdot 1 = 0$	$0 \cdot 1 = 0$
พจน์ที่สอง	$1 \cdot (1 \oplus 1) = 1 \cdot 0 = 0$	$1 \cdot \overline{(1 \oplus 1)} = 1 \cdot 1 = 1$
ได้ B_{out}	`0` — ผิด	`1` — ถูก

เช็กด้วยเลขจริง: 1 − 1 − 1 = −1 ติดลบ**ต้องยืมแน่นอน** คำตอบจึงต้องเป็น 1

วิธีจำ: ตัวลบเต็มมีขีดกลับค่า **2 จุด** — ที่ A และที่วงเล็บ ถ้าขีดได้จุดเดียวแปลว่าลืม

เช็กลิสต์ตัวลบ

1 **ดูขาเข้าก่อนว่าครึ่งหรือเต็ม**

2 ขา (A, B) = ตัวลบครึ่ง · 3 ขา (มี B_{in} ด้วย) = ตัวลบเต็ม

2 **เขียนผลต่างก่อนเลย เพราะง่ายที่สุด**

Δ ใช้ XOR เหมือนผลบวกทุกประการ — ครึ่ง: $A \oplus B$ · เต็ม: $A \oplus B \oplus B_{in}$

3 **เขียนตัวยืม โดยลอกสูตรตัวทดมาแล้วเติมขีด**

ครึ่ง: $A\cdot B \rightarrow \bar{A}\cdot B$ · เต็ม: $A\cdot B + C_{in}(A \oplus B) \rightarrow \bar{A}\cdot B + B_{in}\cdot\overline{(A \oplus B)}$

4 **ถ้าลบหลายบิต ให้ไล่จากขวาไปซ้าย แล้วหยุด**

B_{out} ของหลักขวา → B_{in} ของหลักซ้ายถัดไป · **หยุดตรงนี้** ถ้าตัวยืมออกของหลักซ้ายสุดเป็น 1 แปลว่าคำตอบติดลบ ซึ่งเป็นคนละเรื่องกับที่เรากำลังทำ ไม่ต้องไปคิดต่อ

6 ตัวเปรียบเทียบ — เท่ากันไหม แล้วใครมากกว่า

เริ่มจากการเทียบเลขบัตรสองใบ

สมมติน้องถือบัตรสองใบ แล้วอยากรู้ว่าเลขบนบัตรสองใบนี้**เป็นเลขเดียวกันไหม** น้องทำยังไงครับ

น้องก็เทียบทีละหลักใช่ไหมครับ — หลักแรกตรงกันไหม หลักที่สองตรงกันไหม ไล่ไปจนครบ

แล้วจะสรุปว่า "เป็นเลขเดียวกัน" ได้เมื่อไหร่ — เมื่อ**ตรงกันครบทุกหลัก** ถ้ามีหลักไหนไม่ตรงแม้แต่หลักเดียว ก็ไม่ใช่เลขเดียวกันแล้ว

ลองก่อน (คิดในหัวพอ)

ในบทที่แล้วน้องรู้จักเกต 7 ตัว
คำถามที่ 1: เกตตัวไหนที่ให้ผลเป็น **1 เมื่อขาเข้าสองขาเหมือนกัน**
คำถามที่ 2: ถ้ามีสัญญาณ 4 เส้น แล้วอยากได้ผลเป็น 1 **ก็ต่อเมื่อทั้ง 4 เส้นเป็น 1 หมด** ต้องใช้เกตอะไร

เฉลย:
ข้อ 1 — **XNOR** ครับ (XOR ที่เติมฟองกลับค่า) มันให้ 1 เมื่อขาเข้า**เหมือนกัน**
ข้อ 2 — **AND** ที่มี 4 ขาเข้าครับ เพราะ AND ต้องการ 1 ครบทุกขา
สองคำตอบนี้ต่อกันเป็นวงจรเปรียบเทียบได้เลย ไม่ต้องมีอะไรมากกว่านี้

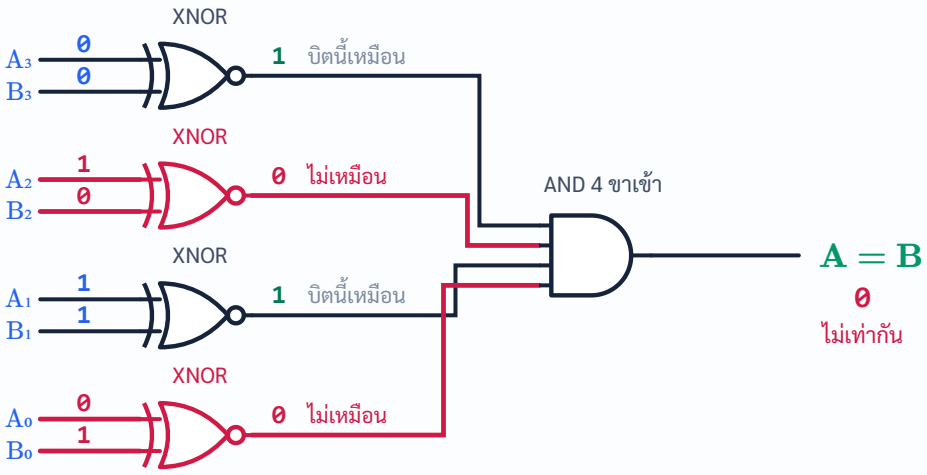
ตั้งชื่อ และประกอบวงจร

วงจรที่เอาเลขสองจำนวนมาเทียบกัน เรียกว่า **ตัวเปรียบเทียบ** หรือ **comparator** ครับ

หน้าที่ของมันมีได้ถึง 3 อย่าง — บอกว่า**เท่ากัน** (A = B) · บอกว่า**ตัวแรกมากกว่า** (A > B) · บอกว่า**ตัวแรกน้อยกว่า** (A < B)

เริ่มจากอย่างแรกก่อน เพราะง่ายที่สุด — **XNOR หนึ่งตัวต่อหนึ่งบิต** แล้วเอาผลทั้งหมดมา **AND** รวมกัน

อ่านวงจรนี้เป็นภาษาคนได้ว่า "บิต 3 เหมือนกัน **และ** บิต 2 เหมือนกัน **และ** บิต 1 เหมือนกัน **และ** บิต 0 เหมือนกัน" ซึ่งก็คือ "เลขสองตัวนี้เท่ากัน" พอดีครับ



ตัวเปรียบเทียบความเท่ากัน 4 บิต · ตัวอย่าง A = 0110 , B = 0011

เส้นแดงคือบิตที่ไม่ตรงกัน · พอมี 0 เข้า AND แม้แต่เส้นเดียว ผลรวมก็เป็น 0 ทันที

บิต	A กับ B	XNOR บอกว่า	ได้
บิต 3	0 กับ 0 — เหมือนกัน	ผ่าน	1
บิต 2	1 กับ 0 — **ต่างกัน**	ไม่ผ่าน	0
บิต 1	1 กับ 1 — เหมือนกัน	ผ่าน	1
บิต 0	0 กับ 1 — **ต่างกัน**	ไม่ผ่าน	0
AND รวมทั้งสี่เส้น — ต้องเป็น 1 ครบทุกเส้นถึงจะได้ 1			0

เช็กคำตอบกลับเข้าโจทย์: วงจรตอบ A = B เป็น 0 แปลว่า "ไม่เท่ากัน"

เลขจริงคือ 0110 = 6 และ 0011 = 3 ซึ่งไม่เท่ากันจริง ๆ ครับ ตรงกัน

ในสไลด์มีวาดอีกแบบด้วย — XOR แล้วต่อ NOR

อีกแบบคือใช้ **XOR** รายบิต (ซึ่งให้ 1 เมื่อ**ต่างกัน**) แล้วเอาทุกเส้นเข้า **NOR**

อ่านเป็นภาษาคนว่า "ไม่มีบิตไหนต่างกันเลย" ซึ่งแปลว่าเท่ากันเหมือนกันครับ
ทั้งสองแบบให้คำตอบตรงกันทุกกรณี เลือกใช้แบบไหนก็ได้ · **จำง่าย ๆ ว่า XNOR คู่กับ AND และ XOR คู่กับ NOR** ห้ามจับคู่สลับกัน

ทีนี้ถามยากขึ้น — ใครมากกว่า

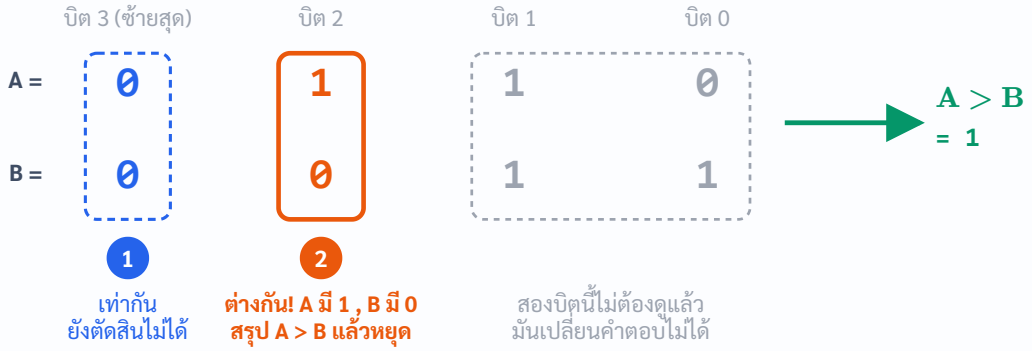
เวลาน้องเทียบว่าเลขไหนมากกว่ากัน เช่น 5,318 กับ 5,291 น้องดูหลักไหนก่อนครับ

น้องดู**หลักซ้ายสุด**ก่อนเสมอ เพราะหลักซ้ายมีค่ามากที่สุด ถ้าหลักซ้ายต่างกันเมื่อไหร่ ก็**ตัดสินได้ทันที**โดยไม่ต้องดูหลักที่เหลือเลย

ในตัวอย่างนั้น หลักพันเท่ากัน (5 ทั้งคู่) หลักร้อยคือ 3 กับ 2 — ต่างกันแล้ว จึงสรุปได้เลยว่า 5,318 มากกว่า โดยไม่ต้องดูหลักสิบกับหลักหน่วย

เลขฐานสองใช้หลักการเดียวกันเป๊ะครับ — **ไล่จากบิตซ้ายสุด เจอคู่แรกที่ต่างกันแล้วหยุด**

และเมื่อเจอคู่ที่ต่างกัน ใครมากกว่า — **ฝั่งที่เป็น 1 มากกว่า** เพราะ 1 มากกว่า 0



กวาดจากบิตซ้ายสุด · ① บิต 3 เท่ากัน จึงไปต่อ · ② บิต 2 ต่างกัน ตัดสินได้ทันทีแล้วหยุด
เลขจริงคือ 6 กับ 3 ซึ่ง 6 มากกว่าจริง

เขียนเป็นนิพจน์ — ตัวเปรียบเทียบขนาด 2 บิต

ทีนี้เอาความคิด "กวาดจากซ้าย" มาเขียนเป็นสูตรครับ ใช้เลข 2 บิตเพื่อให้สั้นก่อน คือ $A = A_1A_0$ และ $B = B_1B_0$ ตามโจทย์ในสไลด์

เริ่มจาก A = B ก่อน เพราะง่ายสุด — ต้องเหมือนกันทั้งสองบิต

$$E = (A_1 \odot B_1) \cdot (A_0 \odot B_0)$$

เครื่องหมาย $\odot$ คือ XNOR · อ่านว่า "บิตบนเหมือน และ บิตล่างเหมือน"

ต่อด้วย A > B — มีสองทางที่ทำให้ A มากกว่าครับ ต้องไล่ให้ครบทั้งสองทาง

ทางที่	เกิดขึ้นเมื่อ	เขียนเป็นนิพจน์
ทางที่ 1	บิตซ้ายตัดสินเลย — A_1 เป็น 1 แต่ B_1 เป็น 0	$A_1 \cdot \bar{B}_1$
ทางที่ 2	บิตซ้าย**เท่ากัน** จึงต้องไปดูบิตขวา แล้วบิตขวา A_0 เป็น 1 ส่วน B_0 เป็น 0	$(A_1 \odot B_1) \cdot A_0 \cdot \bar{B}_0$

$$G = A_1\bar{B}_1 + (A_1 \odot B_1) \cdot A_0\bar{B}_0$$

ทางที่ 1 หรือ ทางที่ 2

ส่วน A < B ใช้ความคิดเดียวกันแต่สลับข้าง — ที่ไหนมี A ก็เปลี่ยนเป็น B ที่ไหนมี B ก็เปลี่ยนเป็น A

$$L = \bar{A}_1 B_1 + (A_1 \odot B_1)\cdot \bar{A}_0 B_0$$

วิธีตรวจว่าสามสูตรนี้ถูกไหม โดยไม่ต้องไล่ 16 แถว

เลขสองจำนวนใด ๆ ต้องเป็น**อย่างใดอย่างหนึ่งเท่านั้น** — มากกว่า หรือ น้อยกว่า หรือ เท่ากัน จะเป็นสองอย่างพร้อมกันไม่ได้ และจะไม่เป็นสักอย่างก็ไม่ได้

เพราะฉะนั้น**ไม่ว่าจะป้อนอะไรเข้าไป ขาออกทั้งสามต้องมีตัวที่เป็น 1 อยู่ตัวเดียวเป๊ะ**
ถ้าน้องลองแทนค่าแล้วได้ 1 สองตัว หรือได้ 0 ทั้งสามตัว แปลว่าสูตรผิดแน่นอนครับ — เป็นวิธีเช็กตัวเองที่เร็วที่สุดในห้องสอบ

ของจริง — ชิป 74HC85 และการต่อพ่วง

ตัวเปรียบเทียบขนาด 4 บิตมีขายเป็นชิปชื่อ **74HC85** ครับ มีขาเข้า A_3–A_0 กับ B_3–B_0 และขาออกสามเส้นคือ A>B , A=B , A<B

ถ้าอยากเทียบเลข 8 บิต ก็ใช้สองตัวต่อพ่วงกัน โดยเอาขาออกสามเส้นของชิปที่ดูแล**บิตล่าง** ไปเสียบขาเข้าสามเส้นของชิปที่ดูแล**บิตบน**

ทำไมต้องต่อทางนั้น — เพราะ**บิตบนสำคัญกว่า**ครับ ชิปตัวบนจะตัดสินก่อน แล้วจะหันไปฟังผลของบิตล่าง**ก็ต่อเมื่อบิตบนของมันเท่ากัน**เท่านั้น ซึ่งก็คือกฎ "กวาดจากซ้าย" ที่เราเพิ่งเรียนไป แต่ขยายไปทั้งชิป

กับดักข้อสอบ · เทียบจากบิตขวาสุด

เวลาบวกเลข เราไล่จาก**ขวาไปซ้าย** พอมาถึงหัวข้อนี้หลายคนเลยไล่ทางเดิมโดยอัตโนมัติ ซึ่ง**ผิด**ครับ

ลองใช้ตัวอย่างเดิม A = 0110 กับ B = 0011 ดู
ถ้าเริ่มจากบิต 0: $A_0 = 0$ ส่วน $B_0 = 1$ — เจอคู่ที่ต่างกันทันที แล้วสรุปว่า **A < B**

แต่ความจริงคือ 6 มากกว่า 3 ครับ — **ตอบผิดกลับด้านเลย**

วิธีจำ: การบวกไล่จากขวา เพราะตัวทดวิ่งไปทางซ้าย · การเทียบไล่จากซ้าย เพราะบิตซ้ายมีน้ำหนักมากกว่า — **คนละทิศกันเสมอ**

เช็กลิสต์ตัวเปรียบเทียบ

1

ดูก่อนว่าโจทย์ถามแค่ "เท่ากันไหม" หรือถามว่า "ใครมากกว่า"

ถามแค่เท่ากัน → ใช้ XNOR รายบิต แล้ว AND รวม · จบแค่นั้น ไม่ต้องทำอะไรเพิ่ม

2

ถ้าถามว่าใครมากกว่า ให้กวาดจากบิตซ้ายสุด

เจอคู่แรกที่ต่างกันเมื่อไหร่ ให้**หยุดทันที** · ฝั่งที่เป็น 1 คือฝั่งที่มากกว่า · ถ้าไล่จนหมดแล้วไม่เจอคู่ที่ต่างเลย แปลว่าเท่ากัน

3

ถ้าต้องเขียนนิพจน์ ให้ไล่ "ทาง" ทีละทาง

บิตซ้ายสุดตัดสินเอง 1 ทาง · บิตซ้ายเท่ากันแล้วบิตถัดมาตัดสิน อีก 1 ทาง · ไล่ลงไปเรื่อย ๆ แล้วเอาทุกทางมา OR กัน

4

ตรวจก่อนส่ง แล้วหยุด

แทนค่าเลขสักคู่แล้วดูว่าขาออกสามเส้นมี 1 อยู่**เส้นเดียว**ไหม · **หยุดตรงนี้** ไม่ต้องไปลดรูปนิพจน์ให้สั้นลงถ้าโจทย์ไม่ได้สั่ง

7 ตัวเข้ารหัส — 10 ปุ่ม เหลือ 4 เส้น

เริ่มจากปุ่มลิฟต์

ลิฟต์ตึกหนึ่งมีปุ่ม 8 ปุ่ม คือชั้น 0 ถึงชั้น 7 · เวลาน้องกดปุ่มชั้น 5 สายไฟจากปุ่มนั้นก็ส่งสัญญาณไปบอกเครื่องควบคุม

ถ้าลากสายจากทุกปุ่มไปเครื่องควบคุมตรง ๆ ต้องใช้สาย **8 เส้น** ใช่ไหมครับ

แต่สายมันแพง และช่องเสียบบนแผงวงจรก็มีจำกัด เราเลยอยาก**ใช้สายให้น้อยที่สุด** แต่ยังบอกได้ครบว่ากดปุ่มไหน

ลองก่อน (คิดในหัวพอ)

ถ้าสายแต่ละเส้นส่งได้แค่ 0 กับ 1
คำถาม: ต้องใช้สาย**กี่เส้น** ถึงจะแยกแยะได้ครบทั้ง 8 ปุ่ม

เฉลย: 3 เส้น ครับ
เพราะสาย 3 เส้น แต่ละเส้นเป็นได้ 2 ค่า จึงมีรูปแบบทั้งหมด 2 × 2 × 2 = 8 แบบ ซึ่งพอดีกับ 8 ปุ่มเป๊ะ และ 8 แบบนั้นก็คือเลขฐานสอง 000, 001, 010, ... , 111 นั่นเอง — แปลว่าเราแค่ส่ง**เลขของชั้น**ไปแทนที่จะส่งว่า "ปุ่มไหนถูกกด"

ตั้งชื่อ และหาว่าเกตไหนต่อกับปุ่มไหน

วงจรที่รับสัญญาณจากหลายเส้น แล้วแปลงเป็น**รหัสเลขฐานสอง**ที่สั้นกว่า เรียกว่า **ตัวเข้ารหัส** หรือ **encoder** ครับ

แบบที่รับ 8 เส้นแล้วคายออก 3 เส้น เรียกว่า **8-to-3 encoder**

ทีนี้มาหากันว่าต้องต่อยังไง — วิธีคิดคือ**ถามทีละเส้นขาออก**ครับ ไม่ใช่ถามทีละปุ่ม

ถามแบบนี้: "เส้นขาออก A_0 จะเป็น 1 ตอนกดปุ่มไหนบ้าง"

A_0 คือบิตขวาสุดของรหัส ซึ่งเป็น 1 ในเลข 1, 3, 5, 7 (คือเลขคี่) เพราะฉะนั้น $\mathbf{A_0 = D_1}$ **หรือ** $\mathbf{D_3}$ **หรือ** $\mathbf{D_5}$ **หรือ** $\mathbf{D_7}$ — คำว่า "หรือ" แปลเป็น OR

ทำแบบเดียวกันกับ A_1 และ A_2 ก็ได้ครบทั้งสามเส้น

ปุ่มที่กด	A_2	A_1	A_0
D_0	0	0	0
D_1	0	0	1
D_2	0	1	0
D_3	0	1	1
D_4	1	0	0
D_5	1	0	1
D_6	1	1	0
D_7	1	1	1

วิธีอ่านตารางนี้ให้ได้นิพจน์: ให้ดู**ลงตามคอลัมน์** ไม่ใช่อ่านตามแถวครับ

คอลัมน์ A_2 เป็น 1 ที่แถว D_4, D_5, D_6, D_7 → $\mathbf{A_2 = D_4 + D_5 + D_6 + D_7}$

คอลัมน์ A_1 เป็น 1 ที่แถว D_2, D_3, D_6, D_7 → $\mathbf{A_1 = D_2 + D_3 + D_6 + D_7}$

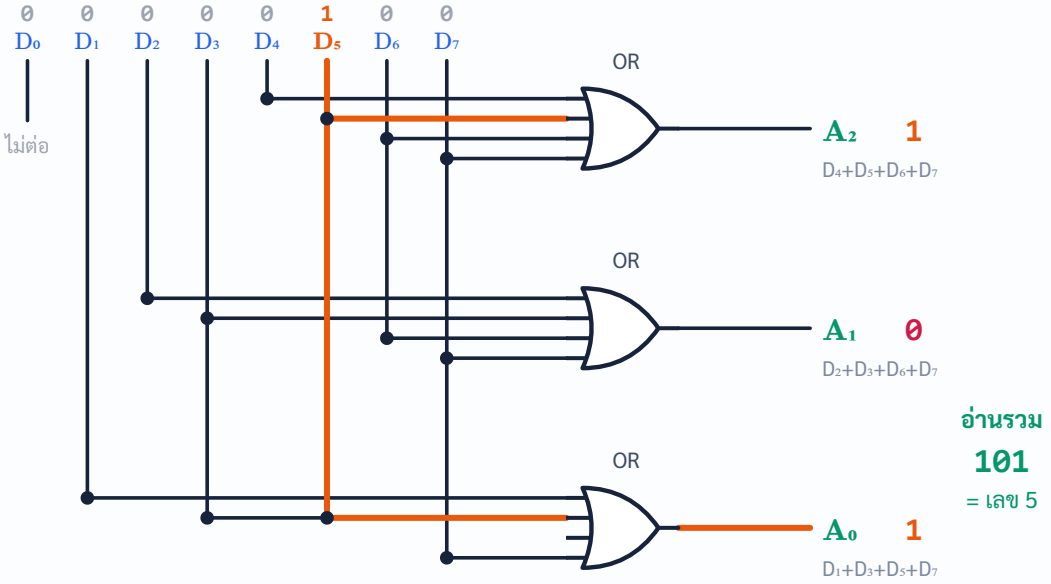
คอลัมน์ A_0 เป็น 1 ที่แถว D_1, D_3, D_5, D_7 → $\mathbf{A_0 = D_1 + D_3 + D_5 + D_7}$

เครื่องหมาย + ในวงจรลอจิกคือ **OR** นะครับ ไม่ใช่การบวกเลข

ปุ่ม D_0 ไม่ได้ต่อเข้าเกตไหนเลย — ไม่ได้ลืมวาด

ดูแถว D_0 ในตารางสิครับ รหัสของมันคือ 000 คือ**ไม่มีบิตไหนเป็น 1 เลย**
แปลว่าถ้าไม่กดอะไรเลย สายทั้งสามก็เป็น 0 อยู่แล้ว ซึ่งอ่านได้ว่า "0" พอดี
ปุ่ม D_0 จึงไม่ต้องต่อไปไหน · นี่คือ**ข้อเสีย**ของตัวเข้ารหัสแบบธรรมดาด้วย — มันแยกไม่ออกระหว่าง "กดปุ่ม 0" กับ "ไม่ได้กดอะไรเลย"

วงจรจริง — พร้อมเดินตัวอย่าง กดปุ่ม D_5



ตัวเข้ารหัส 8-to-3 · เส้นแนวตั้งคือสายจากปุ่มแต่ละปุ่ม จุดดำคือจุดที่สายต่อเข้าเกตจริง ๆ · เส้นที่ตัดกันโดยไม่มีจุดดำแปลว่าไม่ได้ต่อกัน

เส้นส้มคือตอนกดปุ่ม D_5 — มันแตกไปเข้า OR ตัวบนกับ OR ตัวล่าง แต่ไม่ได้เข้า OR ตัวกลาง

ขั้น	ทำอะไร	ได้อะไร
①	กดปุ่ม D_5 — สาย D_5 เป็น 1 ส่วนสายอื่นเป็น 0 หมด	D_5 = `1`
②	OR ตัวบน (A_2) มีขา D_5 ต่ออยู่ — **มีขาที่เป็น 1**	A_2 = `1`
③	OR ตัวกลาง (A_1) ต่อกับ D_2, D_3, D_6, D_7 — **ไม่มี D_5 อยู่เลย** ทุกขาเป็น 0	A_1 = `0`
④	OR ตัวล่าง (A_0) มีขา D_5 ต่ออยู่ — **มีขาที่เป็น 1**	A_0 = `1`

เช็กคำตอบกลับเข้าโจทย์: อ่าน $A_2A_1A_0$ ได้ **101** ซึ่งในฐานสองคือเลข 5

และเรากดปุ่ม D_5 ซึ่งคือชั้น 5 พอดีครับ ตรงกัน

ตัวเข้ารหัสเลขฐานสิบเป็น BCD

ทีนี้เปลี่ยนโจทย์เป็นแป้นตัวเลข 0 ถึง 9 คือ **10 ปุ่ม** ครับ

รหัสที่ใช้ตอนขาออกเรียกว่า **BCD** ซึ่งย่อมาจาก Binary Coded Decimal แปลว่า "เอาเลขฐานสิบหนึ่งหลักมาเขียนเป็นเลขฐานสอง 4 บิต"

เช่น เลข 7 เขียนเป็น BCD คือ 0111 · เลข 9 คือ 1001 — ก็คือเลขฐานสองธรรมดานั่นแหละครับ แต่**ใช้แค่ 0 ถึง 9** ส่วนรหัส 1010 ถึง 1111 ไม่ใช้เลย เพราะไม่มีเลขฐานสิบหลักเดียวที่มีค่าเท่านั้น

วิธีหานิพจน์เหมือนเดิมเป๊ะครับ — ดูลงตามคอลัมน์ว่าเลขไหนบ้างที่บิตนั้นเป็น 1

ปุ่ม	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

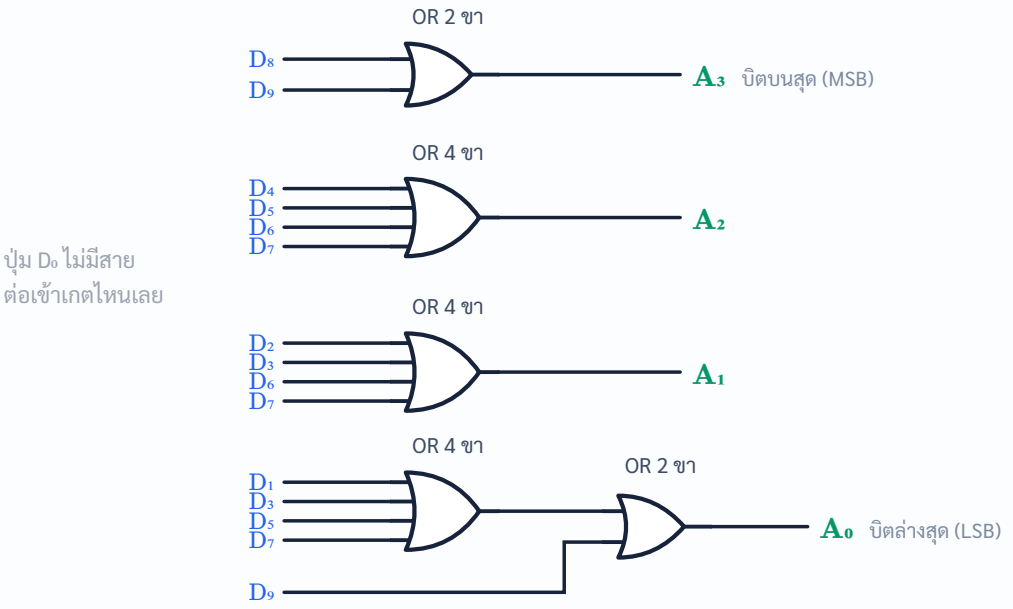
คอลัมน์ A_3 เป็น 1 แค่สองแถว คือ 8 กับ 9 · จึงใช้ OR แค่ **2 ขาเข้า**

คอลัมน์ A_2 เป็น 1 ที่ 4, 5, 6, 7 · ใช้ OR 4 ขาเข้า

คอลัมน์ A_1 เป็น 1 ที่ 2, 3, 6, 7 · ใช้ OR 4 ขาเข้า

คอลัมน์ A_0 เป็น 1 ที่ 1, 3, 5, 7, 9 (เลขคี่ทั้งหมด) · ต้อง OR ถึง **5 ตัว** ซึ่งถ้าในลิ้นชักมีแต่ OR 4 ขา ก็ใช้ OR 4 ขาหนึ่งตัวก่อน แล้วเอาผลไป OR กับตัวที่เหลืออีกทีได้ครับ (OR ต่อกันหลายชั้น ผลเหมือน OR ตัวใหญ่ตัวเดียวเป๊ะ)

และเหมือนเดิมครับ — **ปุ่ม 0 ไม่ต่อเข้าเกตไหนเลย**



ตัวเข้ารหัสเลขฐานสิบเป็น BCD · OR 4 ตัว ตัวละหนึ่งบิตขาออก · ชื่อปุ่มที่เขียนไว้ที่ขาเกตแต่ละขา คือสายที่ต้องลากมาเข้า บิต A_0 ต้องรวมสัญญาณ 5 เส้น จึงใช้ OR 4 ขาหนึ่งตัวแล้วต่อ OR 2 ขาอีกตัว

กับดักข้อสอบ · กดสองปุ่มพร้อมกันแล้วได้คำตอบมั่ว

วงจรนี้ตั้งอยู่บนสมมติฐานเงียบ ๆ ข้อหนึ่ง คือ**มีปุ่มเดียวที่ถูกกดในแต่ละครั้ง** ถ้าสมมติฐานนี้พัง คำตอบก็พังทันที

ลองกด **D_3 พร้อมกับ D_5** ในวงจร 8-to-3 ดูครับ

ขาออก	ต่อกับปุ่มอะไรบ้าง	มี D_3 หรือ D_5 ไหม	ได้
A_2	D_4, D_5, D_6, D_7	มี D_5	1
A_1	D_2, D_3, D_6, D_7	มี D_3	1
A_0	D_1, D_3, D_5, D_7	มีทั้งสองตัว	1

อ่านรวมได้ **111 = เลข 7** — ซึ่ง**ไม่ใช่ทั้ง 3 และไม่ใช่ทั้ง 5** เป็นเลขที่ไม่มีใครกดเลยด้วยซ้ำ

ถ้าไล่ดูทุกคู่ที่เป็นไปได้ จะมีทั้งหมด 28 คู่ และ **9 คู่** ให้คำตอบที่ไม่ตรงกับปุ่มไหนเลย · ที่เหลือบังเอิญตรงกับปุ่มใดปุ่มหนึ่ง ซึ่งก็ยังเดาไม่ได้อยู่ดี

ทางแก้ — ตัวเข้ารหัสแบบมีลำดับความสำคัญ

ทางแก้คือกำหนดกติกาล่วงหน้าว่า ถ้ากดหลายปุ่มพร้อมกัน ให้**ถือเอาปุ่มที่เลขมากที่สุด** แล้วเมินปุ่มที่เหลือไปเลย

วงจรที่มีกติกานี้เรียกว่า **ตัวเข้ารหัสแบบมีลำดับความสำคัญ** หรือ **priority encoder** ครับ

กลับไปที่ตัวอย่างเมื่อกี้ — กด D_3 พร้อม D_5 · ตัวแบบมีลำดับจะตอบ **101 = 5** เพราะ 5 มากกว่า 3 · คำตอบนี้**ใช้งานได้จริง** ต่างจาก 7 ที่ไม่ได้แปลว่าอะไรเลย

ชิปจริงสองตัวที่สไลด์พูดถึง · และเรื่อง "ทำงานตอนเป็น 0"

74HC147 — ตัวเข้ารหัสเลขฐานสิบเป็น BCD แบบมีลำดับความสำคัญ มีขาเข้า 9 ขา (ไม่มีขาของเลข 0 ด้วยเหตุผลที่เราคุยไปแล้ว) และขาออก 4 ขา

74LS148 — ตัวเข้ารหัส 8-to-3 แบบมีลำดับความสำคัญ มีขาช่วยขยายอีกสามขา คือ EI (เปิดใช้งาน) · EO (ส่งต่อ) · GS (บอกว่ามีคนกดอยู่)
เอาสองตัวมาต่อกันได้เป็น 16-to-4 โดยต่อขา EO ของตัวที่ดูแลเลขสูง ไปเข้าขา EI ของตัวที่ดูแลเลขต่ำ

เรื่องสำคัญที่ต้องระวัง: ชิปทั้งสองตัวนี้ใช้ขาเข้าและขาออกแบบ **active-low** ซึ่งแปลว่า "**ทำงานตอนเป็น 0**" ไม่ใช่ตอนเป็น 1 — ขาแบบนี้ในผังวงจรจะมีวงกลมเล็ก ๆ กำกับไว้ที่ขา บอกว่าขานั้นทำงานตอนเป็น 0
ในตารางของชิปจะเขียนด้วยตัว L (low) กับ H (high) แทน 0 กับ 1 และมีขีดบนหัวชื่อขา เช่น $\overline{D}_5$ แปลว่า "กดปุ่ม 5 = ดึงขานี้ลงเป็น 0"
ตรรกะข้างในยังเหมือนเดิมทุกอย่างครับ แค่กลับความหมายของระดับไฟ

เช็กลิสต์ตัวเข้ารหัส

1. **นับขาเข้ากับขาออกก่อน**
   ขาเข้ามากกว่าขาออก = ตัวเข้ารหัส · 10 เข้า 4 ออก = decimal-to-BCD · 8 เข้า 3 ออก = 8-to-3
2. **เขียนตารางรหัส แล้วอ่านลงตามคอลัมน์**
   คอลัมน์ไหนเป็น 1 ที่แถวไหนบ้าง เอาชื่อแถวพวกนั้นมา OR กัน · ทำทีละคอลัมน์จนครบ
3. **วาด OR หนึ่งตัวต่อหนึ่งบิตขาออก**
   จำนวนขาของ OR แต่ละตัว = จำนวน 1 ในคอลัมน์นั้น · **ขาเข้าตัวแรกสุด (D_0 หรือปุ่ม 0) ไม่ต้องต่อ**
4. **ถ้าโจทย์พูดถึงการกดพร้อมกัน ให้นึกถึงลำดับความสำคัญ แล้วหยุด**
   ตอบว่า "แบบธรรมดาจะได้คำตอบมั่ว จึงต้องใช้ priority encoder ที่เลือกเลขมากที่สุด" · **หยุดตรงนี้** ไม่ต้องไปวาดวงจรลำดับความสำคัญเต็ม ๆ ถ้าโจทย์ไม่ได้สั่ง

8 ตัวถอดรหัส — 3 เส้น แตกเป็น 8 เส้น

กลับหัวกลับหางจากหัวข้อที่แล้ว

หัวข้อ 7 เราบีบ 8 ปุ่มให้เหลือสาย 3 เส้น ส่งไปถึงเครื่องควบคุม

แต่ปลายทางต้อง**เอาไปใช้งานจริง**ครับ เช่นสั่งให้ลิฟต์จอดชั้น 5 ก็ต้องมีสายเส้นหนึ่ง ที่วิ่งไปเปิดกลไกของชั้น 5 โดยเฉพาะ

เพราะฉะนั้นปลายทางต้อง**คลายรหัส 3 เส้นกลับออกเป็น 8 เส้น** แล้วทำให้มี**เส้นเดียวที่ทำงาน** คือเส้นของชั้น 5

ลองก่อน (คิดในหัวพอ)

สายสามเส้นที่เข้ามาชื่อ A_2, A_1, A_0 และตอนนี้มีค่า **1, 0, 1** ตามลำดับ
คำถาม: ถ้าน้องอยากได้ไฟดวงหนึ่งที่ติด**เฉพาะตอนที่รหัสเป็น 101 เท่านั้น** น้องต้องเช็กเงื่อนไขอะไรบ้าง และเชื่อมเงื่อนไขพวกนั้นด้วยคำว่าอะไร

เฉลย: ต้องเช็กสามเงื่อนไขครับ — **A_2 เป็น 1** และ **A_1 เป็น 0** และ **A_0 เป็น 1**
เชื่อมด้วยคำว่า "**และ**" ทั้งหมด เพราะต้องจริงครบทั้งสามข้อ ไม่งั้นไฟไม่ติด
คำว่า "และ" แปลเป็นเกต **AND** · ส่วนเงื่อนไข "A_1 เป็น 0" ก็แค่เอา A_1 ไปผ่าน NOT ก่อน

ตั้งชื่อ และหากฎการต่อ

วงจรที่รับรหัสเข้ามาแล้วทำให้**มีขาออกเดียวที่ทำงาน** เรียกว่า **ตัวถอดรหัส** หรือ **decoder** ครับ

แบบที่รับ 3 เส้นแล้วมีขาออก 8 เส้น เรียกว่า **3-to-8 decoder** หรืออีกชื่อว่า **1-of-8 decoder** เพราะมันเลือก 1 เส้นจาก 8 เส้น

กฎการต่อง่ายมากครับ — **AND หนึ่งตัวต่อหนึ่งเส้นขาออก** และ AND ตัวนั้นรับ A ทั้งสามตัวเข้ามา

แต่ละ AND จะต่อกับ **A** ตรง ๆ หรือต่อกับ **$\bar{A}$** (ที่ผ่าน NOT แล้ว) **ขึ้นกับว่ารหัสของเส้นนั้นบิตนี้เป็น 1 หรือ 0**

พูดง่าย ๆ คือ บิตไหนในรหัสเป็น **1** ก็ต่อ A ตัวนั้นเข้าไปเลย · บิตไหนเป็น **0** ก็ต่อ $\bar{A}$ ของตัวนั้นเข้าไปแทน

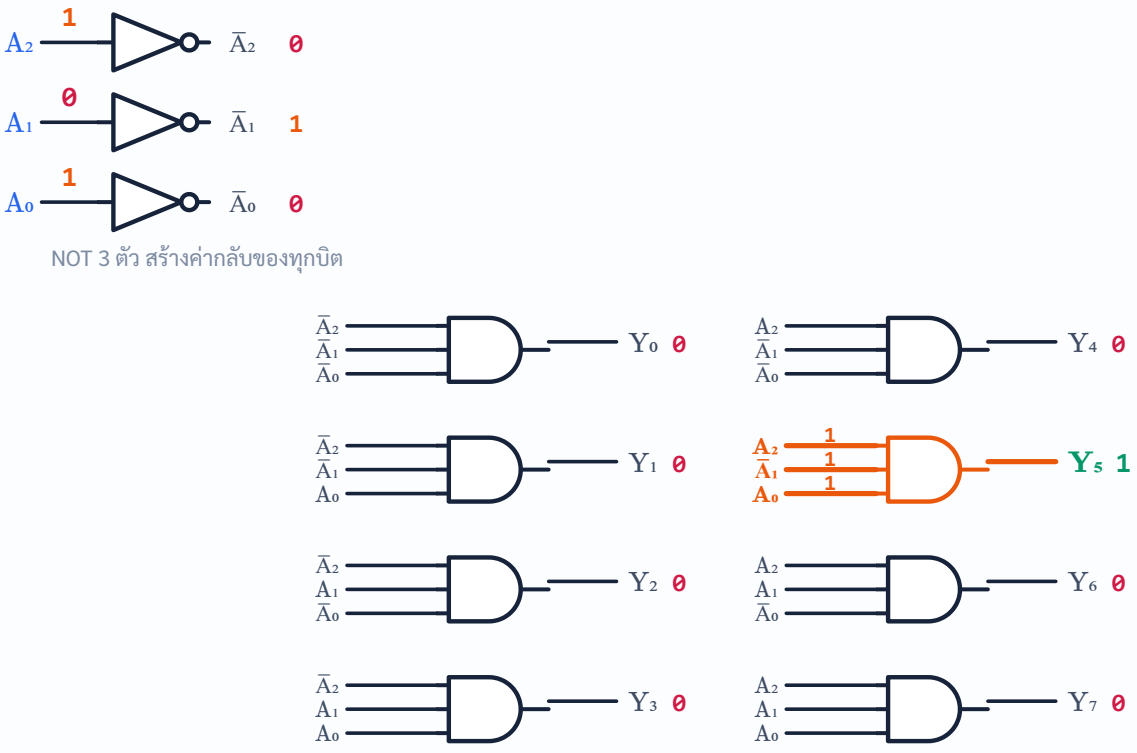
A_2	A_1	A_0	เส้นที่ติด	นิพจน์ของ AND ตัวนั้น
0	0	0	Y_0	$\bar{A}_2 \cdot \bar{A}_1 \cdot \bar{A}_0$
0	0	1	Y_1	$\bar{A}_2 \cdot \bar{A}_1 \cdot A_0$
0	1	0	Y_2	$\bar{A}_2 \cdot A_1 \cdot \bar{A}_0$
0	1	1	Y_3	$\bar{A}_2 \cdot A_1 \cdot A_0$
1	0	0	Y_4	$A_2 \cdot \bar{A}_1 \cdot \bar{A}_0$
1	0	1	Y_5	$A_2 \cdot \bar{A}_1 \cdot A_0$
1	1	0	Y_6	$A_2 \cdot A_1 \cdot \bar{A}_0$
1	1	1	Y_7	$A_2 \cdot A_1 \cdot A_0$

ลองดูคอลัมน์ขวาสุดเทียบกับสามคอลัมน์ซ้ายสิครับ — ตรงไหนเป็น **0** จะมี**ขีดบนหัว** ตรงไหนเป็น **1** จะ**ไม่มีขีด**

ตัวอย่าง Y_5 มีรหัส 1-0-1 จึงได้ A_2 (ไม่มีขีด) · $\bar{A}_1$ (มีขีด) · A_0 (ไม่มีขีด)

สังเกตว่าทั้ง 8 นิพจน์ใช้ตัวแปรครบทั้งสามตัวเสมอ ไม่มีตัวไหนขาดไป — ถ้าน้องเขียนแล้วมี AND ตัวไหนที่มีแค่สองตัวแปร แปลว่าผิดแน่นอน

วงจรจริง — พร้อมเดินตัวอย่าง ป้อน 101



NOT 3 ตัว สร้างค่ากลับของทุกบิต

ตัวถอดรหัส 3-to-8 · NOT 3 ตัวทางซ้ายสร้าง $\bar{A}_2$, $\bar{A}_1$, $\bar{A}_0$ ไว้ให้ใช้ร่วมกัน ชื่อที่เขียนไว้ที่ขาของ AND แต่ละตัว คือสายที่ต้องลากมาจากซ้าย

ป้อน 101 แล้ว AND ตัวส้ม (Y_5) เป็นตัวเดียวที่ขาเข้าเป็น 1 ครบทั้งสามขา จึงมีเส้นเดียวที่ติด

ขั้น	ทำอะไร	ได้อะไร
①	ป้อน A_2=1, A_1=0, A_0=1 · NOT สามตัวสร้างค่ากลับให้ทันที	$\bar{A}_2$= 0 , $\bar{A}_1$= 1 , $\bar{A}_0$= 0
②	AND ของ Y_5 ต่อกับ A_2 , $\bar{A}_1$, A_0 — ค่าคือ 1, 1, 1	Y_5 = 1

③	AND ของ Y_4 ต่อกับ A_2 , $\bar{A}_1$, $\bar{A}_0$ — ขาสุดท้ายเป็น 0	Y_4 = `0`
④	AND ตัวอื่น ๆ ก็มีอย่างน้อยหนึ่งขาที่เป็น 0 เหมือนกันหมด	Y_0–Y_3, Y_6, Y_7 = `0`

เช็กคำตอบกลับเข้าโจทย์: รหัส 101 คือเลข 5 และเส้นที่ติดคือ Y_5 เส้นเดียว ตรงกันครับ

และสังเกตว่า**ไม่ว่าป้อนรหัสอะไรเข้าไป จะมีเส้นที่เป็น 1 อยู่เส้นเดียวเสมอ** — นี่คือวิธีเช็กงานของตัวเองที่เร็วที่สุด ถ้าได้สองเส้นหรือไม่ได้เลย แปลว่าต่อผิด

ขาออกแบบ "ทำงานตอนเป็น 0" — active-low

ชิปตัวถอดรหัสของจริงส่วนใหญ่ **ไม่ได้**ทำให้เส้นที่ถูกเลือกเป็น 1 ครับ มันทำกลับกัน คือทำให้เส้นที่ถูกเลือก**เป็น 0** ส่วนเส้นที่เหลือเป็น 1 หมด

วิธีทำง่ายมาก — **เปลี่ยน AND ทุกตัวเป็น NAND** เท่านั้นเอง เพราะ NAND ก็คือ AND ที่เติมฟองกลับค่าตรงขาออก

เราเรียกขาออกแบบนี้ว่า **active-low** — คำเดียวกับที่เจอตอน 74HC147/74LS148 ในหัวข้อตัวเข้ารหัส แปลตรงตัวว่า "ทำงานตอนระดับไฟต่ำ" และในสไลด์จะเขียนชื่อขาพร้อม**ขีดบนหัว** เช่น $\overline{Y_5}$

ทำไมของจริงชอบทำแบบนี้ — เพราะอุปกรณ์หลายอย่างอย่างหลอด LED ถูกออกแบบให้**ติดเมื่อถูกดึงลงกราวด์** การให้ขาออกเป็น 0 จึงต่อใช้งานได้ตรง ๆ โดยไม่ต้องมีวงจรเพิ่ม

ชิปจริงสองตัวในสไลด์

74HC138 — ตัวถอดรหัส 3-to-8 ขาออก**แบบ active-low** ทั้ง 8 เส้น
มีขาเปิดใช้งานอีก 3 ขา คือ $\overline{E_1}$ และ $\overline{E_2}$ (สองตัวนี้ active-low) กับ E_3 (active-high)
ทั้งสามขาต้องอยู่ในสถานะทำงานพร้อมกัน ชิปถึงจะยอมทำงาน ถ้าขาไหนขาดหนึ่งขา ขาออกจะเป็น 1 หมดทุกเส้น เหมือนไม่มีอะไรถูกเลือก

74HC42 — ตัวถอดรหัส**ฐานสองเป็นฐานสิบ** รับ BCD 4 บิต คายออก 10 เส้น แบบ active-low เช่นกัน · อีกชื่อคือ **4-line-to-10-line** หรือ **1-of-10 decoder**
ข้างในใช้ **NAND 10 ตัว** ตัวละหนึ่งเลข ด้วยกฎการต่อเดียวกับที่เราเพิ่งเรียนเป๊ะ แค่เปลี่ยนจาก 3 ตัวแปรเป็น 4 ตัวแปร

ตัวถอดรหัส BCD เป็นฐานสิบ — และรหัสที่ไม่มีอยู่จริง

ตัวนี้รับ 4 บิต ซึ่งเป็นรหัส BCD คือเลข 0 ถึง 9 แล้วคายออก 10 เส้น

กฎการต่อเหมือนเดิมทุกอย่างครับ เช่น เส้นของเลข 7 มีรหัส 0111 จึงเป็น $\bar{A}_3 \cdot A_2 \cdot A_1 \cdot A_0$ — บิตบนสุดเป็น 0 จึงใส่ขีด ที่เหลือเป็น 1 จึงไม่ใส่

แต่ 4 บิตสร้างรหัสได้ 16 แบบ ในขณะที่เรามีเส้นขาออกแค่ 10 เส้น แล้วรหัส **1010 ถึง 1111** ซึ่งเกินเลข 9 ไปแล้ว จะเกิดอะไรขึ้น

คำตอบคือ **ไม่มีเส้นไหนติดเลยสักเส้น**ครับ (หรือถ้าเป็นแบบ active-low ก็คือเป็น 1 ทั้งสิบเส้น)

เพราะแต่ละ AND ถูกออกแบบมาให้ตรงกับรหัสของตัวเองเป๊ะ ๆ เมื่อรหัสที่ป้อนเข้ามาไม่ตรงกับใครเลย ก็ไม่มีใครติด — ซึ่งเป็นพฤติกรรมที่ปลอดภัยดีครับ ดีกว่าไปติดเส้นมั่ว ๆ

กับดักข้อสอบ · ลืมขีดบนหัว

กับดักอันดับหนึ่งของหัวข้อนี้คือเขียน AND โดย**ไม่ใส่ขีดตรงบิตที่เป็น 0**

สมมติน้องเขียนเส้น Y_5 เป็น $A_2 \cdot A_1 \cdot A_0$ (ลืมขีดบน A_1) แล้วป้อน 101 เข้าไป
จะได้ $1 \cdot \mathbf{0} \cdot 1 = \mathbf{0}$ — ไฟไม่ติด ทั้งที่ควรติด

แล้วเส้นนั้นจะไปติดตอนไหนแทน — ตอนป้อน **111** ครับ ซึ่งชนกับเส้น Y_7 พอดี กลายเป็นว่ามีสองเส้นติดพร้อมกัน

วิธีเช็กเร็ว: หลังเขียนครบทุกเส้นแล้ว ลองนับว่า **มีกี่เส้นที่ทุกตัวแปรไม่มีขีดเลย** — ต้องมีเส้นเดียวเท่านั้น (คือเส้นบนสุด Y_7)
และ**มีกี่เส้นที่ทุกตัวแปรมีขีดหมด** — ต้องมีเส้นเดียวเช่นกัน (คือเส้นล่างสุด Y_0)

เช็กลิสต์ตัวถอดรหัส

1. **นับขาเข้ากับขาออก แล้วแยกให้ออกจากตัวเข้ารหัส**
   ขาออก**มากกว่า**ขาเข้า = ตัวถอดรหัส · 3 เข้า 8 ออก = 3-to-8 · 4 เข้า 10 ออก = BCD เป็นฐานสิบ
2. **วาด NOT ของทุกบิตขาเข้าไว้ก่อน**
   3 บิตใช้ NOT 3 ตัว · 4 บิตใช้ 4 ตัว · วาดไว้ทางซ้ายแล้วใช้ร่วมกันทุกเส้น
3. **เขียน AND ทีละเส้น โดยแปลรหัสเป็นขีด**
   บิตเป็น 1 → ใช้ตัวแปรตรง ๆ · บิตเป็น 0 → ใส่ขีดบนหัว · ทุก AND ต้องมีตัวแปร**ครบทุกตัว**
4. **ถ้าโจทย์บอกว่า active-low ให้เปลี่ยนเป็น NAND แล้วหยุด**
   เปลี่ยน AND เป็น NAND ทุกตัวเท่านั้น ตรรกะข้างในไม่ต้องแก้อะไรเลย · **หยุดตรงนี้** อย่าไปกลับค่าขาเข้าตามไปด้วย ฟองอยู่ที่**ขาออก**อย่างเดียว

9 จอ 7 ขีด — ทำให้เลขโผล่บนหน้าจอจริง

เริ่มจากตัวเลขบนเตาไมโครเวฟ

ลองนึกถึงตัวเลขสีแดงบนเตาไมโครเวฟหรือนาฬิกาปลุกหัวเตียงครับ ตัวเลขพวกนั้นไม่ได้วาดมาเป็นรูปเลข มันคือ**หลอดไฟเส้นตรงสั้น ๆ 7 เส้น** ที่วางเรียงกันเป็นรูปเลข 8 แล้วเลือกจุดบางเส้นให้ติด บางเส้นให้ดับ เลข 1 ก็คือจุดแค่สองเส้นด้านขวา · เลข 7 ก็คือจุดเส้นบนกับสองเส้นขวา · เลข 8 คือจุดหมดทุกเส้น

ลองก่อน (คิดในหัวพอ)

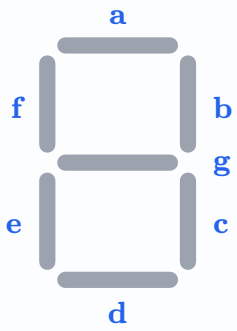
นึกภาพเลข 8 ที่ทำจากขีด 7 เส้นไว้ในหัวครับ
คำถามที่ 1: ถ้าจะแสดงเลข **1** ต้องให้ติดกี่เส้น
คำถามที่ 2: ถ้าจะแสดงเลข **0** ต้องดับเส้นไหน

เฉลย:
ข้อ 1 — **2 เส้น** ครับ คือสองเส้นตั้งด้านขวา
ข้อ 2 — ดับ**เส้นกลาง**เส้นเดียว ที่เหลืออีก 6 เส้นติดหมด
ถ้าเส้นกลางติดด้วย มันจะกลายเป็นเลข 8 ทันที

ตั้งชื่อขีดทั้งเจ็ด

จอแบบนี้เรียกว่า **จอ 7 ขีด** หรือ **7-segment display** และแต่ละขีดมีชื่อเป็นตัวอักษร **a ถึง g** ครับ

การเรียงชื่อมีกฎ: เริ่มที่ขีดบนสุดคือ **a** แล้ววน**ตามเข็มนาฬิกา** ไปเรื่อย ๆ จนครบรอบนอก ส่วนขีดกลางได้ชื่อสุดท้ายคือ **g**



ขีดทั้ง 7 เส้นชื่อ a ถึง g
เรียงตามเข็มนาฬิกาจากขีดบน
คือ a → b → c → d → e → f
แล้วขีดกลางชื่อ g เป็นตัวสุดท้าย

ชื่อของขีดทั้ง 7 · ข้อสอบจะอ้างถึงขีดด้วยตัวอักษรพวกนี้ตลอด ต้องจำตำแหน่งให้ได้ว่าตัวไหนอยู่ตรงไหน

ขีดแต่ละเส้นคือหลอด LED หนึ่งดวงครับ และ LED มี**ขั้ว** — ต้องต่อไฟบวกเข้าขาที่เรียกว่า**แอโนด (A)** และต่อไฟลบหรือกราวด์เข้าขาที่เรียกว่า **แคโทด (K)** ถึงจะติด ถ้าต่อกลับขั้วมันจะไม่ติดเลย

เลข 0 ถึง 9 ต้องติดขีดไหนบ้าง

ตารางนี้คือ**หัวใจของทั้งหัวข้อ**ครับ พี่วาดรูปเลขไว้ให้ด้วย จะได้ไม่ต้องจำ แค่มองรูปแล้วอ่านว่าเส้นไหนติด



เลข 0 ถึง 9 บนจอ 7 ขีด · **เส้นส้ม = ติด** · เส้นเทาจาง = ดับ · ตัวเลขข้างล่างบอกว่าเลขนั้นใช้กี่ขีด

เลข	BCD	ขีดที่ต้องติด	จำนวน
0	0000	a b c d e f	6
1	0001	b c	2
2	0010	a b d e g	5
3	0011	a b c d g	5
4	0100	b c f g	4
5	0101	a c d f g	5
6	0110	a c d e f g	6
7	0111	a b c	3
8	1000	a b c d e f g	7
9	1001	a b c d f g	6

วิธีจำที่ไม่ต้องท่อง: ให้จำแค่**เลข 8 ซึ่งติดหมดทุกขีด** แล้วคิดว่าเลขอื่น ๆ คือ "เลข 8 ที่ดับบางขีด"

เลข 6 คือเลข 8 ที่**ดับขีด b** (เส้นตั้งขวาบน) ขีดเดียว

เลข 9 คือเลข 8 ที่**ดับขีด e** (เส้นตั้งซ้ายล่าง) ขีดเดียว

เลข 0 คือเลข 8 ที่**ดับขีด g** (เส้นกลาง) ขีดเดียว

สังเกตอีกอย่าง: **ขีด c ดับแค่ตอนเลข 2 เลขเดียว** และ **ขีด b ดับแค่ตอนเลข 5 กับ 6** — สองข้อนี้ช่วยเช็กงานได้เร็วมาก

จอสองแบบ · ต่อขั้วรวมไว้คนละข้าง

หลอด LED ทั้ง 7 ดวงในจอไม่ได้แยกกันอิสระครับ ผู้ผลิตจะมัดขาข้างหนึ่งของทุกดวงเข้าด้วยกันเป็น**ขาร่วม** เพื่อให้ตัวจอมีขาน้อยลง

แบบแอโนดร่วม (common anode) — มัดขาบวกไว้ด้วยกันแล้วต่อเข้าไฟบวกค้างไว้
แปลว่าขีดจะติดเมื่อ**ขาอีกข้างถูกดึงลงเป็น 0** จึงต้องใช้ตัวขับแบบ **active-low**

แบบแคโทดร่วม (common cathode) — มัดขาลบไว้ด้วยกันแล้วต่อลงกราวด์ค้างไว้
แปลว่าขีดจะติดเมื่อ**ขาอีกข้างถูกยกขึ้นเป็น 1** จึงต้องใช้ตัวขับแบบ **active-high**
ไม่มีแบบไหนดีกว่าแบบไหนครับ เลือกตามว่าจะใช้ชิปตัวขับรุ่นไหน

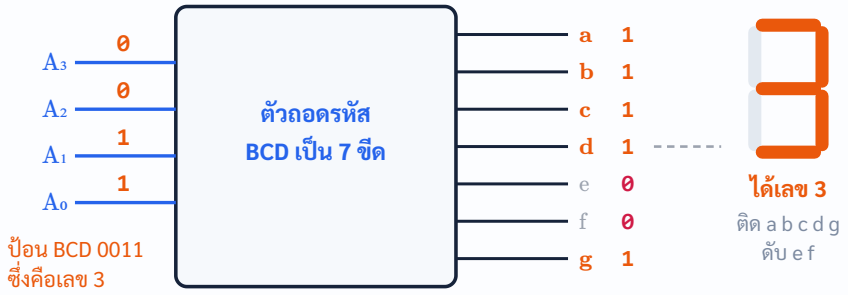
ตัวถอดรหัส BCD เป็น 7 ขีด

ทีนี้มาถึงวงจรจริงครับ โจทย์คือ **รับ BCD 4 บิต แล้วคายออก 7 เส้น** เส้นละหนึ่งขีด

เราเรียกมันว่า **ตัวถอดรหัส BCD เป็น 7 ขีด** หรือ **BCD-to-7-segment decoder**

สิ่งสำคัญที่สุดที่ต้องเข้าใจคือ — **ขาออกแต่ละเส้นเป็นฟังก์ชันของตัวเอง แยกกันคนละอัน** ไม่ได้ทำงานเป็นก้อนเดียว

พูดอีกแบบคือ เราต้องหานิพจน์ **7 อัน** อันละหนึ่งขีด แต่ละอันตอบคำถามว่า "ขีดนี้ติดตอนเลขอะไรบ้าง"



ตัวถอดรหัสมีขาเข้า 4 เส้น (รหัส BCD) และขาออก 7 เส้น (ขีด a ถึง g) · ตัวอย่างป้อน 0011 ซึ่งคือเลข 3 · ขาออกที่เป็น 1 คือ a, b, c, d, g ตรงกับรูปเลข 3 พอดี

หานิพจน์ของแต่ละขีด — ใช้ตารางคาร์นอฟ และของแถมชื่อ don't care

วิธีหานิพจน์คือเอาคอลัมน์ของขีดนั้นจากตารางข้างบน ไปลงตารางคาร์นอฟ (K-map) ขนาด 4 ตัวแปร แล้วจับกลุ่มให้ได้นิพจน์สั้นที่สุด ซึ่งเป็นวิธีจากบทก่อนหน้าครับ

แต่ตัวแปร 4 ตัวสร้างรหัสได้ 16 แบบ ในขณะที่ BCD ใช้แค่ 10 แบบ เหลือรหัส **1010 ถึง 1111 ที่ไม่มีวันเกิดขึ้น**

รหัสที่ไม่มีวันเกิด เราเรียกว่า **don't care** และเขียนแทนด้วยเครื่องหมาย **X** ในตาราง

ของแถมคือ ช่อง X พวกนี้เรา**เลือกให้เป็น 0 หรือ 1 ก็ได้ตามใจ** เลือกอันที่ทำให้กลุ่มใหญ่ขึ้นแล้วนิพจน์สั้นลง — เพราะยังไงมันก็ไม่มีวันเกิด

ถ้าไม่ใช้ don't care ช่วย นิพจน์จะยาวกว่านี้มาก และต้องใช้เกตเยอะกว่าโดยไม่จำเป็นครับ

นิพจน์ทั้ง 7 ขีด (แบบ active-high สำหรับจอแคโทดร่วม)

ตัวแปรคือ D_3 D_2 D_1 D_0 โดย D_3 เป็นบิตบนสุด · ทุกนิพจน์ผ่านการตรวจกับตารางเลข 0–9 ครบทุกเลขแล้ว

ขีด	นิพจน์	ติดตอนเลข
a	$D_3 + D_1 + D_2D_0 + \bar{D}_2\bar{D}_0$	0,2,3,5,6,7,8,9
b	$\bar{D}_2 + D_1D_0 + \bar{D}_1\bar{D}_0$	0,1,2,3,4,7,8,9
c	$D_2 + \bar{D}_1 + D_0$	0,1,3,4,5,6,7,8,9
d	$D_3 + \bar{D}_2\bar{D}_0 + D_1\bar{D}_0 + \bar{D}_2D_1 + D_2\bar{D}_1D_0$	0,2,3,5,6,8,9
e	$\bar{D}_2\bar{D}_0 + D_1\bar{D}_0$	0,2,6,8
f	$D_3 + \bar{D}_1\bar{D}_0 + D_2\bar{D}_1 + D_2\bar{D}_0$	0,4,5,6,8,9
g	$D_3 + D_1\bar{D}_0 + \bar{D}_2D_1 + D_2\bar{D}_1$	2,3,4,5,6,8,9

สังเกตขีด c: นิพจน์สั้นที่สุดในกลุ่ม เพราะมันติดเกือบทุกเลข ดับแค่เลข 2 เลขเดียว · ยิ่งขีดไหนดับน้อย นิพจน์ยิ่งสั้น

เดินตัวอย่างให้ดู — ป้อน BCD 0011 (เลข 3)

แทน D_3=0 , D_2=0 , D_1=1 , D_0=1 ลงไปทีละขีดครับ ไม่ข้ามขั้น

ขีด	แทนค่าแล้วได้	ผล
a	0 + **1** + (0·1) + (1·0) = 0 + 1 + 0 + 0 — พจน์ที่สองเป็น 1	`1` ติด
b	**1** + (1·1) + (0·0) — พจน์แรก $\bar{D}_2$ เป็น 1 อยู่แล้ว	`1` ติด
c	0 + 0 + **1** — พจน์ D_0 เป็น 1	`1` ติด
d	0 + (1·0) + (1·0) + (**1·1**) + (0·0·1) — พจน์ $\bar{D}_2D_1$ เป็น 1	`1` ติด
e	(1·0) + (1·0) = 0 + 0 — ทั้งสองพจน์มี $\bar{D}_0$ ซึ่งเป็น 0	`0` ดับ
f	0 + (0·0) + (0·0) + (0·0) — ทุกพจน์เป็น 0	`0` ดับ
g	0 + (1·0) + (**1·1**) + (0·0) — พจน์ $\bar{D}_2D_1$ เป็น 1	`1` ติด

เช็กคำตอบกลับเข้าโจทย์: ขีดที่ติดคือ **a, b, c, d, g** และดับคือ e, f

เปิดตารางข้างบนไปดูแถวเลข 3 เขียนไว้ว่า "a b c d g" เหมือนกันเป๊ะครับ และถ้ามองรูปเลข 3 ในแถบรูป ก็เห็นว่าเส้นตั้งซ้ายสองเส้น (e กับ f) ดับอยู่จริง

กับดักข้อสอบ · จับคู่ชิปกับจอผิดแบบ

สไลด์พูดถึงชิปสองตัวที่หน้าตาเหมือนกันแต่ใช้แทนกันไม่ได้

ชิป	ขาออกเป็นแบบ	ใช้กับจอแบบ	ขีดติดเมื่อขาออกเป็น
74LS47	active-low	แอโนดร่วม	`0`
74LS48	active-high	แคโทดร่วม	`1`

ถ้าจับคู่ผิด เช่นเอา 74LS47 ไปต่อจอแคโทดร่วม จะเกิดอะไรขึ้น — ขีดที่ควรติดจะดับ และขีดที่ควรดับจะติด **ภาพออกมาเป็นเลขกลับด้าน** เช่นป้อนเลข 1 แล้วได้ภาพเหมือนเลข 8 ที่หายไปสองขีดขวา

วิธีจำ: ตารางของ 74LS47 ในสไลด์เขียนค่าไว้เป็น $\overline{a}$, $\overline{b}$, ... มีขีดบนหัวทุกตัว
เห็นขีดเมื่อไหร่แปลว่า **active-low** และค่าในตารางนั้นคือ**ตรงข้าม**กับตารางที่เราใช้ในหัวข้อนี้ — เลข 1 ที่เราเขียนว่า "ติด" ในตาราง 74LS47 จะเขียนเป็น 0

เช็กลิสต์จอ 7 ขีด

1. **วาดรูปเลขก่อน อย่าเพิ่งเขียนตัวอักษร**
   วาดเลข 8 แล้วดับขีดที่ไม่ใช้ · แล้วค่อยอ่านว่าที่เหลือคือขีดอะไรบ้าง · วิธีนี้ผิดยากกว่าการท่องตาราง

2. **ถ้าโจทย์ให้หานิพจน์ ให้ทำทีละขีด**
   เขียนคอลัมน์ของขีดนั้นลง K-map 4 ตัวแปร · ใส่ **X** ที่ช่อง 10–15 เสมอ · แล้วจับกลุ่มโดยใช้ X ช่วยขยายกลุ่มได้

3. **ตรวจนิพจน์ด้วยการแทนค่าสองสามเลข**
   เลือกเลขที่ขีดนั้น**ติด**หนึ่งเลข และเลขที่**ดับ**หนึ่งเลข แล้วแทนค่าดู · ถ้าผ่านทั้งสอง โอกาสถูกสูงมาก

4. **ดูว่าโจทย์พูดถึงจอแบบไหน แล้วหยุด**
   แอโนดร่วม → ต้อง active-low (74LS47) · แคโทดร่วม → ต้อง active-high (74LS48) · **หยุดตรงนี้**
   ถ้าโจทย์ไม่ได้บอกแบบของจอ ให้ตอบแบบ active-high แล้วเขียนกำกับไว้ว่าสมมติแบบไหน

10 แปลงรหัส binary ↔ Gray — โซ่ XOR สองทิศ

เริ่มจากลูกบิดที่อ่านค่าผิด

ลองนึกถึงแกนหมุนที่มีแผ่นรหัสติดอยู่ แล้วมีเซนเซอร์ 4 ตัวคอยอ่านว่าตอนนี้หมุนมาถึงตำแหน่งที่เท่าไหร่

ถ้าเราเขียนตำแหน่งด้วยเลขฐานสองธรรมดา พอหมุนจากตำแหน่งที่ 7 ไปตำแหน่งที่ 8 รหัสจะเปลี่ยนจาก **0111** เป็น **1000**

ลองก่อน (คิดในหัวพอ)

เทียบ **0111** กับ **1000** ทีละบิตครับ
คำถามที่ 1: มี**กี่บิต**ที่ต้องเปลี่ยนค่า
คำถามที่ 2: ถ้าบิตซ้ายสุดเปลี่ยนเร็วกว่าเพื่อนเสี้ยววินาที เซนเซอร์จะอ่านได้**เลขอะไร** ในเสี้ยววินาทีนั้น

เฉลย:
ข้อ 1 — เปลี่ยน**ทั้ง 4 บิต**พร้อมกันครับ ไม่มีบิตไหนเหมือนเดิมเลย
ข้อ 2 — ถ้าบิตซ้ายสุดขึ้นเป็น 1 ก่อน ส่วนบิตอื่นยังเป็น 1 อยู่ จะอ่านได้ **1111** ซึ่งคือ**เลข 15**
ทั้งที่แกนแค่หมุนจาก 7 ไป 8 แต่เครื่องอ่านได้ 15 ไปแวบหนึ่ง — ถ้าเครื่องเอาค่านั้นไปสั่งงานทันที ก็พังได้เลยครับ

ต้นตอของปัญหาคือ**หลายบิตเปลี่ยนพร้อมกัน** และในโลกจริงไม่มีทางเปลี่ยนพร้อมกันเป๊ะได้

ทางแก้ก็ตรงไปตรงมาครับ — **ออกแบบรหัสใหม่ให้เปลี่ยนทีละหนึ่งบิตเท่านั้น**

ตั้งชื่อรหัสใหม่

รหัสที่**เปลี่ยนทีละ 1 บิต** เมื่อเลขเพิ่มขึ้นทีละ 1 เรียกว่า **รหัสเกรย์** หรือ **Gray code** ครับ

คุณสมบัติข้อเดียวนี้แหละคือเหตุผลทั้งหมดที่รหัสนี้มีอยู่ ไม่มีเหตุผลอื่นเลย

เลข	ฐานสอง	เกรย์	เปลี่ยนบิตไหน
0	0000	0000	—
1	0001	0001	บิต 0
2	0010	0011	บิต 1
3	0011	0010	บิต 0
4	0100	0110	บิต 2
5	0101	0111	บิต 0
6	0110	0101	บิต 1
7	0111	0100	บิต 0
8	1000	1100	บิต 3
9	1001	1101	บิต 0
…	…	…	…
15	1111	1000	บิต 0

ดูคอลัมน์ขวาสุดครับ — **ทุกก้าวเปลี่ยนแค่บิตเดียว** ไม่มียกเว้น

โดยเฉพาะก้าวจาก 7 ไป 8 ที่เคยเป็นปัญหา · ฐานสองเปลี่ยน 4 บิต แต่เกรย์เปลี่ยนแค่ **บิต 3** บิตเดียว (0100 → 1100)

ตอนนี้ถ้าเซนเซอร์อ่านคร่อมจังหวะ มันจะได้ค่าเก่า (7) หรือค่าใหม่ (8) เท่านั้น **ไม่มีทางได้ค่าประหลาดอย่าง 15** เพราะมีบิตเดียวที่กำลังเปลี่ยน

นี่คือเหตุผลที่โรงงานใช้รหัสเกรย์กับตัวอ่านตำแหน่งแกนหมุนครับ

แปลงจากฐานสองเป็นเกรย์

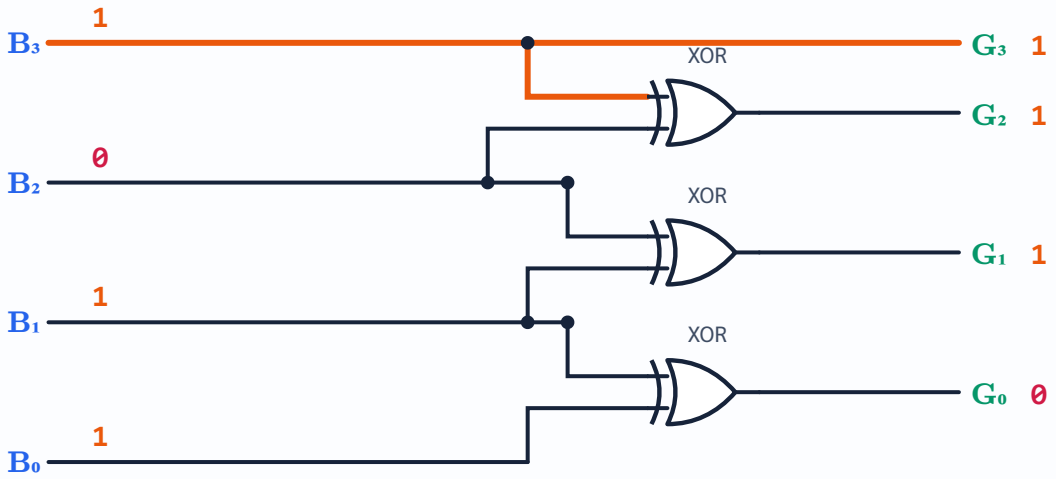
กฎมีแค่สองข้อครับ

ข้อที่ 1: บิตซ้ายสุดของเกรย์ **ลอกมาจากบิตซ้ายสุดของฐานสองตรง ๆ** ไม่ต้องทำอะไรเลย

ข้อที่ 2: บิตที่เหลือ ให้เอา**บิตที่อยู่ติดกันทางซ้ายของมันในเลขฐานสอง** มา XOR กับ**ตัวมันเอง**

$$G_3 = B_3 \cdot G_2 = B_3 \oplus B_2 \cdot G_1 = B_2 \oplus B_1 \cdot G_0 = B_1 \oplus B_0$$

ทุกตัวใช้ "บิตฐานสองคู่ที่ติดกัน" มา XOR กัน



บิตซ้ายสุดผ่านตรง ๆ ไม่ต้องมีเกต · ที่เหลือใช้ XOR ตัวละบิต

ฐานสอง → เกรย์ · **เส้นส้มบนสุด**คือ B_3 ที่วิ่งตรงไปเป็น G_3 และแตกไปเข้า XOR ตัวแรกด้วย
ตัวอย่างป้อน B = 1011 ได้ G = 1110

บิต	คิดยังไง	ได้
G_3	ลอก B_3 มาตรง ๆ = 1	1
G_2	$B_3 \oplus B_2 = 1 \oplus 0$ — ต่างกัน	1
G_1	$B_2 \oplus B_1 = 0 \oplus 1$ — ต่างกัน	1
G_0	$B_1 \oplus B_0 = 1 \oplus 1$ — เหมือนกัน	0

เช็กคำตอบกลับเข้าโจทย์: ได้เกรย์ = **1110** จากฐานสอง 1011

เดี๋ยวเราจะแปลงกลับในหัวข้อย่อยถัดไป ถ้าได้ 1011 คืนมา ก็แปลว่าถูกครับ

แปลงจากเกรย์กลับเป็นฐานสอง

ขากลับใช้ XOR เหมือนกัน แต่มี**จุดที่ต่างกันหนึ่งจุด ซึ่งเป็นจุดที่คนพลาดมากที่สุด** พี่จะเน้นตรงนั้นครับ

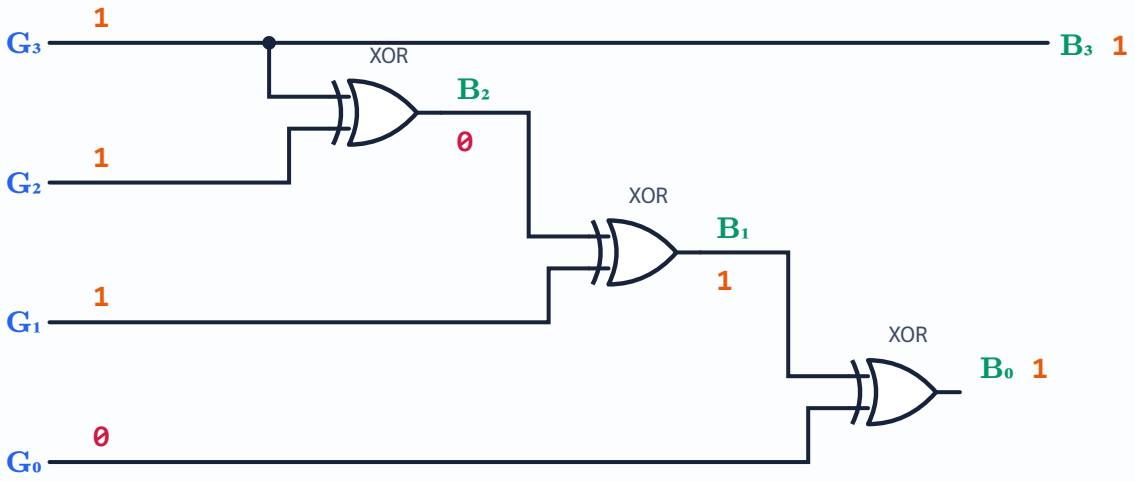
ข้อที่ 1: บิตซ้ายสุดของฐานสอง ลอกมาจากบิตซ้ายสุดของเกรย์ตรง ๆ — เหมือนขาไปเป๊ะ

ข้อที่ 2: บิตที่เหลือ ให้เอา**บิตฐานสองที่เราเพิ่งคำนวณได้** มา XOR กับ**บิตเกรย์ตัวปัจจุบัน**

ขอย้ำคำว่า "**บิตฐานสองที่เราเพิ่งคำนวณได้**" นะครับ — **ไม่ใช่**บิตเกรย์ตัวก่อนหน้า ตรงนี้คือจุดต่างของขากลับ

$$B_3 = G_3 \;\cdot\; B_2 = \mathbf{B_3} \oplus G_2 \;\cdot\; B_1 = \mathbf{B_2} \oplus G_1 \;\cdot\; B_0 = \mathbf{B_1} \oplus G_0$$

ตัวที่ไฮไลต์คือผลลัพธ์ของบรรทัดก่อนหน้า ไม่ใช่บิตเกรย์



สังเกตว่า XOR แต่ละตัวรับ "ผลของตัวก่อนหน้า" ไม่ใช่รับบิตเกรย์สองตัวติดกัน

เกรย์ → ฐานสอง · ผลลัพธ์ของ XOR ตัวหนึ่ง**วิ่งไปเป็นขาเข้าของ XOR ตัวถัดไป** จึงเป็นโซ่ที่ต้องทำเรียงกัน ต่างจากขาไปที่ทำพร้อมกันได้หมด

ตัวอย่างป้อน G = 1110 ได้ B = 1011

บิต	คิดยังไง	ได้
B_3	ลอก G_3 มาตรง ๆ = 1	1
B_2	**B_3** ⊕ G_2 = 1 ⊕ 1 — เหมือนกัน · (ใช้ B_3 ที่เพิ่งได้ ไม่ใช่ G_3)	0
B_1	**B_2** ⊕ G_1 = 0 ⊕ 1 — ต่างกัน · (ใช้ B_2 ที่เพิ่งได้ ไม่ใช่ G_2)	1
B_0	**B_1** ⊕ G_0 = 1 ⊕ 0 — ต่างกัน · (ใช้ B_1 ที่เพิ่งได้ ไม่ใช่ G_1)	1

เช็กคำตอบกลับเข้าโจทย์: ได้ฐานสอง = **1011**

ซึ่งตรงกับเลขที่เราเริ่มต้นในหัวข้อย่อยที่แล้วเป๊ะครับ — แปลงไปแล้วแปลงกลับได้เลขเดิม นี่คือวิธีตรวจงานตัวเองที่ดีที่สุดของหัวข้อนี้

กับดักข้อสอบ · ขากลับใช้บิตเกรย์แทนบิตฐานสอง

ขาไปใช้บิตฐานสองสองตัวติดกัน คนเลยเผลอคิดว่าขากลับก็ใช้บิตเกรย์สองตัวติดกัน

ลองทำผิดวิธีดูกับ G = 1110 ครับ

บิต	วิธีผิด (ใช้บิตเกรย์คู่ติดกัน)	วิธีถูก (ใช้ผลที่เพิ่งได้)
B_3	G_3 = `1`	G_3 = `1`
B_2	$G_3 \oplus G_2 = 1 \oplus 1$ = `0`	$B_3 \oplus G_2 = 1 \oplus 1$ = `0`
B_1	$G_2 \oplus G_1 = 1 \oplus 1$ = `0`	$B_2 \oplus G_1 = 0 \oplus 1$ = `1`
B_0	$G_1 \oplus G_0 = 1 \oplus 0$ = `1`	$B_1 \oplus G_0 = 1 \oplus 0$ = `1`
ได้	**1001** = เลข 9 — ผิด	**1011** = เลข 11 — ถูก

สองบิตแรกบังเอิญเหมือนกัน เพราะ B_3 ก็คือ G_3 อยู่แล้ว **มันเริ่มแตกต่างที่บิตที่สามเป็นต้นไป** — ทำให้คนตรวจงานตัวเองแค่สองบิตแรกแล้วคิดว่าถูก

วิธีกันพลาด: แปลงกลับเสร็จแล้ว ให้**แปลงไปใหม่อีกรอบ** ถ้าได้เกรย์ตัวเดิมคืนมาก็ถูก · ในตัวอย่างนี้ 1001 แปลงไปจะได้ 1101 ไม่ใช่ 1110 จึงรู้ทันทีว่าผิด

เช็กลิสต์การแปลงรหัส

1 ดูทิศก่อนว่าไปทางไหน

ฐานสอง → เกรย์ = ใช้บิต**ฐานสอง**คู่ติดกัน · เกรย์ → ฐานสอง = ใช้**ผลที่เพิ่งได้**กับบิตเกรย์

2 เขียนบิตซ้ายสุดก่อนเลย ทั้งสองทิศ

บิตซ้ายสุดลอกกันตรง ๆ เสมอ ไม่มี XOR · เขียนลงไปได้ทันทีโดยไม่ต้องคิด

3 ไล่จากซ้ายไปขวาทีละบิต

ทั้งสองทิศไล่จากซ้ายไปขวาเหมือนกัน · ขากลับต้องทำเรียงกันจริง ๆ เพราะบิตถัดไปต้องใช้ผลของบิตก่อนหน้า

4 แปลงย้อนกลับเพื่อตรวจ แล้วหยุด

ได้คำตอบแล้วให้แปลงกลับทิศเดิมดู ถ้าได้ตัวตั้งคืนมาก็ถูก · **หยุดตรงนี้** ไม่ต้องไปแปลงเป็นเลขฐานสิบถ้าโจทย์ไม่ได้ถาม

11 MUX กับ DEMUX — สวิตช์เลือกทาง

เริ่มจากสายสัญญาณที่มีเยอะแต่ช่องรับมีช่องเดียว

ทีวีเครื่องหนึ่งมีช่องเสียบ HDMI อยู่ช่องเดียว แต่บ้านมีกล่องอยู่ 4 กล่อง

น้องแก้ปัญหานี้ยังไงครับ — ก็ซื้อกล่องสลับสัญญาณมาต่อคั่น แล้วกดปุ่มเลือกว่าจะให้กล่องไหนผ่านเข้าไปที่ทีวี

สังเกตว่ากล่องสลับนั้นมี**สายข้อมูลเข้า 4 เส้น สายออก 1 เส้น** และมี**ปุ่มควบคุม**ที่บอกว่าจะเลือกเส้นไหน

ลองก่อน (คิดในหัวพอ)

ถ้าจะสั่งกล่องนี้ด้วยสายไฟที่ส่งได้แค่ 0 กับ 1

คำถาม: ต้องใช้สายควบคุม**กี่เส้น** ถึงจะสั่งเลือกได้ครบทั้ง 4 ทาง

เฉลย: 2 เส้น ครับ เพราะ 2 × 2 = 4 พอดี

เป็นคำถามเดียวกับหัวข้อ 7 เป๊ะ ๆ — **สาย n เส้นสั่งได้ 2^n ทาง**

4 ทางใช้ 2 เส้น · 8 ทางใช้ 3 เส้น · 16 ทางใช้ 4 เส้น

ตั้งชื่อ

วงจรที่เลือกสายข้อมูลเส้นหนึ่งจากหลายเส้น ส่งออกไปทางเส้นเดียว เรียกว่า **มัลติเพล็กเซอร์** หรือ **MUX** ครับ · อีกชื่อคือ **data selector**

ขาต่าง ๆ มีชื่อเรียกดังนี้ — **D_0, D_1, D_2, D_3** คือสายข้อมูลเข้า · **S_1, S_0** คือสายเลือก (data-select) · **Y** คือสายข้อมูลออก

กติกาก็ตรงไปตรงมาครับ: **อ่าน S_1S_0 เป็นเลขฐานสอง แล้วเลือก D ตัวที่มีเลขนั้น**

S_1	S_0	อ่านเป็นเลข	ส่งตัวไหนออก
0	0	0	D_0
0	1	1	D_1
1	0	2	D_2
1	1	3	D_3

แล้ววงจรข้างในทำยังไง — คิดแบบนี้ครับ "D_2 จะผ่านออกไปได้ก็ต่อเมื่อ **S_1 เป็น 1 และ S_0 เป็น 0 และตัว D_2 เองเป็น 1**"

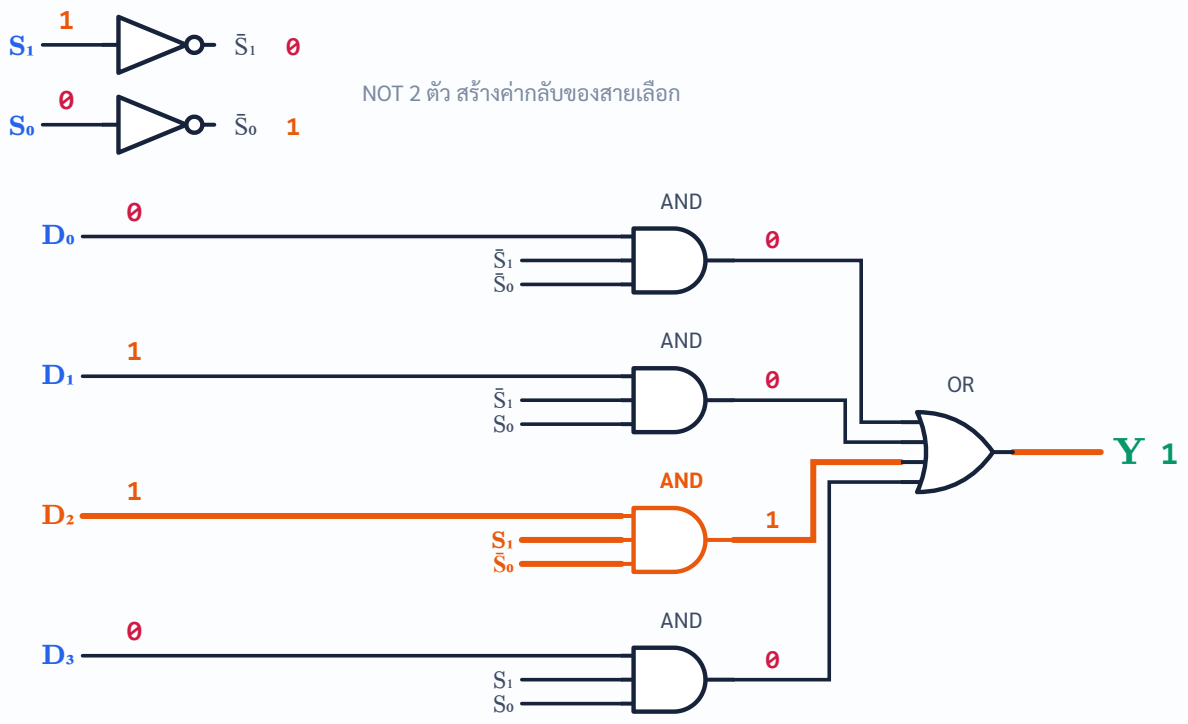
สามเงื่อนไขต่อกันด้วย "และ" → **AND 3 ขาเข้า** หนึ่งตัวต่อหนึ่งสายข้อมูล

แล้วสุดท้ายเอาผลของ AND ทั้งสี่ตัวมา **OR** รวมกัน เพราะ "ออกทางไหนก็ได้ในสี่ทาง"

และเพราะมี AND ได้ 1 ตัวเดียวเสมอ (สายเลือกชี้ไปที่เดียว) OR จึงไม่มีทางปนกัน

$$Y = \bar{S}_1\bar{S}_0 \cdot D_0 + \bar{S}_1 S_0 \cdot D_1 + S_1\bar{S}_0 \cdot D_2 + S_1 S_0 \cdot D_3$$

เทียบกับตัวถอดรหัสในหัวข้อ 8 ดูสิครับ — ส่วน $\bar{S}_1\bar{S}_0$ พวกนี้คือตัวถอดรหัสทั้งดุ้น แค่เอา D มาคูณเพิ่มอีกตัว



MUX 4 ต่อ 1 · **เส้นทางส้ม**คือทางที่ถูกเลือกตอน $S_1S_0 = 10$ · ป้อน D_0=0, D_1=1, D_2=1, D_3=0 แล้วได้ $Y = D_2 = 1$
สังเกตว่า D_1 ก็เป็น 1 เหมือนกัน แต่ AND ของมันมี $\bar{S}_1$ ซึ่งเป็น 0 อยู่ จึงผ่านไม่ได้

ขั้น	ทำอะไร	ได้อะไร
①	ตั้งสายเลือกเป็น S_1=1, S_0=0 · NOT สร้าง $\bar{S}_1$=0 และ $\bar{S}_0$=1	$\bar{S}_1$ = `0` , $\bar{S}_0$ = `1`
②	AND ของ D_2 ได้ขาเข้า S_1=1 , $\bar{S}_0$=1 , D_2=1 — ครบทั้งสาม	ผ่าน `1`
③	AND ของ D_1 ได้ $\bar{S}_1$=0 ทำให้ตายทันที ถึง D_1 จะเป็น 1 ก็ตาม	ตัน `0`
④	OR รวมได้ 0+0+1+0	Y = `1`

เช็กคำตอบกลับเข้าโจทย์: สายเลือกชี้ที่ 2 และ D_2 มีค่า 1 · วงจรตอบ Y = 1 ตรงกันครับ

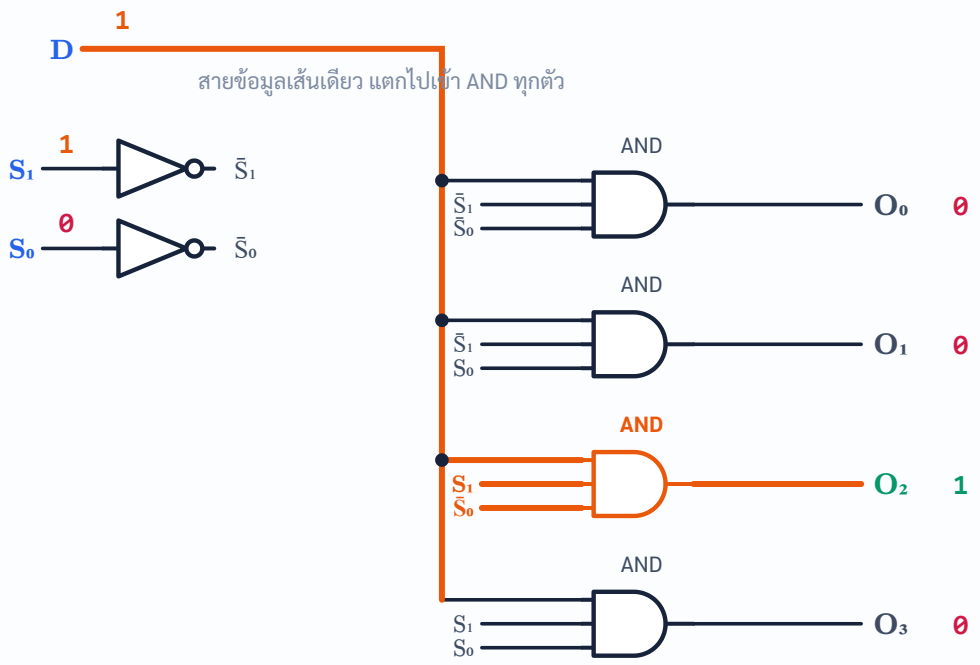
DEMUX — ทำกลับทาง

ดีมัลติเพล็กเซอร์ หรือ **DEMUX** ทำตรงข้ามกับ MUX ทุกอย่างครับ — รับข้อมูลเข้า**เส้นเดียว** แล้วส่งออกไปทาง**เส้นใดเส้นหนึ่งจากหลายเส้น**

อีกชื่อของมันคือ **data distributor** แปลว่าตัวกระจายข้อมูล

ถ้า MUX คือกล่องสลับ HDMI ที่รวมสี่กล่องเข้าทีวีเครื่องเดียว DEMUX ก็คือกล่องที่เอาสัญญาณจากเครื่องเล่นเครื่องเดียว ไปออกจอไหนก็ได้ในสี่จอครับ

วงจรข้างในก็**เหมือน MUX เกือบทั้งหมด** — ใช้ AND 3 ขาต่อหนึ่งเส้นขาออกเหมือนกัน ต่างแค่**ไม่มี OR ปิดท้าย** และสายข้อมูล D เส้นเดียวถูกแยกไปเข้า AND ทุกตัว



DEMUX 1 ต่อ 4 · ป้อน D = 1 และตั้ง $S_1S_0 = 10$ · ข้อมูลออกทาง O_2 เส้นเดียว ส่วนอีกสามเส้นเป็น 0
ถ้าเปลี่ยน D เป็น 0 เส้นที่ถูกเลือกก็จะเป็น 0 ด้วย — มัน**ส่งค่าที่ป้อนเข้ามาจริง ๆ** ไม่ใช่ส่งเลข 1 เฉย ๆ

ท่าที่ข้อสอบชอบออก — เอา MUX มาสร้างฟังก์ชัน

ท่านี้มาจากการสังเกตอย่างหนึ่งครับ — **MUX เลือกค่าตามรหัสที่ป้อนเข้าสายเลือก** ซึ่งฟังดูเหมือน**ตารางค่าความจริง**ที่ "เลือกคำตอบตามแถว" เป๊ะเลย

ถ้าอย่างนั้น เราก็เอา**ตัวแปรของฟังก์ชันไปต่อที่สายเลือก** แล้วเอา**คำตอบของแต่ละแถวไปต่อที่สายข้อมูล**สิครับ — MUX ก็จะกลายเป็นวงจรของฟังก์ชันนั้นทันที

ลองทำจริงกับโจทย์นี้ครับ: **F = Σm(1, 2, 6, 7)** ของตัวแปร A, B, C

สัญลักษณ์ Σm(...) แปลว่า "ฟังก์ชันนี้เป็น 1 ที่แถวหมายเลขเหล่านี้" โดยหมายเลขแถวคือค่าของ ABC อ่านเป็นเลขฐานสอง

แถว	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

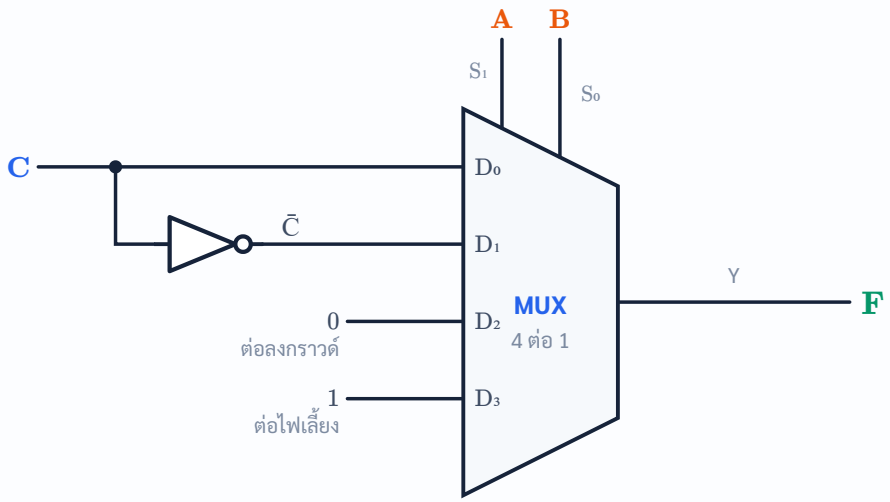
เรามีตัวแปร 3 ตัว แต่ MUX 4 ต่อ 1 มีสายเลือกแค่ 2 เส้น

วิธีแก้คือเอา **A กับ B ไปเข้าสายเลือก** (A ไปที่ S_1 · B ไปที่ S_0) แล้วให้ **C เป็นตัวที่เหลือ**

ทีนี้จับตารางมา**แบ่งเป็นคู่ ๆ** ตามค่า AB จะได้ 4 คู่พอดี แต่ละคู่คือสองแถวที่ AB เหมือนกัน ต่างกันแค่ C

แล้วถามทีละคู่ว่า "**ในคู่นี้ F เกี่ยวกับ C ยังไง**" — คำตอบที่เป็นไปได้มีแค่ 4 แบบ คือ **0 · 1 · C · C̄**

AB	สองแถวที่ตรงกัน	F เป็นอะไรเมื่อ C=0 / C=1	สรุปว่า F คือ	ต่อเข้า
0 0	แถว 0 และ 1	0 / 1 — ตามค่า C เป๊ะ	**C**	D_0
0 1	แถว 2 และ 3	1 / 0 — ตรงข้ามกับ C	**C̄**	D_1
1 0	แถว 4 และ 5	0 / 0 — เป็น 0 ทั้งคู่	**0**	D_2
1 1	แถว 6 และ 7	1 / 1 — เป็น 1 ทั้งคู่	**1**	D_3



ตัวแปร A กับ B ไปที่สายเลือก · ตัวแปร C กับค่าคงที่ 0/1 ไปที่สายข้อมูล

สร้าง F = Σm(1,2,6,7) ด้วย MUX ตัวเดียว + NOT ตัวเดียว · ใช้สัญลักษณ์กล่องของ MUX ตามที่สไลด์วาด (ข้างในคือวงจร AND/OR ในรูปก่อนหน้า)

ลองตรวจดูสองแถวครับ

ป้อน A B C	สายเลือกชี้ที่	ขานั้นต่ออยู่กับ	Y ได้	ตารางบอกว่า
0 1 0 (แถว 2)	AB = 01 → D_1	C̄ ซึ่งตอนนี้ C=0 จึง C̄=1	1	F = 1 ✓

1 0 1 (แถว 5)	AB = 10 → D_2	ค่าคงที่ 0	`0`	F = 0 ✓

เช็กคำตอบกลับเข้าโจทย์: ตรงทั้งสองแถว · ถ้าลองครบทั้ง 8 แถวก็ตรงหมดครับ เพราะเราสร้างการต่อจากตารางนั้นโดยตรง

ถ้าใช้ MUX ที่ใหญ่พอ จะง่ายกว่านี้อีก

ถ้าโจทย์ให้ MUX **8 ต่อ 1** มาใช้กับฟังก์ชัน 3 ตัวแปร จะไม่ต้องคิดเรื่อง C กับ $\bar{C}$ เลยครับ

เพราะสายเลือกมี 3 เส้นพอดีกับตัวแปร 3 ตัว — เอา A, B, C ไปเข้า S_2, S_1, S_0 แล้ว**เอาคอลัมน์ F ในตารางไปต่อขา D_0 ถึง D_7 ตรง ๆ ทีละแถว**

ในโจทย์นี้ก็คือต่อ 0, 1, 1, 0, 0, 0, 1, 1 เรียงกันลงไป

ไม่ต้องใช้ NOT เลยสักตัว และไม่ต้องคิดอะไรทั้งนั้น — แค่ลอกตาราง

กับดักข้อสอบ · สลับลำดับสายเลือก

กับดักที่เจอบ่อยที่สุดคือเอา **A ไปต่อ S_0 และ B ไปต่อ S_1** ซึ่งสลับกัน

ลองดูผลกับแถว 2 (A=0, B=1, C=0) ครับ

ต่อถูก: S_1S_0 = AB = 01 → เลือก D_1 ซึ่งต่อกับ $\bar{C}$ = 1 → F = 1 ✓

ต่อสลับ: S_1S_0 = BA = 10 → เลือก D_2 ซึ่งต่อกับ 0 → F = 0 ✗

วิธีจำ: ตัวแปรที่มีน้ำหนักมากสุดไปที่สายเลือกที่มีน้ำหนักมากสุด — A (บิตซ้ายสุดของเลขแถว) ต้องไปที่ S_1 ซึ่งเป็นบิตซ้ายสุดของสายเลือก

เขียนกำกับไว้ข้างวงจรทุกครั้งว่า "S_1 = A, S_0 = B" จะกันพลาดได้มาก

เช็กลิสต์ MUX / DEMUX

1

ดูทิศของข้อมูลก่อน

หลายเส้นเข้า เส้นเดียวออก = **MUX** · เส้นเดียวเข้า หลายเส้นออก = **DEMUX** · สายเลือกไม่ใช่สายข้อมูล อย่านับรวม

2

นับสายเลือกจากจำนวนทาง

4 ทางใช้ 2 เส้น · 8 ทางใช้ 3 เส้น · ถ้าโจทย์ให้สายเลือกมา n เส้น แปลว่ามี 2^n ทาง

3

ถ้าต้องวาดวงจร ให้วาด AND ต่อหนึ่งทาง

AND แต่ละตัวรับสายเลือกครบทุกเส้น (ใส่ขีดตามรหัสเหมือนตัวถอดรหัส) บวกสายข้อมูลอีกหนึ่ง · MUX จบด้วย OR รวม · DEMUX ไม่มี OR

4

ถ้าโจทย์ให้สร้างฟังก์ชันด้วย MUX ให้จับตารางเป็นคู่

ตัวแปรตัวบน ๆ ไปสายเลือก · ตัวที่เหลือดูว่าแต่ละคู่แถวเป็น **0 · 1 · ตัวแปรนั้น · หรือตัวกลับค่า** แล้วต่อเข้าขาข้อมูล

5

ตรวจสองแถว แล้วหยุด

สุ่มแถวที่ F=1 มาหนึ่งแถว และแถวที่ F=0 มาอีกหนึ่ง แล้วไล่ตามวงจรดู · **หยุดตรงนี้** ไม่ต้องไล่ครบทุกแถวถ้าเวลาไม่พอ สองแถวที่ต่างผลกันจับความผิดได้เกือบหมดแล้ว

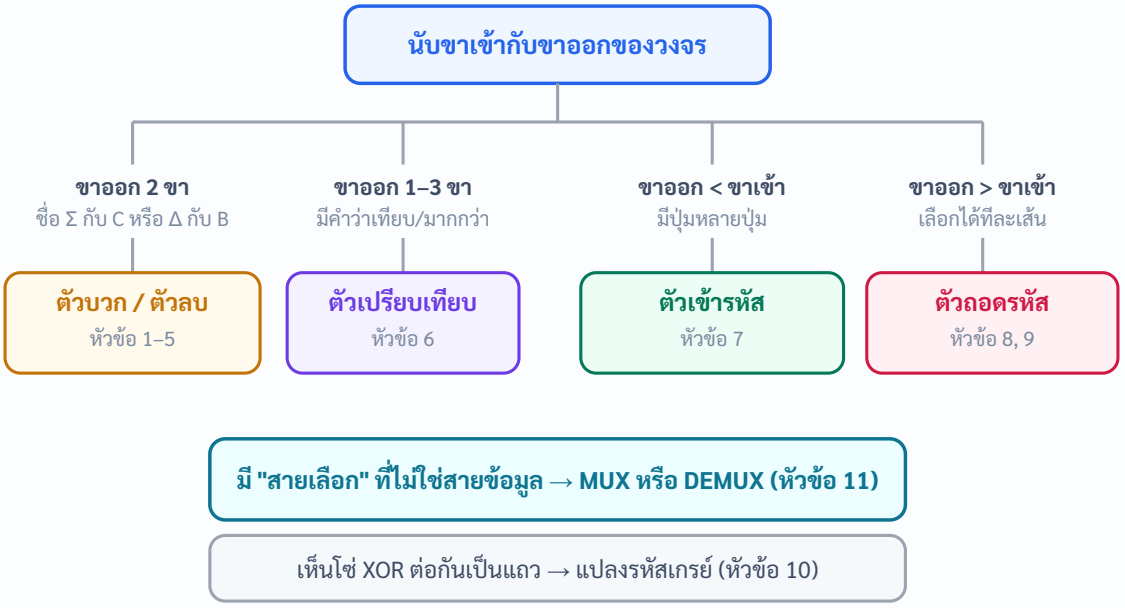
12 เห็นโจทย์แล้วรู้ทันทีว่าเป็นวงจรอะไร

ในห้องสอบ ปัญหาไม่ใช่ว่าน้องจำสูตรไม่ได้ครับ ปัญหาคือ**ดูไม่ออกว่าโจทย์ข้อนี้พูดถึงวงจรตัวไหน** แล้วเลยหยิบสูตรผิดตัวมาใช้

หัวข้อนี้จึงเป็นเครื่องมือจำแนก ไม่ใช่เนื้อหาใหม่ · พี่แนะนำให้อ่านหน้านี้ซ้ำก่อนเข้าห้องสอบครับ

ขั้นที่ 1 — นับขาก่อน แล้วอ่านคำสำคัญ

วิธีที่เร็วที่สุดคือ**นับจำนวนขาเข้ากับขาออก** เพราะวงจรแต่ละชนิดมีรูปร่างของตัวเองที่ซ้ำกันไม่ได้



แผนผังจำแนกวงจร · เริ่มจากนับขา แล้วค่อยดูคำสำคัญในโจทย์

ขั้นที่ 2 — ตารางจำแนกละเอียด

ถ้าโจทย์พูดว่า	คือวงจร	สิ่งแรกที่ต้องเขียนลงกระดาษ
"บวกเลขหนึ่งบิตสองตัว" · มี 2 ขาเข้า	ตัวบวกครึ่ง	$\Sigma = A \oplus B$ และ $C_{out} = A \cdot B$
มีคำว่า C_{in} หรือ "ตัวทดเข้า" · 3 ขาเข้า	ตัวบวกเต็ม	$\Sigma = A \oplus B \oplus C_{in}$ และ $C_{out} = A \cdot B + C_{in}(A \oplus B)$
"บวกเลข n บิต" · ให้วาดเป็นแถว	ตัวบวกขนาน	วางตัวบวกเต็ม n ตัว · ต่อ C_{out} ไป C_{in} ตัวถัดไป · ตัวขวาสุดต่อกราวด์

มีคำว่า "ช้า" "หน่วง" "ns" "worst case"	ตัวทดไล่	เวลารวม = เวลาต่อชั้น × จำนวนบิต
มีคำว่า generation · propagation · look-ahead	ตัวทดมองล่วงหน้า	$C_g = A{\cdot}B$ · $C_p = A + B$ · $C_{out} = C_g + C_pC_{in}$ แล้วกางทีละชั้น (ชั้น k มี k+2 พจน์)
มีคำว่า "ยืม" borrow หรือสัญลักษณ์ Δ	ตัวลบ	Δ เหมือน Σ ทุกตัวอักษร · ตัวยืมคือสูตรตัวทดที่เติมขีด 2 จุด
"เท่ากันไหม" · "ใครมากกว่า" · A>B, A=B, A<B	ตัวเปรียบเทียบ	เท่ากัน = XNOR รายบิตแล้ว AND รวม · มากกว่า = กวาดจากบิตซ้ายสุด
"กดปุ่ม" · "ขาเข้าหลายเส้นเหลือน้อยเส้น" · BCD	ตัวเข้ารหัส	OR หนึ่งตัวต่อหนึ่งบิตขาออก · อ่าน**ลงตามคอลัมน์**ของตารางรหัส
"เลือกเส้นเดียวจากหลายเส้น" · 1-of-8 · 1-of-10	ตัวถอดรหัส	AND หนึ่งตัวต่อหนึ่งเส้น · บิตเป็น 0 ใส่ขีด · active-low ใช้ NAND
มีตัวอักษร a–g หรือรูปเลขบนจอ	ตัวถอดรหัส 7 ขีด	วาดรูปเลขก่อน แล้วอ่านว่าติดขีดไหน · ใช้ don't care ที่ 1010–1111
มีคำว่า Gray หรือ "เปลี่ยนทีละบิต"	ตัวแปลงรหัส	บิตซ้ายสุดลอกตรง ๆ · ที่เหลือ XOR · ขากลับใช้ผลที่เพิ่งได้
มีสาย S ที่ไม่ใช่ข้อมูล · "data select"	MUX / DEMUX	หลายเข้าหนึ่งออก = MUX (จบด้วย OR) · หนึ่งเข้าหลายออก = DEMUX (ไม่มี OR)
ให้ตาราง Σm(...) แล้วบอกว่ามี MUX ให้ใช้	MUX สร้างฟังก์ชัน	ตัวแปรตัวบนไปสายเลือก · แต่ละคู่แถวเป็น 0 / 1 / ตัวแปร / ตัวกลับค่า

ขั้นที่ 3 — สูตรทั้งบทในหน้าเดียว

แผ่นโกงส่วนคำนวณ

วงจร	ผลหลัก	ผลพ่วง (ตัวทด / ตัวยืม)
ตัวบวกครึ่ง	$\Sigma = A \oplus B$	$C_{out} = A \cdot B$
ตัวบวกเต็ม	$\Sigma = A \oplus B \oplus C_{in}$	$C_{out} = A \cdot B + C_{in}(A \oplus B)$
ตัวลบครึ่ง	$\Delta = A \oplus B$	$B_{out} = \bar{A} \cdot B$
ตัวลบเต็ม	$\Delta = A \oplus B \oplus B_{in}$	$B_{out} = \bar{A} \cdot B + B_{in} \cdot \overline{(A \oplus B)}$
look-ahead ชั้น 0	$C_{out0} = C_{g0} + C_{p0}C_{in}$	
look-ahead ชั้น 1	$C_{out1} = C_{g1} + C_{p1}C_{g0} + C_{p1}C_{p0}C_{in}$	
look-ahead ชั้น 2	$C_{out2} = C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{g0} + C_{p2}C_{p1}C_{p0}C_{in}$	
เปรียบเทียบ (2 บิต)	$E = (A_1 \odot B_1)(A_0 \odot B_0) \cdot G = A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0 \cdot L = \bar{A}_1B_1 + (A_1 \odot B_1)\bar{A}_0B_0$	
เข้ารหัส 8-to-3	$A_2 = D_4+D_5+D_6+D_7 \cdot A_1 = D_2+D_3+D_6+D_7 \cdot A_0 = D_1+D_3+D_5+D_7$	
ถอดรหัส 3-to-8	$Y_5 = A_2\bar{A}_1A_0$ (บิต 0 ใส่ขีด) · active-low ก็คือ NAND	
ฐานสอง → เกรย์	$G_3 = B_3 \cdot G_2 = B_3 \oplus B_2 \cdot G_1 = B_2 \oplus B_1 \cdot G_0 = B_1 \oplus B_0$	
เกรย์ → ฐานสอง	$B_3 = G_3 \cdot B_2 = B_3 \oplus G_2 \cdot B_1 = B_2 \oplus G_1 \cdot B_0 = B_1 \oplus G_0$	
MUX 4 ต่อ 1	$Y = \bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3$	
DEMUX 1 ต่อ 4	$O_0 = D \cdot \bar{S}_1\bar{S}_0 \cdot O_1 = D \cdot \bar{S}_1S_0 \cdot O_2 = D \cdot S_1\bar{S}_0 \cdot O_3 = D \cdot S_1S_0$	

ขั้นที่ 4 — กับดักทั้งบท รวมไว้ที่เดียว

สิบข้อที่ทำให้เสียคะแนนบ่อยที่สุด

	กับดัก	วิธีกัน
1	เกตปิดท้ายตัวบวกเต็มใส่ AND แทน OR	ตัวบวกเต็มใช้เกต 5 ตัว และตัวสุดท้ายเป็น **OR เสมอ**
2	อ่านคำตอบของตัวบวกไล่เร็วเกินไป	คำถามเรื่องเวลาให้ตอบด้วย**กรณีแย่ที่สุด** = เวลาต่อชั้น × จำนวนบิต
3	look-ahead ลืมพจน์สุดท้ายที่มี C_{in}	นับพจน์ก่อนส่ง · ชั้น k ต้องมี **k+2 พจน์**
4	ตัวลบเต็มใส่ขีดแค่ที่ A	ต้องมีขีด **2 จุด** — ที่ A และที่วงเล็บ $(A \oplus B)$
5	เปรียบเทียบไล่จากบิตขวาสุด	บวกไล่จากขวา · **เทียบไล่จากซ้าย** — คนละทิศเสมอ
6	ตัวเข้ารหัสธรรมดาถูกกดสองปุ่ม	ตอบว่าได้คำตอบมั่ว จึงต้องใช้ **priority encoder**
7	ตัวถอดรหัสลืมขีดตรงบิตที่เป็น 0	ทุก AND ต้องมีตัวแปร**ครบทุกตัว** และขีดต้องตรงกับรหัส
8	จับคู่ 74LS47 กับจอผิดแบบ	47 = active-low คู่**แอโนดร่วม** · 48 = active-high คู่**แคโทดร่วม**
9	เกรย์ขากลับใช้บิตเกรย์คู่ติดกัน	ขากลับใช้**ผลฐานสองที่เพิ่งได้** · ตรวจด้วยการแปลงไปกลับ
10	MUX สลับสายเลือก S_1 กับ S_0	ตัวแปรน้ำหนักมากสุดไปที่ S ตัวบนสุด · เขียนกำกับไว้ข้างรูป

ขั้นที่ 5 — ลำดับการลงมือในห้องสอบ

ทำตามนี้ทุกข้อ ไม่ต้องคิดใหม่

1. **อ่านโจทย์แล้ววงคำสำคัญก่อนจับปากกาคำนวณ**
   คำอย่าง carry · borrow · select · priority · active-low · look-ahead บอกชนิดวงจรได้ทันทีจากตารางข้างบน

2. **เขียน "สิ่งแรกที่ต้องเขียน" ลงกระดาษทันที**
   ยังไม่ต้องแทนตัวเลข · แค่เขียนสูตรหรือโครงวงจรให้ครบก่อน จะได้เห็นว่าขาดอะไร

3. **แทนค่าทีละบรรทัด ห้ามคิดในใจข้ามขั้น**
   คะแนนส่วนใหญ่อยู่ที่ขั้นตอน ไม่ใช่คำตอบสุดท้าย · เขียนไว้ว่าบรรทัดนี้กำลังทำอะไร

4. **เช็กคำตอบกลับเข้าโจทย์เสมอ**
   แปลงเป็นเลขฐานสิบแล้วบวก/ลบ/เทียบดูจริง · หรือแทนค่าย้อนกลับเข้าวงจร · ใช้เวลา 20 วินาทีแต่กันเสียคะแนนได้ทั้งข้อ

5. **หยุดเมื่อตอบครบสิ่งที่ถูกถาม**
   โจทย์ถามตารางก็ให้ตาราง ถามวงจรก็ให้วงจร · **อย่าลดรูปนิพจน์ อย่าคำนวณเวลา อย่าวาดเพิ่ม** ถ้าโจทย์ไม่ได้สั่ง เพราะทำเกินแล้วผิดก็เสียคะแนนได้

คำสุดท้ายของบทนี้

วงจรทั้ง 12 หัวข้อในบทนี้ประกอบขึ้นจากเกต 7 ตัวที่น้องรู้จักอยู่แล้วทั้งหมดครับ ไม่มีอะไรใหม่เลยในระดับเกต

สิ่งที่ใหม่คือ**วิธีจัดเรียง** — และทุกการจัดเรียงมาจาก**คำถามภาษาคนหนึ่งคำถาม**
"ทดไหม" → AND · "มีสักตัวไหม" → OR · "ต่างกันไหม" → XOR · "เหมือนกันไหม" → XNOR · "ตรงกับรหัสนี้ไหม" → AND ที่ใส่ขีดตามรหัส

ถ้าน้องจำไม่ได้ว่าสูตรหน้าตายังไง **ให้กลับไปตั้งคำถามภาษาคนนั้นใหม่** แล้วแปลเป็นเกตทีละคำ · น้องจะสร้างสูตรขึ้นมาเองได้เสมอ ซึ่งปลอดภัยกว่าการท่องมากครับ

บทที่ 6 · Timing diagram

อ่านและวาดรูปคลื่นของเกต

ทำไมต้องมีบทนี้

ในสไลด์ Midterm Review อาจารย์เขียนไว้ว่าเกตหนึ่งตัววิเคราะห์ได้ **3 ทาง** คือ Boolean expression · Truth table · Timing diagrams สองทางแรกน้องเจอมาทั้งบทที่ 3 แล้วครับ

ทางที่สามคือบทนี้ และเป็นทางที่**คนทำข้อสอบพลาดเยอะที่สุด** ไม่ใช่เพราะมันยาก แต่เพราะมันเป็นวิธีเดียวในสามวิธีที่**ไม่มีใครเคยให้ลองทำ** พอเจอในห้องสอบก็เลยนั่งมองรูปแล้วไม่รู้จะเริ่มตรงไหน

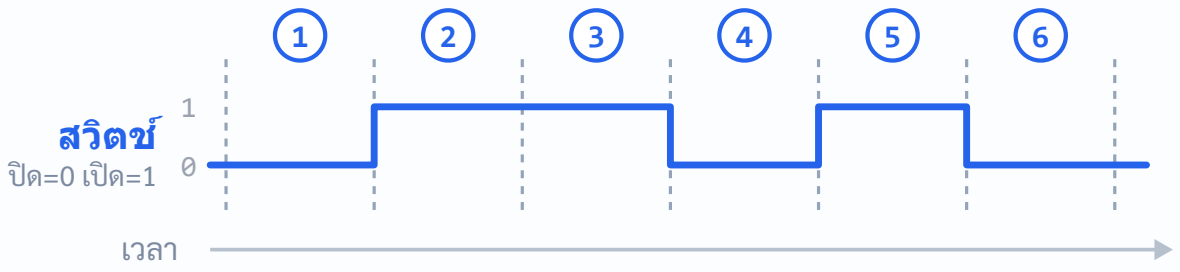
บทนี้จะเริ่มจากสวิตช์ไฟบนผนัง ไม่ใช่จากศัพท์ครับ พอจบบท น้องจะ**วาดขาออกของเกตทั้ง 7 ตัวได้ วาดวงจรสองเกตได้** และ**ทำย้อนทางได้** — คือให้รูปคลื่นขาเข้ากับขาออกมา แล้วบอกได้ว่าเป็นเกตอะไร

1 เริ่มจากสวิตช์ไฟบนผนัง

สมมติว่ามีสวิตช์ไฟอยู่บนผนังหนึ่งตัวครับ น้องเดินเข้าห้อง กดเปิด ปล่อยไว้สักพัก แล้วกดปิด เดินไปทำอย่างอื่น แล้วกลับมากดเปิดอีกรอบ กดปิดอีกรอบ

ทีนี้ถ้าพี่ขอให้น้อง**วาดรูปเล่าให้พี่ฟัง**ว่าเมื่อกี้เกิดอะไรขึ้นบ้าง น้องจะวาดยังไงครับ วิธีที่คนส่วนใหญ่คิดออกเองคือลากเส้นนอนเป็นเวลา แล้วยกเส้นขึ้นตอนไฟติด ลดเส้นลงตอนไฟดับ

ซึ่ง**ถูกแล้วครับ** นั่นแหละคือรูปข้างล่างนี้เป๊ะ ๆ



รูปที่ 1 · สวิตช์ไฟหนึ่งตัวถูกกดเปิด-ปิดไปเรื่อย ๆ · แกนนอนคือเวลา เส้นสูง=เปิด เส้นต่ำ=ปิด · เส้นประแบ่งเวลาออกเป็น 6 ช่วง ①–⑥

รูปแบบนี้แหละครับที่วิศวกรใช้เล่าว่าสายไฟเส้นหนึ่ง "เป็นยังไงบ้างตลอดช่วงเวลาที่ผ่านมา" มันมีชื่อเรียกว่า timing diagram (ไดอะแกรมเวลา) ครับ และคำว่า "รูปคลื่น" ที่พี่จะใช้ต่อจากนี้ ก็คือเส้นหยัก ๆ หนึ่งเส้นในรูปนั้นเอง

กติกาของรูป — มีแค่ 3 ข้อ

① **แกนนอนคือเวลา** เดินจากซ้ายไปขวาเสมอ ไม่มีตัวเลขบอกวินาทีก็ได้ เพราะเราสนใจแค่ "ก่อน-หลัง" ไม่ได้สนใจว่ากี่วินาที

② **เส้นอยู่สูง = 1 (HIGH) · เส้นอยู่ต่ำ = 0 (LOW)** มีแค่สองระดับเท่านั้น ไม่มีระดับกลาง เพราะนี่คือวงจร**ดิจิทัล** — สวิตช์ปิดหรือเปิด ไม่มีครึ่ง ๆ กลาง ๆ

③ **เส้นประแนวตั้งคือจุดที่มีอะไรเปลี่ยน** เส้นประพวกนี้หั่นเวลาออกเป็นช่วง ๆ พี่จะใส่เลขในวงกลม ①②③... กำกับทุกช่วงไว้ให้ ตลอดทั้งบท **"ช่วง ③" หมายถึงแถบเวลาระหว่างเส้นประที่ 3 กับที่ 4**

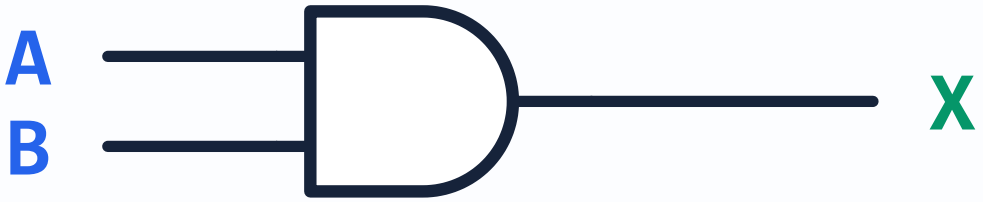
ทำไมต้องหั่นเป็นช่วง

เพราะภายใน**ช่วงเดียวกัน ไม่มีอะไรเปลี่ยนเลย**ครับ ค่าของทุกเส้นคงที่ตลอดทั้งช่วง เพราะฉะนั้นแทนที่จะต้องคิดเรื่อง "เวลา" ซึ่งเป็นของต่อเนื่องไม่รู้จบ เราเหลือแค่**ต้องตอบคำถามเดิม 6 รอบ** รอบละช่วง ซึ่งแต่ละรอบก็คือการเปิดตารางค่าความจริงหนึ่งแถวเท่านั้นเอง **นี่คือหัวใจทั้งหมดของบทนี้** ที่เหลือเป็นรายละเอียด

2 ลองทำเองก่อน — เกต AND กับรูปคลื่นสองเส้น

ทีนี้เอาสวิตช์**สองตัว**มาต่อเข้าเกต AND ครับ สวิตช์ตัวบนเรียก A ตัวล่างเรียก B ไฟที่ออกจากเกตเรียก X

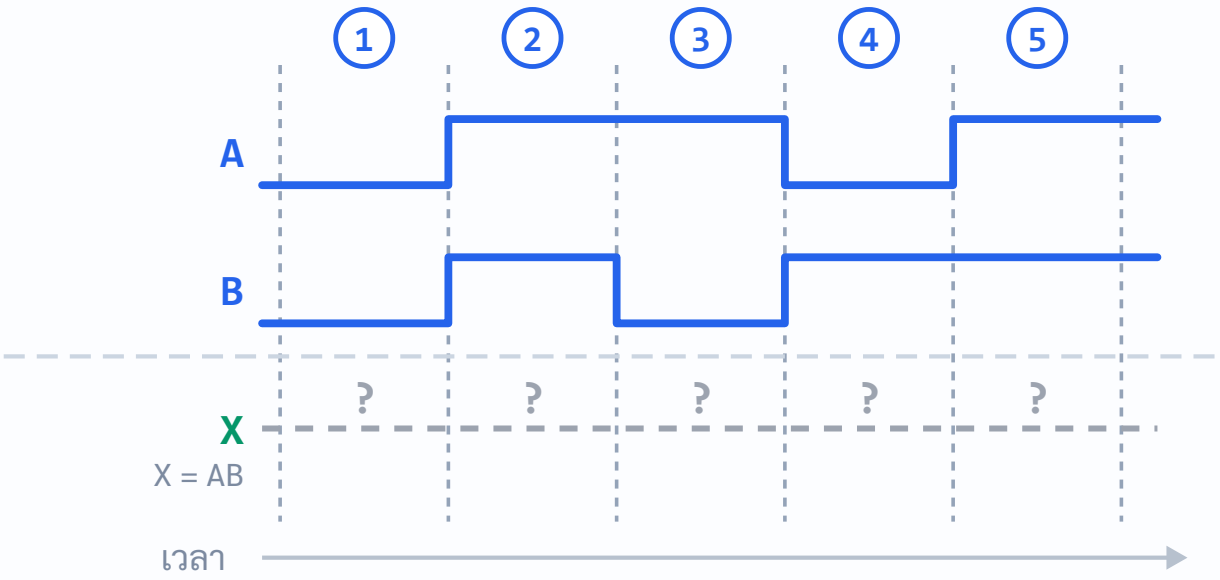
เกต AND ทำอะไร น้องรู้อยู่แล้วจากบทที่ 3 ครับ — **ออก 1 ก็ต่อเมื่อขาเข้าเป็น 1 ทุกขา** ถ้ามีขาไหนเป็น 0 แม้แต่ขาเดียว ผลออกเป็น 0 ทันที เขียนเป็นสูตรคือ $X = AB$



รูปที่ 2 · เกต AND หนึ่งตัว · สวิตช์สองตัวเข้าทางซ้าย ไฟออกทางขวา · $X = AB$

ลองคิดในหัวก่อน ไม่ต้องเขียน — รูปข้างล่างให้ A กับ B มาครบแล้ว ส่วน X ยังว่างอยู่ (เส้นประเทา ๆ ที่มีเครื่องหมาย ? อยู่)

คำถามง่าย ๆ ข้อเดียวครับ: **ช่วงไหนบ้างที่ไฟ X จะติด** ไม่ต้องคิดเป็นสูตร คิดแบบสวิตช์ไฟเลย — ไฟจะติดตอนที่สวิตช์ทั้งสองตัวเปิดพร้อมกันเท่านั้น

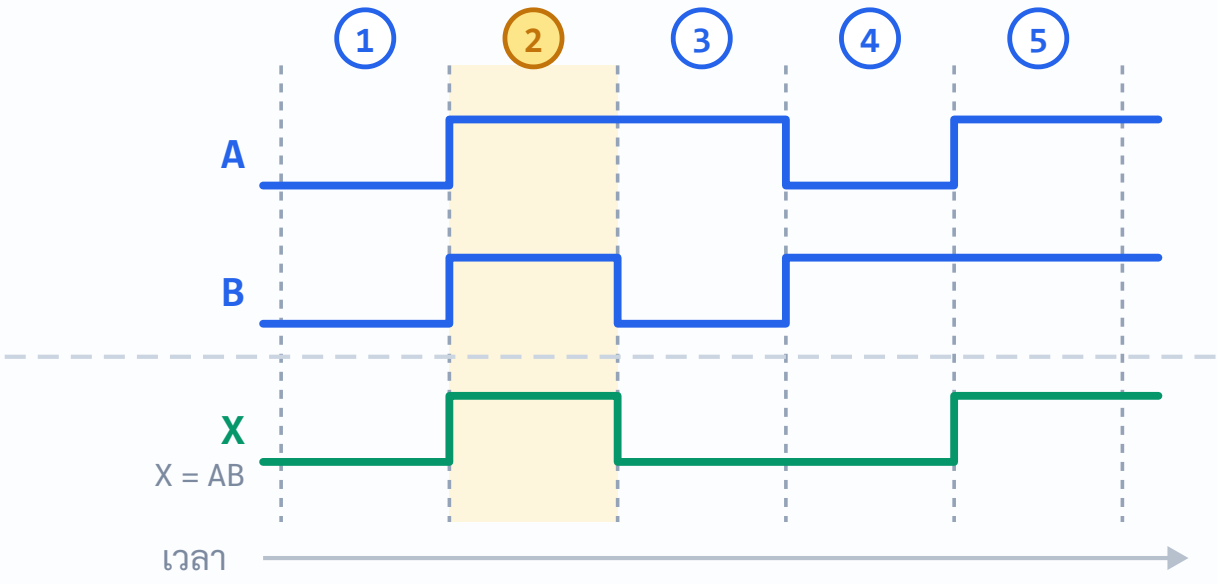


รูปที่ 3 · ขาเข้า A กับ B ของเกต AND ให้มาแล้ว · ลองเติมเส้น X ด้วยตัวเองก่อนเปิดหน้าถัดไป

เฉลย

ตอบ **ช่วง ② กับ ช่วง ⑤** ใช่ไหมครับ

ถ้าตอบได้แบบนั้น แปลว่าน้อง**อ่าน timing diagram เป็นแล้วครับ** — สิ่งที่น้องเพิ่งทำในหัวเมื่อกี้ คือการไล่ดูทีละช่วงแล้วถามว่า "ช่วงนี้ทั้งสองเส้นอยู่สูงพร้อมกันไหม" ซึ่งก็คือ**การเปิดตารางค่าความจริงของ AND ทีละแถว** เพียงแต่แทนที่จะอ่านจากตาราง น้องอ่านจากรูป



รูปที่ 4 · เฉลย — X สูงเฉพาะช่วงที่ A และ B สูงพร้อมกัน คือช่วง ② กับ ⑤ (ช่วง ② ระบายไว้ให้ดู)

ทีนี้ลองเขียนสิ่งที่น้องคิดในหัวออกมาเป็นตาราง จะได้หน้าตาแบบนี้ครับ — **หนึ่งแถวคือหนึ่งช่วงเวลา** ไม่ใช่หนึ่งกรณีเหมือนตารางค่าความจริงปกติ

ช่วงเวลา	A	B	X = AB
①	0	0	0
②	1	1	1
③	1	0	0
④	0	1	0
⑤	1	1	1

สองแถวที่ระบายไว้คือช่วงที่ A กับ B สูงพร้อมกัน · อีกสามช่วง X ต่ำหมด

จุดที่ต้องเข้าใจให้แน่นก่อนไปต่อ

ตารางเมื่อกี้กับตารางค่าความจริงของ AND ที่น้องท่องมา **เป็นของคนละอย่าง**ครับ ต่างกันตรงนี้:

ตารางค่าความจริง ไล่ **ทุกกรณีที่เป็นไปได้** — A, B สองขา ก็ต้องมีครบ 4 แถว 00 · 01 · 10 · 11 เรียงตามลำดับตายตัว และไม่มีแถวไหนซ้ำ

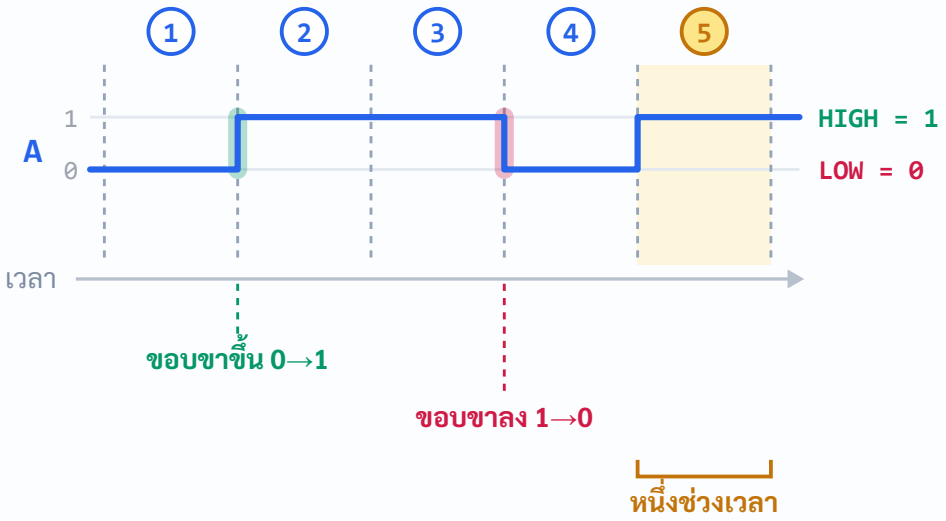
ตารางต่อช่วงเวลา ไล่ **ตามที่เกิดขึ้นจริง** — มีกี่ช่วงก็กี่แถว เรียงตามเวลา **ซ้ำกันได้** (ช่วง ② กับ ⑤ ในรูปที่ 4 เป็นกรณี $A{=}1, B{=}1$ เหมือนกันเป๊ะ) และ**ไม่จำเป็นต้องครบทุกกรณี** — รูปที่ 4 บังเอิญครบทั้ง 4 กรณีพอดี แต่นั่นเป็นเรื่องบังเอิญครับ วงจร 3 ขาเข้าในหัวข้อ 5 มี 6 ช่วง แต่ได้กรณีที่ต่างกันจริงแค่ **5 กรณี จาก 8 กรณี** ที่เป็นไปได้

กับดักข้อแรก — อย่าวาดตามจำนวนแถวของตารางค่าความจริง

คนที่ท่องมาว่า "AND 2 ขา มี 4 แถว" พอเจอรูปที่มี 5 ช่วง จะเผลอวาดขาออกแค่ 4 ช่วงแล้วงงว่าทำไมเหลือ **จำนวนช่วงในรูปไม่เกี่ยวอะไรกับจำนวนแถวของตารางเลยครับ** รูปมีกี่ช่วง ขาออกก็ต้องมีค่ากำกับครบเท่านั้นช่วง — 5 ช่วงก็ 5 ค่า 9 ช่วงก็ 9 ค่า

3 อ่านรูปคลื่นขาเข้าให้ขาด — ตั้งชื่อของทุกชิ้นบนรูป

ก่อนจะวาดขาออกเป็น ต้องอ่านขาเข้าให้ขาดก่อนครับ ซึ่งมีของอยู่แค่ **4 ชิ้น** บนรูป และทั้ง 4 ชิ้นมีชื่อของมันที่อาจารย์จะใช้ในโจทย์



รูปที่ 5 · ของ 4 ชิ้นบนรูปคลื่น — ระดับ HIGH · ระดับ LOW · ขอบขาขึ้น/ขาลง · และหนึ่งช่วงเวลา

ลองคิดในหัวก่อน ไม่ต้องเปิดตารางข้างล่าง — ดูรูปที่ 5 ตรงช่วงที่มีวงกลมสีส้ม (ช่วง 5)

คำถามง่าย ๆ ข้อเดียวครับ: **ช่วงนั้นเส้น A อยู่สูงหรือต่ำ** แล้วก่อนเข้าช่วงนั้นเส้นเพิ่งขึ้นหรือเพิ่งลง ลองทายจากรูปเฉย ๆ ก่อน

ตอบ อยู่สูง กับเพิ่งขึ้นมา ใช่ไหม — สองอย่างนี้แหละคือของที่ตารางข้างล่างจะเรียกชื่อให้ว่า "ระดับ HIGH" กับ "ขอบขาขึ้น"

ชื่อบนรูป	คำที่อาจารย์ใช้	หมายความว่าอะไร
ระดับสูง	`HIGH` / logic 1	ช่วงนั้นสายนี้มีไฟ · ในโจทย์คือ "ตัวแปรนี้มีค่า 1"
ระดับต่ำ	`LOW` / logic 0	ช่วงนั้นสายนี้ไม่มีไฟ · ในโจทย์คือ "ตัวแปรนี้มีค่า 0"
ขอบขาขึ้น	`rising edge` / leading edge	เส้นแนวตั้งที่วิ่งจากต่ำขึ้นสูง คือ**จุดที่ค่าเปลี่ยนจาก 0 เป็น 1**
ขอบขาลง	`falling edge` / trailing edge	เส้นแนวตั้งที่วิ่งจากสูงลงต่ำ คือ**จุดที่ค่าเปลี่ยนจาก 1 เป็น 0**
หนึ่งช่วงเวลา	`time interval`	ช่องระหว่างเส้นประสองเส้นที่ติดกัน · **ภายในช่องนี้ทุกค่าคงที่**

วิธีขีดเส้นประเองเมื่อโจทย์ไม่ได้ขีดมาให้

ข้อสอบบางข้อให้รูปคลื่นมาเฉย ๆ ไม่มีเส้นประครับ ให้น้อง**ขีดเองก่อนทำ** — กวาดตาดูทุกเส้นพร้อมกัน เจอจุดไหนที่มี**เส้นใดเส้นหนึ่งเปลี่ยนระดับ** ให้ลากเส้นประแนวตั้งลงมาตลอดทั้งรูป แล้วไล่ใส่เลข ①②③... ทุกช่อง **ห้ามข้ามขั้นนี้** เพราะเส้นประคือสิ่งที่ทำให้เราตอบทีละช่องได้โดยไม่หลง

กับดัก — เส้นประมาจาก "เส้นใดเส้นหนึ่ง" ไม่ใช่ "เส้นบน"

คนที่ขีดเส้นประตามขอบของเส้น A อย่างเดียวจะได้ช่องน้อยเกินไปครับ แล้วพอภายในช่องหนึ่งของเขา เส้น B ดันเปลี่ยนค่ากลางช่อง ขาออกก็จะเปลี่ยนค่ากลางช่องตามไปด้วย — ซึ่งวาดไม่ได้ เพราะเราตกลงกันไว้ว่าภายในช่องทุกอย่างต้องคงที่ **ลากเส้นประที่ขอบของทุกเส้นรวมกัน แล้วจำนวนช่องจะพอดีเสมอ**

4 วาดขาออกของเกตทั้ง 7 ตัว

ตอนนี้น้องอ่านขาเข้าเป็นแล้ว เหลือแค่**วาดขาออก**ครับ ซึ่งมีวิธีเดียว และใช้ได้กับเกตทุกตัวไม่มียกเว้น

วิธีนั้นคือ **ทำทีละช่วง ไม่ใช่ทีละเส้น** ครับ คนที่พยายามลากเส้นขาออกรวดเดียวตั้งแต่ซ้ายยันขวาจะพังแน่นอน เพราะสมองต้องจำหลายอย่างพร้อมกัน แต่ถ้าทำทีละช่วง แต่ละรอบเหลือแค่คำถามเดียวที่น้องตอบได้อยู่แล้ว

1 **ตั้งลำแสงตรงช่วงที่ 1 แล้วอ่านค่าขาเข้าลงมาในแนวตั้ง**
เอานิ้วหรือไม้บรรทัดทาบแนวตั้งบนช่วงนั้น อ่านจากบนลงล่างว่าแต่ละเส้นสูงหรือต่ำ เขียนไว้ข้าง ๆ เลยครับ เช่น "ช่วง ① : A=0, B=0"

2 **เปิดตารางค่าความจริงแถวที่ตรงกับค่านั้น**
ค่าที่อ่านได้คือ**เลขที่อยู่หัวแถว**ของตารางค่าความจริง · หาแถวนั้น แล้วดูว่าช่องขาออกเป็นอะไร

3 **จุดจุดเดียวลงในแถวขาออก ตรงกลางช่วงนั้น**
จุด ไม่ใช่ลากเส้นครับ — จุดสูงถ้าได้ 1 จุดต่ำถ้าได้ 0 แล้วขยับไปช่วงถัดไป

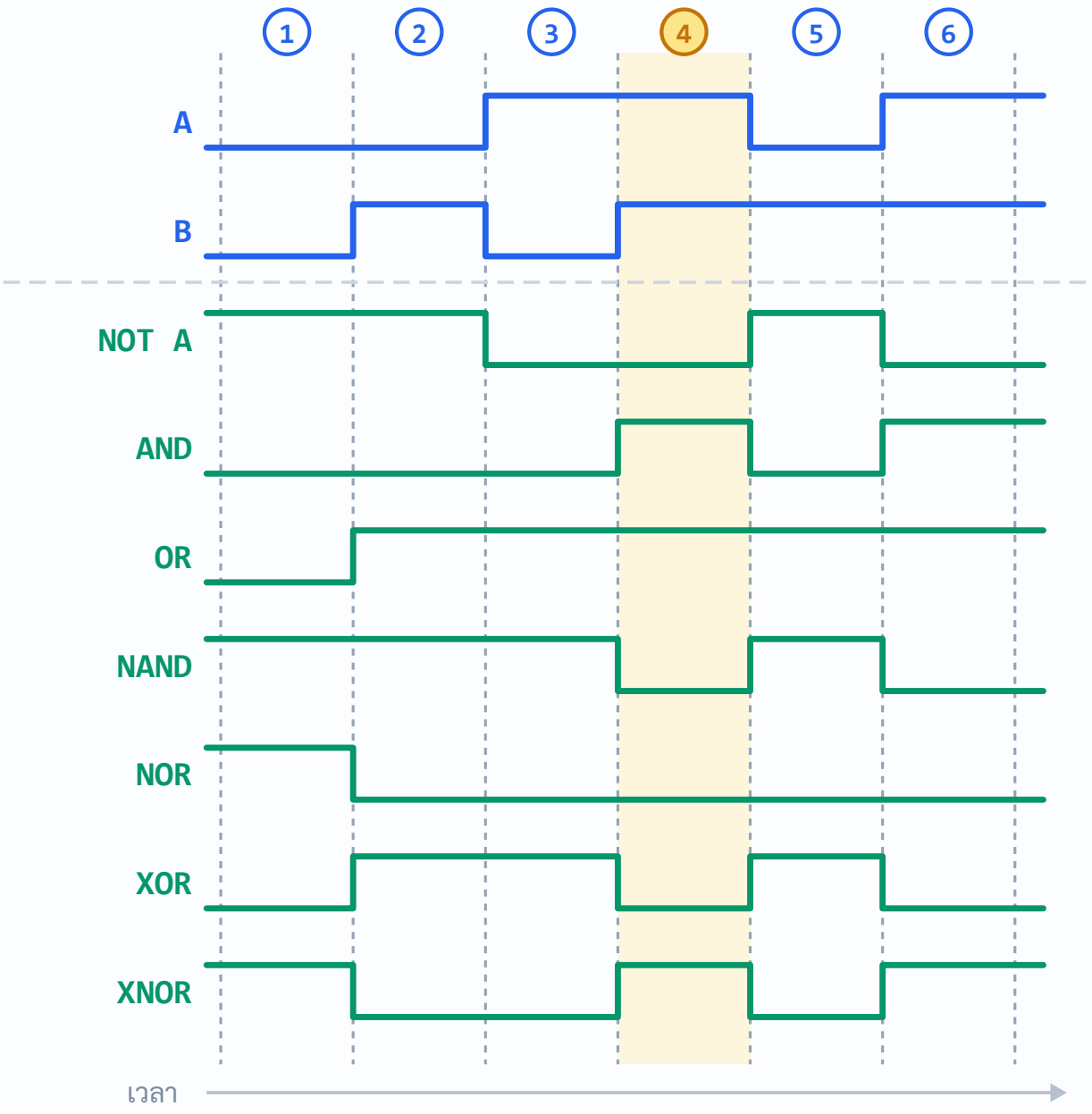
4 **ครบทุกช่วงแล้วค่อยลากเส้นเชื่อมจุด**
ลากนอนจากจุดหนึ่งไปอีกจุด ถ้าสองจุดที่ติดกันอยู่คนละระดับ ให้**ลากแนวตั้งตรงเส้นประพอดี** ไม่ใช่ลากเฉียง เพราะการเปลี่ยนค่าเกิดขึ้นที่เส้นประเป๊ะ ๆ

ทำไมต้องจุดให้ครบก่อนแล้วค่อยลาก

เพราะถ้าลากไปด้วยคิดไปด้วย พอคิดผิดหนึ่งช่วง เส้นที่ลากมาแล้วจะพาให้คิดผิดต่อทั้งแถวครับ การจุดให้ครบก่อนทำให้แต่ละช่วง**ตัดสินใจแยกจากกันจริง ๆ** และถ้าจะตรวจงาน ก็ตรวจทีละจุดได้เลย

ตัวอย่างที่ 1 — ขาเข้าชุดเดียว ป้อนเข้าเกตครบทั้ง 7 ตัว

รูปข้างล่างใช้ขาเข้า A กับ B ชุดเดียวกันตลอด แล้ววาดขาออกของเกตทั้ง 7 ตัวเทียบกันให้ดูครับ **สังเกตว่าเส้นขาเข้าสองเส้นบนสุดเหมือนกันหมด แต่ขาออกข้างล่างไม่ซ้ำกันเลยสักคู่** — นั่นคือเหตุผลที่ข้อสอบถามด้วยรูปคลื่นได้ เพราะรูปคลื่นแยกเกตออกจากกันได้จริง



รูปที่ 6 · ขาเข้าชุดเดียวกัน ป้อนเข้าเกตทั้ง 7 ตัว · ช่วง ④ ที่ระบายไว้คือช่วงที่ A กับ B สูงพร้อมกัน สังเกตว่าเกตแต่ละตัวตอบต่างกันหมด

ลองคิดในหัวก่อน ไม่ต้องเปิดตารางข้างล่าง — ดูรูปที่ 6 เฉพาะแถว A กับ B

คำถามง่าย ๆ ข้อเดียวครับ: **ช่วง ① กับช่วง ④ (วงกลมสีส้ม) เกต AND จะออก 0 หรือ 1** ใช้กฎเดิมจากบทที่ 3 ได้เลย — AND ออก 1 ก็ต่อเมื่อ A กับ B สูงพร้อมกันเท่านั้น

ตอบ ช่วง ① ออก 0 ช่วง ④ ออก 1 ใช่ไหม — เพราะช่วง ① ทั้งคู่ต่ำ ช่วง ④ ทั้งคู่สูง ลองไล่ดูแถว AND ในรูปที่ 6 จะตรงเป๊ะ ที่เหลือใช้หลักเดียวกันนี้กับอีก 6 เกต

เขียนเป็นตารางต่อช่วงจะได้แบบนี้ครับ ทุกตัวเลขในตารางตรงกับทุกเส้นในรูปที่ 6 เป๊ะ

ช่วงเวลา	A	B	NOT A	AND	OR	NAND	NOR	XOR	XNOR
①	0	0	1	0	0	1	1	0	1

ช่วงเวลา	A	B	NOT A	AND	OR	NAND	NOR	XOR	XNOR
②	0	1	1	0	1	1	0	1	0
③	1	0	0	0	1	1	0	1	0
④	1	1	0	1	1	0	0	0	1
⑤	0	1	1	0	1	1	0	1	0
⑥	1	1	0	1	1	0	0	0	1

อ่านทีละคอลัมน์ในแนวตั้ง = ตารางค่าความจริงของเกตตัวนั้น เพียงแต่เรียงตามเวลาแทนที่จะเรียงตาม 00-01-10-11

อ่านรูปให้เร็วขึ้น — ประโยคเดียวต่อหนึ่งเกต

พอเห็นรูปที่ 6 ทั้งแผ่นพร้อมกัน จะจับสังเกตได้ว่าเกตแต่ละตัวมี**ลายเซ็นบนรูปคลื่น**ของมันเอง ตารางข้างล่างคือสิ่งที่ให้มองหาในรูป แทนที่จะต้องเปิดตารางค่าความจริงทุกช่วง

เกต	มองหาอะไรในรูป	ผลที่ได้ในรูปที่ 6	ตรวจซ้ำยังไง
NOT	พลิกเส้นเดิม**บน-ล่าง** ตรงไหนสูงให้เป็นต่ำ	NOT A สูงที่ ①②⑤ ซึ่งคือช่วงที่ A ต่ำพอดี	วางสองเส้นซ้อนกัน ต้องไม่มีช่วงไหนสูงพร้อมกันเลย
AND	สูงเฉพาะช่วงที่**ทุกเส้นสูงพร้อมกัน**	สูงที่ ④⑥ · อีก 4 ช่วงต่ำ	นับช่วงที่ทุกเส้นสูง ต้องได้จำนวนเท่ากับช่วงที่ขาออกสูง
OR	ต่ำเฉพาะช่วงที่**ทุกเส้นต่ำพร้อมกัน** นอกนั้นสูงหมด	ต่ำที่ ① ช่วงเดียว · อีก 5 ช่วงสูง	หาช่วงที่ทุกเส้นต่ำ ต้องเป็นช่วงเดียวกับที่ขาออกต่ำ
NAND	วาด AND ก่อน แล้ว**พลิกบน-ล่าง**	ต่ำที่ ④⑥ · ตรงข้ามกับ AND ทุกช่วง	ทาบกับแถว AND ต้องไม่มีช่วงไหนเหมือนกันเลย
NOR	วาด OR ก่อน แล้ว**พลิกบน-ล่าง**	สูงที่ ① ช่วงเดียว · ตรงข้ามกับ OR ทุกช่วง	ทาบกับแถว OR ต้องไม่มีช่วงไหนเหมือนกันเลย
XOR	สูงเมื่อสองเส้น**ไม่เหมือนกัน** (เส้นหนึ่งสูง อีกเส้นต่ำ)	สูงที่ ②③⑤ ซึ่งเป็นช่วงที่ A กับ B อยู่คนละระดับ	ไล่ดูทีละช่วงว่า A กับ B อยู่ระดับเดียวกันหรือเปล่า
XNOR	สูงเมื่อสองเส้น**เหมือนกัน** (สูงคู่หรือ ต่ำคู่)	สูงที่ ①④⑥	ทาบกับแถว XOR ต้องตรงข้ามกันทุกช่วง

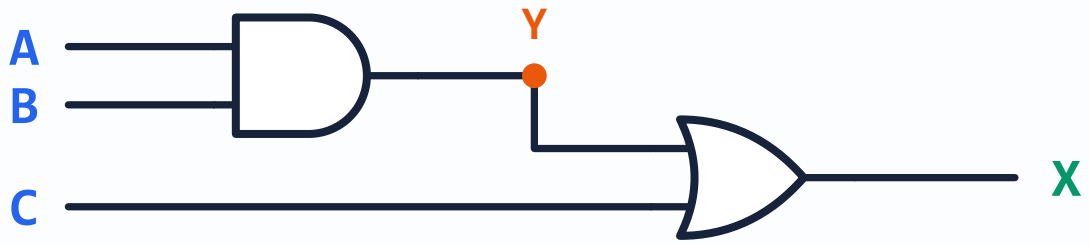
กับดักที่ทำคนเสียคะแนนมากที่สุดในบทนี้

① วาด NAND แล้วลืมพลิก — คนจำนวนมากอ่านโจทย์ว่า NAND แล้วสมองไปหยิบ "ทุกเส้นสูงพร้อมกัน" ของ AND มาใช้ ได้คำตอบเป็นรูป AND เป๊ะ ๆ แล้วส่ง **เสียคะแนนทั้งข้อ** เพราะทุกช่วงผิดหมด วิธีกัน: เขียนคำว่า "พลิก" ตัวโต ๆ ไว้ข้างแถวที่ต้องวาดก่อนลงมือ

② สลับ XOR กับ OR — สองตัวนี้เหมือนกันทุกช่วง **ยกเว้นช่วงที่ทั้งสองเส้นสูงพร้อมกัน** ในรูปที่ 6 คือช่วง ④ กับ ⑥ · OR ให้ 1 แต่ XOR ให้ 0 ถ้าในโจทย์มีช่วงแบบนี้แค่ช่วงเดียว ตอบผิดก็เสียแค่จุดเดียว แต่ถ้าโจทย์ถามว่า "เกตอะไร" ก็ตอบผิดทั้งข้อ (ดูหัวข้อ 6)

5 ตัวอย่างที่ 2 — วงจรที่มีเกตสองตัว

ข้อสอบจริงไม่ค่อยให้เกตตัวเดียวครับ ส่วนใหญ่ให้**วงจรเล็ก** ๆมาแล้วถามขาออกสุดท้าย ซึ่งไม่มีอะไรใหม่เลย แค่ต้อง**วาดจุดกลางก่อน**เท่านั้นเอง



รูปที่ 7 · วงจรสองเกต · จุดสีส้มคือ Y ซึ่งเป็นสายที่โจทย์ไม่ได้ตั้งชื่อให้ เราตั้งเอง · $Y = AB$ แล้ว $X = Y + C$ นั่นคือ $X = AB + C$

ขั้นตอนเพิ่มมาข้อเดียว

ก่อนเริ่มวาด ให้ตั้งชื่อทุกสายที่ออกจากเกตแล้วยังไม่ถึงปลายทาง ในรูปนี้มีสายเดียวคือสายที่ออกจาก AND ตั้งชื่อว่า Y ครับ แล้ว**เพิ่มแถว Y ลงในรูปคลื่นด้วย** — เพิ่มจริง ๆ ไม่ใช่คิดในใจ เพราะแถว Y นี่แหละคือขาเข้าของเกตตัวที่สอง ถ้าไม่มีมันน้องจะไม่มีอะไรไปป้อนเกต OR

เดินให้ดูทีละช่วง ครบทั้ง 6 ช่วง ไม่ข้าม

ขาเข้าที่โจทย์ให้มาคือรูปที่ 8 ข้างล่าง (สามแถวบน) ส่วนสองแถวล่างคือของที่เราต้องเติมเอง พี่จะเดินให้ดูทุกช่วงครับ **ช่วงละสองบรรทัด** — บรรทัดแรกหา Y บรรทัดที่สองหา X

ลองคิดในหัวก่อน ไม่ต้องเขียน — ดูวงจรในรูปที่ 7 อีกที $X = AB + C$

คำถามง่าย ๆ ข้อเดียวครับ: **ถ้าช่วงหนึ่ง $A = 0$ กับ $B = 0$ (สวิตช์ทั้งสองปิด) แต่ $C = 1$ (สวิตช์ C เปิด) ไฟ X จะติดไหม** คิดแบบสวิตช์ไฟเลย ไม่ต้องคิดเป็นสูตร

ตอบ ติด ใช่ไหม — $Y = AB = 0$ ก็จริง แต่เกต OR ที่รับ Y กับ C ขอแค่เส้นใดเส้นหนึ่งสูงก็พอ C สูงอยู่แล้วก็ดันให้ X ติดได้เลยโดยไม่ต้องรอ Y

ช่วง ① — อ่านค่า A=0 B=0 C=1

Y = A·B = 0 · 0 = **0** AND ต้องสูงทั้งคู่ ไม่ใช่

X = Y + C = 0 + 1 = **1** OR ขอแค่เส้นเดียวสูง C สูงอยู่

ช่วง ② — อ่านค่า A=1 B=1 C=0

Y = A·B = 1 · 1 = **1** สูงทั้งคู่พอดี

X = Y + C = 1 + 0 = **1** คราวนี้ Y เป็นคนดันขึ้น

ช่วง ③ — อ่านค่า A=1 B=0 C=0

Y = A·B = 1 · 0 = **0** B ต่ำ AND ก็ตก

X = Y + C = 0 + 0 = **0** ไม่มีใครสูงเลย

ช่วง ④ — อ่านค่า A=0 B=1 C=0

Y = A·B = 0 · 1 = **0** คราวนี้ A ต่ำแทน

X = Y + C = 0 + 0 = **0** ยังไม่มีใครสูง

ช่วง ⑤ — อ่านค่า A=1 B=1 C=1

Y = A·B = 1 · 1 = **1** สูงทั้งคู่อีกรอบ

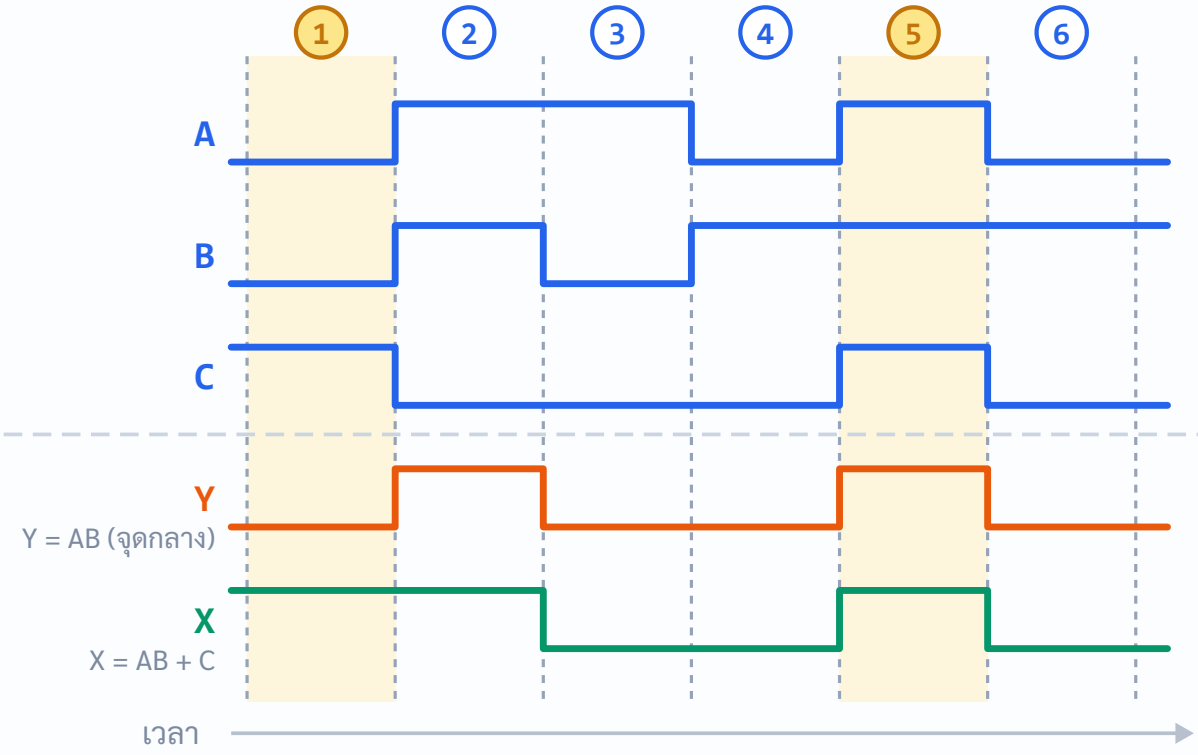
X = Y + C = 1 + 1 = **1** สูงทั้งคู่ OR ก็ยังได้ 1 ไม่ใช่ 2

ช่วง ⑥ — อ่านค่า A=0 B=1 C=0

Y = A·B = 0 · 1 = **0** เหมือนช่วง ④ เป๊ะ

X = Y + C = 0 + 0 = **0** ได้ค่าเดิมเพราะขาเข้าชุดเดิม

เอาค่าที่ได้ไปจุดลงรูป แล้วลากเส้นเชื่อม จะได้แบบนี้ครับ



รูปที่ 8 · วงจรสองเกต X = AB + C · แถวสีส้ม Y คือจุดกลางที่ต้องวาดก่อน · ช่วง ① ที่ระบายไว้ X สูงทั้งที่ Y ต่ำ เพราะ C เป็นคนดันขึ้นไปเอง

ช่วงเวลา	A	B	C	Y = AB	X = Y + C
①	0	0	1	0	1
②	1	1	0	1	1
③	1	0	0	0	0
④	0	1	0	0	0
⑤	1	1	1	1	1
⑥	0	1	0	0	0

ต้องเติมคอลัมน์ Y ให้ครบทั้ง 6 ช่วงก่อน แล้วค่อยเติมคอลัมน์ X · ข้ามขั้นนี้เมื่อไรพลาดทันที

ตรวจคำตอบกลับเข้าโจทย์เดิม

คำตอบคือ X สูงที่ช่วง ①②⑤ ทีนี้ตรวจด้วยวิธีที่สอง โดยไม่ต้องคำนวณใหม่ครับ:

โจทย์คือ $X = AB + C$ แปลว่า X ต้องสูง**ทุกช่วงที่ Y สูง บวกกับทุกช่วงที่ C สูง**

จากรูปที่ 8 · Y สูงที่ ②⑤ · C สูงที่ ①⑤ · เอาสองชุดมารวมกันได้ ①②⑤ — **ตรงกับคำตอบที่เราวาดพอดี** ถ้าไม่ตรงเมื่อไร แปลว่ามีช่วงใดช่วงหนึ่งจุดผิด กลับไปไล่ใหม่ทีละช่วงได้เลย

กับดักของวงจรหลายเกต

ช่วง ① คือช่วงที่ดักคนเยอะที่สุดครับ — A กับ B **ต่ำทั้งคู่** คนที่มองผ่าน ๆ จะรีบสรุปว่า "ขาเข้าต่ำหมด ขาออกก็ต้องต่ำ" แล้ววาด X ต่ำ แต่ C **สูงอยู่** และ C ไม่ได้ผ่านเกต AND เลย มันวิ่งตรงเข้า OR **เกต OR ขอแค่ขาเดียวสูงก็พอ** ผลลัพธ์จึงสูง · วิธีกันคือ**อย่าข้ามการเขียนแถว** Y พอมีแถว Y อยู่บนกระดาษ น้องจะเห็นเองว่าเกต OR รับ Y กับ C ไม่ได้รับ A กับ B

6 ตัวอย่างที่ 3 — ทำย้อนทาง ให้รูปคลื่นมาครบ แล้วถามว่าเกตอะไร

สองหัวข้อที่แล้วเราเดินจากซ้ายไปขวา คือ**รู้เกต แล้วหาขาออก**ครับ ข้อสอบชอบถามอีกทางหนึ่งด้วย คือให้รูปคลื่น**ครบทั้งขาเข้าและขาออก** แล้วถามว่าข้างในกล่องเป็นเกตอะไร



รูปที่ 9 · โจทย์แบบย้อนทาง — รู้ A รู้ B รู้ X ครบ แต่ไม่รู้ว่าข้างในกล่องเป็นเกตอะไร

ฟังดูเหมือนยากกว่า แต่จริง ๆ **ง่ายกว่าครับ** เพราะเราไม่ต้องสร้างอะไรใหม่เลย แค่**แปลรูปกลับเป็นตาราง** แล้วเทียบกับตารางค่าความจริงที่น้องท่องมา

1 **ทำตารางต่อช่วงให้ครบก่อน**

อ่านทุกช่วงในแนวตั้ง ได้ A B X ครบทุกช่วง · ขั้นนี้ยังไม่ต้องคิดอะไรเลย แค่คัดลอกจากรูป

2 **ตัดแถวที่ซ้ำกันออก เหลือแต่กรณีที่ต่างกัน**

ช่วงที่มี A, B เหมือนกันเป๊ะให้เก็บไว้แถวเดียว · ตรงนี้ได้ของแถมด้วย — **ถ้าสองช่วงมีขาเข้าเหมือนกันแต่ขาออกต่างกัน แปลว่าอ่านรูปผิด** กลับไปดูใหม่ทันที

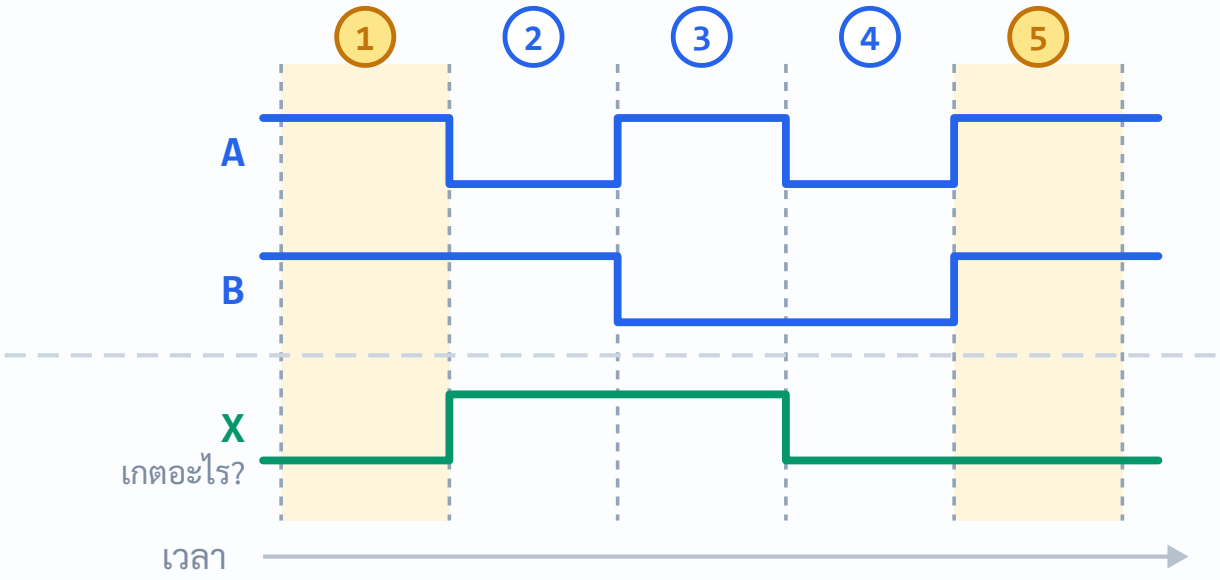
3 **เช็กว่าได้กรณีครบ 4 กรณีหรือเปล่า**

ขาเข้า 2 ขา มี 4 กรณี (00 · 01 · 10 · 11) · **ถ้าครบ 4 จะฟันธงได้ตัวเดียว** ถ้าไม่ครบ จะเหลือคำตอบที่เป็นไปได้มากกว่าหนึ่ง และต้องตอบไปตามนั้น

4 **เทียบกับตารางค่าความจริงของเกตทั้ง 7 ตัว**

ดูคอลัมน์ขาออกของตารางที่เราเพิ่งได้ แล้วหาว่าตรงกับเกตไหน · **เริ่มจากกรณี** $A{=}1, B{=}1$ **ก่อน** เพราะกรณีนี้แยกเกตออกจากกันได้ไวที่สุด

ลงมือกับรูปจริง



รูปที่ 10 · คราวนี้ให้ครบทั้งขาเข้าและขาออก แล้วถามว่าเป็นเกตอะไร · ช่วง ① กับ ⑤ ที่ระบายไว้คือช่วงชี้ขาด

ลองคิดในหัวก่อน ไม่ต้องเขียน — ดูรูปที่ 10 ให้ครบ ทั้ง A B X

คำถามง่าย ๆ ข้อเดียวครับ: **กล่อง ? ข้างในเป็นเกตอะไร** ลองทายดูก่อนเลย ใช้ตารางค่าความจริง 7 ตัวที่ท่องมาจากบทที่ 3 ช่วยคิดได้

ตอบ XOR ใช่ไหม — สังเกตช่วง ① กับ ⑤ ที่ A, B สูงทั้งคู่แต่ X กลับต่ำ มีแค่ XOR (กับ NAND) เท่านั้นที่ทำแบบนี้ได้ ขั้นตอนถัดไปจะพิสูจน์ให้ดูแบบเป็นระบบ

ขั้นที่ 1 — คัดลอกจากรูปลงตาราง ได้ตามนี้ครับ

ช่วงเวลา	A	B	X	กรณีนี้คือ
①	1	1	**0**	A=1, B=1
②	0	1	**1**	A=0, B=1
③	1	0	**1**	A=1, B=0
④	0	0	**0**	A=0, B=0
⑤	1	1	**0**	A=1, B=1 (ซ้ำกับ ①)

ขั้นที่ 2 — ช่วง ⑤ ซ้ำกับช่วง ① (ขาเข้า 1,1 เหมือนกัน) และขาออกก็เป็น 0 เหมือนกันทั้งคู่ **สอดคล้องกัน แปลว่าอ่านรูปไม่ผิด** ตัดช่วง ⑤ ทิ้ง เหลือ 4 แถว

ขั้นที่ 3 — 4 แถวที่เหลือคือกรณี 11 · 01 · 10 · 00 **ครบทั้ง 4 กรณีพอดี** แปลว่าข้อนี้ฟันธงได้ตัวเดียว ไม่มีคำตอบอื่น

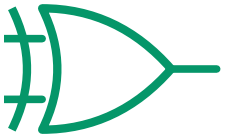
ขั้นที่ 4 — เรียงใหม่ตามลำดับมาตรฐาน 00 · 01 · 10 · 11 แล้วเทียบ:

A	B	X ที่อ่านได้
0	0	0
0	1	1
1	0	1
1	1	0

อ่านออกมาเป็นประโยค

"ออก 1 เมื่อขาเข้า**ไม่เหมือนกัน** · ออก 0 เมื่อขาเข้า**เหมือนกัน**"

นี่คือคำนิยามของ **XOR** ตรงตัวครับ

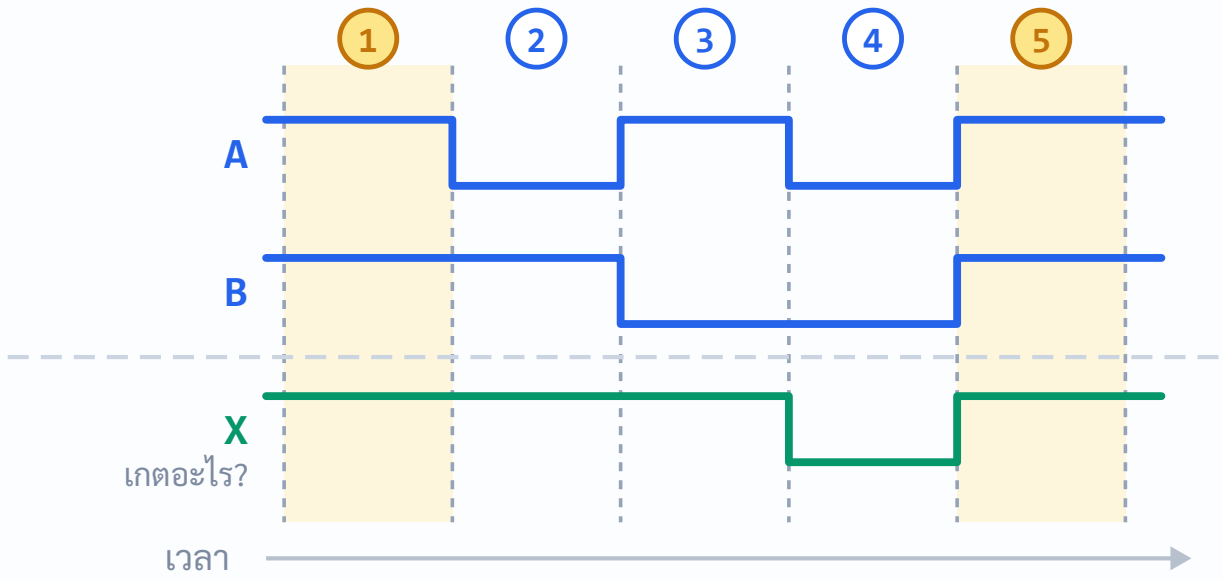


เฉลยตัวอย่างที่ 3

คำตอบ: เกต XOR · $X = A \oplus B$ · ตรวจซ้ำเร็ว ๆ ด้วยช่วงที่ชี้ขาด — ช่วง ① กับ ⑤ ขาเข้าสูงทั้งคู่ แต่ขาออก**ต่ำ** ซึ่งมีแค่ XOR กับ NAND ที่ทำแบบนี้ · แล้วช่วง ④ ขาเข้าต่ำทั้งคู่ ขาออก**ต่ำ** ซึ่ง NAND ทำไม่ได้ (NAND ต้องออก 1) เหลือ XOR ตัวเดียว

เดินให้เห็นตอนพัง — ตัวลวงที่ต่างกันแค่สองช่วง

ทีนี้พี่จะเปลี่ยนโจทย์นิดเดียวครับ **ขาเข้า A กับ B เหมือนเดิมเป๊ะทุกช่วง** เปลี่ยนแค่ขาออก แล้วดูว่าคำตอบเปลี่ยนไหม



รูปที่ 11 · ขาเข้าชุดเดิมเป๊ะ แต่ขาออกต่างจากรูปที่ 10 แค่ช่วง ① กับ ⑤ · ต่างแค่สองช่วงนี้ คำตอบก็เปลี่ยนจาก XOR เป็น OR

ช่วงเวลา	A	B	X ในรูปที่ 10	X ในรูปที่ 11
①	1	1	0	1
②	0	1	1	1
③	1	0	1	1
④	0	0	0	0
⑤	1	1	0	1

สองคอลัมน์ขวาต่างกันแค่ช่วง ① กับ ⑤ ซึ่งเป็นสองช่วงที่ A กับ B สูงพร้อมกันพอดี

ตารางข้างบนเอาขาออกของทั้งสองรูปมาวางเทียบกัน **ช่วง ②③④ เหมือนกันทุกประการ** ต่างกันแค่ **ช่วง ① กับ ⑤** — และสองช่วงนั้นคือช่วงที่ A กับ B สูงพร้อมกันพอดี

ทำขั้นที่ 1–4 ซ้ำกับรูปที่ 11 จะได้ตารางกรณี 00→0 · 01→1 · 10→1 · **11→1** ซึ่งอ่านเป็นประโยคได้ว่า "ออก 1 เมื่อมีขาไหนสูงก็ได้" นั่นคือ **เกต OR** ครับ ไม่ใช่ XOR

นี่คือจุดที่คนตอบผิดจริง ๆ ในข้อสอบ

ถ้าน้องเผลอดูแค่ "ขาออกสูงตอนที่ขาเข้าสูงอย่างน้อยหนึ่งเส้น" น้องจะตอบ OR ทั้งสองรูป เพราะข้อสังเกตนั้นเป็นจริงในรูปที่ 10 ด้วย (ช่วง ②③ ขาออกสูง และมีขาเข้าสูงหนึ่งเส้นจริง)

วิธีกันพลาดมีข้อเดียว: ดูช่วงที่ขาเข้าสูงพร้อมกันก่อนเสมอ ช่วงแบบนั้นแยก XOR ออกจาก OR ได้ทันที (XOR ให้ 0 · OR ให้ 1) และแยก AND ออกจาก NAND ได้ด้วย ถ้าในโจทย์**ไม่มีช่วงไหนเลย**ที่ขาเข้าสูงพร้อมกัน แปลว่าโจทย์ข้อนั้น**แยก XOR กับ OR ไม่ออกจริง ๆ** ให้ตอบไปว่าเป็นได้ทั้งสองตัว พร้อมบอกเหตุผล — นั่นคือคำตอบที่ถูก ไม่ใช่การเดา

7 สรุปเป็นขั้นตอนที่ทำตามได้ในห้องสอบ

ทั้งบทย่อลงมาเป็นสองเช็กลิสต์ครับ ใช้อันไหนขึ้นอยู่กับว่าโจทย์ให้อะไรมา

โจทย์ให้ "วงจร + ขาเข้า" → หาขาออก

1. ตั้งชื่อ**ทุกสายกลาง**ในวงจร (Y, Z, ...) แล้วเพิ่มเป็นแถวในรูป
2. ลากเส้นประแนวตั้งที่**ขอบของทุกเส้นรวมกัน** ใส่เลข ①②③... ทุกช่อง
3. ไล่**ทีละช่อง** อ่านค่าขาเข้าในแนวตั้ง → เปิดตารางค่าความจริง → **จุด**ลงแถวขาออก
4. วาดสายกลางให้ครบทุกช่องก่อน แล้วค่อยขยับไปเกตถัดไป
5. ครบทุกช่องแล้วค่อย**ลากเส้นเชื่อมจุด** ขึ้น-ลงตรงเส้นประเท่านั้น
6. **ตรวจกลับ**: เอาสูตรมาอ่านเป็นประโยค แล้วนับช่วงเทียบกับรูปที่วาด

หยุดตรงนี้ — ไม่ต้องย่อสูตร ไม่ต้องทำ K-map โจทย์ไม่ได้ถาม

โจทย์ให้ "ขาเข้า + ขาออก" → หาว่าเกตอะไร

1. คัดลอกทุกช่วงลงตาราง A B X · ยังไม่ต้องคิด
2. ตัดแถวที่ขาเข้าซ้ำกันออก · **ถ้าขาเข้าซ้ำแต่ขาออกต่าง = อ่านรูปผิด** กลับไปดูใหม่
3. นับว่าได้กี่กรณีจาก 4 กรณี · ไม่ครบ = ฟันธงตัวเดียวไม่ได้
4. **ดูกรณี** $A{=}1, B{=}1$ **ก่อน** แยกได้ไวที่สุด (XOR/NAND/NOR ให้ 0 · AND/OR/XNOR ให้ 1)
5. เทียบคอลัมน์ขาออกกับตารางค่าความจริงทั้ง 7 ตัว
6. **ตรวจกลับ**: เอาเกตที่ตอบไปวาดขาออกใหม่ ต้องได้รูปเดิมทุกช่วง

หยุดตรงนี้ — ถ้ากรณีไม่ครบ 4 ให้ตอบเป็น "ได้ทั้ง ... หรือ ..." อย่าเดาเอาตัวเดียว

เชื่อมกลับไปบทที่ 3

ตลอดทั้งบทนี้ไม่มีความรู้ใหม่เลยสักข้อครับ — ตารางค่าความจริงของเกตทั้ง 7 ตัวมาจากบทที่ 3 ทั้งหมด บทนี้เพิ่มมาแค่**วิธีอ่านและวาดรูป**เท่านั้น เพราะฉะนั้นถ้าตอนทำโจทย์ติดตรงไหน ให้กลับไปดูว่าติดที่**ตารางค่าความจริง** (ต้องกลับไปทบทวนบทที่ 3) หรือติดที่**การอ่านรูป** (กลับมาอ่านหัวข้อ 3 ของบทนี้) · แยกให้ออกแล้วจะแก้ถูกที่

สามบรรทัดสุดท้ายก่อนเข้าห้องสอบ

① นับช่วงในรูปก่อนลงมือ แล้วขาออกต้องมีค่าครบเท่านั้นช่วง
② มีเกตกี่ตัวในวงจร ก็ต้องมีแถวในรูปเพิ่มเท่านั้นแถว (รวมสายกลาง)
③ เจอ NAND หรือ NOR ให้เขียนคำว่า "พลิก" ไว้ข้างแถวก่อนเริ่มวาด

ภาคผนวก ก · เฉลย Assignment 1 และ 2 ทุกข้อ

ภาคผนวกนี้ทำโจทย์ **Assignment 1 ทั้ง 15 ข้อ** และ **Assignment 2 ทั้ง 10 ข้อ** ให้ดูทุกข้อย่อย ไม่ข้ามข้อไหนเลย ทุกข้อเขียนด้วยรูปแบบเดียวกัน 5 ส่วน อ่านเรียงลงมาได้เลย

อ่านยังไง

① **โจทย์** — ยกข้อความอังกฤษเดิมมาให้ครบ แล้วแปลไทยข้างล่าง เพราะข้อสอบจริงใช้คำอังกฤษชุดนี้

② **วิธีไหน บทไหน** — บอกชื่อวิธี กับบทของเล่มนี้ที่สอนวิธีนั้น (ถ้ายังไม่เข้าใจให้ย้อนไปอ่านบทนั้นก่อน ภาคผนวกนี้ไม่สอนซ้ำ)

③ **ทำให้ดูทุกบรรทัด** — ไม่มีขั้นไหนคิดในใจ ตัวทด ตัวยืม การกลับบิต เขียนออกมาหมด

④ **คำตอบในกรอบเขียว** — ลอกลงกระดาษคำตอบได้เลย

⑤ **จุดที่มักพลาดข้อนี้** — บรรทัดเดียว อ่านก่อนเข้าห้องสอบ

บทไหนสอนอะไร

บทที่ 1 · ตัวเลขในเครื่อง — น้ำหนักประจำหลัก แปลงฐาน เลขมีจุด บวกลบคูณหารฐานสอง คอมพลีเมนต์ เลขมีเครื่องหมายและพิสัย ทศนิยมลอยตัว

บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด — BCD, Gray code, พาริตี, CRC, Hamming code

1 Assignment 1 — ฐานเลข คอมพลีเมนต์ และเลขคณิตฐานสอง (15 ข้อ)

ข้อ 1 · เขียนเลขฐานสิบในรูปน้ำหนักประจำหลัก

โจทย์ Assignment #1 · Q1

> **Describe the following decimal numbers using the positional numbering system:**
> a. 10724_{10} b. 1638_{10} c. 297_{10}

ไทย: เขียนเลขฐานสิบสามจำนวนนี้ออกมาในรูป “ผลรวมของ ตัวเลขแต่ละหลัก × น้ำหนักประจำหลัก” — คำว่า `describe ... using the positional numbering system` ในวิชานี้แปลว่า **กางหลักออกมาให้เห็น** ไม่ได้สั่งให้แปลงฐาน

วิธีที่ใช้: น้ำหนักประจำหลัก (`positional notation`) น้ำหนักของหลักที่ k คือ ฐาน^{k} โดยนับหลักหน่วยเป็น k = 0 · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

เลขหนึ่งตัวไม่ได้มีค่าที่ตัวมันเอง มันมีค่าตาม **ตำแหน่งที่มันยืน** เลข 7 ในข้อ a ไม่ได้แปลว่าเจ็ด แต่แปลว่าเจ็ดร้อย เพราะมันยืนอยู่หลักที่ 2 นับจากขวา



a. 10724_{10}

นับหลักจากขวามาซ้าย เริ่มที่ k = 0 ได้ทั้งหมด 5 หลัก คือ k = 4, 3, 2, 1, 0 แล้วคูณทีละหลัก ไม่ข้ามหลักไหน

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
หลักหมื่น	**1**	$10^4 = 10{,}000$	$1 \times 10{,}000 =$ **10,000**
หลักพัน	**0**	$10^3 = 1{,}000$	$0 \times 1{,}000 =$ **0**
หลักร้อย	**7**	$10^2 = 100$	$7 \times 100 =$ **700**
หลักสิบ	**2**	$10^1 = 10$	$2 \times 10 =$ **20**
หลักหน่วย	**4**	$10^0 = 1$	$4 \times 1 =$ **4**
รวมทุกหลัก			**10,724**

เขียนเป็นบรรทัดเดียวตามที่โจทย์ขอ

$$10724_{10} = 1\times10^4 + 0\times10^3 + 7\times10^2 + 2\times10^1 + 4\times10^0$$

เช็กกลับ: 10000 + 0 + 700 + 20 + 4 = 10724 ตรงกับโจทย์

b. 1638_{10}

เลขนี้มี 4 หลัก หลักซ้ายสุดจึงเป็น k = 3

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
หลักพัน	**1**	$10^3 = 1{,}000$	$1 \times 1{,}000 =$ **1,000**

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
หลักร้อย	**6**	$10^2 = 100$	$6 \times 100 =$ **600**
หลักสิบ	**3**	$10^1 = 10$	$3 \times 10 =$ **30**
หลักหน่วย	**8**	$10^0 = 1$	$8 \times 1 =$ **8**
รวมทุกหลัก			**1,638**

$$1638_{10} = 1\times10^3 + 6\times10^2 + 3\times10^1 + 8\times10^0$$

เช็กกลับ: 1000 + 600 + 30 + 8 = 1638 ตรงกับโจทย์

c. 297_{10}

เลขนี้มี 3 หลัก หลักซ้ายสุดจึงเป็น k = 2

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
หลักร้อย	**2**	$10^2 = 100$	$2 \times 100 =$ **200**
หลักสิบ	**9**	$10^1 = 10$	$9 \times 10 =$ **90**
หลักหน่วย	**7**	$10^0 = 1$	$7 \times 1 =$ **7**
รวมทุกหลัก			**297**

$$297_{10} = 2\times10^2 + 9\times10^1 + 7\times10^0$$

เช็กกลับ: 200 + 90 + 7 = 297 ตรงกับโจทย์

คำตอบ

a. $1\times10^4 + 0\times10^3 + 7\times10^2 + 2\times10^1 + 4\times10^0 = 10724$

b. $1\times10^3 + 6\times10^2 + 3\times10^1 + 8\times10^0 = 1638$

c. $2\times10^2 + 9\times10^1 + 7\times10^0 = 297$

จุดที่มักพลาดข้อนี้

ตัดพจน์ของเลขศูนย์ทิ้ง — ข้อ a ต้องมี **0×10^3** อยู่ด้วย ถ้าหายไปจำนวนหลักจะเลื่อน คนตรวจถือว่าเขียนน้ำหนักผิด

ข้อ 2 · กางหลักเลขฐานสอง แล้วหาค่าฐานสิบ

โจทย์ Assignment #1 · Q2

> **Describe the following binary numbers using the positional numbering system and determine their equivalent decimal values:**
> a. 11101_2 b. 10010_2 c. 01001_2

ไทย: กางหลักของเลขฐานสองสามจำนวนนี้ แล้วบอกว่ามีค่าเท่ากับเลขฐานสิบเท่าไร — ข้อนี้มีสองคำสั่งในข้อเดียว ต้องตอบทั้งรูปกางหลัก และ ตัวเลขฐานสิบ

วิธีที่ใช้: น้ำหนักประจำหลักเหมือนข้อ 1 เป๊ะ แค่เปลี่ยนฐานจาก 10 เป็น 2 น้ำหนักจึงเป็น 1, 2, 4, 8, 16 · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ฐานสองมีตัวเลขให้ใช้แค่ 0 กับ 1 เพราะฉะนั้นการคูณในแต่ละหลักมีแค่สองแบบ คือ **เอาน้ำหนักมาทั้งก้อน** (เมื่อบิตเป็น 1) หรือ **ไม่เอาเลย** (เมื่อบิตเป็น 0) — บวกเฉพาะหลักที่เป็น 1 ก็พอ

① วางหลัก ② ใส่น้ำหนัก 2 ยกกำลัง k ③ คูณแล้วบวก

1	1	1	0	1
2^4 =16	2^3 =8	2^2 =4	2^1 =2	2^0 =1
16	8	4	0	1

รวม = 29

ข้อ a — บวกเฉพาะช่องที่บิตเป็น 1 คือ 16 + 8 + 4 + 1

a. 11101_2

มี 5 บิต หลักซ้ายสุดคือ k = 4 น้ำหนัก $2^4 = 16$

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 4	**1**	$2^4 = 16$	$1 \times 16 =$ **16**
k = 3	**1**	$2^3 = 8$	$1 \times 8 =$ **8**
k = 2	**1**	$2^2 = 4$	$1 \times 4 =$ **4**
k = 1	**0**	$2^1 = 2$	$0 \times 2 =$ **0**
k = 0	**1**	$2^0 = 1$	$1 \times 1 =$ **1**
รวมทุกหลัก			**29**

$$11101_2 = 1\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$

$$= 16 + 8 + 4 + 0 + 1 = \mathbf{29}_{10}$$

เช็กกลับ: 29 มีอยู่ในช่วง 16 ถึง 31 ซึ่งเป็นช่วงของเลข 5 บิตที่ขึ้นต้นด้วย 1 พอดี

b. 10010_2

บิตที่เป็น 1 มีสองตัว อยู่ที่ k = 4 และ k = 1

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 4	**1**	$2^4 = 16$	1 × 16 = **16**
k = 3	**0**	$2^3 = 8$	0 × 8 = **0**
k = 2	**0**	$2^2 = 4$	0 × 4 = **0**
k = 1	**1**	$2^1 = 2$	1 × 2 = **2**
k = 0	**0**	$2^0 = 1$	0 × 1 = **0**
รวมทุกหลัก			**18**

$$10010_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$$

$$= 16 + 0 + 0 + 2 + 0 = \mathbf{18}_{10}$$

เช็กกลับ: บิตขวาสุดเป็น 0 แปลว่าเลขนี้ต้องเป็นจำนวนคู่ — 18 เป็นจำนวนคู่ ถูกต้อง

c. 01001_2

บิตซ้ายสุดเป็น 0 มันไม่ได้เพิ่มค่าอะไร แต่ **ห้ามตัดทิ้งตอนเขียนกางหลัก** เพราะโจทย์ให้มา 5 บิต

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 4	**0**	$2^4 = 16$	0 × 16 = **0**
k = 3	**1**	$2^3 = 8$	1 × 8 = **8**
k = 2	**0**	$2^2 = 4$	0 × 4 = **0**
k = 1	**0**	$2^1 = 2$	0 × 2 = **0**
k = 0	**1**	$2^0 = 1$	1 × 1 = **1**
รวมทุกหลัก			**9**

$$01001_2 = 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$$

$$= 0 + 8 + 0 + 0 + 1 = \mathbf{9}_{10}$$

เช็กกลับ: 8 + 1 = 9 และ 9 น้อยกว่า 16 สมกับที่บิต 2^4 เป็น 0

คำตอบ

a. $1\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = \mathbf{29_{10}}$

b. $1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = \mathbf{18_{10}}$

c. $0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = \mathbf{9_{10}}$

จุดที่มักพลาดข้อนี้

ไล่น้ำหนักผิดทาง — น้ำหนักมากอยู่ **ซ้าย** น้อยอยู่ **ขวา** ถ้าไล่ 1, 2, 4, 8, 16 จากซ้ายไปขวา ข้อ a จะได้ 23 แทนที่จะเป็น 29

ข้อ 3 · กางหลักเลขฐานแปด แล้วหาค่าฐานสิบ

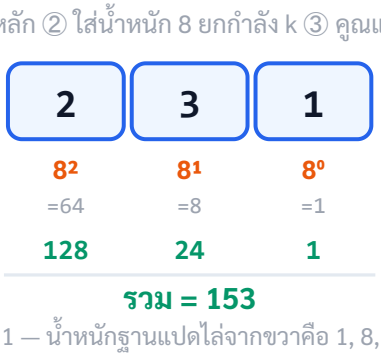
โจทย์ Assignment #1 · Q3

> **Describe the following octal numbers using the positional numbering system and determine their equivalent decimal values:**
> 1. 231_8 2. 64_8 3. 157_8

ไทย: กางหลักเลขฐานแปดสามจำนวนนี้ แล้วบอกค่าฐานสิบ (โจทย์ข้อนี้ใช้เลข 1. 2. 3. แทน a. b. c. แต่ความหมายเหมือนกัน)

วิธีที่ใช้: น้ำหนักประจำหลักฐาน 8 คือ 1, 8, 64, 512 ... · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ฐานแปดใช้ตัวเลข 0 ถึง 7 เท่านั้น ไม่มีเลข 8 ไม่มีเลข 9 ถ้าเห็นเลข 8 หรือ 9 ในโจทย์ที่บอกว่าเป็นฐานแปด แปลว่าโจทย์พิมพ์ผิด



1. 231_8

มี 3 หลัก หลักซ้ายสุดคือ k = 2 น้ำหนัก $8^2 = 64$

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**2**	$8^2 = 64$	2 × 64 = **128**

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 1	**3**	$8^1 = 8$	3 × 8 = **24**
k = 0	**1**	$8^0 = 1$	1 × 1 = **1**
รวมทุกหลัก			**153**

$$231_8 = 2\times8^2 + 3\times8^1 + 1\times8^0$$

$$= 128 + 24 + 1 = \mathbf{153}_{10}$$

เช็กกลับ: 153 หารด้วย 8 ได้ 19 เศษ 1 (ตรงกับหลักหน่วย 1) แล้ว 19 หารด้วย 8 ได้ 2 เศษ 3 (ตรงกับหลักกลาง 3 และหลักบน 2)

2. 64_8

มีแค่ 2 หลัก หลักซ้ายคือ k = 1 น้ำหนัก 8

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 1	**6**	$8^1 = 8$	6 × 8 = **48**
k = 0	**4**	$8^0 = 1$	4 × 1 = **4**
รวมทุกหลัก			**52**

$$64_8 = 6\times8^1 + 4\times8^0$$

$$= 48 + 4 = \mathbf{52}_{10}$$

เช็กกลับ: 52 หารด้วย 8 ได้ 6 เศษ 4 ตรงกับตัวเลขในโจทย์พอดี

ระวัง เลข “64” ตัวนี้ ไม่ใช่ หกสิบสี่ เพราะมันเป็นฐานแปด ค่าจริงคือ 52

3. 157_8

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**1**	$8^2 = 64$	1 × 64 = **64**
k = 1	**5**	$8^1 = 8$	5 × 8 = **40**
k = 0	**7**	$8^0 = 1$	7 × 1 = **7**
รวมทุกหลัก			**111**

$$157_8 = 1\times8^2 + 5\times8^1 + 7\times8^0$$

$$= 64 + 40 + 7 = \mathbf{111}_{10}$$

เช็กกลับ: 111 หารด้วย 8 ได้ 13 เศษ 7 แล้ว 13 หารด้วย 8 ได้ 1 เศษ 5 อ่านเศษจากล่างขึ้นบนได้ 157 ตรงกับโจทย์

คำตอบ

1. $2\times8^2 + 3\times8^1 + 1\times8^0 = \mathbf{153_{10}}$

2. $6\times8^1 + 4\times8^0 = \mathbf{52_{10}}$

3. $1\times8^2 + 5\times8^1 + 7\times8^0 = \mathbf{111_{10}}$

จุดที่มักพลาดข้อนี้

อ่านเลขฐานแปดเป็นเลขฐานสิบโดยอัตโนมัติ — 64_8 คือ 52 ไม่ใช่ 64 ต้องมองตัวห้อยของฐานทุกครั้งก่อนคิด

ข้อ 4 · กางหลักเลขฐานสิบหก แล้วหาค่าฐานสิบ

โจทย์ Assignment #1 · Q4

> **Describe the following hexadecimal numbers using the positional numbering system and determine their equivalent decimal values:**
> 1. $C98_{16}$ 2. $BD2_{16}$ 3. $70E_{16}$

ไทย: กางหลักเลขฐานสิบหกสามจำนวนนี้ แล้วบอกค่าฐานสิบ

วิธีที่ใช้: แปลงตัวอักษรเป็นตัวเลขก่อน (A=10 ... F=15) แล้วใช้น้ำหนัก 1, 16, 256 · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ฐานสิบหกต้องการตัวเลข 16 ตัว แต่เรามีสัญลักษณ์ตัวเลขแค่ 0–9 จึงหยิบตัวอักษรมาใช้ต่อ

สัญลักษณ์	A	B	C	D	E	F
ค่า	10	11	12	13	14	15

① วางหลัก ② ใส่น้ำหนัก 16 ยกกำลัง k ③ คูณแล้วบวก

C	9	8
16^2	16^1	16^0
=256	=16	=1
3072	144	8

รวม = 3224

ข้อ 1 — C มีค่า 12 คูณน้ำหนัก 256 ได้ 3072

1. $C98_{16}$

ขั้นแรกเปลี่ยน C เป็น 12 ก่อน แล้วค่อยคูณน้ำหนัก

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**C**	$16^2 = 256$	C × 256 = **3072**
k = 1	**9**	$16^1 = 16$	9 × 16 = **144**
k = 0	**8**	$16^0 = 1$	8 × 1 = **8**
รวมทุกหลัก			**3224**

$$C98_{16} = 12\times16^2 + 9\times16^1 + 8\times16^0$$

$$= 3072 + 144 + 8 = \mathbf{3224}_{10}$$

เช็กกลับ: 3224 หารด้วย 16 ได้ 201 เศษ 8 (ตรงกับหลักหน่วย 8) · 201 หารด้วย 16 ได้ 12 เศษ 9 (ตรงกับ 9) · เหลือ 12 คือ C

2. $BD2_{16}$

B คือ 11 และ D คือ 13

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**B**	$16^2 = 256$	B × 256 = **2816**
k = 1	**D**	$16^1 = 16$	D × 16 = **208**
k = 0	**2**	$16^0 = 1$	2 × 1 = **2**
รวมทุกหลัก			**3026**

$$BD2_{16} = 11\times16^2 + 13\times16^1 + 2\times16^0$$

$$= 2816 + 208 + 2 = \mathbf{3026}_{10}$$

เช็กกลับ: 3026 หารด้วย 16 ได้ 189 เศษ 2 · 189 หารด้วย 16 ได้ 11 เศษ 13 · เหลือ 11 คือ B อ่านขึ้นมาได้ B D 2 ตรงกับโจทย์

3. $70E_{16}$

E คือ 14 ส่วนหลักกลางเป็น 0 ต้องเขียนพจน์ไว้ด้วย

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**7**	$16^2 = 256$	7 × 256 = **1792**
k = 1	**0**	$16^1 = 16$	0 × 16 = **0**
k = 0	**E**	$16^0 = 1$	E × 1 = **14**

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
รวมทุกหลัก			**1806**

$$70E_{16} = 7\times16^2 + 0\times16^1 + 14\times16^0$$

$$= 1792 + 0 + 14 = \mathbf{1806}_{10}$$

เช็กกลับ: 1806 หารด้วย 16 ได้ 112 เศษ 14 (คือ E) · 112 หารด้วย 16 ได้ 7 เศษ 0 · เหลือ 7 อ่านขึ้นมาได้ 7 0 E ตรงกับโจทย์

คำตอบ

1. $12\times16^2 + 9\times16^1 + 8\times16^0 = \mathbf{3224_{10}}$

2. $11\times16^2 + 13\times16^1 + 2\times16^0 = \mathbf{3026_{10}}$

3. $7\times16^2 + 0\times16^1 + 14\times16^0 = \mathbf{1806_{10}}$

จุดที่มักพลาดข้อนี้

ลืมแปลงตัวอักษรก่อนคูณ — เขียน “C×256” แล้วค้างอยู่แค่นั้น ต้องเปลี่ยนเป็น 12 ก่อนเสมอ และ C คือ 12 ไม่ใช่ 11 (A เริ่มที่ 10)

ข้อ 5 · เลขที่มีจุด — กางหลักทั้งซ้ายและขวาของจุด

โจทย์ Assignment #1 · Q5

> **Describe the following numbers using the positional numbering system and determine their equivalent decimal values:**
> 1. 1010.01_2 2. 74.23_8 3. 910.85_{10} 4. $C7.A1_{16}$

ไทย: ข้อนี้รวมสี่ฐานไว้ในข้อเดียว แต่ละจำนวนมีจุด ให้กางหลักทั้งฝั่งซ้ายและฝั่งขวาของจุด แล้วบอกค่าฐานสิบ

วิธีที่ใช้: น้ำหนักประจำหลักที่ขยายไปทางขวาของจุดด้วยเลขชี้กำลัง **ติดลบ** (k = −1, −2, …) · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

กฎเดิมไม่เปลี่ยนเลย หลักหน่วยคือ k = 0 เดินไปทางซ้ายเลขชี้กำลังเพิ่มทีละหนึ่ง เดินไปทางขวาเลขชี้กำลัง **ลดทีละหนึ่ง** พอเลยจุดไปก็กลายเป็นติดลบ

เลขชี้กำลังติดลบไม่ได้แปลว่าค่าติดลบ 2^{-1} แปลว่า “หนึ่งส่วนสอง” = 0.5 ซึ่งยังเป็นบวกอยู่



1. 1010.01_2

ฝั่งซ้ายของจุดมี 4 บิต (k = 3, 2, 1, 0) ฝั่งขวามี 2 บิต (k = −1, −2)

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 3	**1**	$2^3 = 8$	1 × 8 = **8**
k = 2	**0**	$2^2 = 4$	0 × 4 = **0**
k = 1	**1**	$2^1 = 2$	1 × 2 = **2**
k = 0	**0**	$2^0 = 1$	0 × 1 = **0**
k = -1	**0**	$2^{-1} = 1/2 = 0.5$	0 × 0.5 = **0**
k = -2	**1**	$2^{-2} = 1/4 = 0.25$	1 × 0.25 = **0.25**
รวมทุกหลัก			**10.25**

$$1010.01_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}$$

$$= 8 + 0 + 2 + 0 + 0 + 0.25 = \mathbf{10.25}_{10}$$

เช็กกลับ: ส่วนจำนวนเต็ม $1010_2 = 8 + 2 = 10$ และส่วนเศษ $.01_2$ = หนึ่งส่วนสี่ = 0.25 รวมได้ 10.25

2. 74.23_8

ฐานแปด น้ำหนักฝั่งขวาของจุดคือ 1/8 = 0.125 และ 1/64 = 0.015625

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 1	**7**	$8^1 = 8$	7 × 8 = **56**
k = 0	**4**	$8^0 = 1$	4 × 1 = **4**
k = -1	**2**	$8^{-1} = 1/8 = 0.125$	2 × 0.125 = **0.25**
k = -2	**3**	$8^{-2} = 1/64 = 0.015625$	3 × 0.015625 = **0.046875**
รวมทุกหลัก			**60.296875**

$$74.23_8 = 7\times8^1 + 4\times8^0 + 2\times8^{-1} + 3\times8^{-2}$$

$$= 56 + 4 + 0.25 + 0.046875 = \mathbf{60.296875}_{10}$$

เช็กกลับ: ส่วนเศษอย่างเดียวคือ 0.25 + 0.046875 = 0.296875 ซึ่งน้อยกว่า 1 พอดี ถ้าบวกแล้วเกิน 1 แปลว่าคำนวณผิด

3. 910.85_{10}

ข้อนี้เป็นฐานสิบอยู่แล้ว ค่าฐานสิบจึงเท่าเดิม แต่โจทย์ยังสั่งให้ **กางหลัก** อยู่ ต้องเขียนออกมาให้ครบ

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 2	**9**	$10^2 = 100$	9 × 100 = **900**
k = 1	**1**	$10^1 = 10$	1 × 10 = **10**
k = 0	**0**	$10^0 = 1$	0 × 1 = **0**
k = -1	**8**	$10^{-1} = 1/10 = 0.1$	8 × 0.1 = **0.8**
k = -2	**5**	$10^{-2} = 1/100 = 0.01$	5 × 0.01 = **0.05**
รวมทุกหลัก			**910.85**

$$910.85_{10} = 9\times10^{2} + 1\times10^{1} + 0\times10^{0} + 8\times10^{-1} + 5\times10^{-2}$$

$$= 900 + 10 + 0 + 0.8 + 0.05 = \mathbf{910.85}_{10}$$

เช็กกลับ: ได้ตัวเลขเดิมกลับมา ถูกต้อง เพราะการกางหลักไม่ได้เปลี่ยนค่า แค่เขียนให้เห็นที่มา

4. $C7.A1_{16}$

แปลงตัวอักษรก่อน C = 12 และ A = 10 · น้ำหนักฝั่งขวาคือ 1/16 = 0.0625 และ 1/256 = 0.00390625

หลัก	ตัวเลข	น้ำหนักประจำหลัก	คูณกันได้
k = 1	**C**	$16^1 = 16$	C × 16 = **192**
k = 0	**7**	$16^0 = 1$	7 × 1 = **7**
k = -1	**A**	$16^{-1} = 1/16 = 0.0625$	A × 0.0625 = **0.625**
k = -2	**1**	$16^{-2} = 1/256 = 0.00390625$	1 × 0.00390625 = **0.00390625**
รวมทุกหลัก			**199.62890625**

$$C7.A1_{16} = 12\times16^{1} + 7\times16^{0} + 10\times16^{-1} + 1\times16^{-2}$$

$$= 192 + 7 + 0.625 + 0.00390625 = \mathbf{199.62890625}_{10}$$

เช็กกลับ: ส่วนจำนวนเต็ม $C7_{16} = 192 + 7 = 199$ และส่วนเศษ 0.625 + 0.00390625 = 0.62890625 ซึ่งน้อยกว่า 1 รวมได้ 199.62890625

คำตอบ

1. $1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2} = \mathbf{10.25_{10}}$

2. $7\times8^1 + 4\times8^0 + 2\times8^{-1} + 3\times8^{-2} = \mathbf{60.296875_{10}}$

3. $9\times10^2 + 1\times10^1 + 0\times10^0 + 8\times10^{-1} + 5\times10^{-2} = \mathbf{910.85_{10}}$

4. $12\times16^1 + 7\times16^0 + 10\times16^{-1} + 1\times16^{-2} = \mathbf{199.62890625_{10}}$

จุดที่มักพลาดข้อนี้

เริ่มนับหลังจุดเป็น k = 0 — ผิด หลังจุดต้องเริ่มที่ **k = −1** ทันที ถ้าเริ่มที่ 0 ข้อ 1 จะได้ 0.5 แทนที่จะเป็น 0.25

ข้อ 6 · แปลงเลขฐานสิบไปเป็นฐานที่กำหนด

โจทย์ Assignment #1 · Q6

> **Convert the following decimal numbers to the specified base:**
> a. 6781_{10} ⇒ base 8 b. 0.2890625_{10} ⇒ base 16

ไทย: ข้อ a แปลงจำนวนเต็มไปฐานแปด · ข้อ b แปลง เศษ (เลขหลังจุด) ไปฐานสิบหก — สองข้อนี้ใช้วิธี **คนละวิธี** อย่าสลับกัน

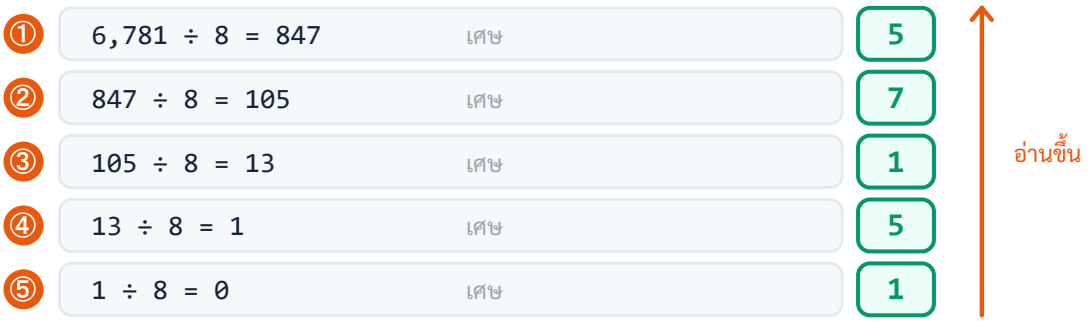
วิธีที่ใช้: จำนวนเต็มใช้ **หารเก็บเศษ** แล้วอ่านเศษจากล่างขึ้นบน · เศษส่วนใช้ **คูณเก็บส่วนจำนวนเต็ม** แล้วอ่านจากบนลงล่าง · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ทำไมจำนวนเต็มต้องหาร แต่เศษต้องคูณ — เพราะการหารด้วยฐานคือการ “ปอก” หลักขวาสุดออกมาทีละหลัก ส่วนการคูณด้วยฐานคือการ “ดัน” หลักซ้ายสุดของเศษให้ข้ามจุดขึ้นมาทีละหลัก

a. 6781_{10} ⇒ ฐาน 8

หารด้วย 8 ไปเรื่อย ๆ จดเศษไว้ทุกรอบ หยุดเมื่อผลหารเป็น 0

① หารด้วย 8 ② จดเศษไว้ ③ เอาผลหารไปหารต่อ จนผลหารเป็น 0



อ่านเศษจากล่างขึ้นบน = 15175_8

รอบที่	เอาอะไรหาร 8	ผลหาร	เศษ (คือหลักของฐานแปด)
1	6,781	847	5
2	847	105	7
3	105	13	1
4	13	1	5
5	1	0	1

อ่านเศษ **จากรอบสุดท้ายขึ้นมารอบแรก** ได้ $1 \cdot 5 \cdot 1 \cdot 7 \cdot 5$

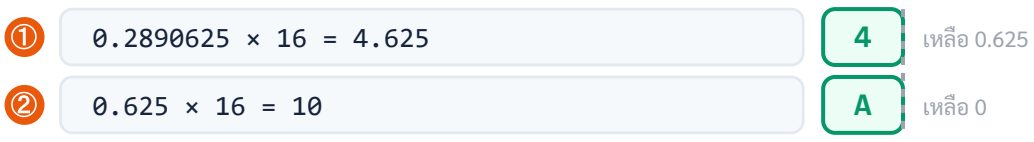
$$6781_{10} = 15175_8$$

เช็กกลับด้วยน้ำหนักประจำหลัก $1\times4096 + 5\times512 + 1\times64 + 7\times8 + 5\times1 = 4096 + 2560 + 64 + 56 + 5 = \mathbf{6781}$ ตรงกับโจทย์

b. 0.2890625_{10} ⇒ ฐาน 16

คราวนี้ไม่มีส่วนจำนวนเต็มเลย มีแต่เศษ จึงใช้วิธีคูณ เอาเศษคูณ 16 แล้วดู “ส่วนจำนวนเต็ม” ที่โผล่ขึ้นมา นั่นคือหลักแรกหลังจุด

① คูณเศษด้วย 16 ② หลักที่ได้คือส่วนจำนวนเต็ม ③ เอาเศษที่เหลือไปคูณต่อ



อ่านหลักจากบนลงล่าง

รอบที่	เอาอะไรคูณ 16	ได้เท่าไร	หลักที่ได้ (ส่วนจำนวนเต็ม)	เศษที่เหลือไปทำต่อ
1	0.2890625	4.625	4	0.625
2	0.625	10	A	0

รอบที่ 2 เศษที่เหลือกลายเป็น 0 พอดี แปลว่า **จบ** ไม่ต้องทำต่อ อ่านหลักจากบนลงล่างได้ 4 แล้ว 10 ซึ่ง 10 ในฐานสิบหกเขียนเป็น **A**

$$0.2890625_{10} = \mathbf{0.4A}_{16}$$

เช็กกลับ $4\times16^{-1} + 10\times16^{-2} = 0.25 + 0.0390625 = \mathbf{0.2890625}$ ตรงกับโจทย์

คำตอบ

a. $6781_{10} = \mathbf{15175_8}$

b. $0.2890625_{10} = \mathbf{0.4A_{16}}$

จุดที่มักพลาดข้อนี้

อ่านเศษผิดทาง — ของข้อ a ต้องอ่าน **จากล่างขึ้นบน** (ถ้าอ่านบนลงล่างจะได้ 57151 ซึ่งผิด) ส่วนข้อ b ต้องอ่าน **จากบนลงล่าง** สองข้อนี้ทิศตรงข้ามกัน

ข้อ 7 · แปลงระหว่างฐานสองกับฐานสิบหกด้วยการจับกลุ่ม

โจทย์ Assignment #1 · Q7

> **Convert the following numbers to the specified base:**
> a. $1000010111110110_2 \Rightarrow$ base 16 b. $D5C9_{16} \Rightarrow$ base 2

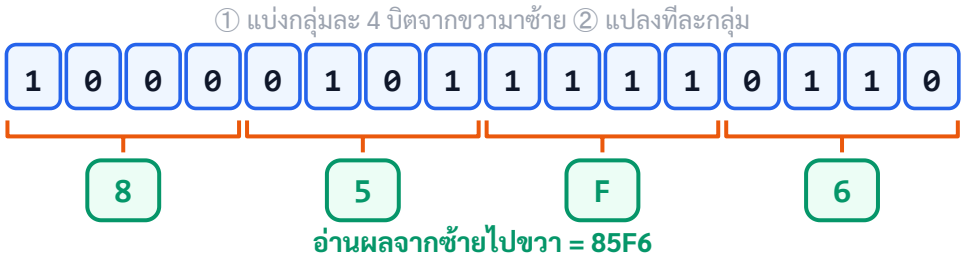
ไทย: a แปลงเลขฐานสอง 16 บิตเป็นฐานสิบหก · b แปลงฐานสิบหกกลับเป็นฐานสอง

วิธีที่ใช้: จับกลุ่มทีละ 4 บิต (เพราะ $2^4 = 16$ พอดี) ไม่ต้องแวะผ่านฐานสิบเลย · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

เลขฐานสิบหกหนึ่งหลักเก็บค่าได้ 16 แบบ และบิต 4 ตัวก็เก็บค่าได้ 16 แบบเท่ากันพอดี เพราะฉะนั้น **4 บิต = 1 หลักฐานสิบหก** เสมอ ไม่มีข้อยกเว้น

a. $1000010111110110_2 \Rightarrow$ ฐาน 16

นับบิตก่อน มี 16 บิต หาร 4 ลงตัวพอดี ได้ 4 กลุ่ม ไม่ต้องเติมศูนย์หน้า



แปลงทีละกลุ่ม แต่ละกลุ่มใช้น้ำหนัก 8 4 2 1

กลุ่มบิต	คิดเป็นเลขฐานสิบ	เขียนเป็นฐานสิบหก
`1000`	1×8 + 0×4 + 0×2 + 0×1 = 8	`8`
`0101`	0×8 + 1×4 + 0×2 + 1×1 = 5	`5`
`1111`	1×8 + 1×4 + 1×2 + 1×1 = 15	`F`
`0110`	0×8 + 1×4 + 1×2 + 0×1 = 6	`6`

$$1000010111110110_2 = \mathbf{85F6}_{16}$$

เช็กกลับ: แตก 85F6 กลับเป็นบิต 8=1000, 5=0101, F=1111, 6=0110 ต่อกันได้ 1000 0101 1111 0110 ตรงกับโจทย์เดิมทุกบิต

b. $D5C9_{16}$ ⇒ ฐาน 2

ทิศกลับกัน หลักฐานสิบหกหนึ่งหลักแตกเป็น 4 บิต **ต้องเป็น 4 บิตเสมอ** แม้ค่าจะน้อย ก็ต้องเติมศูนย์ข้างหน้าให้ครบ 4

ตัวอักษร/เลข	มีค่าเท่าไร	เขียนเป็น 4 บิต
D	13	`1101`
5	5	`0101`
C	12	`1100`
9	9	`1001`

เอามาต่อกันตามลำดับเดิม D → 5 → C → 9

`1` `1` `0` `1` `0` `1` `0` `1` `1` `1` `0` `0` `1` `0` `0` `1`

ต่อกันได้ 16 บิต เรียงตามลำดับตัวอักษรในโจทย์

$$D5C9_{16} = \mathbf{1101010111001001}_2$$

เช็กกลับ: จับกลุ่มละ 4 จากขวา ได้ 1101 | 0101 | 1100 | 1001 = 13, 5, 12, 9 = D, 5, C, 9 ตรงกับโจทย์

คำตอบ

a. $1000010111110110_2 = \mathbf{85F6}_{16}$

b. $D5C9_{16}$ = **1101010111001001_2**

จุดที่มักพลาดข้อนี้

จับกลุ่มจากซ้ายแทนที่จะจับจากขวา — ถ้าจำนวนบิตไม่ใช่ตัวคูณของ 4 (เช่น 14 บิต) การจับจากซ้ายจะทำให้ทุกกลุ่มเลื่อน ต้อง **เริ่มนับจากขวาเสมอ** แล้วเติมศูนย์ที่หัวถ้าขาด

ข้อ 8 · หา 1's complement

โจทย์ Assignment #1 · Q8

> **Determine the 1's complement of the following binary numbers:**
> a. 110 b. 11010111

ไทย: หา 1's complement (คอมพลีเมนต์ที่หนึ่ง) ของเลขฐานสองสองจำนวนนี้

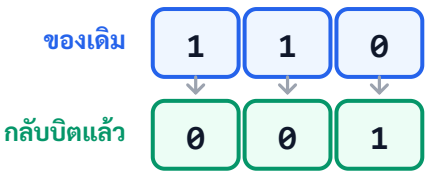
วิธีที่ใช้: กลับทุกบิต 0 เป็น 1 และ 1 เป็น 0 — ไม่ทำอะไรอย่างอื่นเลย และจำนวนบิตต้องเท่าเดิม · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

คำว่าคอมพลีเมนต์ที่หนึ่งฟังดูน่ากลัว แต่สิ่งที่ต้องทำมีแค่อย่างเดียวคือ **พลิกสวิตช์ทุกตัว** ที่เคยปิดให้เปิด ที่เคยเปิดให้ปิด

a. 110

มี 3 บิต ไล่พลิกทีละตัวจากซ้ายไปขวา ไม่ข้ามตัวไหน

บิตที่ (นับจากซ้าย)	1	2	3
ของเดิม	1	1	0
กลับบิต (0↔1)	0	0	1



บิตที่ 1: 1→0 · บิตที่ 2: 1→0 · บิตที่ 3: 0→1

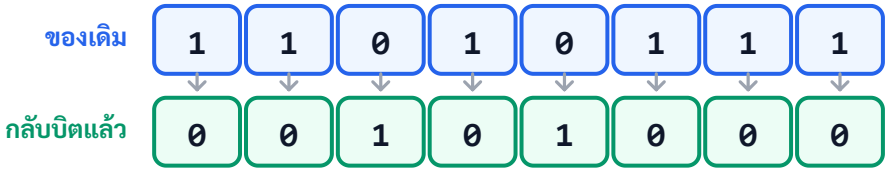
1's complement ของ 110 = **001**

เช็กกลับ: เอาของเดิมบวกกับผลลัพธ์ 110 + 001 = 111 ซึ่งเป็นเลขที่มี 1 ครบทุกบิต — นี่คือลักษณะประจำตัวของคู่ 1’s complement เสมอ

b. 11010111

มี 8 บิต ผลลัพธ์ก็ต้องมี 8 บิต ห้ามตัดศูนย์หน้าทิ้ง

บิตที่ (นับจากซ้าย)	1	2	3	4	5	6	7	8
ของเดิม	1	1	0	1	0	1	1	1
กลับบิต (0↔1)	0	0	1	0	1	0	0	0



พลิกครบทั้ง 8 บิต ศูนย์สองตัวหน้าผลลัพธ์ห้ามตัดทิ้ง

1’s complement ของ 11010111 = **00101000**

เช็กกลับ: 11010111 + 00101000 = 11111111 ได้ 1 ครบทั้ง 8 บิต ถูกต้อง

คำตอบ

a. `001`

b. `00101000`

จุดที่มักพลาดข้อนี้

ตัดเลขศูนย์ข้างหน้าผลลัพธ์ทิ้ง — ข้อ b ต้องตอบ **00101000** ครบ 8 บิต ถ้าตอบ 101000 ถือว่าผิด เพราะจำนวนบิตเปลี่ยนไปแล้ว

ข้อ 9 · หา 2’s complement

โจทย์ Assignment #1 · Q9

> **Determine the 2’s complement of the following binary numbers:**
> a. 111 b. 1101

ไทย: หา 2’s complement (คอมพลีเมนต์ที่สอง) ของเลขฐานสองสองจำนวนนี้

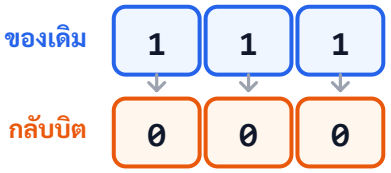
วิธีที่ใช้: ① กลับทุกบิตก่อน (ได้ 1’s complement) ② แล้วบวก 1 เข้าไปที่บิตขวาสุด · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

2’s complement คือ 1’s complement ที่บวกหนึ่งเข้าไปอีกทีเดียวเท่านั้น ทำสองขั้นตามลำดับ ห้ามสลับ



a. 111

ขั้นที่ 1 กลับบิต — 1 ทั้งสามตัวกลายเป็น 0 ทั้งสามตัว



ได้ 1’s complement = 000

ขั้นที่ 2 บวกหนึ่ง — บวกที่บิตขวาสุด

บวกหนึ่งแบบตั้งหลัก (ตัวทดสีแดง)

```
  000
+   1
-----
  001
```

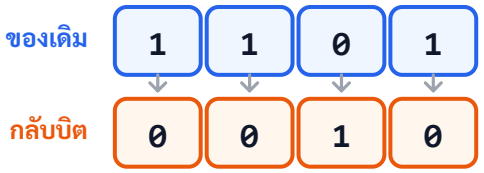
บิตขวาสุด 0 + 1 = 1 ไม่มีตัวทด บิตที่เหลือคงเดิมเป็น 0

2’s complement ของ 111 = **001**

เช็กกลับ: ของเดิมบวกผลลัพธ์ 111 + 001 = 1000 — ตัวทดที่ล้นออกไปหนึ่งตัว เหลือ 3 บิตล่างเป็น 000 พอดี นี่คือคุณสมบัติของ 2’s complement (บวกกันแล้วได้ศูนย์)

b. 1101

ขั้นที่ 1 กลับบิต



ได้ 1’s complement = 0010

ขั้นที่ 2 บวกหนึ่ง

บวกหนึ่งแบบตั้งหลัก (ตัวทดสีแดง)

```
  0010
+    1
------
  0011
```

บิตขวาสุด 0 + 1 = 1 · บิตอื่นไม่เปลี่ยน ได้ 0011

2’s complement ของ 1101 = **0011**

เช็กกลับ: 1101 + 0011 = 10000 ตัวทดล้นออกไปหนึ่งตัว เหลือ 4 บิตล่างเป็น 0000 ถูกต้อง

ทางลัดที่ใช้ได้จริงในห้องสอบ

ไล่จากขวามาซ้าย **ลอก 0 มาเรื่อย ๆ จนเจอ 1 ตัวแรก ลอก 1 ตัวนั้นมาด้วย แล้วบิตที่เหลือทางซ้ายให้กลับหมด**

ลองกับ 1101 — ขวาสุดเป็น 1 ลอกมาเลยได้ 1 แล้วบิตที่เหลือ 110 กลับเป็น 001 ต่อกันได้ **0011** ตรงกับที่ทำยาว ๆ ข้างบน

คำตอบ

a. `001`

b. `0011`

จุดที่มักพลาดข้อนี้

บวกหนึ่งก่อนแล้วค่อยกลับบิต — ข้อ b ถ้าทำสลับจะได้ 1101 + 1 = 1110 แล้วกลับบิตเป็น **0001** ซึ่งไม่ใช่คำตอบ · ต้อง **กลับบิตก่อน แล้วค่อยบวกหนึ่ง** เสมอ

ข้อ 10 · เขียน −45 เป็นเลข 8 บิต ทั้งสามแบบ

โจทย์ Assignment #1 · Q10

> **Express -45_{10} as an 8-bit number in sign-magnitude, 1's complement, and 2's complement forms.**

ไทย: เขียนเลข −45 ให้อยู่ในรูปเลข 8 บิต สามแบบ คือ แบบเครื่องหมาย-ขนาด, แบบ 1's complement และแบบ 2's complement

วิธีที่ใช้: เขียน ขนาด 45 เป็นเลขฐานสอง 8 บิตก่อน แล้วค่อยทำสามแบบจากตัวเดียวกันนั้น · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ทั้งสามแบบเริ่มจากของชิ้นเดียวกันคือ **ค่าบวก 45 ในรูป 8 บิต** เพราะฉะนั้นทำขั้นนี้ให้เสร็จก่อน แล้วสามคำตอบจะตามมาเอง

ขั้นที่ 1 · เขียน 45 เป็นเลขฐานสอง

หารด้วย 2 เก็บเศษไปเรื่อย ๆ

① หารด้วย 2 ② จดเศษไว้ ③ เอาผลหารไปหารต่อ จนผลหารเป็น 0



อ่านเศษจากล่างขึ้นบน = 101101_2

อ่านเศษจากล่างขึ้นบนได้ 101101 ซึ่งมี 6 บิต โจทย์ต้องการ 8 บิต จึง **เติมศูนย์ข้างหน้าอีกสองตัว**

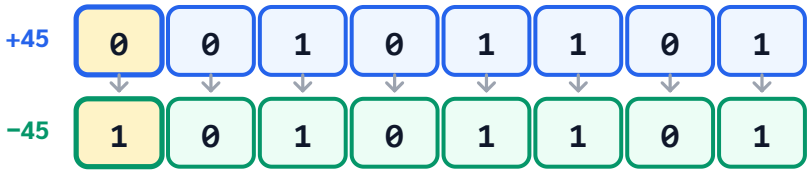


45 ในรูป 8 บิต — สองบิตหน้าที่เติมเข้าไปเป็นศูนย์

เช็กกลับ 32 + 8 + 4 + 1 = 45 ถูกต้อง

แบบที่ 1 · sign-magnitude (เครื่องหมาย-ขนาด)

แบบนี้ตรงไปตรงมาที่สุด **บิตซ้ายสุดเก็บเครื่องหมาย** 0 คือบวก 1 คือลบ ส่วนอีก 7 บิตที่เหลือเก็บขนาดตามเดิม ไม่แตะต้องเลย

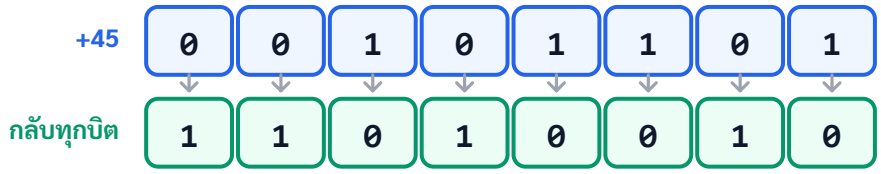


เปลี่ยนแค่บิตซ้ายสุดจาก 0 เป็น 1 บิตอื่นเหมือนเดิมทั้งหมด

−45 แบบ sign-magnitude = **10101101**

แบบที่ 2 · 1’s complement

เอาค่าบวก 8 บิตมากลับทุกบิต



0↔1 ทั้ง 8 ตำแหน่ง

−45 แบบ 1’s complement = **11010010**

แบบที่ 3 · 2’s complement

เอาผลของแบบที่ 2 มาบวกหนึ่ง

บวกหนึ่งที่บิตขวาสุด

```
  11010010
+ 00000001
----------
  11010011
```

บิตขวาสุด 0 + 1 = 1 ไม่มีตัวทด บิตอื่นคงเดิม

−45 แบบ 2’s complement = **11010011**

เช็กกลับ: ทำ 2’s complement ของ 11010011 อีกครั้ง ต้องได้ 45 กลับมา — กลับบิตได้ 00101100 บวกหนึ่งได้ **00101101** = 45 ถูกต้อง

ทั้งสามแบบวางเรียงกัน

แบบ	บิต 8 ตัว	บิตซ้ายสุดบอกอะไร
sign-magnitude	`10101101`	1 = ลบ · 7 บิตที่เหลืออ่านเป็นขนาด 45 ตรง ๆ
1’s complement	`11010010`	1 = ลบ · ต้องกลับบิตก่อนถึงจะอ่านขนาดได้
2’s complement	`11010011`	1 = ลบ · ต้องกลับบิตแล้วบวกหนึ่งถึงจะอ่านขนาดได้

คำตอบ

sign-magnitude = `10101101`

1’s complement = `11010010`

2’s complement = `11010011`

จุดที่มักพลาดข้อนี้

ทำคอมพลีเมนต์จาก 101101 ที่ยังไม่เติมศูนย์ครบ 8 บิต — ต้องเติมเป็น **00101101** ก่อนเสมอ ไม่งั้น 1’s complement จะได้ 010010 ซึ่งผิดทั้งค่าและจำนวนบิต

ข้อ 11 · อ่านค่าเลข 8 บิต ภายใต้ข้อตกลงสามแบบ

โจทย์ Assignment #1 · Q11

> **Consider the following 8-bit binary numbers. Determine the equivalent decimal values assuming sign-magnitude, 1’s complement, and 2’s complement forms.**
> a. 00000000_2 b. 10010100_2

ไทย: บิตชุดเดียวกันนี้ ถ้าตีความว่าเป็นแบบเครื่องหมาย-ขนาด / แบบ 1’s complement / แบบ 2’s complement จะมีค่าเท่าไรในแต่ละแบบ — ข้อนี้กลับทิศกับข้อ 10 คือให้บิตมา แล้วถามค่า

วิธีที่ใช้: ดูบิตซ้ายสุดก่อนว่าบวกหรือลบ ถ้าลบให้ **แปลงกลับ** ตามกติกาของแต่ละแบบเพื่อหาขนาด แล้วค่อยใส่เครื่องหมายลบ · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ข้อสำคัญที่ต้องเข้าใจก่อนคือ บิตชุดเดียวกัน **ไม่ได้มีค่าเดียว** ค่าของมันขึ้นกับว่าเราตกลงกันว่าจะอ่านด้วยข้อตกลงแบบไหน เหมือนตัวเลข 12 ที่แปลว่าสิบสองถ้าอ่านแบบฐานสิบ แต่แปลว่าสิบถ้าอ่านแบบฐานแปด

a. 00000000_2

บิตซ้ายสุดเป็น **0** แปลว่าเป็นจำนวนบวกในทุกแบบ ส่วนที่เหลืออีก 7 บิตก็เป็นศูนย์หมด ขนาดจึงเป็น 0



บิตเครื่องหมาย (ตัวแรกสุด) เป็น 0 = บวก

ตีความแบบ	ทำอะไร	ได้ค่า
sign-magnitude	เครื่องหมาย 0 = บวก · ขนาด 0000000 = 0	**+0**
1’s complement	บิตเครื่องหมายเป็น 0 อ่านตรง ๆ ได้ 0	**+0**
2’s complement	บิตเครื่องหมายเป็น 0 อ่านตรง ๆ ได้ 0	**0**

$$00000000_2 = \mathbf{0} \text{ ทั้งสามแบบ}$$

เช็กกลับ: ทั้งสามแบบให้ค่าเดียวกันคือศูนย์ ไม่ขัดกันเอง

เกร็ดที่ข้อสอบชอบถามต่อ

ศูนย์มี **สองหน้า** ในแบบ sign-magnitude และแบบ 1’s complement คือ 00000000 (บวกศูนย์) กับ 11111111 (ลบศูนย์) ส่วนแบบ 2’s complement มีศูนย์หน้าเดียว นี่คือเหตุผลที่เครื่องจริงเลือกใช้ 2’s complement

b. 10010100_2

บิตซ้ายสุดเป็น **1** แปลว่าเป็นจำนวน **ลบ** ในทุกแบบ แต่วิธีหาขนาดต่างกันสามแบบ

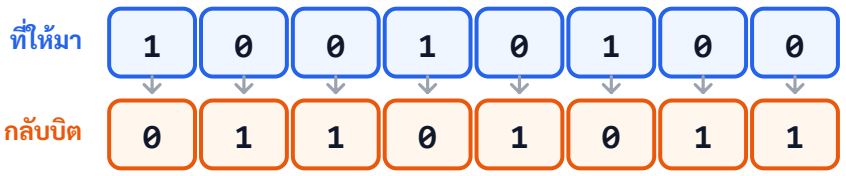


บิตเครื่องหมายเป็น 1 = ลบ ทั้งสามแบบ

แบบที่ 1 sign-magnitude — ตัดบิตเครื่องหมายทิ้ง แล้วอ่าน 7 บิตที่เหลือตรง ๆ

$$0010100_2 = 16 + 4 = 20 \;\rightarrow\; \text{ค่าคือ } \mathbf{-20}$$

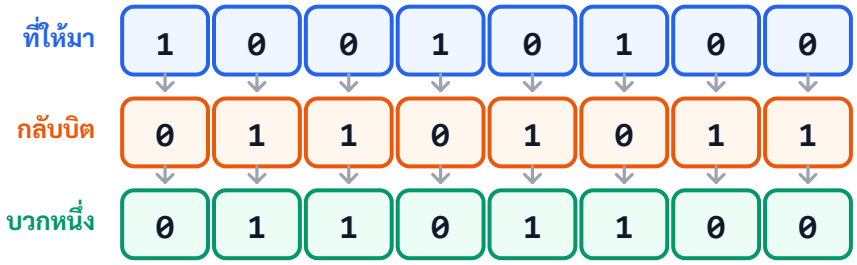
แบบที่ 2 1’s complement — ต้องกลับทุกบิตก่อน ผลที่ได้คือ “ขนาด”



01101011 คือขนาดของจำนวนลบตัวนี้

$$01101011_2 = 64 + 32 + 8 + 2 + 1 = 107 \;\rightarrow\; \text{ค่าคือ } \mathbf{-107}$$

แบบที่ 3 2’s complement — กลับทุกบิตแล้ว **บวกหนึ่ง** ผลที่ได้คือขนาด



01101011 + 1 = 01101100

$$01101100_2 = 64 + 32 + 8 + 4 = 108 \;\rightarrow\; \text{ค่าคือ } \mathbf{-108}$$

เช็กกลับด้วยวิธีที่สอง — แบบ 2’s complement อ่านได้อีกทางคือให้บิตซ้ายสุดมีน้ำหนัก **ติดลบ**

$$-128 + 16 + 4 = -108$$

$$(-128\times1) + (64\times0) + (32\times0) + (16\times1) + (8\times0) + (4\times1) + (2\times0) + (1\times0)$$

ได้ −108 เท่ากัน ยืนยันว่าถูก

คำตอบ

a. 00000000 → sign-magnitude = **+0** · 1’s complement = **+0** · 2’s complement = **0**

b. 10010100 → sign-magnitude = **−20** · 1’s complement = **−107** · 2’s complement = **−108**

พิสัยของเลข 8 บิต (ข้อสอบถามบ่อย)

แบบ	ค่าน้อยสุด	ค่ามากสุด	จำนวนค่าที่ใช้ได้จริง
sign-magnitude	−127	+127	255 (เพราะศูนย์ซ้ำสองหน้า)
1’s complement	−127	+127	255 (ศูนย์ซ้ำสองหน้าเหมือนกัน)
2’s complement	−128	+127	256 (ครบทุกรูปแบบ ไม่ซ้ำ)

บิต 8 ตัวสร้างรูปแบบได้ $2^8 = 256$ แบบ แบบ 2’s complement ใช้ครบทั้ง 256 จึงลงไปได้ถึง −128 ขณะที่อีกสองแบบเสียไปหนึ่งแบบให้ศูนย์ที่ซ้ำ

จุดที่มักพลาดข้อนี้

ลืมใส่เครื่องหมายลบหลังแปลงเสร็จ — ค่าที่ได้จากการกลับบิตคือ **ขนาด** ไม่ใช่คำตอบ ต้องเติมลบข้างหน้าเสมอเมื่อบิตซ้ายสุดเป็น 1

ข้อ 12 · บวกเลขฐานสอง

โจทย์ Assignment #1 · Q12

> **Add the following binary numbers.**
> a. 111+110 b. 1001 + 101 c. 1101+1011

ไทย: บวกเลขฐานสองสามคู่นี้ (เป็นเลขไม่มีเครื่องหมาย ให้ตอบเป็นเลขฐานสอง)

วิธีที่ใช้: บวกทีละหลักจากขวาไปซ้าย มีกติกาแค่สี่บรรทัด 0+0=0, 0+1=1, 1+0=1, **1+1=10** (เขียน 0 ทดไป 1) · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ฐานสองมีเลขแค่ 0 กับ 1 พอบวกได้ 2 มันก็เต็มหลักทันที ต้องทดขึ้นหลักถัดไป เหมือนฐานสิบที่พอได้ 10 ก็ทด แค่จุดที่เต็มมาถึงเร็วกว่ามาก

ตารางบวกฐานสอง — จำสี่บรรทัดนี้พอ			
0 + 0 = 0	0 + 1 = 1	1 + 0 = 1	**1 + 1 = 10** (ทด 1)
ถ้ามีตัวทดมาด้วย: **1 + 1 + 1 = 11** (เขียน 1 ทด 1)			

a. 111 + 110

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
  1 1
    1 1 1
+   1 1 0
-------------
  1 1 0 1
```

หลักที่	บิตบน	บิตล่าง	ตัวทดที่รับมา	เขียนลง	ทดไป
1 (นับจากขวา)	1	0	0	**1**	0
2 (นับจากขวา)	1	1	0	**0**	1
3 (นับจากขวา)	1	1	1	**1**	1

ตัวทดของหลักซ้ายสุดยังเหลืออยู่ 1 จึงต้อง **เขียนเพิ่มอีกหนึ่งหลัก** ผลลัพธ์จึงยาวกว่าตัวตั้ง

$$111 + 110 = \mathbf{1101}_2$$

เช็กด้วยฐานสิบ: $7 + 6 = 13$ และ $1101_2 = 13$ ตรงกัน

b. 1001 + 101

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
      1
  1 0 0 1
+   1 0 1
---------
  1 1 1 0
```

หลักที่	บิตบน	บิตล่าง	ตัวทดที่รับมา	เขียนลง	ทดไป
1 (นับจากขวา)	1	1	0	**0**	1
2 (นับจากขวา)	0	0	1	**1**	0
3 (นับจากขวา)	0	1	0	**1**	0
4 (นับจากขวา)	1	0	0	**1**	0

$$1001 + 101 = \mathbf{1110}_2$$

เช็กด้วยฐานสิบ: $9 + 5 = 14$ และ $1110_2 = 14$ ตรงกัน

c. 1101 + 1011

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
  1 1 1 1
    1 1 0 1
+   1 0 1 1
-----------
  1 1 0 0 0
```

หลักที่	บิตบน	บิตล่าง	ตัวทดที่รับมา	เขียนลง	ทดไป
1 (นับจากขวา)	1	1	0	**0**	1
2 (นับจากขวา)	0	1	1	**0**	1
3 (นับจากขวา)	1	0	1	**0**	1
4 (นับจากขวา)	1	1	1	**1**	1

ตัวทดของหลักซ้ายสุดยังเหลืออยู่ 1 จึงต้อง **เขียนเพิ่มอีกหนึ่งหลัก** ผลลัพธ์จึงยาวกว่าตัวตั้ง

$$1101 + 1011 = \mathbf{11000}_2$$

เช็กด้วยฐานสิบ: $13 + 11 = 24$ และ $11000_2 = 24$ ตรงกัน

คำตอบ

a. $111 + 110 = \mathbf{1101_2}$ $(7 + 6 = 13)$

b. $1001 + 101 = \mathbf{1110_2}$ $(9 + 5 = 14)$

c. $1101 + 1011 = \mathbf{11000_2}$ $(13 + 11 = 24)$

จุดที่มักพลาดข้อนี้

ลืมเขียนตัวทดตัวสุดท้ายที่ล้นออกมาทางซ้าย — ข้อ c ต้องได้ **11000** ห้าบิต ถ้าตอบ 1000 สี่บิตคือหล่นตัวทดไปหนึ่งตัว ค่าจะกลายเป็น 8 แทนที่จะเป็น 24

ข้อ 13 · ลบเลขฐานสอง

โจทย์ Assignment #1 · Q13

> **Subtract the following binary numbers.**
> a. 1110-11 b. 1100-1001 c. 11010-10111

ไทย: ลบเลขฐานสองสามคู่นี้แบบตรง ๆ (ยังไม่ใช้คอมพลีเมนต์ ข้อนั้นอยู่ใน Assignment 2)

วิธีที่ใช้: ลบทีละหลักจากขวาไปซ้าย ถ้าหลักไหนบนน้อยกว่าล่าง ให้ **ยืมจากหลักซ้าย** · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

การยืมในฐานสองได้มา **2 หน่วย** ไม่ใช่ 10 หน่วยเหมือนฐานสิบ เพราะหลักถัดไปทางซ้ายมีค่าเป็นสองเท่าของหลักปัจจุบัน

ตารางลบฐานสอง			
$0 - 0 = 0$	$1 - 0 = 1$	$1 - 1 = 0$	**$0 - 1 = 1$ แล้วยืม 1**

ก่อนลงมือทุกครั้ง ให้ **เติมศูนย์หน้าตัวลบให้ยาวเท่าตัวตั้ง** ก่อน เช่นข้อ a คือ 1110 − 0011 จะได้ไม่หลงหลัก

a. 1110 − 11

เติมศูนย์หน้าตัวลบก่อน 11 กลายเป็น **0011**

ลบแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวยืม

```
    1 1
  1 1 1 0
- 0 0 1 1
---------
  1 0 1 1
```

หลักที่	บิตบน	บิตล่าง	ตัวยืมค้างมา	เขียนลง	ต้องยืมต่อไหม
1 (นับจากขวา)	0	1	0	**1**	ยืม 1
2 (นับจากขวา)	1	1	1	**1**	ยืม 1
3 (นับจากขวา)	1	0	1	**0**	—
4 (นับจากขวา)	1	0	0	**1**	—

$$1110 - 11 = \mathbf{1011}_2$$

เช็กด้วยฐานสิบ: $14 - 3 = 11$ และ $1011_2 = 11$ ตรงกัน

เช็กอีกทางด้วยการบวกกลับ: 1011 + 0011 = 1110 ได้ตัวตั้งเดิมคืนมา

b. 1100 − 1001

ลบแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวยืม

```
    1 1
  1 1 0 0
- 1 0 0 1
---------
  0 0 1 1
```

หลักที่	บิตบน	บิตล่าง	ตัวยืมค้างมา	เขียนลง	ต้องยืมต่อไหม
1 (นับจากขวา)	0	1	0	**1**	ยืม 1
2 (นับจากขวา)	0	0	1	**1**	ยืม 1
3 (นับจากขวา)	1	0	1	**0**	—
4 (นับจากขวา)	1	1	0	**0**	—

$$1100 - 1001 = \mathbf{0011}_2$$

เช็กด้วยฐานสิบ: $12 - 9 = 3$ และ $0011_2 = 3$ ตรงกัน

เช็กอีกทางด้วยการบวกกลับ: 0011 + 1001 = 1100 ได้ตัวตั้งเดิมคืนมา

c. 11010 − 10111

ลบแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวยืม

```
    1 1 1
  1 1 0 1 0
- 1 0 1 1 1
-----------
  0 0 0 1 1
```

หลักที่	บิตบน	บิตล่าง	ตัวยืมค้างมา	เขียนลง	ต้องยืมต่อไหม
1 (นับจากขวา)	0	1	0	**1**	ยืม 1
2 (นับจากขวา)	1	1	1	**1**	ยืม 1
3 (นับจากขวา)	0	1	1	**0**	ยืม 1
4 (นับจากขวา)	1	0	1	**0**	—
5 (นับจากขวา)	1	1	0	**0**	—

$$11010 - 10111 = \mathbf{00011}_2$$

เช็กด้วยฐานสิบ: $26 - 23 = 3$ และ $00011_2 = 3$ ตรงกัน

เช็กอีกทางด้วยการบวกกลับ: 00011 + 10111 = 11010 ได้ตัวตั้งเดิมคืนมา

คำตอบ

a. 1110 − 11 = $\mathbf{1011_2}$ (14 − 3 = 11)

b. 1100 − 1001 = $\mathbf{0011_2}$ (12 − 9 = 3)

c. 11010 − 10111 = $\mathbf{00011_2}$ (26 − 23 = 3)

จุดที่มักพลาดข้อนี้

ยืมแล้วลืมหักหลักที่ให้ยืม — เวลายืม หลักซ้ายต้องลดลง 1 ด้วยเสมอ (ในตารางข้างบนคือช่อง “ตัวยืมค้างมา” ของรอบถัดไป) ถ้าลืมหัก ข้อ c จะได้ 01101 แทนที่จะเป็น 00011

ข้อ 14 · คูณเลขฐานสอง

โจทย์ Assignment #1 · Q14

> **Perform the following binary multiplications.**
> a. 1001 x 11 b. 1101 x 1101 c. 1110 x 1101

ไทย: คูณเลขฐานสองสามคู่นี้ ตอบเป็นเลขฐานสอง

วิธีที่ใช้: ตั้งคูณเหมือนฐานสิบ แต่ง่ายกว่ามาก เพราะตัวคูณมีแค่ 0 กับ 1 · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ตัวคูณเป็น 1 แปลว่า **ลอกตัวตั้งลงมาทั้งแถว** ตัวคูณเป็น 0 แปลว่า **ลงศูนย์ทั้งแถว** ไม่มีสูตรคูณให้ท่องเลย

ที่เหลือคือ **เลื่อนหลัก** — แถวของตัวคูณหลักที่สองต้องเลื่อนซ้ายหนึ่งช่อง หลักที่สามเลื่อนสองช่อง เหมือนฐานสิบเป๊ะ แล้วจบด้วยการบวกทุกแถว

a. 1001 × 11

หลักของตัวคูณ	บิต	ทำอะไร	ได้แถวย่อย
1 (นับจากขวา)	1	ลอก 1001 ลงมา แล้วเลื่อนซ้าย 0 ช่อง	`1001`
2 (นับจากขวา)	1	ลอก 1001 ลงมา แล้วเลื่อนซ้าย 1 ช่อง	`10010`

คูณแบบตั้งคูณ

```
    1 0 0 1
x       1 1
-----------

    1 0 0 1
  1 0 0 1 0
-----------

  1 1 0 1 1
```

บวกเฉพาะแถวที่ไม่ใช่ศูนย์ ทีละคู่: **1001 + 10010 = 11011**

$$1001 \times 11 = \mathbf{11011}_{2}$$

เช็กด้วยฐานสิบ: $9 \times 3 = 27$ และ $11011_2 = 27$ ตรงกัน

b. 1101 × 1101

หลักของตัวคูณ	บิต	ทำอะไร	ได้แถวย่อย
1 (นับจากขวา)	1	ลอก 1101 ลงมา แล้วเลื่อนซ้าย 0 ช่อง	`1101`

หลักของตัวคูณ	บิต	ทำอะไร	ได้แถวย่อย
2 (นับจากขวา)	0	ตัวคูณเป็น 0 แถวนี้เป็นศูนย์ทั้งแถว	`00000`
3 (นับจากขวา)	1	ลอก 1101 ลงมา แล้วเลื่อนซ้าย 2 ช่อง	`110100`
4 (นับจากขวา)	1	ลอก 1101 ลงมา แล้วเลื่อนซ้าย 3 ช่อง	`1101000`

คูณแบบตั้งคูณ

```
          1 1 0 1
x         1 1 0 1
-----------------

          1 1 0 1
        0 0 0 0 0
      1 1 0 1 0 0
    1 1 0 1 0 0 0
-----------------

  1 0 1 0 1 0 0 1
```

บวกเฉพาะแถวที่ไม่ใช่ศูนย์ ทีละคู่: **1101 + 110100 = 1000001** · **1000001 + 1101000 = 10101001**

$$1101 \times 1101 = \mathbf{10101001}_{2}$$

เช็กด้วยฐานสิบ: $13 \times 13 = 169$ และ $10101001_2 = 169$ ตรงกัน

c. 1110 × 1101

หลักของตัวคูณ	บิต	ทำอะไร	ได้แถวย่อย
1 (นับจากขวา)	1	ลอก 1110 ลงมา แล้วเลื่อนซ้าย 0 ช่อง	`1110`
2 (นับจากขวา)	0	ตัวคูณเป็น 0 แถวนี้เป็นศูนย์ทั้งแถว	`00000`
3 (นับจากขวา)	1	ลอก 1110 ลงมา แล้วเลื่อนซ้าย 2 ช่อง	`111000`
4 (นับจากขวา)	1	ลอก 1110 ลงมา แล้วเลื่อนซ้าย 3 ช่อง	`1110000`

คูณแบบตั้งคูณ

```
          1 1 1 0
x         1 1 0 1
-----------------
          1 1 1 0
        0 0 0 0 0
      1 1 1 0 0 0
    1 1 1 0 0 0 0
-----------------
  1 0 1 1 0 1 1 0
```

บวกเฉพาะแถวที่ไม่ใช่ศูนย์ ทีละคู่: **1110 + 111000 = 1000110 · 1000110 + 1110000 = 10110110**

$$1110 \times 1101 = \mathbf{10110110}_{2}$$

เช็กด้วยฐานสิบ: $14 \times 13 = 182$ และ $10110110_2 = 182$ ตรงกัน

คำตอบ

a. $1001 \times 11 = \mathbf{11011_2}$ ($9 \times 3 = 27$)

b. $1101 \times 1101 = \mathbf{10101001_2}$ ($13 \times 13 = 169$)

c. $1110 \times 1101 = \mathbf{10110110_2}$ ($14 \times 13 = 182$)

จุดที่มักพลาดข้อนี้

ลืมเลื่อนหลักของแถวที่ตัวคูณเป็น 0 — แถวศูนย์ก็ต้องนับตำแหน่งด้วย ข้อ b และ c ตัวคูณ 1101 มีศูนย์อยู่หลักที่สอง ถ้าข้ามแถวนั้นไปเฉย ๆ แถวที่สามจะเลื่อนผิดไปหนึ่งช่อง ผลคูณจะเล็กลงครึ่งหนึ่ง

ข้อ 15 · หารเลขฐานสอง

โจทย์ Assignment #1 · Q15

> **Divide the followig binary numbers.** (สะกดตามต้นฉบับ)
> a. 100 ÷ 10 b. 1001 ÷ 11 c. 1100 ÷ 100

ไทย: หารเลขฐานสองสามคู่นี้ ตอบเป็นเลขฐานสอง

> **วิธีที่ใช้:** ตั้งหารยาวเหมือนฐานสิบ แต่ผลหารแต่ละหลักมีแค่ **1 (ลบได้)** หรือ **0 (ลบไม่ได้)** · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ขั้นตอนวนซ้ำแค่สามอย่าง ① **ดึงบิตถัดไปลงมา** ② **ถามว่าตัวหารลบได้ไหม** ③ **ลบได้ใส่ 1 แล้วลบ ลบไม่ได้ใส่ 0** วนจนหมดบิต

ผลหารที่อ่านได้จากการตั้งหารจะมี **จำนวนหลักเท่ากับตัวตั้ง** เสมอ เช่นข้อ a ได้ 010 สามหลัก เวลาตอบให้ตัดศูนย์ข้างหน้าออกเหลือ **10** (ค่าเท่ากัน)

a. 100 ÷ 10

หารยาวฐานสอง

```
     0 1 0
     -----
10 ) 1 0 0
     1 0
   - 1 0
     ---
       0
```

รอบที่	ตัวตั้งที่มีอยู่ตอนนี้	เอา 10 ลบได้ไหม	บิตผลหาร	เหลือเท่าไร
1	**1**	ลบไม่ได้ (น้อยกว่า 10)	**0**	**1**
2	**10**	ลบได้	**1**	**0**
3	**0**	ลบไม่ได้ (น้อยกว่า 10)	**0**	**0**

$$100 \div 10 = \mathbf{10}_{2} \text{ เศษ } 0$$

เช็กด้วยฐานสิบ: 4 ÷ 2 = 2 เศษ 0 ตรงกัน

เช็กกลับด้วยการคูณ: 10 × 10 = 100 ได้ตัวตั้งเดิม

b. 1001 ÷ 11

หารยาวฐานสอง

```
     0 0 1 1
     -------
11 ) 1 0 0 1
     1 0 0
     - 1 1
     -----
         1
         1 1
       - 1 1
         ---
           0
```

รอบที่	ตัวตั้งที่มีอยู่ตอนนี้	เอา 11 ลบได้ไหม	บิตผลหาร	เหลือเท่าไร
1	**1**	ลบไม่ได้ (น้อยกว่า 11)	**0**	**1**
2	**10**	ลบไม่ได้ (น้อยกว่า 11)	**0**	**10**
3	**100**	ลบได้	**1**	**1**
4	**11**	ลบได้	**1**	**0**

$$1001 \div 11 = \mathbf{11}_2 \text{ เศษ } 0$$

เช็กด้วยฐานสิบ: 9 ÷ 3 = 3 เศษ 0 ตรงกัน

เช็กกลับด้วยการคูณ: 11 × 11 = 1001 ได้ตัวตั้งเดิม

c. 1100 ÷ 100

หารยาวฐานสอง

```
      0 0 1 1
      -------
100 ) 1 1 0 0
      1 1 0
    - 1 0 0
      -----
        1 0
        1 0 0
      - 1 0 0
        -----
            0
```

รอบที่	ตัวตั้งที่มีอยู่ตอนนี้	เอา 100 ลบได้ไหม	บิตผลหาร	เหลือเท่าไร
1	`1`	ลบไม่ได้ (น้อยกว่า 100)	**0**	`1`
2	`11`	ลบไม่ได้ (น้อยกว่า 100)	**0**	`11`
3	`110`	ลบได้	**1**	`10`
4	`100`	ลบได้	**1**	`0`

$$1100 \div 100 = \mathbf{11}_2 \text{ เศษ } 0$$

เช็กด้วยฐานสิบ: 12 ÷ 4 = 3 เศษ 0 ตรงกัน

เช็กกลับด้วยการคูณ: 11 × 100 = 1100 ได้ตัวตั้งเดิม

ทางลัดที่เห็นได้ทันที

การหารด้วย 10_2 (คือ 2) เท่ากับ **เลื่อนบิตไปทางขวาหนึ่งช่อง** และหารด้วย 100_2 (คือ 4) เท่ากับเลื่อนขวาสองช่อง

ข้อ a: 100 เลื่อนขวา 1 ช่องได้ **10** · ข้อ c: 1100 เลื่อนขวา 2 ช่องได้ **11** ตรงกับที่ตั้งหารยาวไว้ข้างบนทั้งคู่ ใช้เช็กคำตอบเร็ว ๆ ได้

คำตอบ

a. 100 ÷ 10 = $\mathbf{10_2}$ เศษ 0 (4 ÷ 2 = 2)

b. 1001 ÷ 11 = $\mathbf{11_2}$ เศษ 0 (9 ÷ 3 = 3)

c. 1100 ÷ 100 = $\mathbf{11_2}$ เศษ 0 (12 ÷ 4 = 3)

จุดที่มักพลาดข้อนี้

ลืมใส่บิต 0 ในผลหารตอนที่ลบไม่ได้ — ต้องใส่ 0 แล้วดึงบิตถัดไปลงมา ถ้าข้ามไปเฉย ๆ ผลหารจะสั้นไปหนึ่งหลักและค่าหายไปครึ่งหนึ่ง

2 Assignment 2 — ทศนิยมลอยตัว เลขมีเครื่องหมาย และรหัสตรวจข้อผิดพลาด (10 ข้อ)

ข้อ 1 · เขียน −99.03125 เป็นทศนิยมลอยตัวแบบ single precision

โจทย์ Assignment #2 · Q1

> **Express the following real numbers as single-precision floating points: -99.03125_{10}**

ไทย: เขียนจำนวนจริง −99.03125 ให้อยู่ในรูปทศนิยมลอยตัวความละเอียดเดี่ยว (32 บิต) ตามมาตรฐาน IEEE 754

วิธีที่ใช้: สี่ขั้นตามสไลด์ ① หาบิตเครื่องหมาย ② แปลงเป็นฐานสองแล้วทำให้เป็นรูป $1.xxx \times 2^e$ ③ บวกไบแอส 127 เข้ากับ e ④ ต่อสามส่วนเข้าด้วยกัน · **สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

เลข 32 บิตถูกซอยเป็นสามช่อง ช่องละหน้าที่ ถ้าจำภาพนี้ได้ ข้อสอบข้อนี้จะกลายเป็นการเติมช่องว่างเฉย ๆ



S = เครื่องหมาย · E = เลขชี้กำลังบวกไบแอส 127 · F = ตัวเลขหลังจุดของรูป 1.xxx

ยังว่างอยู่ — เราจะเติมทีละช่อง

ขั้นที่ 1 · บิตเครื่องหมาย

เลขในโจทย์เป็นจำนวน **ลบ** บิตเครื่องหมายจึงเป็น **1** (ถ้าเป็นบวกจะเป็น 0)

ตั้งแต่นี้ไปเราทำงานกับ **ขนาด 99.03125** อย่างเดียว ลืมเครื่องหมายลบไปก่อน แล้วค่อยเอากลับมาตอนประกอบ

ขั้นที่ 2 · แปลงส่วนจำนวนเต็ม 99 เป็นฐานสอง

① หารด้วย 2 ② จดเศษไว้ ③ เอาผลหารไปหารต่อ จนผลหารเป็น 0



อ่านเศษจากล่างขึ้นบน = 1100011_2

อ่านเศษจากล่างขึ้นบนได้ **1100011** · เช็กกลับ 64 + 32 + 2 + 1 = 99 ถูกต้อง

ขั้นที่ 3 · แปลงส่วนเศษ 0.03125 เป็นฐานสอง

คูณด้วย 2 แล้วดูว่ามี 1 โผล่ข้ามจุดมาไหม ทำจนเศษเหลือศูนย์

① คูณเศษด้วย 2 ② หลักที่ได้คือส่วนจำนวนเต็ม ③ เอาเศษที่เหลือไปคูณต่อ

①	`0.03125 × 2 = 0.0625`	`0`	เหลือ 0.0625
②	`0.0625 × 2 = 0.125`	`0`	เหลือ 0.125
③	`0.125 × 2 = 0.25`	`0`	เหลือ 0.25
④	`0.25 × 2 = 0.5`	`0`	เหลือ 0.5
⑤	`0.5 × 2 = 1`	`1`	เหลือ 0

อ่านหลักจากบนลงล่าง

อ่านหลักจากบนลงล่างได้ **00001** · เช็กกลับ $0.00001_2 = 1/32 = 0.03125$ ถูกต้อง

$$99.03125_{10} = \mathbf{1100011.00001}_2$$

ขั้นที่ 4 · ทำให้เป็นรูปนอร์มัลไลซ์ 1.xxx × 2^e

เลื่อนจุดไปทางซ้ายจนเหลือเลข 1 ตัวเดียวหน้าจุด ต้องเลื่อนกี่ช่อง เลขชี้กำลังก็เท่านั้น

ก่อนเลื่อน	1	1	0	0	0	1	1	.	0	0	0	0	1

จุดอยู่หลังบิตที่ 7

$$1100011.00001_2 = \mathbf{1.10001100001} \times 2^6$$

เลื่อนจุดมาทางซ้าย 6 ช่อง → เลขชี้กำลัง e = 6

เช็กกลับ: เลื่อนจุดกลับไปขวา 6 ช่องจาก 1.10001100001 ได้ 1100011.00001 ตรงกับก่อนหน้า

ขั้นที่ 5 · เลขชี้กำลังบวกไบแอส

ช่อง E ไม่ได้เก็บ e ตรง ๆ แต่เก็บ **e + 127** เพื่อให้เลขชี้กำลังติดลบเก็บได้โดยไม่ต้องมีเครื่องหมายอีกตัว

$$E = 6 + 127 = \mathbf{133} = \mathbf{10000101}_2$$

เช็กกลับ 128 + 4 + 1 = 133 ถูกต้อง และต้องยาว 8 บิตพอดี

ขั้นที่ 6 · ช่อง F (ตัวเลขหลังจุด)

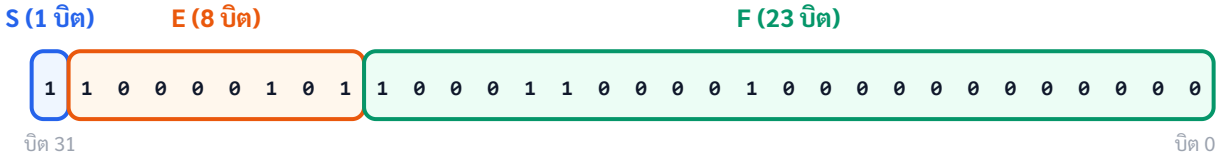
ช่อง F เก็บเฉพาะ **ส่วนที่อยู่หลังจุด** ของ 1.10001100001 ส่วนเลข 1 หน้าจุดไม่ต้องเก็บ เพราะรูปนอร์มัลไลซ์มี 1 หน้าจุดเสมออยู่แล้ว (เรียกว่าบิตที่ซ่อนไว้)

ตัวเลขหลังจุดคือ 10001100001 ซึ่งมี 11 บิต แต่ช่อง F ยาว 23 บิต จึง **เติมศูนย์ต่อท้าย** อีก 12 บิต



11 บิตแรกมาจากโจทย์ · 12 บิตหลัง (สีส้ม) คือศูนย์ที่เติมเข้าไป

ขั้นที่ 7 · ประกอบสามช่องเข้าด้วยกัน



S = เครื่องหมาย · E = เลขชี้กำลังบวกไบแอส 127 · F = ตัวเลขหลังจุดของรูป 1.xxx

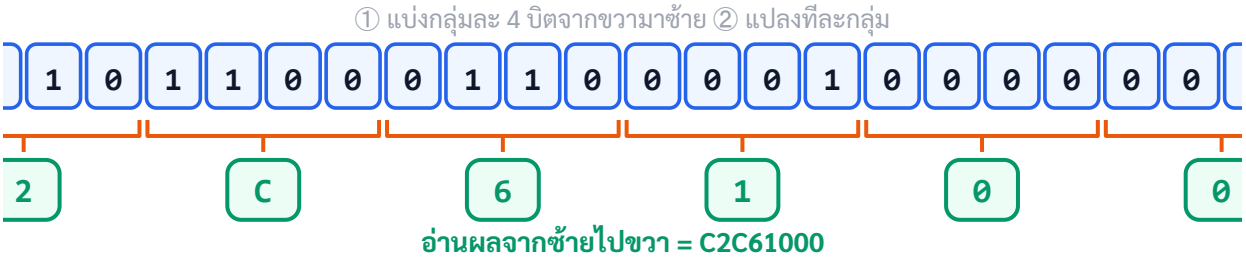
S = 1 · E = 10000101 · F = 10001100001 แล้วตามด้วยศูนย์

1 10000101 10001100001000000000000

นับความยาวเช็ก 1 + 8 + 23 = **32 บิต** ครบพอดี

ขั้นที่ 8 · เขียนเป็นฐานสิบหกให้อ่านง่าย (ถ้าโจทย์อยากได้)

จับกลุ่มละ 4 บิตจากซ้าย



$$= \mathbf{C2C61000}_{16}$$

คำตอบ

บิตเครื่องหมาย S = **1** (เพราะเป็นจำนวนลบ)

เลขชี้กำลัง E = **10000101** (คือ 133 = 6 + 127)

ช่อง F = **10001100001000000000000**

รวม 32 บิต = **11000010110001100001000000000000**

เขียนเป็นฐานสิบหก = $\mathbf{C2C61000}_{16}$

จุดที่มักพลาดข้อนี้

ใส่เลข 1 หน้าจุดลงไปในช่อง F ด้วย — ห้ามใส่ ช่อง F เก็บเฉพาะหลังจุด และอย่าลืมเติมศูนย์ต่อท้ายให้ครบ **23 บิต**

ข้อ 2 · บวกเลขแบบ 2’s complement

โจทย์ Assignment #2 · Q2

> **Perform addition in the 2’s complement form:**
> a. 00010110 + 00110011 b. 01110000 + 10101111

ไทย: บวกเลข 8 บิตสองคู่นี้ โดยถือว่าเป็นเลขแบบ 2’s complement (คือบิตซ้ายสุดเป็นบิตเครื่องหมาย)

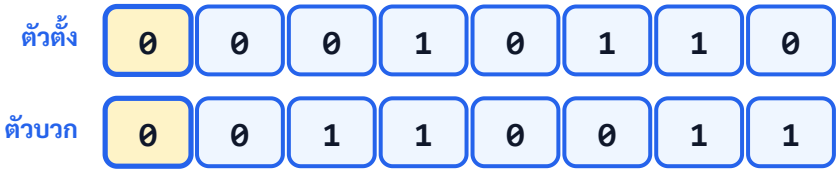
วิธีที่ใช้: บวกตรง ๆ ทุกบิตรวมทั้งบิตเครื่องหมาย แล้ว **ทิ้งตัวทดที่ล้นออกจากบิตซ้ายสุด · สอนไว้ใน บทที่ 1 · ตัวเลขในเครื่อง**

ข้อดีทั้งหมดของ 2’s complement อยู่ตรงนี้ — เครื่องไม่ต้องรู้ว่าเลขบวกหรือลบ บวกไปเหมือนกันหมด แล้วคำตอบออกมาถูกเอง

แต่มี **สองเรื่องที่ต้องเช็กทุกครั้ง** คือ ตัวทดที่ล้นออกมาต้องทิ้ง และต้องดูว่าผลลัพธ์ ล้น (`overflow`) ออกนอกพิสัย −128 ถึง +127 หรือเปล่า

a. 00010110 + 00110011

ก่อนบวก อ่านค่าก่อนว่าเรากำลังบวกอะไรกับอะไร — บิตซ้ายสุดของทั้งคู่เป็น 0 แปลว่าบวกทั้งคู่



บิตซ้ายสุดเป็น 0 ทั้งคู่ = จำนวนบวกทั้งคู่

$00010110_2 = 16 + 4 + 2 = \mathbf{22}$ และ $00110011_2 = 32 + 16 + 2 + 1 = \mathbf{51}$

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
    1 1   1 1
  0 0 0 1 0 1 1 0
+ 0 0 1 1 0 0 1 1
-----------------
  0 1 0 0 1 0 0 1
```

ไม่มีตัวทดล้นออกจากบิตซ้ายสุด ผลลัพธ์คือ 01001001 ครบ 8 บิตพอดี

บิตซ้ายสุดของผลลัพธ์เป็น **0** แปลว่าคำตอบเป็นบวก ถูกต้องตามที่ควรจะเป็น (บวก + บวก ต้องได้บวก)

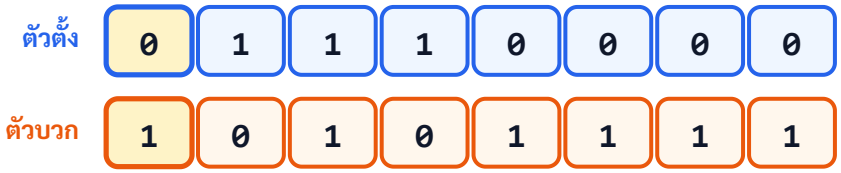
$$00010110 + 00110011 = \mathbf{01001001}_2$$

เช็กด้วยฐานสิบ: 22 + 51 = 73 และ $01001001_2 = 64 + 8 + 1 = 73$ ตรงกัน

เช็กเรื่องล้น: 73 ยังอยู่ในพิสัย −128 ถึง +127 จึง **ไม่ล้น**

b. 01110000 + 10101111

ก่อนบวก บิตซ้ายสุดของตัวแรกเป็น 0 (บวก) ของตัวที่สองเป็น 1 (ลบ)



ตัวล่างบิตซ้ายสุดเป็น 1 = จำนวนลบ

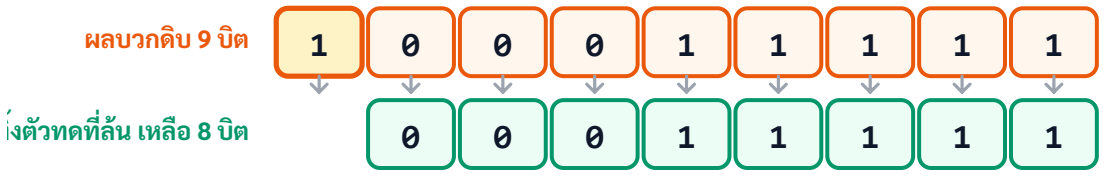
ตัวบน $01110000_2 = 64 + 32 + 16 = \mathbf{112}$

ตัวล่างเป็นลบ ต้องทำ 2's complement กลับเพื่อหาขนาด — กลับบิตได้ 01010000 บวกหนึ่งได้ **01010001** = 64 + 16 + 1 = 81 เพราะฉะนั้นตัวล่างคือ **−81**

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
  1 1 1
    0 1 1 1 0 0 0 0
+   1 0 1 0 1 1 1 1
-----------------
  1 0 0 0 1 1 1 1 1
```

คราวนี้ **มีตัวทดล้นออกจากบิตซ้ายสุด 1 ตัว** ผลบวกดิบยาว 9 บิตคือ 100011111



ตัวทดที่ล้นออกจากบิตเครื่องหมายให้ทิ้งไปเลย ไม่ใช่ความผิดพลาด

บิตซ้ายสุดของผลลัพธ์เป็น **0** แปลว่าคำตอบเป็นบวก ซึ่งสมเหตุสมผลเพราะ 112 มากกว่า 81

$$01110000 + 10101111 = \mathbf{00011111}_2$$

เช็กด้วยฐานสิบ: $112 + (-81) = 31$ และ $00011111_2 = 16 + 8 + 4 + 2 + 1 = 31$ ตรงกัน

เช็กเรื่องล้น: 31 อยู่ในพิสัย จึง **ไม่ล้น**

ตัวทดล้น กับ การล้นของค่า ไม่ใช่เรื่องเดียวกัน

ตัวทดล้นออกจากบิตซ้ายสุด — เกิดขึ้นได้ตลอด ให้ทิ้งทิ้งไปเลย ไม่ได้แปลว่าผิด (อย่างข้อ b)

ค่าล้น (overflow) — เกิดเมื่อบวกเลข เครื่องหมายเดียวกัน แล้วได้ผลลัพธ์เครื่องหมายตรงข้าม เช่น บวก + บวก แล้วได้ลบ นั่นคือผิดจริง ต้องรายงาน

ข้อ b บวกเลขคนละเครื่องหมาย จึง **เป็นไปไม่ได้ที่จะล้น** ไม่ว่าผลจะออกมาเท่าไร

คำตอบ

a. 00010110 + 00110011 = **01001001_2** = +73 · ไม่ล้น

b. 01110000 + 10101111 = **00011111_2** = +31 · ทิ้งตัวทดที่ล้นออกไป 1 ตัว · ไม่ล้น

จุดที่มักพลาดข้อนี้

เก็บตัวทดตัวที่ 9 ไว้ในคำตอบ — ข้อ b ต้องตอบ **00011111** แปดบิต ไม่ใช่ 100011111 เก้าบิต (ซึ่งจะกลายเป็น 287)

ข้อ 3 · ลบเลขแบบ 2's complement

โจทย์ Assignment #2 · Q3

> **Perform each subtraction in the 2's complement form:**
> a. 00110011 – 00010000 b. 01100101 - 11101000

ไทย: ลบเลข 8 บิตสองคู่นี้ โดยใช้วิธีของ 2's complement

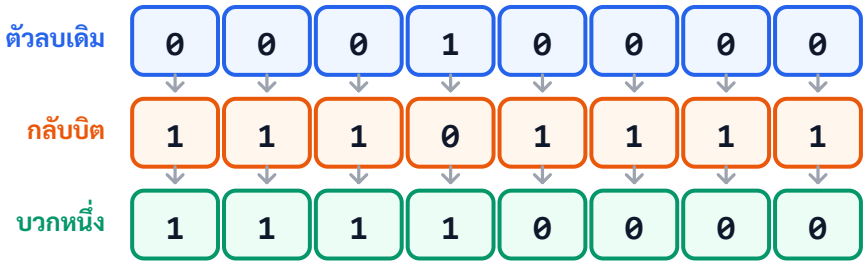
วิธีที่ใช้: เปลี่ยนการลบให้เป็นการบวก — ทำ 2's complement ของตัวลบ แล้วบวก จากนั้นทิ้งตัวทดที่ล้น · **สอนไว้ในบทที่ 1 · ตัวเลขในเครื่อง**

เครื่องจริงไม่มีวงจรลบ มีแต่วงจรบวก เพราะฉะนั้นมันแก้ปัญหาด้วยการเปลี่ยน A − B ให้เป็น A + (−B) แล้วใช้วงจรบวกตัวเดิม

ขั้นตอนมีแค่สามขั้น ① **ทำ 2's complement ของตัวหลัง** ② **บวก** ③ **ทิ้งตัวทดที่ล้น** — ตัวหน้าไม่ต้องแตะเลย

a. 00110011 – 00010000

ขั้นที่ 1 ทำ 2's complement ของตัวลบ 00010000



ได้ 11110000 ซึ่งแทนค่า −16

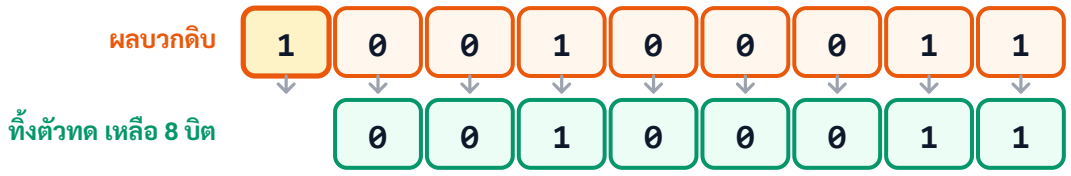
เช็กว่าถูก: 11110000 บิตซ้ายสุดเป็น 1 แปลว่าลบ · กลับบิตได้ 00001111 บวกหนึ่งได้ 00010000 = 16 เพราะฉะนั้น 11110000 คือ **−16** ถูกต้อง

ขั้นที่ 2 เปลี่ยนโจทย์เป็นการบวก 00110011 + 11110000

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
  1 1 1 1
    0 0 1 1 0 0 1 1
+   1 1 1 1 0 0 0 0
  ---------------
  1 0 0 1 0 0 0 1 1
```

ขั้นที่ 3 ผลบวกดิบยาว 9 บิต มีตัวทดล้นออกมาหนึ่งตัว ให้ทิ้ง



คำตอบคือแปดบิตล่าง

$$00110011 - 00010000 = \mathbf{00100011}_2$$

เช็กด้วยฐานสิบ: 51 − 16 = 35 และ $00100011_2 = 32 + 2 + 1 = 35$ ตรงกัน

เช็กอีกทาง: ลบตรง ๆ แบบข้อ 13 ของ Assignment 1 ก็ได้ 00110011 − 00010000 = 00100011 เหมือนกัน

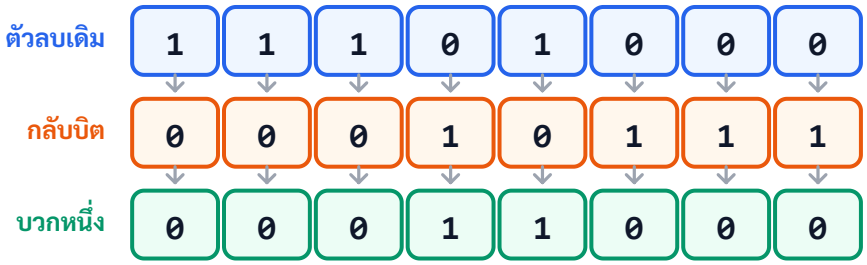
b. 01100101 − 11101000

อ่านโจทย์ก่อน ตัวหน้า 01100101 บิตซ้ายสุดเป็น 0 = บวก มีค่า 64 + 32 + 4 + 1 = **101**

ตัวหลัง 11101000 บิตซ้ายสุดเป็น 1 = **ลบ** — กลับบิตได้ 00010111 บวกหนึ่งได้ 00011000 = 16 + 8 = 24 เพราะฉะนั้นตัวหลังคือ **−24**

โจทย์จึงแปลว่า 101 − (−24) ซึ่งเท่ากับ 101 + 24 = 125 — จำตัวเลขนี้ไว้เช็กตอนท้าย

ขั้นที่ 1 ทำ 2's complement ของตัวลบ 11101000



ได้ 00011000 ซึ่งแทนค่า +24 — ลบของลบกลายเป็นบวก

ขั้นที่ 2 บวก 01100101 + 00011000

บวกแบบตั้งหลัก · ตัวเลขสีแดงข้างบนคือตัวทด

```
  0 1 1 0 0 1 0 1
+ 0 0 0 1 1 0 0 0
-----------------
  0 1 1 1 1 1 0 1
```

ขั้นที่ 3 คราวนี้ผลบวกยาว 8 บิตพอดี **ไม่มีตัวทดล้นออกมา** จึงไม่ต้องทิ้งอะไร

$$01100101 - 11101000 = \mathbf{01111101}_2$$

เช็กด้วยฐานสิบ: 101 − (−24) = 125 และ $01111101_2 = 64 + 32 + 16 + 8 + 4 + 1 = 125$ ตรงกัน

เช็กเรื่องล้น: บิตซ้ายสุดของผลลัพธ์เป็น 0 = บวก ซึ่งถูกต้อง และ 125 ยังไม่เกิน +127 จึง **ไม่ล้น** (ถ้าคำตอบเกิน 127 เมื่อไร บิตซ้ายสุดจะกลายเป็น 1 แล้วอ่านเป็นค่าลบทันที นั่นคือสัญญาณว่าล้น)

คำตอบ

a. 00110011 − 00010000 = $\mathbf{00100011_2}$ = +35

b. 01100101 − 11101000 = $\mathbf{01111101_2}$ = +125

จุดที่มักพลาดข้อนี้

ทำ 2’s complement ผิดตัว — ต้องทำกับ **ตัวหลัง (ตัวลบ)** เท่านั้น ถ้าเผลอไปทำกับตัวหน้าด้วย ข้อ b จะได้เครื่องหมายกลับข้างทั้งหมด

ข้อ 4 · BCD — เลขฐานสิบที่เก็บทีละหลัก

โจทย์ Assignment #2 · Q4

> **Convert the following numbers using binary-coded decimals:**
> a. 011100001001.0010_{BCD} ⇒ decimal numbers b. 8013_{10} ⇒ BCD

ไทย: a อ่านรหัส BCD ที่ให้มาเป็นเลขฐานสิบ · b เขียนเลข 8013 ให้อยู่ในรูป BCD

> **วิธีที่ใช้:** BCD เก็บเลขฐานสิบ **ทีละหลัก หลักละ 4 บิต** — แปลงทีละกลุ่มสี่ ไม่ต้องคิดรวมทั้งก้อน · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

เรื่องที่สับสนที่สุดของข้อนี้คือ BCD **ไม่ใช่** เลขฐานสองธรรมดา มันเป็นแค่ “วิธีเขียนเลขฐานสิบด้วยบิต” เลข 8013 ในรูป BCD จึงยาวกว่า 8013 ในรูปฐานสองแท้ ๆ

และเพราะแต่ละกลุ่มแทนเลขฐานสิบหนึ่งหลัก ค่าของกลุ่มจึง **ต้องไม่เกิน 9** เสมอ กลุ่มอย่าง 1010 ถึง 1111 เป็นรหัสที่ใช้ไม่ได้ใน BCD

a. 011100001001.0010_{BCD} ⇒ เลขฐานสิบ

แบ่งกลุ่มละ 4 จากจุดออกไปทั้งสองทาง ฝั่งซ้ายมี 12 บิตได้ 3 กลุ่ม ฝั่งขวามี 4 บิตได้ 1 กลุ่ม



ส่วนหลังจุด



กลุ่มที่	บิต 4 ตัว	คิดเป็นเลขฐานสิบ	ไม่เกิน 9 ใช่ไหม
จำนวนเต็มที่ 1	`0111`	**7**	ใช่ (ถูกต้อง)
จำนวนเต็มที่ 2	`0000`	**0**	ใช่ (ถูกต้อง)
จำนวนเต็มที่ 3	`1001`	**9**	ใช่ (ถูกต้อง)
หลังจุด	`0010`	**2**	ใช่ (ถูกต้อง)

ประกอบกลับตามตำแหน่งเดิม สามหลักหน้าจุดคือ 7, 0, 9 และหนึ่งหลักหลังจุดคือ 2

$$011100001001.0010_{BCD} = \mathbf{709.2}_{10}$$

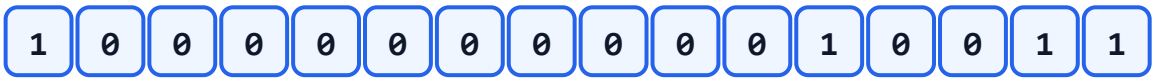
เช็กกลับ: เขียน 709.2 กลับเป็น BCD — 7 = 0111, 0 = 0000, 9 = 1001, 2 = 0010 ต่อกันได้ 011100001001.0010 ตรงกับโจทย์ทุกบิต

b. 8013_{10} ⇒ BCD

แยกเลขออกเป็นหลัก ๆ ก่อน 8 | 0 | 1 | 3 แล้วแปลงทีละหลักให้ได้หลักละ 4 บิตเป๊ะ

หลักฐานสิบ	เขียนเป็น 4 บิต	เช็กกลับ
8	`1000`	8 = 8
0	`0000`	0 = 0
1	`0001`	1 = 1
3	`0011`	2 + 1 = 3

เอามาต่อกันตามลำดับเดิม



8 = 1000 · 0 = 0000 · 1 = 0001 · 3 = 0011

$$8013_{10} = \mathbf{1000\ 0000\ 0001\ 0011}_{BCD}$$

เทียบกับฐานสองแท้ ๆ ให้เห็นความต่าง

BCD ของ 8013 = 1000 0000 0001 0011 — ยาว **16 บิต** อ่านหลักฐานสิบออกได้ทันที

ฐานสองแท้ของ 8013 = 1111101001101 — ยาว **13 บิต** สั้นกว่า แต่ต้องคำนวณก่อนถึงจะรู้ว่าเป็นเลขอะไร

BCD ยอมเปลืองบิตเพื่อแลกกับความง่ายตอนเอาไปขึ้นจอเจ็ดส่วน นี่คือเหตุผลที่มันมีอยู่

คำตอบ

a. $011100001001.0010_{BCD} = \mathbf{709.2_{10}}$

b. $8013_{10} = \mathbf{1000\ 0000\ 0001\ 0011_{(BCD)}}$

จุดที่มักพลาดข้อนี้

อ่าน BCD เป็นเลขฐานสองธรรมดาแล้วบวกน้ำหนักรวมทั้งก้อน — ข้อ a ต้องอ่าน **ทีละกลุ่มสี่** ได้ 709.2 ไม่ใช่เอา 011100001001 ไปคิดเป็นเลขฐานสองตัวเดียว

ข้อ 5 · แปลงเลขฐานสองเป็นรหัส Gray

โจทย์ Assignment #2 · Q5

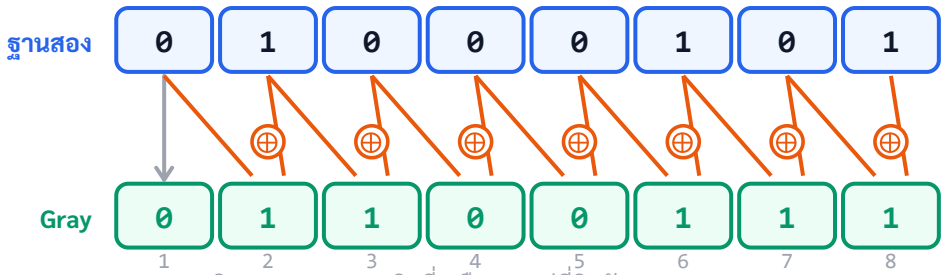
> **Convert the following binary number to Gray code:**
> a. 01000101_2 ⇒ Gray b. 01110110_2 ⇒ Gray

ไทย: แปลงเลขฐานสองสองจำนวนนี้ให้เป็นรหัส Gray

วิธีที่ใช้: บิตซ้ายสุด **ลอกลงมาตรง ๆ** · บิตถัดไปได้จากการ XOR บิตฐานสองสองตัวที่ติดกัน · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

XOR อ่านง่าย ๆ ว่า **“เหมือนกันได้ 0 ต่างกันได้ 1”** จำแค่นี้พอ ไม่ต้องท่องตาราง

a. 01000101_2 ⇒ Gray



บิตแรกลอกลงมา · บิตที่เหลือ XOR คู่ที่ติดกันของแถวบน

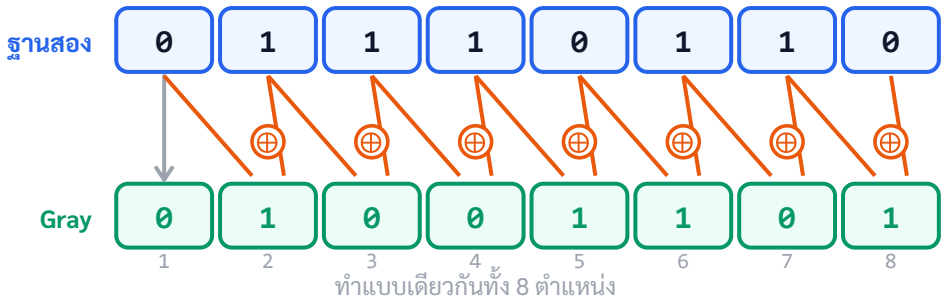
บิตที่ (นับจากซ้าย)	คิดจากอะไร	คิดยังไง	ได้
1	บิตที่ 1 ของฐานสอง	ลอกลงมาตรง ๆ ไม่ต้อง XOR	**0**
2	บิตที่ 1 กับบิตที่ 2	0 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
3	บิตที่ 2 กับบิตที่ 3	1 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
4	บิตที่ 3 กับบิตที่ 4	0 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**0**
5	บิตที่ 4 กับบิตที่ 5	0 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**0**
6	บิตที่ 5 กับบิตที่ 6	0 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
7	บิตที่ 6 กับบิตที่ 7	1 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**

บิตที่ (นับจากซ้าย)	คิดจากอะไร	คิดยังไง	ได้
8	บิตที่ 7 กับบิตที่ 8	0 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**

$$01000101_2 = \mathbf{01100111}_{\text{Gray}}$$

เช็กกลับ: แปลง Gray กลับเป็นฐานสอง (วิธีของข้อ 6) — 0 · 0⊕1=1 · 1⊕1=0 · 0⊕0=0 · 0⊕0=0 · 0⊕1=1 · 1⊕1=0 · 0⊕1=1 ได้ **01000101** ตรงกับโจทย์

b. 01110110_2 ⇒ Gray



ทำแบบเดียวกันทั้ง 8 ตำแหน่ง

บิตที่ (นับจากซ้าย)	คิดจากอะไร	คิดยังไง	ได้
1	บิตที่ 1 ของฐานสอง	ลอกลงมาตรง ๆ ไม่ต้อง XOR	**0**
2	บิตที่ 1 กับบิตที่ 2	0 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
3	บิตที่ 2 กับบิตที่ 3	1 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**0**
4	บิตที่ 3 กับบิตที่ 4	1 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**0**
5	บิตที่ 4 กับบิตที่ 5	1 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
6	บิตที่ 5 กับบิตที่ 6	0 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**
7	บิตที่ 6 กับบิตที่ 7	1 ⊕ 1 (เหมือนกันได้ 0 ต่างกันได้ 1)	**0**
8	บิตที่ 7 กับบิตที่ 8	1 ⊕ 0 (เหมือนกันได้ 0 ต่างกันได้ 1)	**1**

$$01110110_2 = \mathbf{01001101}_{\text{Gray}}$$

เช็กกลับด้วยวิธีเดียวกับข้อ a จะได้ 01110110 คืนมา

ทำไมต้องมีรหัสนี้

รหัส Gray ออกแบบให้เลขที่ติดกัน **ต่างกันแค่บิตเดียว** เสมอ ตัวอย่างเช่นจาก 3 ไป 4 ในฐานสองคือ 011 → 100 ซึ่งเปลี่ยนสามบิตพร้อมกัน ถ้าเซนเซอร์อ่านไม่พร้อมกันจะได้ค่ามั่ว แต่ในรหัส Gray คือ 010 → 110 เปลี่ยนบิตเดียว อ่านผิดก็เพี้ยนแค่หนึ่งขั้น

คำตอบ

a. 01000101_2 = **`01100111₍Gray₎`**

b. 01110110_2 = **`01001101₍Gray₎`**

จุดที่มักพลาดข้อนี้

ไป XOR กับบิตของ ผลลัพธ์ แทนที่จะเป็นบิตของ ตัวตั้ง — ขาไป (ฐานสอง→Gray) ใช้บิตแถวบนสองตัวที่ติดกันเสมอ ส่วนการใช้ผลลัพธ์เป็นเรื่องของขากลับในข้อ 6

ข้อ 6 · แปลงรหัส Gray กลับเป็นเลขฐานสอง

โจทย์ Assignment #2 · Q6

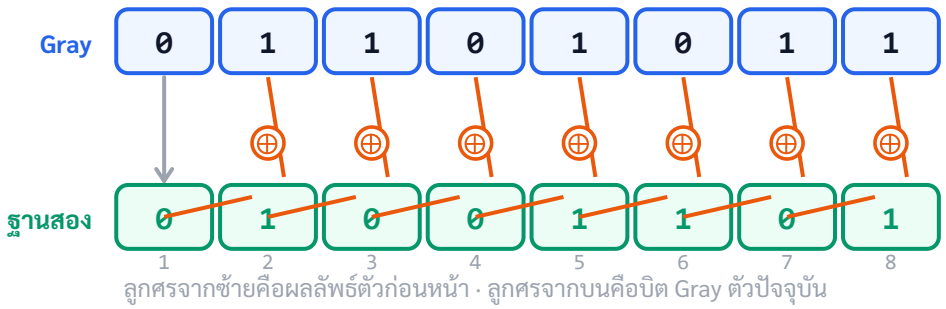
> **Convert the following Gray code to binary numbers:**
> a. 01101011_{Gray} ⇒ base 2 b. 11011111_{Gray} ⇒ base 2

ไทย: แปลงรหัส Gray สองชุดนี้กลับเป็นเลขฐานสอง

> **วิธีที่ใช้:** บิตซ้ายสุดลอกลงมา · บิตถัดไป = **บิตผลลัพธ์ตัวก่อนหน้า** XOR **บิต Gray ตัวปัจจุบัน** · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

ขากลับต่างจากขาไปตรงเดียว — ขาไปใช้บิตของ **ตัวตั้ง** สองตัวที่ติดกัน แต่ขากลับใช้บิตของ **ผลลัพธ์ที่เพิ่งได้** มา XOR กับ Gray ตัวถัดไป เพราะฉะนั้นต้องทำเรียงจากซ้ายไปขวา ข้ามไม่ได้

a. 01101011_{Gray} ⇒ ฐาน 2



ลูกศรจากซ้ายคือผลลัพธ์ตัวก่อนหน้า · ลูกศรจากบนคือบิต Gray ตัวปัจจุบัน

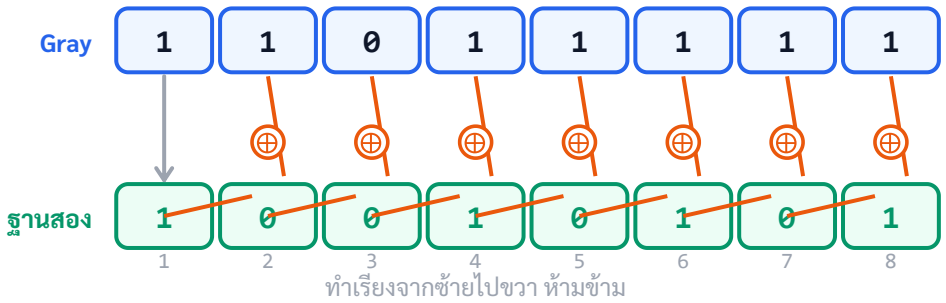
บิตที่	คิดจากอะไร	คิดยังไง	ได้บิตฐานสอง
1	บิตที่ 1 ของ Gray	ลอกลงมาตรง ๆ	**0**
2	ผลลัพธ์บิตที่ 1 กับ Gray บิตที่ 2	$0 \oplus 1$	**1**

บิตที่	คิดจากอะไร	คิดยังไง	ได้บิตฐานสอง
3	ผลลัพธ์บิตที่ 2 กับ Gray บิตที่ 3	$1 \oplus 1$	**0**
4	ผลลัพธ์บิตที่ 3 กับ Gray บิตที่ 4	$0 \oplus 0$	**0**
5	ผลลัพธ์บิตที่ 4 กับ Gray บิตที่ 5	$0 \oplus 1$	**1**
6	ผลลัพธ์บิตที่ 5 กับ Gray บิตที่ 6	$1 \oplus 0$	**1**
7	ผลลัพธ์บิตที่ 6 กับ Gray บิตที่ 7	$1 \oplus 1$	**0**
8	ผลลัพธ์บิตที่ 7 กับ Gray บิตที่ 8	$0 \oplus 1$	**1**

$$01101011_{\text{Gray}} = \mathbf{01001101}_{2}$$

เช็กกลับด้วยวิธีขาไปของข้อ 5: เอา 01001101 ไปทำ Gray — $0 \cdot 0 \oplus 1 = 1 \cdot 1 \oplus 0 = 1 \cdot 0 \oplus 0 = 0 \cdot 0 \oplus 1 = 1 \cdot 1 \oplus 1 = 0 \cdot 1 \oplus 0 = 1 \cdot 0 \oplus 1 = 1$ ได้ **01101011** ตรงกับโจทย์

b. 11011111_{Gray} ⇒ ฐาน 2



บิตที่	คิดจากอะไร	คิดยังไง	ได้บิตฐานสอง
1	บิตที่ 1 ของ Gray	ลอกลงมาตรง ๆ	**1**
2	ผลลัพธ์บิตที่ 1 กับ Gray บิตที่ 2	$1 \oplus 1$	**0**
3	ผลลัพธ์บิตที่ 2 กับ Gray บิตที่ 3	$0 \oplus 0$	**0**
4	ผลลัพธ์บิตที่ 3 กับ Gray บิตที่ 4	$0 \oplus 1$	**1**
5	ผลลัพธ์บิตที่ 4 กับ Gray บิตที่ 5	$1 \oplus 1$	**0**
6	ผลลัพธ์บิตที่ 5 กับ Gray บิตที่ 6	$0 \oplus 1$	**1**
7	ผลลัพธ์บิตที่ 6 กับ Gray บิตที่ 7	$1 \oplus 1$	**0**
8	ผลลัพธ์บิตที่ 7 กับ Gray บิตที่ 8	$0 \oplus 1$	**1**

$$11011111_{\text{Gray}} = \mathbf{10010101}_{2}$$

เช็กกลับ: เอา 10010101 ไปทำ Gray ได้ 11011111 ตรงกับโจทย์

เช็กเร็วในห้องสอบ

นับจำนวนเลข 1 ใน Gray จากซ้ายมาถึงตำแหน่งที่สนใจ ถ้านับได้ **จำนวนคี่** บิตฐานสองตำแหน่งนั้นเป็น 1 ถ้าได้ **จำนวนคู่** เป็น 0

ลองข้อ b ตำแหน่งที่ 4: Gray สี่ตัวแรกคือ 1101 มีเลข 1 อยู่ 3 ตัว เป็นจำนวนคี่ บิตฐานสองตำแหน่งที่ 4 จึงเป็น **1** — ตรงกับบิตที่ 4 ของคำตอบ 10010101 พอดี

คำตอบ

a. $01101011_{(\text{Gray})} = \mathbf{01001101_2}$

b. $11011111_{(\text{Gray})} = \mathbf{10010101_2}$

จุดที่มักพลาดข้อนี้

ใช้บิต Gray สองตัวที่ติดกันมา XOR กัน (คือทำแบบขาไป) — ขากลับต้องใช้ **ผลลัพธ์ตัวก่อนหน้า** เสมอ ถ้าทำผิดทาง ข้อ a จะได้ 01011110 ซึ่งไม่ใช่คำตอบ

ข้อ 7 · ใส่บิตพาริตีคู่

โจทย์ Assignment #2 · Q7

> **Assign the proper even parity bit to the following code groups:**
> a. 101101_2 b. 1000111001001_2

ไทย: เติมบิตพาริตีแบบ **คู่** ให้รหัสสองชุดนี้ — คือเติมบิตหนึ่งตัวเพื่อให้จำนวนเลข 1 ทั้งหมด (รวมบิตที่เติม) เป็นจำนวนคู่

> **วิธีที่ใช้:** นับจำนวนเลข 1 ในข้อมูล ถ้าได้จำนวน **คู่** ใส่บิตพาริตี 0 ถ้าได้ **คี่** ใส่ 1 · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

บิตพาริตีไม่ได้เก็บข้อมูลอะไรเลย มันมีหน้าที่เดียวคือ **ทำให้จำนวนเลข 1 ออกมาเป็นคู่** ตามที่ตกลงกันไว้ ถ้าปลายทางนับแล้วไม่เป็นคู่ แปลว่ามีบิตพลิกระหว่างทาง

ตามสไลด์ของวิชานี้ บิตพาริตีจะ **เติมไว้ข้างหน้า** (ซ้ายสุด) ของรหัส

a. 101101_2

นับเลข 1 ทีละตำแหน่ง ไม่นับข้ามตัว

ตำแหน่ง	1	2	3	4	5	6
บิต	1	0	1	1	0	1
นับสะสม	1	1	2	3	3	4

ได้ทั้งหมด **4 ตัว** อยู่ที่ตำแหน่ง 1 และ 3 และ 4 และ 6 ซึ่งเป็นจำนวน **คู่** อยู่แล้ว

ระบบพาริตีคู่ต้องการให้จำนวนเลข 1 เป็นคู่ ตอนนี้มันคู่อยู่แล้ว จึงไม่ต้องเพิ่มอะไร ใส่บิตพาริตีเป็น **0**



บิตสีส้มตัวแรกคือบิตพาริตีที่เติมเข้าไป

$$\text{บิตพาริตี} = 0 \cdot \text{รหัสที่ส่งออกไป} = 0101101$$

เช็กกลับ: นับเลข 1 ในรหัส 7 บิตที่ส่ง ได้ 4 ตัว เป็นจำนวนคู่ ถูกต้อง

b. 1000111001001_2

ชุดนี้ยาว 13 บิต นับให้ครบทุกตำแหน่ง

ตำแหน่ง	1	2	3	4	5	6	7	8	9	10	11	12	13
บิต	1	0	0	0	1	1	1	0	0	1	0	0	1
นับสะสม	1	1	1	1	2	3	4	4	4	5	5	5	6

ได้ทั้งหมด **6 ตัว** อยู่ที่ตำแหน่ง 1, 5, 6, 7, 10, 13 ซึ่งเป็นจำนวน **คู่**

เพราะฉะนั้นบิตพาริตีเป็น **0** เหมือนข้อ a



รหัสที่ส่งยาว 14 บิต

$$\text{บิตพาริตี} = 0 \cdot \text{รหัสที่ส่งออกไป} = 01000111001001$$

เช็กกลับ: นับเลข 1 ในรหัส 14 บิตที่ส่ง ได้ 6 ตัว เป็นจำนวนคู่ ถูกต้อง

ถ้าโจทย์เปลี่ยนเป็นพาริตีคี่

คำตอบจะกลับกันทันที — ทั้งสองข้อมีเลข 1 เป็นจำนวนคู่อยู่แล้ว ถ้าต้องทำให้เป็น **คี่** ก็ต้องเติมบิตพาริตีเป็น **1** ทั้งคู่

คำตอบ

a. บิตพาริตีคู่ = `0` · รหัสที่ส่ง = `0101101`

b. บิตพาริตีคู่ = `0` · รหัสที่ส่ง = `01000111001001`

จุดที่มักพลาดข้อนี้

นับเลข 1 ตกหล่นในชุดที่ยาว — ข้อ b มี 13 บิต ให้ไล่นับทีละตัวแล้วจดเลขสะสมไว้ อย่านับด้วยสายตารวดเดียว

ข้อ 8 · ระบบพาริตีคี่ — ชุดไหนมีบิตผิด

โจทย์ Assignment #2 · Q8

> **An odd parity system receives the following code groups. Determine which groups (if any) contain a single bit error:**
> a. 110011_2 b. 1100010101010_2

ไทย: ระบบนี้ใช้พาริตี **คี่** และได้รับรหัสสองชุดนี้มา ให้บอกว่าชุดไหน (ถ้ามี) มีบิตผิดหนึ่งบิต

วิธีที่ใช้: นับเลข 1 ทั้งชุด (รวมบิตพาริตีที่ติดมาด้วย) ถ้าได้ **คี่** = ผ่าน ถ้าได้ **คู่** = มีบิตผิด · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

ข้อนี้กลับทิศกับข้อ 7 — ข้อ 7 เราเป็นฝ่ายส่ง ต้องคิดบิตพาริตีขึ้นมา ข้อ 8 เราเป็นฝ่ายรับ ได้รหัสมาทั้งชุดแล้ว หน้าที่เดียวคือ **นับแล้วตัดสิน**

และเพราะระบบนี้ตกลงกันไว้ว่าใช้พาริตี **คี่** รหัสที่ถูกต้องจะต้องมีเลข 1 เป็นจำนวนคี่เสมอ

a. 110011_2

ตำแหน่ง	1	2	3	4	5	6
บิต	1	1	0	0	1	1
นับสะสม	1	2	2	2	3	4

นับได้ **4 ตัว** ซึ่งเป็นจำนวน **คู่**

แต่ระบบนี้ต้องการจำนวน **คี่** เพราะฉะนั้นชุดนี้ **ผิด**



นับเลข 1 ได้ 4 ตัว = คู่ → ขัดกับข้อตกลงพาริตีคี่

a. มีเลข 1 จำนวน 4 ตัว (คู่) → **มีบิตผิด**

b. 1100010101010_2

ตำแหน่ง	1	2	3	4	5	6	7	8	9	10	11	12	13
บิต	1	1	0	0	0	1	0	1	0	1	0	1	0
นับสะสม	1	2	2	2	2	3	3	4	4	5	5	6	6

นับได้ **6 ตัว** ซึ่งเป็นจำนวน **คู่** อีกเหมือนกัน



นับเลข 1 ได้ 6 ตัว = คู่ → ขัดกับข้อตกลงพาริตีคี่

b. มีเลข 1 จำนวน 6 ตัว (คู่) → **มีบิตผิด**

พาริตีบอกได้แค่ไหน

พาริตีบอกได้ว่า **"มีบิตผิด"** เท่านั้น มัน บอกไม่ได้ ว่าผิดที่ตำแหน่งไหน และแก้ให้ไม่ได้ด้วย

ยิ่งกว่านั้น ถ้าพลิก **สองบิต** พร้อมกัน จำนวนเลข 1 จะกลับมาเป็นคู่/คี่เหมือนเดิม พาริตีจะจับไม่ได้เลย — นี่คือเหตุผลที่ต้องมี CRC (ข้อ 9) และ Hamming (ข้อ 10) ตามมา

คำตอบ

a. 110011 — เลข 1 มี 4 ตัว เป็นจำนวนคู่ ขัดกับพาริตีคี่ → **มีบิตผิด**

b. 1100010101010 — เลข 1 มี 6 ตัว เป็นจำนวนคู่ ขัดกับพาริตีคี่ → **มีบิตผิด**

สรุป: **ผิดทั้งสองชุด** แต่บอกไม่ได้ว่าบิตไหนผิด

จุดที่มักพลาดข้อนี้

เผลอตัดสินด้วยเกณฑ์พาริตี **คู่** เพราะเพิ่งทำข้อ 7 มา — ข้อนี้โจทย์บอกชัดว่าเป็น `odd parity system` จำนวนคู่คือผิด ไม่ใช่ถูก

ข้อ 9 · CRC — หาบิตตรวจสอบ แล้วเช็กฝั่งผู้รับ

โจทย์ Assignment #2 · Q9

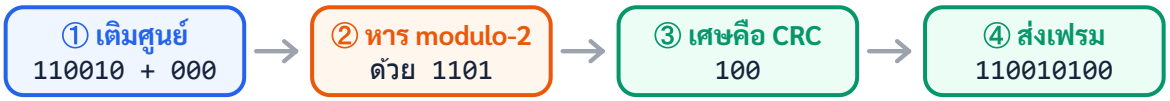
> **Determine the transmitted CRC for the following data (D) 110010 and generator code (G) 1101. Also verify that the received data 110011100 at the receiver has an error or not.**

ไทย: ข้อมูล D = 110010 ตัวสร้าง G = 1101 ให้หา CRC ที่ส่งออกไป แล้วตรวจว่าเฟรมที่ผู้รับได้ คือ 110011100 มีข้อผิดพลาดหรือไม่

วิธีที่ใช้: หารแบบ modulo-2 (คือ **XOR ไม่มีการยืม**) สี่ขั้นตามสไลด์ · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

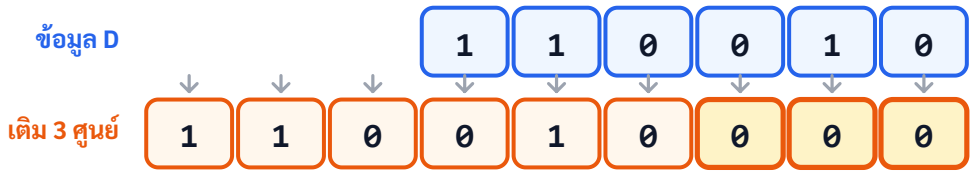
CRC ทำงานด้วยความคิดเดียว — **ผู้ส่งจงใจแต่งเฟรมให้หารด้วย G ลงตัวพอดี** ถ้าปลายทางหารแล้วยังลงตัว ก็แปลว่าบิตน่าจะรอดมาครบ ถ้าเหลือเศษ แปลว่ามีอะไรเพี้ยนระหว่างทาง

คำว่าหารแบบ modulo-2 ฟังดูยาก แต่ทำจริงง่ายกว่าหารปกติ เพราะ **ไม่มีการยืม ไม่มีการเทียบว่าใครมากกว่าใคร** ดูแค่บิตซ้ายสุดของหน้าต่าง ถ้าเป็น 1 ก็ XOR ด้วย G ถ้าเป็น 0 ก็ข้าม



ขั้นที่ 1 · เติมศูนย์ต่อท้ายข้อมูล

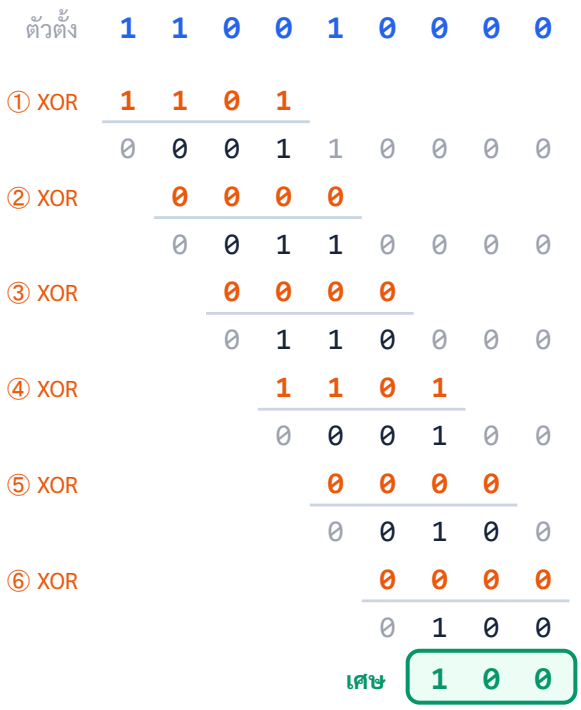
G มี 4 บิต เพราะฉะนั้นเติมศูนย์ **r = 4 − 1 = 3 ตัว**



ได้ตัวตั้ง 9 บิตสำหรับเอาไปหาร

ขั้นที่ 2 · หารด้วย G = 1101 แบบ modulo-2

ทุกแถวคือการ XOR หนึ่งครั้ง ทำไล่จากซ้ายไปขวา ตัวเลขสีส้มคือ G ที่เอามา XOR ตัวเลขสีเทาคือบิตที่ยังไม่ได้แตะ

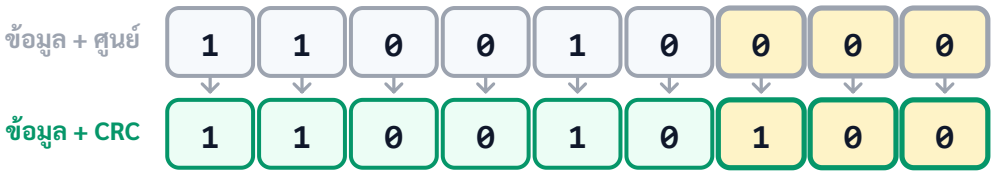


รอบที่	บิตซ้ายสุดของหน้าต่าง	เอาอะไรมา XOR	หน้าต่างก่อน	หลัง XOR
1	1	1101	1100	0001
2	0	0000 (ข้ามรอบนี้)	0011	0011
3	0	0000 (ข้ามรอบนี้)	0110	0110
4	1	1101	1100	0001
5	0	0000 (ข้ามรอบนี้)	0010	0010
6	0	0000 (ข้ามรอบนี้)	0100	0100

ทำครบ 6 รอบ (เท่ากับจำนวนบิตของข้อมูล) เศษที่เหลือคือ **3 บิตขวาสุด**

CRC = **100**

ขั้นที่ 3 · เอา CRC ไปแทนที่ศูนย์ที่เติมไว้



นี่คือเฟรมที่ส่งออกไปจริง

เฟรมที่ส่ง = **110010100**

เช็กกลับ: เอา 110010100 หารด้วย 1101 ต้องได้เศษ **000** พอดี เพราะผู้ส่งแต่งไว้ให้ลงตัว — ลองหารดูได้เศษ **000** จริง

ขั้นที่ 4 · ฝั่งผู้รับ ตรวจเฟรม 110011100

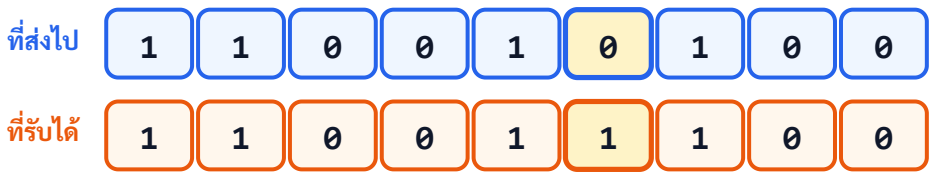
ผู้รับไม่ต้องรู้อะไรเลยนอกจาก G ทำอย่างเดียวคือหารทั้งเฟรมด้วย G แล้วดูเศษ



รอบที่	บิตซ้ายสุดของหน้าต่าง	เอาอะไรมา XOR	หน้าต่างก่อน	หลัง XOR
1	1	1101	1100	0001
2	0	0000 (ข้ามรอบนี้)	0011	0011
3	0	0000 (ข้ามรอบนี้)	0111	0111
4	1	1101	1111	0010
5	0	0000 (ข้ามรอบนี้)	0100	0100
6	1	1101	1000	0101

เศษที่ผู้รับได้ = **101** ซึ่ง **ไม่ใช่ 000**

เศษไม่เป็นศูนย์ แปลว่า **เฟรมนี้มีข้อผิดพลาด** ผู้รับต้องขอให้ส่งใหม่



เทียบกันแล้วต่างกันที่บิตที่ 6 นับจากซ้าย — 0 กลายเป็น 1

เทียบบิตต่อบิตจะเห็นว่าต่างกันจริงที่ตำแหน่งที่ 6 ซึ่งยืนยันว่าการตรวจด้วย CRC ให้คำตอบถูกต้อง

CRC ทำอะไรได้ ทำอะไรไม่ได้

CRC **ตรวจจับ** ข้อผิดพลาดได้ดีมาก โดยเฉพาะบิตที่พลิกติดกันเป็นช่วง (`burst error`)

แต่มัน **ไม่บอกตำแหน่ง** และ **แก้ไม่ได้** — ถ้าอยากได้ถึงขั้นแก้ ต้องใช้ Hamming code ในข้อ 10

คำตอบ

CRC ที่คำนวณได้ = `100`

เฟรมที่ส่งออกไป = `110010100`

ตรวจเฟรมที่รับมา 110011100 → เศษ = `101 ≠ 000` → **มีข้อผิดพลาด**

จุดที่มักพลาดข้อนี้

เติมศูนย์ผิดจำนวน — ต้องเติมเท่ากับ **จำนวนบิตของ G ลบหนึ่ง** ข้อนี้ G มี 4 บิตจึงเติม 3 ตัว (ไม่ใช่ 4) และตอนหารให้ดูแค่ **บิตซ้ายสุดของหน้าต่าง** ห้ามเทียบว่าตัวไหนมากกว่ากันแบบหารปกติ

ข้อ 10 · Hamming code — เข้ารหัส แล้วหาบิตที่ผิดให้เจอ

โจทย์ Assignment #2 · Q10

> **Encode 7-bit data 1011001 in Hamming code using even parity. Also verify that the received data 10101101110 at the receiver has an error or not.**

ไทย: เข้ารหัสข้อมูล 7 บิต 1011001 ด้วย Hamming code แบบพาริตีคู่ แล้วตรวจว่าข้อมูลที่ผู้รับได้ คือ 10101101110 มีข้อผิดพลาดหรือไม่

วิธีที่ใช้: หาจำนวนบิตพาริตีจาก $2^p \geq d + p + 1$ · วางบิตพาริตีไว้ที่ตำแหน่งกำลังสอง · พาริตีแต่ละตัวคุมกลุ่มที่ซ้อนทับกัน · **สอนไว้ใน บทที่ 2 · รหัสและการตรวจจับข้อผิดพลาด**

พาริตีธรรมดาในข้อ 8 บอกได้แค่ว่า “มีอะไรผิด” · Hamming เพิ่มบิตพาริตีหลายตัว แต่ละตัวคุม **คนละกลุ่มที่ซ้อนทับกัน** พอมีบิตพลิก ชุดของตัวตรวจที่ไม่ผ่านจะชี้ตรงไปที่ **เลขตำแหน่ง** ของบิตนั้นพอดี

ขั้นที่ 1 · ต้องใช้บิตพาริตีกี่ตัว

ข้อมูลมี d = 7 บิต ลองแทนค่า p ไปทีละตัวจนเงื่อนไขเป็นจริง

ลอง p	2^p	d + p + 1	พอไหม
3	8	7 + 3 + 1 = 11	8 < 11 → **ไม่พอ**
4	16	7 + 4 + 1 = 12	16 ≥ 12 → **พอ**

ใช้บิตพาริตี **4 ตัว** รหัสรวมจึงยาว 7 + 4 = **11 บิต**

ขั้นที่ 2 · วางบิตลงตำแหน่ง

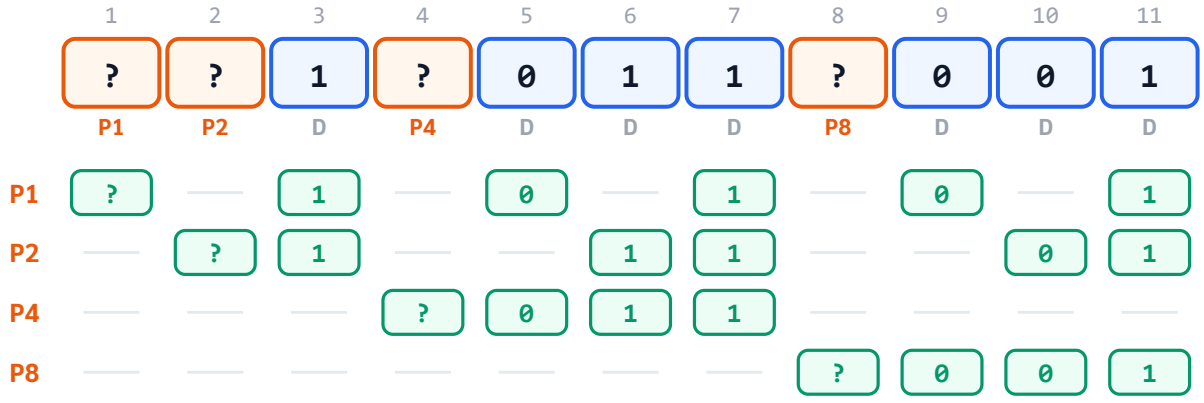
ตำแหน่งที่เป็นกำลังของสอง คือ **1, 2, 4, 8** สงวนไว้ให้บิตพาริตี ตำแหน่งที่เหลือ **3, 5, 6, 7, 9, 10, 11** ใส่ข้อมูลเรียงจากซ้ายไปขวา

ตำแหน่ง	1	2	3	4	5	6	7	8	9	10	11
เป็นช่องอะไร	P1	P2	D	P4	D	D	D	P8	D	D	D
ใส่อะไรลงไป	?	?	1	?	0	1	1	?	0	0	1

ข้อมูล 1011001 ถูกแจกลงเจ็ดช่องตามลำดับ — 1 ไปตำแหน่ง 3 · 0 ไปตำแหน่ง 5 · 1 ไปตำแหน่ง 6 · 1 ไปตำแหน่ง 7 · 0 ไปตำแหน่ง 9 · 0 ไปตำแหน่ง 10 · 1 ไปตำแหน่ง 11

ขั้นที่ 3 · ใครคุมใคร

กฎเดียว: **พาริตีที่ตำแหน่ง k คุมทุกตำแหน่งที่ต้องใช้ k ไปประกอบ** เช่น ตำแหน่ง 6 = 4 + 2 จึงถูก P4 และ P2 คุม ส่วนตำแหน่ง 7 = 4 + 2 + 1 ถูกคุมถึงสามตัว



ช่องสีเขียวคือตำแหน่งที่พาริตีแถวนั้นต้องเอามานับรวม

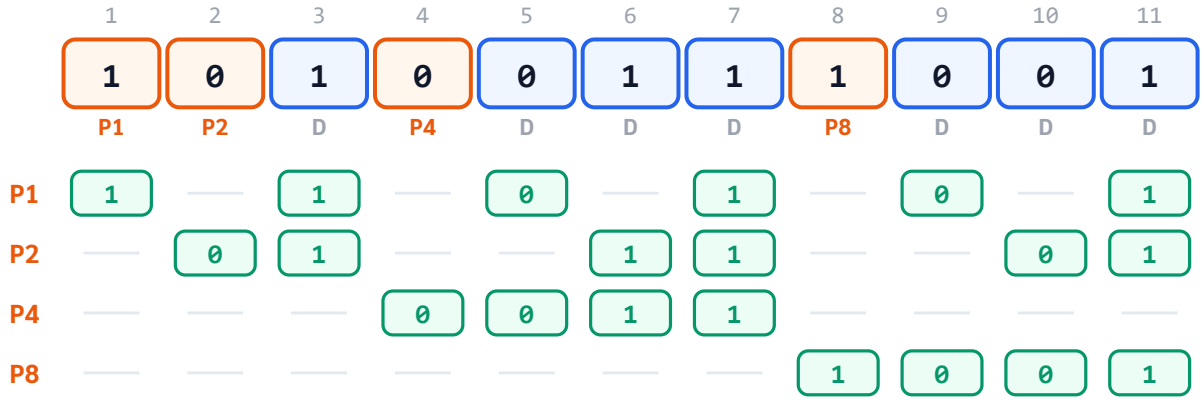
ขั้นที่ 4 · คิดค่าพาริตีทีละตัว (พาริตีคู่)

พาริตีคู่แปลว่าในกลุ่มที่มันคุม จำนวนเลข 1 ต้องออกมาเป็นคู่ เพราะฉะนั้นถ้านับบิตข้อมูลได้คี่ ก็ใส่ 1 ถ้าได้คู่ ก็ใส่ 0

พาริตี	คุมตำแหน่งไหนบ้าง	บิตข้อมูลที่มันคุม (ไม่รวมตัวมันเอง)	นับเลข 1 ได้	ต้องใส่ค่าอะไร
P1	1, 3, 5, 7, 9, 11	`1 0 1 0 1`	3 (คี่)	**1**
P2	2, 3, 6, 7, 10, 11	`1 1 1 0 1`	4 (คู่)	**0**
P4	4, 5, 6, 7	`0 1 1`	2 (คู่)	**0**

พาริตี	คุมตำแหน่งไหนบ้าง	บิตข้อมูลที่มันคุม (ไม่รวมตัวมันเอง)	นับเลข 1 ได้	ต้องใส่ค่าอะไร
P8	8, 9, 10, 11	**0 0 1**	1 (คี่)	**1**

ได้ P1 = 1 · P2 = 0 · P4 = 0 · P8 = 1 เติมลงช่องที่ว่างไว้



รหัสที่ส่งออกไป = 10100111001

รหัส Hamming ของ 1011001 = **10100111001**

เช็กกลับ: ลองตรวจรหัสที่เพิ่งสร้าง ทุกกลุ่มต้องมีเลข 1 เป็นจำนวนคู่

ตัวตรวจ	ดูตำแหน่ง	บิตที่อ่านได้ (รวมบิตพาริตีเอง)	นับเลข 1	ผลตรวจ
C1	1, 3, 5, 7, 9, 11	**1 1 0 1 0 1**	4 (คู่)	**0** ผ่าน
C2	2, 3, 6, 7, 10, 11	**0 1 1 1 0 1**	4 (คู่)	**0** ผ่าน
C4	4, 5, 6, 7	**0 0 1 1**	2 (คู่)	**0** ผ่าน
C8	8, 9, 10, 11	**1 0 0 1**	2 (คู่)	**0** ผ่าน

ผ่านครบทั้งสี่ตัว ซินโดรมเป็น 0000 แปลว่ารหัสที่สร้างถูกต้อง

ขั้นที่ 5 · ฝั่งผู้รับ ตรวจ 10101101110

ผู้รับเอารหัสที่ได้มาทั้ง 11 บิต มาตรวจด้วยกลุ่มเดิมทั้งสี่กลุ่ม คราวนี้ **นับรวมบิตพาริตีเข้าไปด้วย**



รหัสที่ผู้รับได้ พร้อมเลขตำแหน่งใต้แต่ละบิต

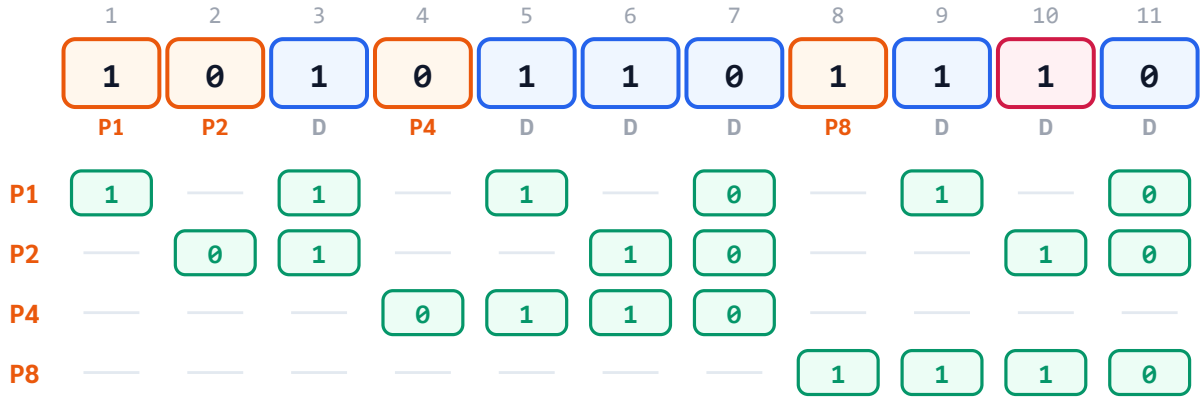
ตัวตรวจ	ดูตำแหน่ง	บิตที่อ่านได้ (รวมบิตพาริตีเอง)	นับเลข 1	ผลตรวจ
C1	1, 3, 5, 7, 9, 11	**1 1 1 0 1 0**	4 (คู่)	**0** ผ่าน
C2	2, 3, 6, 7, 10, 11	**0 1 1 0 1 0**	3 (คี่)	**1** ไม่ผ่าน
C4	4, 5, 6, 7	**0 1 1 0**	2 (คู่)	**0** ผ่าน

ตัวตรวจ	ดูตำแหน่ง	บิตที่อ่านได้ (รวมบิตพาริตีเอง)	นับเลข 1	ผลตรวจ
C8	8, 9, 10, 11	**1 1 1 0**	3 (คี่)	**1** ไม่ผ่าน

เอาผลตรวจมาเรียงเป็นเลขฐานสอง โดยให้ C8 อยู่ซ้ายสุด

$$\text{ซินโดรม} = C8\ C4\ C2\ C1 = 1010_2 = \mathbf{10}$$

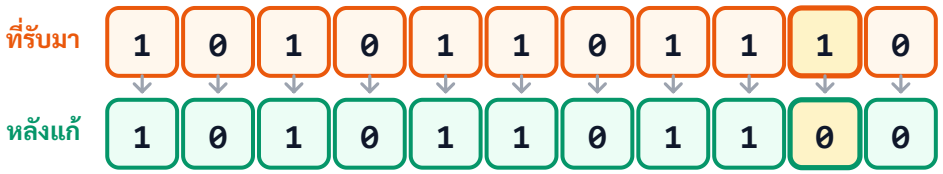
คิดค่า: 8 + 2 = 10 · ไม่ใช่ศูนย์ แปลว่ามีบิตผิด และเลขนี้คือ **ตำแหน่งของบิตที่ผิด**



ซินโดรม 1010 = 10 → บิตตำแหน่งที่ 10 คือตัวที่ผิด

ขั้นที่ 6 · แก้บิตนั้นแล้วอ่านข้อมูลกลับ

พลิกบิตตำแหน่งที่ 10 จาก 1 เป็น 0



แก้บิตเดียวตามที่ซินโดรมชี้

ตรวจซ้ำ: รหัสหลังแก้ให้ซินโดรม 0000 แปลว่าผ่านแล้ว · ดึงบิตข้อมูลจากตำแหน่ง 3, 5, 6, 7, 9, 10, 11 ได้ **1110100**

ข้อสังเกตที่ควรรู้ (กันงง)

รหัสที่ผู้รับได้ในโจทย์ (10101101110) **ไม่ใช่** รหัสที่เราเข้ารหัสไว้ตอนต้นข้อ (10100111001) ที่พลิกไปหนึ่งบิต — เทียบบิตต่อบิตแล้วต่างกันถึง 5 ตำแหน่ง คือตำแหน่งที่ 5, 7, 9, 10, 11

โจทย์ถามสองคำถามที่แยกกัน คำถามหลังคือ “เอาวิธีตรวจของ Hamming ไปใช้กับรหัสชุดนี้ แล้วได้ผลว่าอย่างไร” ซึ่งคำตอบคือ **ซินโดรม 1010 ชี้ไปที่ตำแหน่งที่ 10**

และนี่คือข้อจำกัดที่สไลด์เตือนไว้พอดี — Hamming **แก้ได้แค่บิตเดียว** ถ้าพลิกมากกว่าหนึ่งบิต ซินโดรมจะชี้ผิดที่ แล้วเราจะ “แก้” จนได้รหัสที่ถูกตามกติกา แต่เป็นคนละข้อมูลกับที่ส่งมา

คำตอบ

จำนวนบิตพาริตี = **4** (เพราะ $2^4 = 16 \geq 12$) · รหัสยาว **11 บิต**

P1 = 1 · P2 = 0 · P4 = 0 · P8 = 1

รหัส Hamming ของ 1011001 = **10100111001**

ตรวจรหัสที่รับมา 10101101110 → ซินโดรม = **1010_2 = 10 → มีข้อผิดพลาดที่บิตตำแหน่งที่ 10**

แก้แล้วได้ 10101101100 (ข้อมูล 1110100)

จุดที่มักพลาดข้อนี้

นับตำแหน่งจากขวา — Hamming ในวิชานี้นับ **ตำแหน่งที่ 1 คือบิตซ้ายสุด** ถ้านับกลับทาง ซินโดรมจะชี้ไปคนละบิต · และตอนเข้ารหัส พาริตีแต่ละตัว **ไม่นับตัวเอง** แต่ตอนตรวจฝั่งผู้รับ **ต้องนับตัวเองด้วย**

ทวนก่อนเข้าห้องสอบ · สิบบรรทัด

① กางหลัก = ตัวเลข × ฐาน^k · หลังจุดเริ่มที่ k = −1

② ฐานสิบ → ฐานอื่น: จำนวนเต็ม **หารเก็บเศษ อ่านขึ้น** · เศษ **คูณเก็บจำนวนเต็ม อ่านลง**

③ ฐานสอง ↔ ฐานสิบหก: จับกลุ่มละ 4 **จากขวา** เสมอ

④ 1’s complement = กลับบิต · 2’s complement = กลับบิตแล้วบวกหนึ่ง (ห้ามสลับลำดับ)

⑤ เลข 8 บิต 2’s complement มีพิสัย −128 ถึง +127 · อีกสองแบบ −127 ถึง +127

⑥ บวก/ลบแบบ 2’s complement: ลบ = บวกกับ 2’s complement ของตัวหลัง แล้ว **ทิ้งตัวทดที่ล้น**

⑦ BCD = หลักละ 4 บิต ค่าต้องไม่เกิน 9 · Gray ขาไปใช้บิตตัวตั้ง ขากลับใช้บิตผลลัพธ์

⑧ พาริตีบอกได้แค่ว่าผิด ไม่บอกตำแหน่ง · พลิกสองบิตพร้อมกันจับไม่ได้

⑨ CRC: เติมศูนย์ (บิตของ G ลบหนึ่ง) → หาร XOR → เศษคือ CRC · ผู้รับได้เศษ 0 = ผ่าน

⑩ Hamming: พาริตีอยู่ตำแหน่งกำลังสอง · ซินโดรมที่อ่านได้คือ **เลขตำแหน่งที่ผิด**

ภาคผนวก ข · เฉลย Assignment 3 และ 4 ทุกข้อ

ภาคผนวกนี้เป็น**เฉลยล้วน ๆ** ของใบงาน Assignment #3 และ Assignment #4 — ครบทุกข้อ ทุกข้อย่อย ไม่มีข้อไหนถูกข้าม ส่วนที่เป็น**การสอน**ว่าแต่ละวิธีคืออะไร ทำไมถึงทำแบบนั้น อยู่ใน **บทที่ 3 (ลอจิกเกต · พีชคณิตบูลีน · วงจรผสม)** ถ้าอ่านเฉลยข้อไหนแล้วสะดุด ให้ย้อนกลับไปอ่านหัวข้อที่เฉลยข้อนั้นชี้ให้ก่อน

อ่านเฉลยนี้ยังไง

ทุกข้อเรียงเหมือนกันหมด 5 ส่วน:

① **โจทย์** — ยกข้อความอังกฤษเดิมมาให้ แล้วแปลไทยกำกับ (คำอังกฤษในข้อสอบสำคัญ ต้องอ่านออก)
② **ใช้วิธีไหน** — บอกชื่อวิธี และชี้ว่าบทที่ 3 หัวข้อไหนสอนไว้
③ **เดินทีละบรรทัด** — ทุกบรรทัดมีป้ายบอกว่า “บรรทัดนี้ทำอะไร ใช้กฎชื่ออะไร” ไม่มีบรรทัดไหนข้าม
④ **กล่องคำตอบ** — สีเขียว คือสิ่งที่เขียนลงกระดาษคำตอบจริง ๆ
⑤ **จุดที่มักพลาดข้อนี้** — หนึ่งบรรทัด อ่านก่อนเข้าห้องสอบ

เครื่องหมายที่ใช้ในภาคผนวกนี้

ขีดบน $\overline{X}$ คือ NOT · จุดหรือเขียนติดกัน AB คือ AND · เครื่องหมายบวก $A + B$ คือ OR · Σm คือผลรวมของ minterm · ΠM คือผลคูณของ maxterm · คำว่า HIGH = 1 และ LOW = 0

ตัวเลขทุกตัวในภาคผนวกนี้ถูกตรวจด้วยโปรแกรม `check_a34.py` ซึ่งไล่ค่าอินพุต**ทุกชุดที่เป็นไปได้** เทียบนิพจน์ต้นฉบับกับคำตอบทีละแถว (รวม 93 รายการ)

#3 Assignment #3 — ลอจิกเกต · De Morgan · พีชคณิตบูลีน

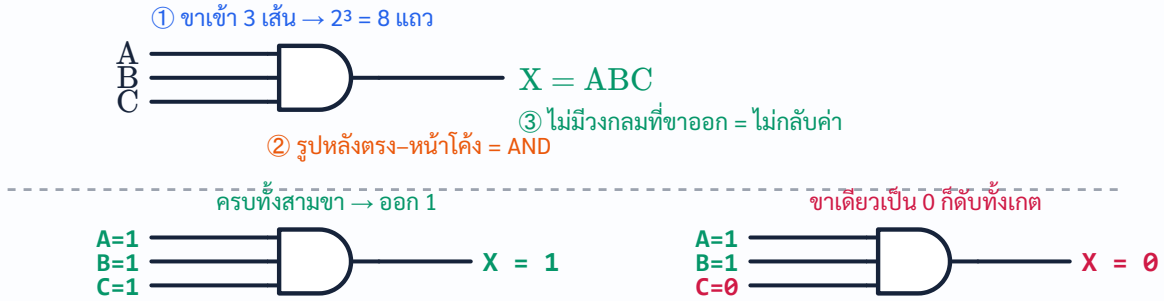
ข้อ 1 · ตารางค่าความจริงของ AND 3 อินพุต

① โจทย์

“Write a truth table for the following 3-input AND gate.” — พร้อมรูปเกต AND ที่มีขาเข้า A, B, C และเขียนกำกับขาออกไว้ว่า $X = ABC$ ใต้รูปมีตารางเปล่าที่พิมพ์คอลัมน์อินพุตมาให้แล้ว 8 แถว (*inputs go here*) เหลือช่องขวาสุดว่างไว้ให้เราเติม (*output goes here*)

แปลไทย: **จงเขียนตารางค่าความจริงของเกต AND ชนิด 3 อินพุตตัวนี้** — คือเติมค่า X ให้ครบทั้ง 8 แถว

② ใช้วิธีไหน ใช้ **นิยามของเกต AND** ไล่ทีละแถว — บทที่ 3 หัวข้อ “เกต AND: ต้องครบทุกขาถึงจะออก 1” สอนวิธีอ่านเกตและวิธีเรียงแถวของตารางไว้



เกต AND 3 อินพุตของข้อนี้ — สายทุกเส้นมีชื่อกำกับ และไล่ค่าให้ดู 2 แถวคือแถวที่ได้ 1 กับแถวที่ใกล้เคียงที่สุดที่ได้ 0

③ เดินทีละแถว กฎของ AND มีประโยคเดียว: $X = 1$ **เฉพาะตอนที่ขาเข้าทุกขาเป็น 1 นอกนั้นเป็น 0** เพราะฉะนั้นไล่ทั้ง 8 แถวได้ดังนี้ (คอลัมน์ขวาสุดคือเหตุผลของแถวนั้น)

A	B	C	$X = ABC$	ทำไมได้ค่านี้
0	0	0	**0**	มี 0 อยู่ → ไม่ครบ
0	0	1	**0**	A, B เป็น 0
0	1	0	**0**	A, C เป็น 0
0	1	1	**0**	A เป็น 0 ตัวเดียวก็พอ
1	0	0	**0**	B, C เป็น 0
1	0	1	**0**	B เป็น 0
1	1	0	**0**	C เป็น 0
1	1	1	**1**	ครบทั้งสามขา → แถวเดียวที่ได้ 1

เช็กกลับเข้าโจทย์: โจทย์บอกว่า $X = ABC$ ซึ่งเป็น AND ล้วน ๆ ของสามตัว AND จะได้ 1 ก็ต่อเมื่อครบทุกตัว → ในตาราง 8 แถวจึงต้องมีเลข 1 **แถวเดียว** และต้องเป็นแถวล่างสุด ($ABC = 111$) — ตรงกับที่เติมไว้

④ ตอบ

คอลัมน์ X จากบนลงล่าง = **0 0 0 0 0 0 0 1**

เขียนเป็นสัญกรณ์ minterm ได้ว่า $X = \sum m(7)$ — เป็น 1 ที่แถว $ABC = 111$ แถวเดียว

จุดที่มักพลาดข้อนี้

เผลอเติมเลข 1 ให้แถวที่ “มี 1 อยู่บ้าง” — AND ไม่ใช่ “มี 1 ก็พอ” (นั่นคือ OR) AND ต้องครบทุกขาเท่านั้น

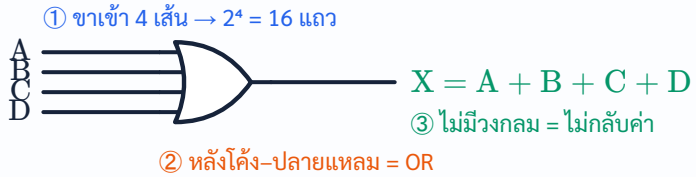
ข้อ 2 · ตารางค่าความจริงของ OR 4 อินพุต

① โจทย์

“Write a truth table for the following 4-input OR gate.” — รูปเกต OR ขาเข้า A, B, C, D ขาออกเขียนไว้ว่า $X = A + B + C + D$ ตารางที่ให้มามี 16 แถว เติมคอลัมน์ X

แปลไทย: **จงเขียนตารางค่าความจริงของเกต OR ชนิด 4 อินพุตตัวนี้**

② ใช้วิธีไหน ใช้ **นิยามของเกต OR** ไล่ทีละแถว — บทที่ 3 หัวข้อ “เกต OR: ขาเดียวเป็น 1 ก็พอ” และหัวข้อ “จำนวนแถว $= 2^n$”



เกต OR 4 อินพุตของข้อนี้ — สายทุกเส้นมีชื่อกำกับ ①②③ คือสามอย่างที่ต้องอ่านจากรูปก่อนลงมือเติมตาราง

วิธีจำที่เร็วที่สุด: OR 4 ขาจะเป็น **0 แถวเดียว** คือแถวที่ขาเข้าเป็น 0 หมด ($ABCD = 0000$) — อีก 15 แถวที่เหลือเป็น **1** ทั้งหมด

③ เดินทีละแถว กฎของ OR: $X = 0$ **เฉพาะตอนที่ขาเข้าทุกขาเป็น 0 นอกนั้นเป็น 1** (พูดอีกแบบคือ “เจอ 1 ที่ไหนก็จบ ออก 1 ทันที”)

A	B	C	D	X	ทำไมได้ค่านี้
0	0	0	0	0	ไม่มี 1 เลยสักขา → แถวเดียวที่ได้ 0
0	0	0	1	1	D เป็น 1
0	0	1	0	1	C เป็น 1
0	0	1	1	1	C, D เป็น 1
0	1	0	0	1	B เป็น 1
0	1	0	1	1	B, D เป็น 1
0	1	1	0	1	B, C เป็น 1
0	1	1	1	1	B, C, D เป็น 1
1	0	0	0	1	A เป็น 1
1	0	0	1	1	A, D เป็น 1
1	0	1	0	1	A, C เป็น 1
1	0	1	1	1	A, C, D เป็น 1
1	1	0	0	1	A, B เป็น 1
1	1	0	1	1	A, B, D เป็น 1
1	1	1	0	1	A, B, C เป็น 1
1	1	1	1	1	ครบทุกขา

เช็กกลับเข้าโจทย์: $X = A + B + C + D$ เป็น OR ล้วน ๆ ของสี่ตัว → ต้องมีเลข 0 **แถวเดียว** คือแถวบนสุด และต้องมีเลข 1 ทั้งหมด $16 - 1 = 15$ แถว — นับในตารางแล้วได้ 15 แถวพอดี

④ ตอบ

คอลัมน์ X จากบนลงล่าง = **`0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1`**

เขียนสั้น ๆ ได้ว่า $X = 0$ ที่ $ABCD = 0000$ เท่านั้น ที่เหลือ $X = 1$ ทั้งหมด

จุดที่มักพลาดข้อนี้

เขียนอินพุต 16 แถวสลับลำดับจนหลุด — ให้ไล่เหมือน**นับเลขฐานสอง** 0000, 0001, 0010, … , 1111 คอลัมน์ขวาสุดสลับทุกแถว ถัดมาสลับทุก 2 แถว ทุก 4 แถว และทุก 8 แถว

ข้อ 3 · ใช้ทฤษฎีบทของ De Morgan ทั้ง 9 ข้อย่อย

① โจทย์

“Apply De Morgan’s laws to the following expressions:” แล้วให้นิพจน์มา 9 อัน (a. ถึง i.)

แปลไทย: **จงใช้ทฤษฎีบทของเดอมอร์แกนกับนิพจน์ต่อไปนี้** — คือ “ดันขีดบนเข้าไปข้างใน” จนกระทั่ง**ไม่มีขีดบนเส้นไหนคลุมมากกว่าหนึ่งตัวอักษร** นั่นคือจุดที่ถือว่าทำเสร็จ

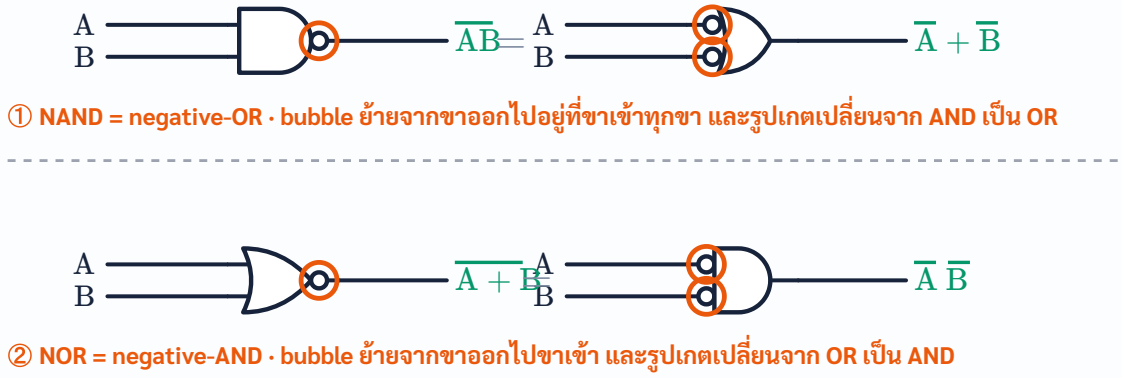
② ใช้วิธีไหน ใช้ **ทฤษฎีบทเดอมอร์แกน (De Morgan’s theorems)** สองข้อ บวกกับ **กฎขีดบนสองชั้น** $\overline{\overline{X}} = X$ — บทที่ 3 หัวข้อ “เดอมอร์แกน: ขีดบนแตกได้ แต่ต้องสลับเครื่องหมาย” และหัวข้อ “dual symbol ของ NAND/NOR”

สองสูตรที่ใช้ตลอดทั้งข้อ 3

$$\overline{X+Y} = \overline{X}\,\overline{Y} \quad \text{และ} \quad \overline{X\,Y} = \overline{X}+\overline{Y}$$

อ่านเป็นประโยคเดียว: ตัดขีดยาวหนึ่งเส้นออก → แจกขีดสั้นให้ทุกก้อนข้างใน → แล้ว**สลับเครื่องหมายตรงกลาง** ($+$ กลายเป็น $\cdot$, $\cdot$ กลายเป็น $+$)

X กับ Y ในสูตรจะใหญ่แค่ไหนก็ได้ — เป็นตัวอักษรเดียว เป็นก้อน ABC หรือเป็นวงเล็บทั้งอันก็ใช้ได้ ข้อนี้ทั้ง 9 ข้อย่อยคือการใช้สองสูตรนี้ซ้ำ ๆ



เดอมอร์แกนในรูปของเกต — วงกลมส้มคือ bubble ที่กำลังถูกย้าย สายทุกเส้นมีชื่อกำกับ (ชื่อตัวแปรที่ขาเข้า นิพจน์ผลลัพธ์ที่ขาออก)

แถวบนของรูปคือสูตร $\overline{XY} = \overline{X} + \overline{Y}$ ที่วาดเป็นรูป แถวล่างคือสูตร $\overline{X+Y} = \overline{X}\,\overline{Y}$ — สองรูปในแต่ละแถวคือ**เกตตัวเดียวกัน** เขียนได้สองหน้าตา เรียกว่า **dual symbol** ของกันและกัน

③ ทั้งสองแถวบอกเรื่องเดียวกันประโยคเดียว: **ย้าย bubble ข้ามตัวเกตเมื่อไร ต้องสลับรูปเกตเมื่อนั้น** — ลืมสลับคือที่มาของคำตอบผิดเกือบทั้งหมดในข้อนี้

③ เดินทีละบรรทัด ครบทั้ง 9 ข้อย่อย แต่ละตารางอ่านจากบนลงล่าง บรรทัดบนสุดคือโจทย์ บรรทัดล่างสุดคือคำตอบ (แถบเขียว) ช่องขวาบอกว่าบรรทัดนั้นทำอะไร

a.) $\overline{ABC + DEF}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{ABC + DEF}$	โจทย์ — ขีดยาวหนึ่งเส้นคลุม “ก้อน ABC **บวก** ก้อน DEF” มองเป็น $\overline{X + Y}$ โดย $X = ABC, Y = DEF$
②	$= \overline{ABC} \cdot \overline{DEF}$	**เดอมอร์แกนสูตร 1** $\overline{X + Y} = \overline{X}\,\overline{Y}$ — ตัดขีดยาวทิ้ง แจกขีดสั้นให้สองก้อน แล้วสลับ $+$ เป็น $\cdot$
③	$= (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$	**เดอมอร์แกนสูตร 2** $\overline{XY} = \overline{X} + \overline{Y}$ ใช้กับแต่ละก้อน — ขีดที่คลุมสามตัวแตกเป็นขีดทีละตัว และ $\cdot$ สลับเป็น $+$ **คำตอบข้อ a**

เช็กว่าจบจริง: ไล่ดูทั้งบรรทัด ③ ไม่มีขีดบนเส้นไหนคลุมเกินหนึ่งตัวอักษรแล้ว → ใช้เดอมอร์แกนจนสุดทาง

b.) $\overline{A\overline{B} + \overline{C}D + EF}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{A\overline{B} + \overline{C}D + EF}$	โจทย์ — ขีดยาวคลุม**สามก้อนที่บวกกัน** ($A\overline{B}$, $\overline{C}D$, EF) ส่วนขีดสั้นบน B กับ C เป็นของเดิมในก้อน อย่าไปยุ่งตอนนี้
②	$= \overline{A\overline{B}} \cdot \overline{\overline{C}D} \cdot \overline{EF}$	**เดอมอร์แกนสูตร 1** (ใช้กับสามก้อนได้เลย) — ขีดยาวหายไป ทุกก้อนได้ขีดของตัวเอง $+$ กลายเป็น $\cdot$
③	$= (\overline{A} + \overline{\overline{B}})(\overline{\overline{C}} + \overline{D})(\overline{E} + \overline{F})$	**เดอมอร์แกนสูตร 2** กับทุกก้อน — ก้อนละสองตัว แตกเป็นบวกกัน (ตอนนี้ B กับ C มีขีดสองชั้นแล้ว)
④	$= (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$	**กฎขีดสองชั้น** $\overline{\overline{X}} = X$ — ขีดสองเส้นซ้อนกันหักล้างกันหายไป **คำตอบข้อ b**

เช็กว่าจบจริง: เหลือขีดบนแค่บน A, D, E, F ตัวละหนึ่งเส้น ไม่มีเส้นไหนคลุมสองตัว → จบ

c.) $\overline{\overline{ABC} + D + E}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{\overline{ABC} + D + E}$	โจทย์ — **ขีดสองชั้นไม่ได้ซ้อนกันตรง ๆ** ขีดชั้นนอกคลุมทั้ง $\overline{ABC} + D + E$ ส่วนขีดชั้นในคลุมแค่ ABC ให้ทำชั้นนอกก่อนเสมอ
②	$= \overline{\overline{ABC}} \cdot \overline{D} \cdot \overline{E}$	**เดอมอร์แกนสูตร 1** กับสามก้อน ($\overline{ABC}$, D, E) — ตอนนี้ก้อนแรกกลายเป็นขีดซ้อนขีดจริง ๆ แล้ว

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
③	$= ABC\,\overline{D}\,\overline{E}$	**กฎขีดสองชั้น** $\overline{\overline{ABC}} = ABC$ — ขีดคู่หายไปพร้อมกันทั้งคู่ **คำตอบข้อ c**

เช็กกลับ: ลองค่า $A = B = C = 1,\ D = E = 0$ — โจทย์เดิม: $\overline{ABC} = \overline{1} = 0$ ข้างในวงเล็บได้ $0 + 0 + 0 = 0$ ขีดนอกทำให้เป็น $1 \cdot$ คำตอบ: $1 \cdot 1 \cdot 1 \cdot \overline{0} \cdot \overline{0} = 1$ — ตรงกัน

d.) $\overline{\overline{(A + B)} + \overline{C}}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{\overline{(A + B)} + \overline{C}}$	โจทย์ — ขีดชั้นนอกคลุมทั้งหมด ข้างในมีสองก้อนบวกกันคือ $\overline{(A + B)}$ กับ $\overline{C}$
②	$= \overline{\overline{(A + B)}} \cdot \overline{\overline{C}}$	**เดอมอร์แกนสูตร 1** — ตัดขีดนอก แจกให้สองก้อน $+$ เป็น $\cdot$ ทั้งสองก้อนกลายเป็นขีดซ้อนขีด
③	$= (A + B)\,C$	**กฎขีดสองชั้น** ทั้งสองก้อน — $\overline{\overline{(A + B)}} = A + B$ และ $\overline{\overline{C}} = C$ **คำตอบข้อ d**
④	$= AC + BC$	**กฎการแจกแจง (distributive)** — ไม่จำเป็นต้องทำ แต่ถ้าโจทย์ข้อถัดไปอยากได้รูปผลบวกของผลคูณ (SOP) ก็กระจายวงเล็บแบบนี้

ข้อสังเกต: ข้อนี้ขีดบน**หายหมด** — เพราะขีดนอกกับขีดในหักล้างกันพอดีทั้งสองก้อน ถ้าได้คำตอบที่ยังมีขีดเหลือ แปลว่าลืมกฎขีดสองชั้นไปหนึ่งที่

e.) $\overline{(\overline{A} + B) + CD}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{(\overline{A} + B) + CD}$	โจทย์ — ขีดยาวคลุม “วงเล็บ $(\overline{A} + B)$ **บวก** ก้อน CD ”
②	$= \overline{(\overline{A} + B)} \cdot \overline{CD}$	**เดอมอร์แกนสูตร 1** — แจกขีดให้สองก้อน $+$ เป็น $\cdot$
③	$= \left(\overline{\overline{A}} \cdot \overline{B}\right) \cdot \overline{CD}$	**เดอมอร์แกนสูตร 1 อีกครั้ง** กับก้อนซ้าย เพราะข้างในวงเล็บเป็น**ผลบวก** $(\overline{A} + B)$
④	$= A\,\overline{B} \cdot (\overline{C} + \overline{D})$	ก้อนซ้าย: **กฎขีดสองชั้น** $\overline{\overline{A}} = A$ · ก้อนขวา: **เดอมอร์แกนสูตร 2** เพราะ CD เป็นผลคูณ **คำตอบข้อ e**
⑤	$= A\overline{B}\,\overline{C} + A\overline{B}\,\overline{D}$	**กฎการแจกแจง** — รูปทางเลือกแบบ SOP (เขียนแบบ ④ ก็ถือว่าตอบแล้ว)

กับดักที่ซ่อนอยู่: ก้อนซ้ายข้างในเป็น “บวก” ต้องใช้**สูตร 1** ส่วนก้อนขวาข้างในเป็น “คูณ” ต้องใช้**สูตร 2** — ก้อนสองก้อนในบรรทัดเดียวกันอาจใช้คนละสูตร ให้ดูเครื่องหมาย**ข้างใต้ขีดนั้น**ทุกครั้ง

f.) $\overline{ABC} + \overline{(\overline{D} + E)}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{ABC} + \overline{(\overline{D} + E)}$	โจทย์ — สังเกตว่า**ไม่มีขีดเส้นไหนคลุมทั้งนิพจน์** มีขีดสองเส้นแยกกันอยู่คนละก้อน เครื่องหมาย $+$ ตรงกลาง**อยู่นอกขีด** จึงแตะไม่ได้
②	$\overline{ABC} \;=\; \overline{A} + \overline{B} + \overline{C}$	ทำก้อนซ้ายก่อน — **เดอมอร์แกนสูตร 2** (ข้างใต้ขีดเป็นผลคูณ)
③	$\overline{\overline{D} + E} \;=\; \overline{\overline{D}} \cdot \overline{E} \;=\; D\,\overline{E}$	ทำก้อนขวา — **เดอมอร์แกนสูตร 1** (ข้างใต้ขีดเป็นผลบวก) แล้ว**ขีดสองชั้น**บน D หักล้างกัน
④	$= \overline{A} + \overline{B} + \overline{C} + D\,\overline{E}$	เอาสองก้อนมาบวกกันคืนตามเครื่องหมายเดิม **คำตอบข้อ f**

เช็กกลับ: ลอง $A = B = C = 1, D = 0, E = 0$ — โจทย์เดิม: $\overline{ABC} = 0$, $\overline{\overline{D} + E} = \overline{1 + 0} = 0$ รวม $0 + 0 = 0$ · คำตอบ: $0 + 0 + 0 + (0 \cdot 1) = 0$ — ตรงกัน

g.) $\overline{A + B + C} + \overline{\overline{D}E}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{A + B + C} + \overline{\overline{D}E}$	โจทย์ — ขีดสองเส้นแยกกันอีกเหมือนข้อ f เครื่องหมาย $+$ ตรงกลางอยู่นอกขีด
②	$\overline{A + B + C} \;=\; \overline{A}\,\overline{B}\,\overline{C}$	ก้อนซ้าย — **เดอมอร์แกนสูตร 1** (ข้างใต้ขีดเป็นผลบวกสามตัว) $+$ ทุกตัวกลายเป็น $\cdot$
③	$\overline{\overline{D}E} \;=\; \overline{\overline{D}} + \overline{E} \;=\; D + \overline{E}$	ก้อนขวา — **เดอมอร์แกนสูตร 2** (ข้างใต้ขีดเป็นผลคูณ $\overline{D} \cdot E$) แล้ว**ขีดสองชั้น**บน D หักล้างกัน
④	$= \overline{A}\,\overline{B}\,\overline{C} + D + \overline{E}$	ประกอบคืนด้วย $+$ ตัวเดิม **คำตอบข้อ g**

เทียบกับข้อ f ให้เห็นชัด: f มี $\overline{ABC}$ (คูณใต้ขีด) จึงแตกเป็น**บวก** · g มี $\overline{A + B + C}$ (บวกใต้ขีด) จึงแตกเป็น**คูณ** — สองข้อนี้หน้าตาคล้ายกันมากแต่ผลลัพธ์กลับกัน

h.) $\overline{(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})}$	โจทย์ — ใต้ขีดยาวคือ “วงเล็บ **คูณ** วงเล็บ” มองเป็น $\overline{XY}$ โดย X, Y คือวงเล็บแต่ละอัน
②	$= \overline{(A + \overline{B} + \overline{C})} \;+\; \overline{(\overline{A} + B + \overline{C})}$	**เดอมอร์แกนสูตร 2** $\overline{XY} = \overline{X} + \overline{Y}$ — คูณระหว่างวงเล็บกลายเป็นบวก

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
③	$= \overline{A}\,\overline{B}\,\overline{C} \;+\; \overline{A}\,B\,\overline{C}$	**เดอมอร์แกนสูตร 1** กับวงเล็บทั้งสองอัน — ข้างในเป็นผลบวก จึงแตกเป็นผลคูณ ทุกตัวได้ขีดเพิ่มหนึ่งชั้น
④	$= \overline{A}BC \;+\; A\,\overline{B}\,C$	**กฎขีดสองชั้น** — $\overline{\overline{B}} = B, \overline{\overline{C}} = C, \overline{\overline{A}} = A$ **คำตอบข้อ h**

เช็กกลับ: คำตอบเป็น 1 แค่สองแถวคือ $ABC = 011$ และ $ABC = 101$ · ลองแทน $ABC = 011$ ในโจทย์เดิม: วงเล็บแรก $= 0 + \overline{1} + \overline{1} = 0$ → ผลคูณเป็น 0 → ขีดนอกทำให้เป็น **1** ตรงกัน

i.) $\overline{\overline{A}BC + \overline{A}\,\overline{B}\,\overline{C}}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{\overline{A}BC + \overline{A}\,\overline{B}\,\overline{C}}$	โจทย์ — ขีดยาวคลุม “ก้อน $\overline{A}BC$ **บวก** ก้อน $\overline{A}\,\overline{B}\,\overline{C}$”
②	$= \overline{\overline{A}BC} \;\cdot\; \overline{\overline{A}\,\overline{B}\,\overline{C}}$	**เดอมอร์แกนสูตร 1** — แจกขีดให้สองก้อน $+$ เป็น $\cdot$
③	$= (\overline{\overline{A}} + \overline{B} + \overline{C})(\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})$	**เดอมอร์แกนสูตร 2** กับทั้งสองก้อน — ข้างใต้ขีดเป็นผลคูณสามตัว แตกเป็นบวกสามตัว
④	$= (A + \overline{B} + \overline{C})(A + B + C)$	**กฎขีดสองชั้น** ทุกที่ที่ขีดซ้อนกัน **คำตอบข้อ i**

ข้อสังเกตที่ใช้ตรวจได้เร็ว: โจทย์เป็น**ผลบวกของผลคูณ** (SOP) เมื่อเติมขีดบนทั้งอัน คำตอบต้องออกมาเป็น**ผลคูณของผลบวก** (POS) เสมอ — ถ้าคำตอบยังเป็น SOP อยู่ แปลว่าทำผิดที่ไหนสักแห่ง

④ ตอบ ข้อ 3 ทั้งเก้าข้อย่อย

a.)	$\overline{ABC + DEF}$	$= (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$
b.)	$\overline{A\overline{B} + \overline{C}D + EF}$	$= (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$
c.)	$\overline{\overline{ABC} + D + E}$	$= ABC\,\overline{D}\,\overline{E}$
d.)	$\overline{\overline{(A + B)} + \overline{C}}$	$= (A + B)C \;=\; AC + BC$
e.)	$\overline{(\overline{A} + B) + CD}$	$= A\overline{B}(\overline{C} + \overline{D}) \;=\; A\overline{B}\,\overline{C} + A\overline{B}\,\overline{D}$
f.)	$\overline{ABC} + \overline{(\overline{D} + E)}$	$= \overline{A} + \overline{B} + \overline{C} + D\overline{E}$
g.)	$\overline{A + B + C} + \overline{\overline{D}E}$	$= \overline{A}\,\overline{B}\,\overline{C} + D + \overline{E}$
h.)	$\overline{(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})}$	$= \overline{A}BC + A\overline{B}C$
i.)	$\overline{\overline{A}BC + \overline{A}\,\overline{B}\,\overline{C}}$	$= (A + \overline{B} + \overline{C})(A + B + C)$

ทั้ง 9 บรรทัดนี้ถูกตรวจด้วยโปรแกรมแล้ว: แทนค่าอินพุต**ทุกชุด** (ข้อที่มี 6 ตัวแปรคือ 64 ชุด) นิพจน์ซ้ายกับขวาให้ค่าตรงกันทุกชุด

จุดที่มักพลาดข้อนี้

แตกขีดบนแล้ว**ลืมสลับเครื่องหมาย** — เขียน $\overline{A+B} = \overline{A} + \overline{B}$ (ผิด) ที่ถูกคือ $\overline{A}\,\overline{B}$ ให้จำว่า “ขีดแตกเมื่อไร เครื่องหมายต้องพลิกเมื่อนั้น”

ข้อ 4 · ลดรูปด้วยพีชคณิตบูลีน (a และ b)

① โจทย์

“Using Boolean algebra techniques, simplify the following expressions:”

a.) $A\overline{B} + A\,\overline{(B+C)} + B\,\overline{(B+C)}$ b.) $\overline{AB + AC} + \overline{A}\,\overline{B}\,C$

แปลไทย: **จงลดรูปนิพจน์ต่อไปนี้ด้วยเทคนิคพีชคณิตบูลีน** — “ลดรูป” หมายถึงทำให้เหลือจำนวนพจน์และจำนวนตัวอักษรน้อยที่สุดโดยค่าทุกแถวในตารางยังเท่าเดิม

② ใช้วิธีไหน ใช้ **กฎพีชคณิตบูลีน** — เดอมอร์แกนเพื่อเปิดขีดบนก่อน แล้วตามด้วยกฎ complement ($X\overline{X} = 0$), กฎการดึงตัวร่วม, กฎ $1 + X = 1$ และกฎการดูดกลืน (absorption) — บทที่ 3 หัวข้อ “กฎพีชคณิตบูลีน 12 ข้อ” และ “ลำดับที่ควรลงมือ: เปิดขีดก่อน แล้วค่อยรวมพจน์”

a.) $A\overline{B} + A\,\overline{(B+C)} + B\,\overline{(B+C)}$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$A\overline{B} + A\,\overline{(B+C)} + B\,\overline{(B+C)}$	โจทย์ — มีขีดบนคลุม $(B+C)$ อยู่สองที่ เปิดขีดให้หมดก่อน ห้ามเริ่มจากการรวมพจน์
②	$= A\overline{B} + A\overline{B}\,\overline{C} + B\,\overline{B}\,\overline{C}$	**เดอมอร์แกนสูตร 1** $\overline{B+C} = \overline{B}\,\overline{C}$ — แทนลงไปทั้งสองที่ ตอนนี้ไม่มีขีดคลุมเกินหนึ่งตัวแล้ว
③	$= A\overline{B} + A\overline{B}\,\overline{C} + 0$	**กฎ complement** $B\,\overline{B} = 0$ — ตัวแปรคูณกับตัวมันเองที่กลับค่าแล้ว ไม่มีทางเป็น 1 พร้อมกัน พจน์ที่สามจึงเป็น 0 เสมอ
④	$= A\overline{B} + A\overline{B}\,\overline{C}$	**กฎเอกลักษณ์ของการบวก** $X + 0 = X$ — พจน์ 0 ทิ้งได้
⑤	$= A\overline{B}\,(1 + \overline{C})$	**ดึงตัวร่วม (factoring)** — ทั้งสองพจน์มี $A\overline{B}$ เหมือนกัน ดึงออกมานอกวงเล็บ
⑥	$= A\overline{B} \cdot 1 = A\overline{B}$	**กฎ** $1 + X = 1$ (บวกกับ 1 ได้ 1 เสมอ) แล้ว $X \cdot 1 = X$ **คำตอบข้อ a**

เช็กกลับเข้าโจทย์ — ไล่ทั้ง 8 แถว (คอลัมน์กลางคือของกลางทางที่ใช้ คอลัมน์สองอันขวาคือนิพจน์เดิมกับคำตอบ ต้องตรงกันทุกแถว)

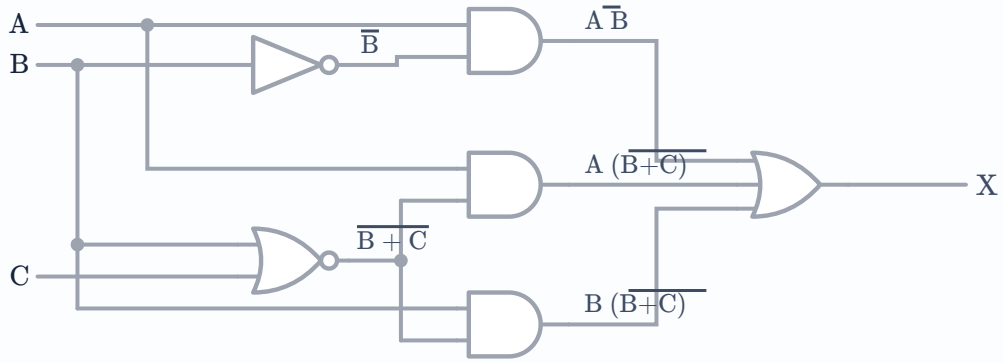
A	B	C	$\overline{B}$	$\overline{B+C}$	นิพจน์เดิม	$A\overline{B}$
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0

สองแถวที่ระบายเขียวคือแถวที่นิพจน์เป็น 1 — คือแถว $A = 1, B = 0$ ทั้งสองแถว ไม่ว่า C จะเป็นอะไร นั่นคือความหมายของ $A\overline{B}$ พอดี

คอลัมน์ “นิพจน์เดิม” กับ “$A\overline{B}$” **ตรงกันครบทั้ง 8 แถว** → การลดรูปไม่ได้ทำให้วงจรเปลี่ยนพฤติกรรม

A	B	C	$\overline{B}$	$\overline{B+C}$	นิพจน์เดิม	$A\overline{B}$
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	1	0	1	1
1	1	0	0	0	0	0
1	1	1	0	0	0	0

ก่อนลดรูป — 6 เกต



หลังลดรูป — 2 เกต (ให้ผลเหมือนกันทั้ง 8 แถว)



วงจรของข้อ a ก่อนและหลังลดรูป — สายทุกเส้นมีชื่อกำกับด้วยนิพจน์ที่มันพาไป (เกตสีเทาคือของเดิมที่ถูกตัดทิ้ง) · จุดกลมบนสายคือจุดแยกสาย ส่วนสายที่ตัดผ่านกันเฉย ๆ โดยไม่มีจุดกลม คือ**ไม่ได้ต่อถึงกัน**

สังเกตว่าตัวแปร C **หายไปทั้งตัว** — วงจรจริงจึงไม่ต้องเดินสาย C เข้ามาเลย นี่คือสิ่งที่การลดรูปซื้อให้เรา

b.) $\overline{AB + AC} + \overline{A}\,\overline{B}\,C$

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{AB + AC} + \overline{A}\,\overline{B}\,C$	โจทย์ — ขีดยาวคลุม $AB + AC$ เท่านั้น ส่วน $\overline{A}\,\overline{B}C$ อยู่นอกขีด เปิดขีดก่อนเหมือนเดิม
②	$= \overline{AB} \cdot \overline{AC} + \overline{A}\,\overline{B}\,C$	**เดอมอร์แกนสูตร 1** $\overline{X+Y} = \overline{X}\,\overline{Y}$ โดย $X = AB$, $Y = AC$
③	$= (\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A}\,\overline{B}\,C$	**เดอมอร์แกนสูตร 2** กับทั้งสองก้อน — $\overline{AB} = \overline{A} + \overline{B}$ และ $\overline{AC} = \overline{A} + \overline{C}$
④	$= \overline{A} + \overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C$	**กฎการแจกแจงแบบที่สอง** $(X+Y)(X+Z) = X + YZ$ โดย $X = \overline{A},\ Y = \overline{B},\ Z = \overline{C}$ — วงเล็บสองอันยุบเหลือพจน์เดียวบวกผลคูณ

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
⑤	$= \overline{A} + \overline{B}\,\overline{C}$	**กฎการดูดกลืน (absorption)** $X + XY = X$ โดย $X = \overline{A}, Y = \overline{B}C$ — พจน์ $\overline{A}\,\overline{B}C$ มี $\overline{A}$ อยู่ในตัว จึงไม่เคยทำให้ผลลัพธ์เป็น 1 ในแถวที่ $\overline{A}$ ยังไม่เป็น 1 อยู่แล้ว **คำตอบข้อ b**

เช็กกลับเข้าโจทย์ — ไล่ทั้ง 8 แถว

A	B	C	$\overline{AB + AC}$	$\overline{A}\,\overline{B}C$	นิพจน์เดิม	$\overline{A} + \overline{B}\,\overline{C}$
0	0	0	1	0	**1**	**1**
0	0	1	1	1	**1**	**1**
0	1	0	1	0	**1**	**1**
0	1	1	1	0	**1**	**1**
1	0	0	1	0	**1**	**1**
1	0	1	0	0	**0**	**0**
1	1	0	0	0	**0**	**0**
1	1	1	0	0	**0**	**0**

ดูคอลัมน์ $\overline{A}\,\overline{B}C$ ให้ดี — มันเป็น 1 อยู่**แถวเดียว** (แถว $ABC = 001$) และแถวนั้นนิพจน์เดิมก็เป็น 1 อยู่แล้วจากพจน์อื่น พจน์นี้จึง**ไม่ได้เพิ่มอะไรเลย** — นั่นคือความหมายของกฎการดูดกลืนในขั้น ⑤

คอลัมน์ “นิพจน์เดิม” กับ “$\overline{A} + \overline{B}\,\overline{C}$” ตรงกันครบ 8 แถว

④ ตอบ ข้อ 4

a.) $A\overline{B} + A\,\overline{(B + C)} + B\,\overline{(B + C)} = \mathbf{A\overline{B}}$

b.) $\overline{AB + AC} + \overline{A}\,\overline{B}\,C = \mathbf{\overline{A} + \overline{B}\,\overline{C}}$

จุดที่มักพลาดข้อนี้

รีบจับพจน์มารวมกันตั้งแต่ยังมีขีดบนคลุมหลายตัวอยู่ — **ต้องเปิดขีดบนให้หมดก่อนเสมอ** แล้วค่อยใช้กฎรวมพจน์ ไม่งั้นจะเผลอ “ตัด” สิ่งที่ตัดไม่ได้

ข้อ 5 · ตัดพจน์ consensus ออก

① โจทย์

“Eliminate the consensus terms from the following Boolean expressions.”

$$\overline{A}\,\overline{C}D + \overline{A}BD + BCD + ABC + AC\overline{D}$$

แปลไทย: **จงตัดพจน์ consensus ออกจากนิพจน์บูลีนต่อไปนี้** — พจน์ consensus คือพจน์ที่**ลบทิ้งได้โดยค่าทุกแถวในตารางไม่เปลี่ยน**

② ใช้วิธีไหน ใช้ **ทฤษฎีบท consensus** — บทที่ 3 หัวข้อ “พจน์ที่ไม่เคยออกเสียง: consensus theorem”

สูตร consensus และวิธีหาพจน์ consensus ของคู่หนึ่ง ๆ

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

วิธีหาเป็นขั้นตอน — หยิบพจน์มาทีละ**คู่** แล้วถาม 2 คำถาม:

คำถามที่ 1: คู่นี้มีตัวแปรที่ปรากฏ**คนละขั้ว พอดีหนึ่งตัว** ไหม (ตัวหนึ่งมีขีด อีกตัวไม่มีขีด)? ถ้าไม่มี หรือมีมากกว่าหนึ่งตัว → คู่นี้ไม่มี consensus ข้ามไปได้เลย

คำถามที่ 2: ถ้ามีพอดีหนึ่งตัว → **ลบตัวแปรตัวนั้นทิ้ง** แล้วเอาตัวอักษรที่เหลือของทั้งสองพจน์มาคูณกัน (ตัวซ้ำนับครั้งเดียว) นั่นคือพจน์ consensus ของคู่นี้ · ถ้าพจน์นั้น**มีอยู่จริง**ในนิพจน์ → มันคือพจน์ที่ตัดได้

③ ไล่ทุกคู่ ไม่ข้ามคู่ไหน นิพจน์มี 5 พจน์ ตั้งชื่อว่า T_1 ถึง T_5 ตามลำดับที่เขียนมา จับคู่ได้ทั้งหมด 10 คู่ ต้องดูให้ครบ

คู่ที่หยิบมา	ตัวแปรที่ขัดกัน	ผลของคำถามที่ 2
$T_1, T_2 : \overline{A}\,\overline{C}D$, $\overline{A}BD$	ไม่มีเลย	ไม่มี consensus
$T_1, T_3 : \overline{A}\,\overline{C}D$, BCD	$\overline{C}$ กับ C — **หนึ่งตัวพอดี**	ลบ C ทิ้ง เหลือ $\overline{A}D$ กับ BD เอามาคูณกันได้ $\overline{A}BD$ → **คือ T_2 ซึ่งมีอยู่จริง**
$T_1, T_4 : \overline{A}\,\overline{C}D$, ABC	A และ C — สองตัว	ไม่มี consensus (ต้องขัดกันตัวเดียวเท่านั้น)
$T_1, T_5 : \overline{A}\,\overline{C}D$, $AC\overline{D}$	A, C, D — สามตัว	ไม่มี consensus
$T_2, T_3 : \overline{A}BD$, BCD	ไม่มีเลย	ไม่มี consensus
$T_2, T_4 : \overline{A}BD$, ABC	$\overline{A}$ กับ A — หนึ่งตัวพอดี	ลบ A ทิ้ง เหลือ BD กับ BC คูณกันได้ BCD → คือ T_3 ซึ่งมีอยู่จริง
$T_2, T_5 : \overline{A}BD$, $AC\overline{D}$	A และ D — สองตัว	ไม่มี consensus
$T_3, T_4 : BCD$, ABC	ไม่มีเลย	ไม่มี consensus

คู่ที่หยิบมา	ตัวแปรที่ขัดกัน	ผลของคำถามที่ 2
$T_3, T_5 : BCD$, $AC\overline{D}$	D กับ $\overline{D}$ — **หนึ่งตัวพอดี**	ลบ D ทิ้ง เหลือ BC กับ AC คูณกันได้ ABC → **คือ T_4 ซึ่งมีอยู่จริง**
$T_4, T_5 : ABC$, $AC\overline{D}$	ไม่มีเลย	ไม่มี consensus

ได้ผู้ต้องสงสัย 3 ตัวคือ T_2, T_3, T_4 แต่**ตัดพร้อมกันทั้งสามไม่ได้** — ลองตัดหมดแล้วเหลือ $\overline{A}\,\overline{C}D + AC\overline{D}$ ซึ่งแถว $ABCD = 0111$ กลายเป็น 0 ทั้งที่เดิมเป็น 1 ให้เลือกชุดที่ทำให้**เหลือพจน์น้อยที่สุด** คือ**ตัด T_2 กับ T_4 ทิ้ง เก็บ T_3 ไว้** เพราะเมื่อ T_2 กับ T_4 หายไปแล้ว T_3 ก็ไม่ได้เป็นพจน์ consensus ของใครอีกต่อไป

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$\overline{A}\,\overline{C}D + \overline{A}BD + BCD + ABC + AC\overline{D}$	โจทย์ (T_1 ถึง T_5 ตามลำดับ)
②	$= \overline{A}\,\overline{C}D + BCD + ABC + AC\overline{D}$	**ทฤษฎีบท consensus** กับคู่ T_1, T_3 — $\overline{A}BD$ เป็น consensus ของ $\overline{A}\,\overline{C}D$ กับ BCD จึงลบทิ้งได้
③	$= \overline{A}\,\overline{C}D + BCD + AC\overline{D}$	**ทฤษฎีบท consensus** กับคู่ T_3, T_5 — ABC เป็น consensus ของ BCD กับ $AC\overline{D}$ จึงลบทิ้งได้
④	$\overline{A}\,\overline{C}D + BCD + AC\overline{D}$	ไล่คู่ที่เหลืออีก 3 คู่: (T_1, T_3) ให้ $\overline{A}BD$ ซึ่งไม่อยู่แล้ว · (T_3, T_5) ให้ ABC ซึ่งไม่อยู่แล้ว · (T_1, T_5) ขัดกันสามตัว → **ตัดต่อไม่ได้อีก** **คำตอบ**

เช็กกลับเข้าโจทย์ — ไล่ทั้ง 16 แถว คอลัมน์ 5 อันกลางคือค่าของแต่ละพจน์ สองคอลัมน์ที่หัวตารางเป็นสีชมพูคือ **พจน์ที่จะตัดทิ้ง** — ทุกแถวที่มันเป็น 1 จะมีพจน์อื่นเป็น 1 อยู่ด้วยเสมอ จึงตัดได้

A	B	C	D	T_1	T_2	T_3	T_4	T_5	X
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	1	1	0	0	1
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	1
1	0	1	1	0	0	0	0	0	0

A	B	C	D	T_1	T_2	T_3	T_4	T_5	X
1	1	0	0	0	0	0	0	0	**0**
1	1	0	1	0	0	0	0	0	**0**
1	1	1	0	0	0	0	1	1	**1**
1	1	1	1	0	0	1	1	0	**1**

อ่านตารางยืนยันคำตอบ: T_2 เป็น 1 ที่แถว 0101 กับ 0111 — แถวแรกมี T_1 เป็น 1 อยู่แล้ว แถวหลังมี T_3 เป็น 1 อยู่แล้ว · T_4 เป็น 1 ที่แถว 1110 กับ 1111 — แถวแรกมี T_5 อยู่แล้ว แถวหลังมี T_3 อยู่แล้ว · ตัดสองพจน์นี้ออกแล้วคอลัมน์ X ยัง**เหมือนเดิมทั้ง 16 แถว** คือเป็น 1 ที่แถว 0001, 0101, 0111, 1010, 1110, 1111 รวม 6 แถว

④ ตอบ ข้อ 5

$$\overline{A}\,\overline{C}D + \overline{A}BD + BCD + ABC + AC\overline{D} \;=\; \mathbf{\overline{A}\,\overline{C}D + BCD + AC\overline{D}}$$

พจน์ที่ตัดทิ้งคือ $\overline{A}BD$ (consensus ของ $\overline{A}\,\overline{C}D$ กับ BCD) และ ABC (consensus ของ BCD กับ $AC\overline{D}$)

ถ้าเจอคำตอบอีกแบบในเฉลยของเพื่อน

มีอีกทางเลือกที่ค่าทุกแถวเท่าเดิมเหมือนกัน คือตัดเฉพาะ BCD ทิ้ง เหลือ $\overline{A}\,\overline{C}D + \overline{A}BD + ABC + AC\overline{D}$ (ตรวจด้วยโปรแกรมแล้วว่าเท่ากันจริง) แต่แบบนั้นเหลือ **4 พจน์** ไม่ใช่ 3 · โปรแกรมไล่ทุกชุดย่อยของทั้ง 5 พจน์แล้วพบว่า**ชุดที่เล็กที่สุดมีชุดเดียว** คือ $\{\overline{A}\,\overline{C}D,\ BCD,\ AC\overline{D}\}$ — ตอบแบบนี้ปลอดภัยที่สุด

จุดที่มักพลาดข้อนี้

เห็นพจน์ที่หน้าตาคล้าย ๆ กันแล้วตัดทิ้งเลย — ต้องยืนยันก่อนว่าคู่ที่สร้างมันขึ้นมามีตัวแปรขัดกัน**พอดีหนึ่งตัว** และตัวมันเอง**มีอยู่จริง**ในนิพจน์

#4 Assignment #4 — minterm · SOP/POS · วงจรผสม · dual symbol

ข้อ 1 · เขียน X และ $\overline{X}$ ด้วยสัญกรณ์ minterm (a, b, c)

① โจทย์

“Express X and $\overline{X}$ defined by the following truth tables using the minterm notation.” แล้วให้ตารางค่าความจริง 3 ตัวแปร (A, B, C) มา 3 ตาราง คือข้อ a, b, c พร้อมช่องว่างให้เติมว่า $X(A, B, C) =$ ..... , $\overline{X}(A, B, C) =$ .....

แปลไทย: **จงเขียน X และ $\overline{X}$ ที่นิยามด้วยตารางค่าความจริงต่อไปนี้ ในรูปสัญกรณ์ minterm**

② ใช้วิธีไหน ใช้ **การอ่าน minterm จากตาราง** — บทที่ 3 หัวข้อ “minterm คือชื่อประจำแถว” และหัวข้อ “X กับ $\overline{X}$ แบ่งแถวกันคนละครึ่ง”

ตารางชื่อประจำแถว ใช้ได้กับทั้ง 3 ข้อย่อย

ตัวแปร 3 ตัวมี 8 แถว แต่ละแถวมี “ชื่อ” ของตัวเองชื่อว่า m_0 ถึง m_7 — เลขห้อยคือค่าของ ABC เมื่ออ่านเป็นเลขฐานสอง (A คือหลักค่าประจำ 4, B คือ 2, C คือ 1) ส่วน “หน้าตา” ของ minterm คือผลคูณของตัวแปรทั้งสาม โดย**ตัวไหนเป็น 0 ให้ใส่ขีดบน**

แถวที่	A	B	C	ชื่อ	หน้าตาของ minterm
0	0	0	0	m_0	$\overline{A}\,\overline{B}\,\overline{C}$
1	0	0	1	m_1	$\overline{A}\,\overline{B}\,C$
2	0	1	0	m_2	$\overline{A}\,B\,\overline{C}$
3	0	1	1	m_3	$\overline{A}\,B\,C$
4	1	0	0	m_4	$A\,\overline{B}\,\overline{C}$
5	1	0	1	m_5	$A\,\overline{B}\,C$
6	1	1	0	m_6	$A\,B\,\overline{C}$
7	1	1	1	m_7	$A\,B\,C$

กฎการอ่านสองบรรทัด: $X = \sum m$ ของ**ทุกแถวที่ X เป็น 1** · $\overline{X} = \sum m$ ของ**ทุกแถวที่ X เป็น 0** — สองรายการนี้รวมกันต้องได้ m_0 ถึง m_7 ครบ 8 ตัวพอดี ใช้เป็นการตรวจคำตอบได้ทันที

③ เดินทีละแถว ครบทั้ง 3 ข้อย่อย

a.) ตารางแรก

แถว	A	B	C	X	อ่านได้ว่า
0	0	0	0	**0**	m_0 เข้าฝั่ง $\overline{X}$
1	0	0	1	**1**	m_1 เข้าฝั่ง X

แถวเขียว (5 แถว) คือแถวที่ $X = 1 \rightarrow X = \sum m(1, 2, 3, 4, 5)$

แถวที่เหลือ (3 แถว) คือแถวที่ $X = 0 \rightarrow \overline{X} = \sum m(0, 6, 7)$

ตรวจ: $5 + 3 = 8$ ครบทุกแถว ไม่มีแถวไหนตกหล่นหรือถูกนับซ้ำ

แถว	A	B	C	X	อ่านได้ว่า
2	0	1	0	**1**	m_2 เข้าฝั่ง X
3	0	1	1	**1**	m_3 เข้าฝั่ง X
4	1	0	0	**1**	m_4 เข้าฝั่ง X
5	1	0	1	**1**	m_5 เข้าฝั่ง X
6	1	1	0	**0**	m_6 เข้าฝั่ง $\overline{X}$
7	1	1	1	**0**	m_7 เข้าฝั่ง $\overline{X}$

ถ้าโจทย์อยากได้รูปเต็ม ๆ ก็คือ

$$X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + A\overline{B}C$$

$$\overline{X} = \overline{A}\,\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

b.) ตารางที่สอง

แถว	A	B	C	X	อ่านได้ว่า
0	0	0	0	**0**	m_0 เข้าฝั่ง $\overline{X}$
1	0	0	1	**0**	m_1 เข้าฝั่ง $\overline{X}$
2	0	1	0	**1**	m_2 เข้าฝั่ง X
3	0	1	1	**1**	m_3 เข้าฝั่ง X
4	1	0	0	**0**	m_4 เข้าฝั่ง $\overline{X}$
5	1	0	1	**1**	m_5 เข้าฝั่ง X
6	1	1	0	**0**	m_6 เข้าฝั่ง $\overline{X}$
7	1	1	1	**1**	m_7 เข้าฝั่ง X

$$X = \sum m(2,3,5,7) \;\cdot\; \overline{X} = \sum m(0,1,4,6)$$

ตรวจ: $4 + 4 = 8$ ครบ

รูปเต็ม:

$$X = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

$$\overline{X} = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + AB\overline{C}$$

c.) ตารางที่สาม

แถว	A	B	C	X	อ่านได้ว่า
0	0	0	0	**1**	m_0 เข้าฝั่ง X
1	0	0	1	**0**	m_1 เข้าฝั่ง $\overline{X}$
2	0	1	0	**0**	m_2 เข้าฝั่ง $\overline{X}$
3	0	1	1	**0**	m_3 เข้าฝั่ง $\overline{X}$
4	1	0	0	**0**	m_4 เข้าฝั่ง $\overline{X}$
5	1	0	1	**1**	m_5 เข้าฝั่ง X
6	1	1	0	**1**	m_6 เข้าฝั่ง X
7	1	1	1	**0**	m_7 เข้าฝั่ง $\overline{X}$

$$X = \sum m(0,5,6) \;\cdot\; \overline{X} = \sum m(1,2,3,4,7)$$

ตรวจ: $3 + 5 = 8$ ครบ

รูปเต็ม:

$$X = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + AB\overline{C}$$

$$\overline{X} = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + ABC$$

④ ตอบ ข้อ 1

	$X(A,B,C)$	$\overline{X}(A,B,C)$
a.)	$\sum m(1,2,3,4,5)$	$\sum m(0,6,7)$
b.)	$\sum m(2,3,5,7)$	$\sum m(0,1,4,6)$
c.)	$\sum m(0,5,6)$	$\sum m(1,2,3,4,7)$

จุดที่มักพลาดข้อนี้

นับเลขแถวผิดเพราะเริ่มที่ 1 — **แถวบนสุดคือ m_0 ไม่ใช่ m_1** และเลขห้อยมาจากการอ่าน ABC เป็นเลขฐานสอง ไม่ใช่ลำดับที่นั่งในตาราง

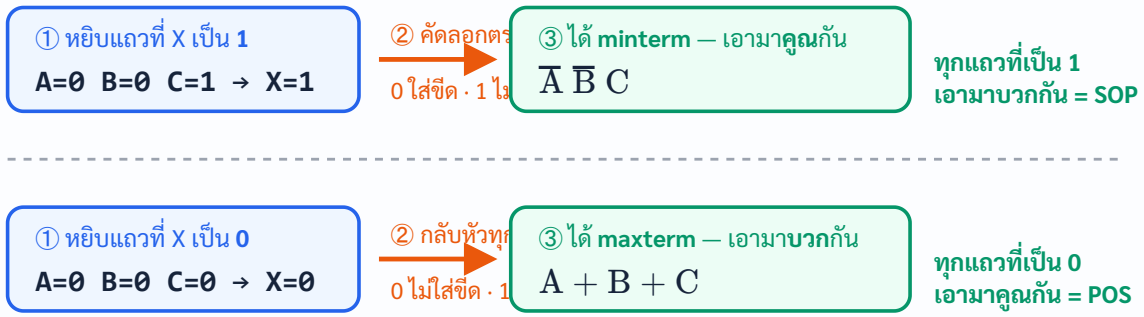
ข้อ 2 · แปลงตารางค่าความจริงเป็นนิพจน์ ทั้งแบบ SOP และ POS (a, b, c)

① โจทย์

“Convert the following truth tables to Boolean expressions in both SOP and POS forms.” แล้วให้ตาราง 3 ตัวแปรมา 3 ตาราง (a, b, c)

แปลไทย: **จงแปลงตารางค่าความจริงต่อไปนี้เป็นนิพจน์บูลีน ทั้งในรูป SOP และ POS** — SOP (sum-of-products) คือ “ผลบวกของผลคูณ” เช่น $AB + \overline{A}C$ · POS (product-of-sums) คือ “ผลคูณของผลบวก” เช่น $(A + B)(\overline{A} + C)$ โจทย์ต้องการ**ทั้งสองแบบของทุกตาราง** รวม 6 คำตอบ

② ใช้วิธีไหน ใช้ **การอ่าน minterm จากแถวที่เป็น 1 (ได้ SOP) และอ่าน maxterm จากแถวที่เป็น 0 (ได้ POS)** — บทที่ 3 หัวข้อ “ตารางเป็นนิพจน์: SOP กับ POS อ่านคนละฝั่งของตารางเดียวกัน”



กติกาการอ่านตารางสองทาง — ①②③ คือลำดับที่ต้องทำ (สีฟ้า = ของตั้งต้น · ส้ม = สิ่งที่กำลังทำ · เขียว = ผลลัพธ์)

ทำไม maxterm ต้องกลับหัว? เพราะ $(A + B + C)$ จะเป็น **0** ได้แถวเดียวเท่านั้น คือแถวที่ A, B, C เป็น 0 หมดพอดี — เราจึงใช้มัน “ปิด” แถวนั้นแถวเดียว แล้วเอาทุกก้อนมาคูณกันเพื่อปิดทุกแถวที่ต้องเป็น 0 ให้ครบ

③ เดินทีละแถว ครบทั้ง 3 ข้อย่อย

a.) ตารางแรก

แถว	A	B	C	X	เขียนเป็นก้อนอะไร
0	0	0	0	**0**	maxterm $A + B + C$
1	0	0	1	**1**	minterm $\overline{A}\,\overline{B}C$
2	0	1	0	**0**	maxterm $A + \overline{B} + C$
3	0	1	1	**0**	maxterm $A + \overline{B} + \overline{C}$
4	1	0	0	**0**	maxterm $\overline{A} + B + C$
5	1	0	1	**0**	maxterm $\overline{A} + B + \overline{C}$

SOP — เอาเฉพาะ 2 แถวเขียวมาบวกกัน

$$X = \sum m(1, 6) = \overline{A}\,\overline{B}C + AB\overline{C}$$

POS — เอา 6 แถวที่เหลือมาคูณกัน

$$X = \prod M(0, 2, 3, 4, 5, 7)$$
$$= (A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})$$
$$\times (\overline{A} + B + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$$

ตรวจ: $2 + 6 = 8$ แถวครบ · และเลขแถวของ SOP กับ POS ต้อง**ไม่ซ้ำกันเลย** — ดูแล้วไม่ซ้ำจริง

แถว	A	B	C	X	เขียนเป็นก้อนอะไร
6	1	1	0	**1**	minterm $AB\overline{C}$
7	1	1	1	**0**	maxterm $\overline{A}+\overline{B}+\overline{C}$

b.) ตารางที่สอง

แถว	A	B	C	X	เขียนเป็นก้อนอะไร
0	0	0	0	**1**	minterm $\overline{A}\,\overline{B}\,\overline{C}$
1	0	0	1	**0**	maxterm $A+B+\overline{C}$
2	0	1	0	**0**	maxterm $A+\overline{B}+C$
3	0	1	1	**0**	maxterm $A+\overline{B}+\overline{C}$
4	1	0	0	**1**	minterm $A\overline{B}\,\overline{C}$
5	1	0	1	**1**	minterm $A\overline{B}C$
6	1	1	0	**0**	maxterm $\overline{A}+\overline{B}+C$
7	1	1	1	**0**	maxterm $\overline{A}+\overline{B}+\overline{C}$

SOP

$$X=\sum m(0,4,5)=\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$$

POS

$$X=\prod M(1,2,3,6,7)$$
$$=(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})$$
$$\times(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$$

ตรวจ: $3+5=8$ ครบ

c.) ตารางที่สาม

แถว	A	B	C	X	เขียนเป็นก้อนอะไร
0	0	0	0	**0**	maxterm $A+B+C$
1	0	0	1	**1**	minterm $\overline{A}\,\overline{B}C$
2	0	1	0	**1**	minterm $\overline{A}B\overline{C}$
3	0	1	1	**0**	maxterm $A+\overline{B}+\overline{C}$
4	1	0	0	**1**	minterm $A\overline{B}\,\overline{C}$
5	1	0	1	**0**	maxterm $\overline{A}+B+\overline{C}$
6	1	1	0	**0**	maxterm $\overline{A}+\overline{B}+C$
7	1	1	1	**0**	maxterm $\overline{A}+\overline{B}+\overline{C}$

SOP

$$X=\sum m(1,2,4)=\overline{A}\,\overline{B}C+\overline{A}B\overline{C}+A\overline{B}\,\overline{C}$$

POS

$$X=\prod M(0,3,5,6,7)$$
$$=(A+B+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})$$
$$\times(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$$

ตรวจ: $3+5=8$ ครบ

④ ตอบ ข้อ 2

	SOP	POS
a.)	$\overline{A}\,\overline{B}C+AB\overline{C}$	$(A+B+C)(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$
b.)	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	$(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$

	SOP	POS
c.)	$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C}$	$(A+B+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$

ทั้ง 6 นิพจน์ถูกตรวจด้วยโปรแกรมแล้ว: แทนค่า A, B, C ครบ 8 ชุด ได้คอลัมน์ X ตรงกับตารางโจทย์ทุกแถว

จุดที่มักพลาดข้อนี้

เขียน POS โดย**ลืมกลับขีด** — แถว $ABC = 010$ ที่เป็น 0 ต้องได้ $A + \overline{B} + C$ (กลับหัว) ไม่ใช่ $\overline{A} + B + \overline{C}$ (ซึ่งเป็นวิธีของ minterm)

ข้อ 3 · สร้างตารางค่าความจริงจากนิพจน์ (a, b, c, d)

① โจทย์

“Construct the truth tables that are described by the following Boolean expressions:”

a.) $X = A\overline{B}C + B\overline{C} + ABC$ · b.) $X = EF + \overline{E}FG\overline{H} + H$

c.) $X = \overline{I}JK + IJ$ · d.) $X = LMN + L\overline{M}\,\overline{N} + \overline{L}M\overline{N}$

แปลไทย: **จงสร้างตารางค่าความจริงตามนิพจน์บูลีนต่อไปนี้** — ข้อนี้เดินกลับทางกับข้อ 2 คือมีนิพจน์แล้วต้องเติมตาราง

② ใช้วิธีไหน ใช้ **การแตกคอลัมน์ทีละพจน์แล้วค่อย OR รวม** — บทที่ 3 หัวข้อ “อ่านนิพจน์ SOP เป็นตาราง: ทำทีละพจน์ ห้ามคิดรวบ”

ขั้นตอนที่ใช้ซ้ำทั้งสี่ข้อย่อย

ขั้น ① นับตัวแปร → เขียนแถวให้ครบ 2^n แถว (a, c, d มี 3 ตัวแปร = 8 แถว · b มี 4 ตัวแปร = 16 แถว)

ขั้น ② ทำคอลัมน์ให้**ทีละพจน์** — แต่ละพจน์เป็นผลคูณ จึงเป็น 1 ได้เฉพาะแถวที่ “ตัวไม่มีขีดเป็น 1 และตัวมีขีดเป็น 0” ครบทุกตัวในพจน์นั้น

ขั้น ③ คอลัมน์ X = OR ของทุกคอลัมน์พจน์ — **มีพจน์ไหนเป็น 1 สักพจน์ก็พอ**

ขั้น ④ ตรวจกลับ: นับว่าแต่ละพจน์ทำให้กี่แถวเป็น 1 (พจน์ที่มีตัวอักษรครบ n ตัว จะเป็น 1 ได้**แถวเดียว** · ขาดไป 1 ตัวจะเป็น 1 ได้ 2 แถว)

a.) $X = A\overline{B}C + B\overline{C} + ABC$ (3 ตัวแปร → 8 แถว)

A	B	C	$A\overline{B}C$	$B\overline{C}$	ABC	X
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	1	0	1
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	0	1
1	1	0	0	1	0	1
1	1	1	0	0	1	1

ไล่ทีละพจน์:

$A\overline{B}C$ มีครบ 3 ตัว → เป็น 1 **แถวเดียว** คือแถว $ABC = 101$

$B\overline{C}$ มี 2 ตัว (ขาด A) → เป็น 1 **สองแถว** คือ 010 และ 110

ABC มีครบ 3 ตัว → เป็น 1 แถวเดียว คือ 111

ตรวจกลับ: $1 + 2 + 1 = 4$ ครั้งที่มีพจน์เป็น 1 และไม่มีแถวไหนถูกนับซ้ำ → ในคอลัมน์ X ต้องมีเลข 1 พอดี **4 แถว** — นับได้ 4 แถวจริง

คำตอบ: คอลัมน์ X = `0 0 1 0 0 1 1 1` หรือ $X = \sum m(2, 5, 6, 7)$

b.) $X = EF + \overline{E}FG\overline{H} + H$ (4 ตัวแปร → 16 แถว)

ไล่ทีละพจน์:

EF มี 2 ตัว (ขาด G, H) → เป็น 1 **4 แถว** คือทุกแถวที่ $EF = 11$

E	F	G	H	EF	$\overline{E}FG\overline{H}$	H	X
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1

$\overline{E}FG\overline{H}$ มีครบ 4 ตัว → เป็น 1 **แถวเดียว** คือ $EFGH = 0110$

H มีตัวเดียว (ขาด 3 ตัว) → เป็น 1 **8 แถว** คือทุกแถวที่ $H = 1$

ตรวจกลับ: แถวที่ $H = 1$ มี 8 แถว บวกแถวที่ $EF = 11$ แต่ $H = 0$ อีก 2 แถว (1100, 1110) บวกแถว 0110 อีก 1 แถว รวม $8 + 2 + 1 = 11$ แถวที่เป็น 1 — นับในคอลัมน์ X ได้ 11 แถวพอดี

คำตอบ: $X = \sum m(1, 3, 5, 6, 7, 9, 11, 12, 13, 14, 15)$

c.) $X = \overline{I}JK + IJ$ **(3 ตัวแปร → 8 แถว)**

I	J	K	$\overline{I}JK$	IJ	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	1	1

ไล่ทีละพจน์:

$\overline{I}JK$ ครบ 3 ตัว → 1 แถวเดียว คือ $IJK = 011$

IJ ขาด K → 1 สองแถว คือ 110 และ 111

ตรวจกลับ: $1 + 2 = 3$ แถว — คอลัมน์ X มี 1 อยู่ 3 แถวพอดี

คำตอบ: $X = \sum m(3, 6, 7)$

ของแถม: สังเกตจากตารางว่า X เป็น 1 เฉพาะแถวที่ $J = 1$ และ ($I = 1$ หรือ $K = 1$) → ลดรูปได้เป็น $X = IJ + JK$ (ตรวจด้วยโปรแกรมแล้วว่าเท่ากันทั้ง 8 แถว) — ข้อนี้ไม่ได้ถามให้ลดรูป แต่ถ้าข้อสอบถามต่อ ก็อ่านจากตารางได้เลย

d.) $X = LMN + L\overline{M}\,\overline{N} + \overline{L}M\overline{N}$ **(3 ตัวแปร → 8 แถว)**

L	M	N	LMN	$L\overline{M}\,\overline{N}$	$\overline{L}M\overline{N}$	X
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	1
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	1

ไล่ทีละพจน์: ทั้งสามพจน์มีตัวอักษรครบ 3 ตัว → แต่ละพจน์เป็น 1 ได้**แถวเดียว** เท่านั้น

LMN → แถว 111 · $L\overline{M}\,\overline{N}$ → แถว 100 · $\overline{L}M\overline{N}$ → แถว 010

ตรวจกลับ: สามพจน์ สามแถว ไม่ซ้ำกัน → คอลัมน์ X ต้องมี 1 พอดี 3 แถว — ได้ 3 แถวจริง

คำตอบ: $X = \sum m(2, 4, 7)$

ระวังแถว 110**:** หลายคนเผลอเติม 1 เพราะเห็น L กับ M เป็น 1 — แต่ LMN ต้องการ $N = 1$ ด้วย และ $\overline{L}M\overline{N}$ ต้องการ $L = 0$ แถวนี้จึงเป็น **0**

④ ตอบ ข้อ 3

	คอลัมน์ X จากบนลงล่าง	เขียนเป็น minterm
a.)	0 0 1 0 0 1 1 1	$X = \sum m(2, 5, 6, 7)$
b.)	0 1 0 1 0 1 1 1 0 1 0 1 1 1 1 1	$X = \sum m(1, 3, 5, 6, 7, 9, 11, 12, 13, 14, 15)$
c.)	0 0 0 1 0 0 1 1	$X = \sum m(3, 6, 7)$
d.)	0 0 1 0 1 0 0 1	$X = \sum m(2, 4, 7)$

จุดที่มักพลาดข้อนี้

เติม 1 ให้แถวที่ “ตัวแปรในพจน์เป็น 1 บางตัว” — พจน์เป็นผลคูณ ต้อง**ครบทุกตัวอักษรในพจน์นั้น** (ตัวมีขีดต้องเป็น 0 ด้วย) ถึงจะได้ 1

ข้อ 4 · ออกแบบนิพจน์ 3 ตัวแปร ที่ออก HIGH เมื่ออินพุตส่วนใหญ่เป็น HIGH

① โจทย์

“Design a three variable Boolean expression whose output is HIGH when a majority of the inputs are HIGH. Express the logic in both SOP and POS forms.” พร้อมตารางเปล่า 8 แถว ($A, B, C \rightarrow X$) ให้เติม

แปลไทย: **จงออกแบบนิพจน์บูลีน 3 ตัวแปร ที่เอาต์พุตเป็น HIGH เมื่ออินพุต “ส่วนใหญ่” เป็น HIGH และเขียนคำตอบทั้งในรูป SOP และ POS**

② ใช้วิธีไหน ใช้ **แปลคำพูดเป็นตารางก่อน แล้วค่อยอ่านตารางเป็น SOP/POS** (วิธีเดียวกับข้อ 2) — บทที่ 3 หัวข้อ “จากโจทย์คำพูดสู่ตาราง” และหัวข้อ “วงจรสองชั้น AND–OR”

③ เดินทีละขั้น

1

แปลคำว่า “majority” ให้เป็นตัวเลขก่อน

อินพุตมี 3 ตัว “เสียงข้างมาก” คือ**มากกว่าครึ่ง** ของ 3 ซึ่งก็คือ **ตั้งแต่ 2 ตัวขึ้นไป** (2 ตัวหรือ 3 ตัว) — เพราะ 3 เป็นเลขคี่ จึงไม่มีกรณีเสมอ ไม่ต้องตัดสินใจอะไรเพิ่ม

เงื่อนไขที่จะใช้จริงจึงเป็นประโยคเดียว: $X = 1$ **เมื่อจำนวนอินพุตที่เป็น 1 มีตั้งแต่ 2 ตัวขึ้นไป**

2

ไล่ทั้ง 8 แถว นับจำนวนเลข 1 ในแต่ละแถว

ไม่ต้องคิดนิพจน์ตอนนี้ แค่ “นับ” อย่างเดียว แล้วเติม X ตามเกณฑ์ในขั้นที่ 1

3

อ่านแถวที่เป็น 1 → SOP · อ่านแถวที่เป็น 0 → POS

ใช้กติกาเดียวกับข้อ 2 เป๊ะ ๆ แล้วค่อยลดรูป SOP ให้สั้นลงถ้าทำได้

แถว	A	B	C	จำนวน 1	X	เขียนเป็นก้อนอะไร
0	0	0	0	0 ตัว	0	maxterm $A+B+C$
1	0	0	1	1 ตัว	0	maxterm $A+B+\overline{C}$
2	0	1	0	1 ตัว	0	maxterm $A+\overline{B}+C$
3	0	1	1	2 ตัว	1	minterm $\overline{A}BC$
4	1	0	0	1 ตัว	0	maxterm $\overline{A}+B+C$
5	1	0	1	2 ตัว	1	minterm $A\overline{B}C$
6	1	1	0	2 ตัว	1	minterm $AB\overline{C}$
7	1	1	1	3 ตัว	1	minterm ABC

SOP (ยังไม่ลดรูป)

$$X = \sum m(3,5,6,7)$$
$$= \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

POS

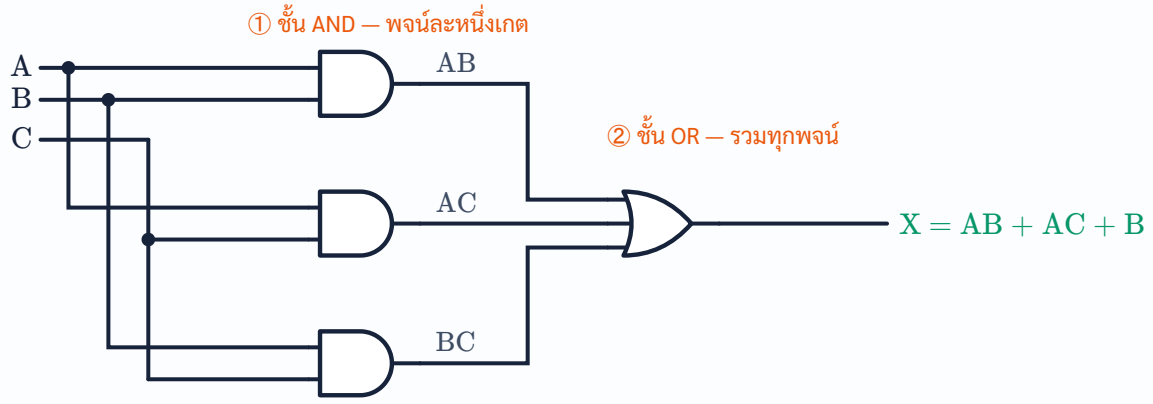
$$X = \prod M(0,1,2,4)$$
$$= (A+B+C)(A+B+\overline{C})$$
$$\times(A+\overline{B}+C)(\overline{A}+B+C)$$

ตรวจ: $4+4=8$ แถวครบ · และถ้ามองแบบนับ: แถวที่มี 1 อยู่ 2 ตัวมี 3 แถว แถวที่มี 3 ตัวมี 1 แถว รวม 4 แถว — ตรงกับจำนวน minterm

ลดรูป SOP ให้สั้นลง (ทำต่ออีกนิดเดียว ได้วงจรเล็กลงมาก)

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$	SOP ที่อ่านจากตารางตรง ๆ
②	$= \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC + ABC + ABC$	**กฎ idempotent** $X = X + X + X$ — คัดลอกพจน์ ABC เพิ่มอีกสองชุด (บวกตัวเองซ้ำไม่เปลี่ยนค่า) เพื่อเอาไปจับคู่กับพจน์อื่น
③	$= (\overline{A}BC + ABC) + (A\overline{B}C + ABC) + (AB\overline{C} + ABC)$	จัดกลุ่มใหม่ — **กฎการสลับที่/จัดหมู่** ของการบวก
④	$= BC(\overline{A}+A) + AC(\overline{B}+B) + AB(\overline{C}+C)$	**ดึงตัวร่วม** ในแต่ละวงเล็บ
⑤	$= BC\cdot 1 + AC\cdot 1 + AB\cdot 1 = AB + AC + BC$	**กฎ complement** $\overline{X}+X=1$ แล้ว $X\cdot 1 = X$ **SOP รูปสั้น**

เช็กกลับเข้าโจทย์: $AB+AC+BC$ อ่านเป็นภาษาคนได้ว่า “มีคู่ไหนสักคู่ที่เป็น 1 พร้อมกัน” — ถ้ามี 1 อยู่ 2 ตัวขึ้นไป ย่อมมีอย่างน้อยหนึ่งคู่เสมอ และถ้ามี 1 ตัวเดียวหรือไม่มีเลย ก็ไม่มีคู่ไหนครบ ตรงกับนิยามของ majority พอดี (โปรแกรมตรวจแล้วว่าให้ค่าตรงกับตารางทั้ง 8 แถว)

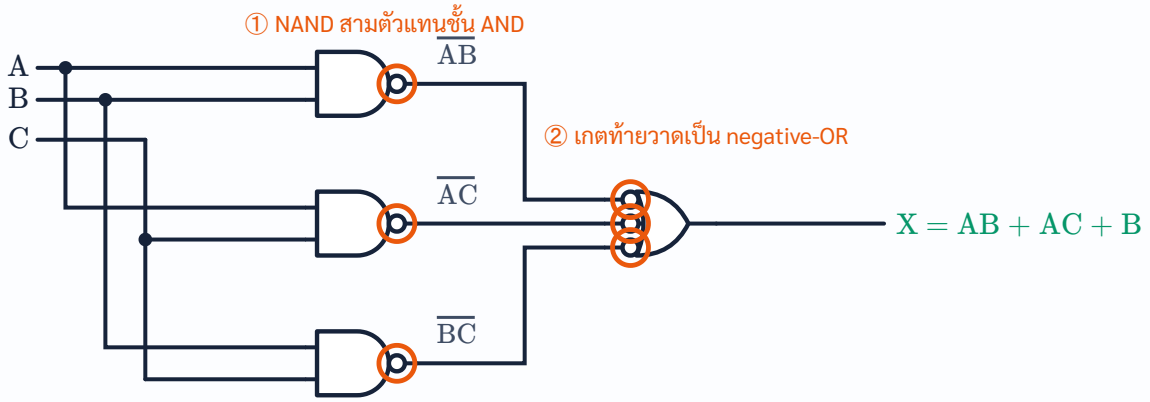


วงจรสองชั้น AND–OR ของ $X = AB + AC + BC$ — สายทุกเส้นมีชื่อกำกับ จุดดำคือจุดแยกสาย (ไม่ใช่จุดตัดกันเฉย ๆ)

ของแถมที่ข้อสอบชอบถามต่อ: วงจรเดียวกันนี้ด้วย NAND ล้วน

เกต NAND เป็น **เกตสากล (universal gate)** — สร้างเกตอื่นได้หมด: $\overline{A} = \overline{A \cdot A}$ (ป้อนสัญญาณเดียวเข้าทั้งสองขา) · $AB = \overline{\overline{AB} \cdot \overline{AB}}$ · $A + B = \overline{\overline{A} \cdot \overline{B}}$ (ตรวจด้วยโปรแกรมแล้วทั้งสามอัน) เพราะแบบนี้วงจร SOP สองชั้น AND–OR จึงแปลงเป็น **NAND–NAND** ได้ทันที โดยไม่ต้องคิดใหม่เลย

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$X = AB + AC + BC$	วงจร AND–OR เดิม
②	$X = \overline{\overline{AB + AC + BC}}$	**กฎขีดสองชั้น** — เติมขีดสองชั้นคลุมทั้งก้อน ค่าไม่เปลี่ยน (นี่คือกลเม็ดของการแปลงเป็น NAND)
③	$X = \overline{\overline{AB} \cdot \overline{AC} \cdot \overline{BC}}$	**เดอมอร์แกนสูตร 1** กับขีดชั้นใน — ผลบวกสามก้อนกลายเป็นผลคูณของสามก้อนที่มีขีดของตัวเอง
④	$X = \text{NAND}(\overline{AB},\ \overline{AC},\ \overline{BC})$	อ่านบรรทัด ③ เป็นเกต: ข้างในคือ NAND 2 ขา สามตัว ข้างนอกคือ NAND 3 ขาอีกหนึ่งตัว **วงจร NAND ล้วน 4 เกต**



③ วงกลมส้ม 6 วง = bubble ที่มาเจอกันแล้วหักล้างกันพอดีทีละคู่ — จึงอ่านผลลัพธ์ได้เป็น AND แล้ว OR เหมือนวงจรเดิ

วงจร NAND ล้วนของ $X = AB + AC + BC$ — เกตท้ายคือ NAND 3 ขา แต่**วาดด้วย dual symbol** (negative-OR) เพื่อให้เห็น bubble หักล้างกันกับ bubble ขาออกของ NAND สามตัวหน้า

④ ตอบ ข้อ 4

ตาราง คอลัมน์ X จากบนลงล่าง = `0 0 0 1 0 1 1 1`

$$\textbf{SOP}: \; X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \; = \; \mathbf{AB + AC + BC}$$

$$\textbf{POS}: \; X = (\mathbf{A + B + C})(\mathbf{A + B + \overline{C}})(\mathbf{A + \overline{B} + C})(\mathbf{\overline{A} + B + C})$$

(ถ้าอาจารย์ขอ “SOP ตามตาราง” ให้เขียนรูปสี่พจน์ · ถ้าขอ “รูปที่สั้นที่สุด” ให้เขียน $AB + AC + BC$ — ทั้งสองถูกและให้ค่าเท่ากันทุกแถว)

จุดที่มักพลาดข้อนี้

ตีความ “majority” ว่า “ทุกตัวเป็น HIGH” (นั่นคือ ABC อย่างเดียว) หรือ “อย่างน้อยหนึ่งตัว” (นั่นคือ OR) — majority ของ 3 ตัวคือ **ตั้งแต่ 2 ตัวขึ้นไป** ต้องได้ 1 ทั้งหมด 4 แถว ไม่ใช่ 1 แถวและไม่ใช่ 7 แถว

ข้อ 5 · นิพจน์ 4 ตัวแปร ที่ออก HIGH เมื่อจำนวน HIGH กับจำนวน LOW ไม่เท่ากัน

① โจทย์

“Derive a four variable Boolean expression whose output is HIGH when the number of HIGH inputs and the number of LOW inputs are NOT equal. Express the logic in both SOP and POS forms.” พร้อมตารางเปล่า 16 แถว ($A, B, C, D \rightarrow X$)

แปลไทย: **จงหานิพจน์บูลีน 4 ตัวแปร ที่เอาต์พุตเป็น HIGH เมื่อ “จำนวนอินพุตที่เป็น HIGH” กับ “จำนวนอินพุตที่เป็น LOW” ไม่เท่ากัน และเขียนทั้งในรูป SOP และ POS**

② ใช้วิธีไหน ใช้ **แปลคำพูดเป็นเงื่อนไขนับเลขก่อน แล้วค่อยอ่านตารางเป็น SOP/POS** (วิธีเดียวกับข้อ 4) — บทที่ 3 หัวข้อ “จากโจทย์คำพูดสู่ตาราง” และหัวข้อ “เลือกฝั่งที่สั้นกว่า: SOP หรือ POS”

③ เดินทีละขั้น

1 เปลี่ยนโจทย์ให้เหลือเงื่อนไขเดียว

อินพุตมี 4 ตัว ถ้ามี HIGH อยู่ h ตัว แล้ว LOW ย่อมมี $4 - h$ ตัวเสมอ (สองจำนวนนี้บวกกันได้ 4 ตลอด)

“เท่ากัน” จึงแปลว่า $h = 4 - h$ นั่นคือ $2h = 4$ นั่นคือ $h = 2$ **พอดีเท่านั้น**

โจทย์ถาม “ไม่เท่ากัน” → $X = 0$ **เฉพาะแถวที่มีเลข 1 อยู่ 2 ตัวพอดี · แถวอื่นเป็น 1 หมด**

2 นับก่อนว่ามีกี่แถวที่ “มี 1 อยู่ 2 ตัวพอดี”

เลือก 2 ตำแหน่งจาก 4 ตำแหน่ง ได้ $\frac{4 \times 3}{2} = 6$ แบบ — คือ 6 แถว ได้แก่ $0011,\ 0101,\ 0110,\ 1001,\ 1010,\ 1100$

เพราะฉะนั้นในตาราง 16 แถว ต้องมีเลข 0 อยู่ **6 แถว** และเลข 1 อยู่ **10 แถว** — ตัวเลขนี้ใช้ตรวจตัวเองตอนเติมเสร็จ

3 อ่านออกมาทั้งสองแบบ แล้วเลือกใช้แบบที่สั้นกว่า

SOP ต้องเขียน 10 พจน์ · POS ต้องเขียน 6 ก้อน — **ข้อนี้ POS สั้นกว่าเกือบครึ่ง** เพราะฝั่งที่เป็น 0 มีน้อยกว่า (โจทย์ขอทั้งสองแบบ จึงต้องเขียนให้ครบ แต่ถ้าต้องต่อวงจรจริงให้เลือกฝั่ง POS)

แถว	A	B	C	D	HIGH	LOW	เท่ากันไหม	X	เขียนเป็นก้อนอะไร
0	0	0	0	0	0	4	ไม่เท่า	1	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$
1	0	0	0	1	1	3	ไม่เท่า	1	$\overline{A}\,\overline{B}\,\overline{C}D$
2	0	0	1	0	1	3	ไม่เท่า	1	$\overline{A}\,\overline{B}C\overline{D}$
3	0	0	1	1	2	2	**เท่ากัน**	0	$A + B + \overline{C} + \overline{D}$
4	0	1	0	0	1	3	ไม่เท่า	1	$\overline{A}B\overline{C}\,\overline{D}$

แถว	A	B	C	D	HIGH	LOW	เท่ากันไหม	X	เขียนเป็นก้อนอะไร
5	0	1	0	1	2	2	**เท่ากัน**	0	$A+\overline{B}+C+\overline{D}$
6	0	1	1	0	2	2	**เท่ากัน**	0	$A+\overline{B}+\overline{C}+D$
7	0	1	1	1	3	1	ไม่เท่า	1	$\overline{A}BCD$
8	1	0	0	0	1	3	ไม่เท่า	1	$A\overline{B}\,\overline{C}\,\overline{D}$
9	1	0	0	1	2	2	**เท่ากัน**	0	$\overline{A}+B+C+\overline{D}$
10	1	0	1	0	2	2	**เท่ากัน**	0	$\overline{A}+B+\overline{C}+D$
11	1	0	1	1	3	1	ไม่เท่า	1	$A\overline{B}CD$
12	1	1	0	0	2	2	**เท่ากัน**	0	$\overline{A}+\overline{B}+C+D$
13	1	1	0	1	3	1	ไม่เท่า	1	$AB\overline{C}D$
14	1	1	1	0	3	1	ไม่เท่า	1	$ABC\overline{D}$
15	1	1	1	1	4	0	ไม่เท่า	1	$ABCD$

เช็กกลับเข้าโจทย์: นับเลข 0 ในคอลัมน์ X ได้ **6 แถว** ตรงกับที่คำนวณไว้ในขั้นที่ 2 และทั้ง 6 แถวนั้น คอลัมน์ HIGH กับ LOW เป็น 2 กับ 2 เหมือนกันหมด · เลข 1 มี $16-6=10$ แถว

④ ตอบ ข้อ 5

ตาราง คอลัมน์ X จากบนลงล่าง = `1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1`

SOP $X=\sum m(0,1,2,4,7,8,11,13,14,15)$

$$X=\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}+\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}\,\overline{B}C\overline{D}+\overline{A}B\overline{C}\,\overline{D}+\overline{A}BCD$$
$$+A\overline{B}\,\overline{C}\,\overline{D}+A\overline{B}CD+AB\overline{C}D+ABC\overline{D}+ABCD$$

POS $X=\prod M(3,5,6,9,10,12)$

$$X=(A+B+\overline{C}+\overline{D})(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$
$$\times(\overline{A}+B+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+\overline{B}+C+D)$$

อ่านคำตอบให้เข้าใจ ไม่ใช่แค่จำ

ก้อน POS แต่ละก้อนคือ “ประตูปิด” ของแถวที่ต้องเป็น 0 หนึ่งแถวพอดี เช่น $(A+B+\overline{C}+\overline{D})$ จะเป็น 0 ได้เฉพาะตอน $A=0,B=0,C=1,D=1$ — ซึ่งก็คือแถว 0011 ที่มี 1 อยู่สองตัวพอดี มี 6 แถวที่ต้องปิด จึงต้องมี 6 ก้อนคูณกัน ไม่ขาดไม่เกิน

จุดที่มักพลาดข้อนี้

ตีความ “not equal” เป็นการเทียบค่าอินพุตกันเอง (เช่น $A\neq B$) — โจทย์เทียบ**จำนวนนับ** ของ HIGH กับ LOW ซึ่งในกรณี 4 ตัวแปร จะแปลว่า “เป็น 0 เฉพาะแถวที่มีเลข 1 อยู่ 2 ตัวพอดี” เท่านั้น

ข้อ 6 · วาดวงจรใหม่ด้วย dual symbol และหานิพจน์เอาต์พุต (a, b)

① โจทย์

“Redraw the circuits below and develop the output expressions using appropriate dual symbols.” แล้วให้วงจรมาสองวง — วง a ประกอบด้วยเกต **NAND ล้วน 5 ตัว** ชื่อ G_1 ถึง G_5 · วง b เป็นวงจรที่ต่อเหมือนกันทุกประการ แต่เปลี่ยนเป็นเกต **NOR ล้วน**

แปลไทย: **จงวาดวงจรข้างล่างนี้ใหม่ และหานิพจน์เอาต์พุต โดยใช้สัญลักษณ์คู่ (dual symbol) ให้เหมาะสม** — “เหมาะสม” หมายถึงวาดให้ bubble ที่**ขาออก**ของเกตหนึ่ง ไปเจอ bubble ที่**ขาเข้า**ของเกตถัดไป เพื่อให้มันหักล้างกัน แล้วอ่านนิพจน์ออกมาได้ตรง ๆ โดยไม่ต้องใช้พีชคณิตช่วย

② ใช้วิธีไหน ใช้ **การวิเคราะห์วงจรทีละเกต (ตั้งชื่อทุกสายก่อน)** แล้วตามด้วย **การวาดใหม่ด้วย dual symbol** — บทที่ 3 หัวข้อ “วิเคราะห์วงจรผสม: ตั้งชื่อสายแล้วไล่จากซ้ายไปขวา” และหัวข้อ “dual symbol กับการหักล้างของ bubble”

สองข้อเท็จจริงที่ใช้ตลอดข้อ 6

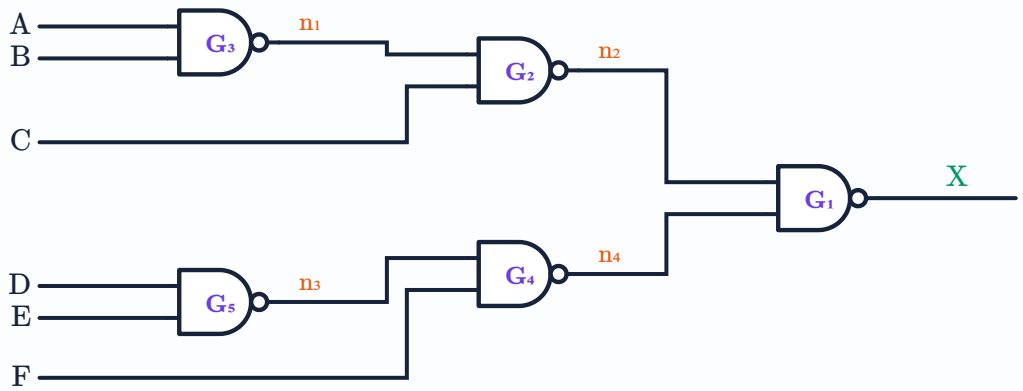
1. เกตหนึ่งตัววาดได้สองหน้าตา (ดูรูปในข้อ 3 ของ Assignment #3 ประกอบ)

NAND = รูป AND + bubble ขาออก = รูป OR + bubble ขาเข้าทุกขา (เรียกว่า **negative-OR**)

NOR = รูป OR + bubble ขาออก = รูป AND + bubble ขาเข้าทุกขา (เรียกว่า **negative-AND**)

2. bubble สองลูกที่ต่อชนกันบนสายเดียวกัน = กลับค่าสองครั้ง = ไม่กลับเลย — ลบทิ้งได้ทั้งคู่ (นี่คือกฎขีดสองชั้น $\overline{\overline{X}} = X$ ในรูปของวงจร)

a.) วงจร NAND ล้วน — ขั้นแรก: ตั้งชื่อทุกสายแล้วไล่ทีละเกต



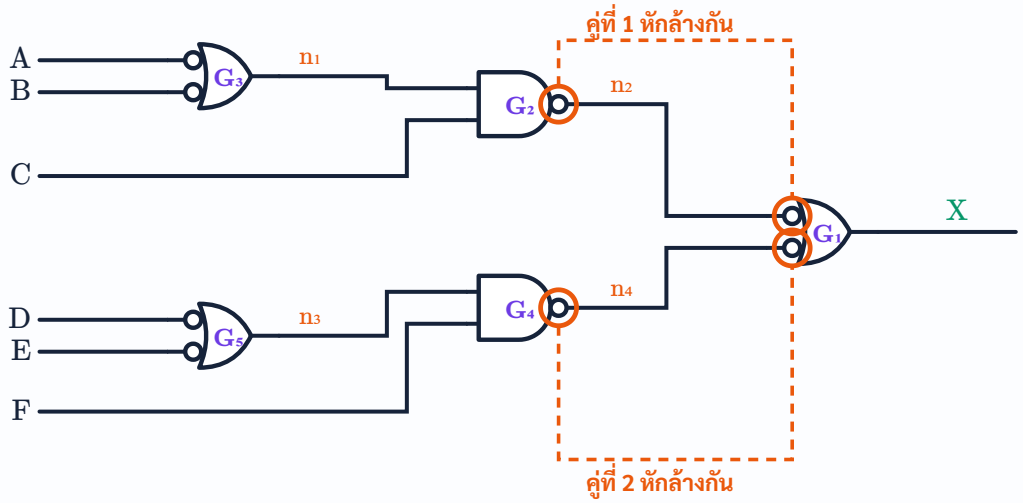
วงจร a ตามโจทย์ — ตั้งชื่อสายภายในว่า n_1, n_2, n_3, n_4 (สีส้ม) สายขาเข้าคือ A ถึง F สายขาออกคือ X · เกตทั้ง 5 ตัวเป็น NAND

สาย	ออกจากเกต	อินพุตของเกตนั้น	นิพจน์ที่สายนี้พา (เขียนจากนิยามของ NAND: คูณกันแล้วกลับค่า)
n_1	G_3	A, B	$n_1 = \overline{AB}$
n_2	G_2	n_1, C	$n_2 = \overline{n_1\, C} = \overline{\overline{AB}\, C}$
n_3	G_5	D, E	$n_3 = \overline{DE}$
n_4	G_4	n_3, F	$n_4 = \overline{n_3\, F} = \overline{\overline{DE}\, F}$
X	G_1	n_2, n_4	$X = \overline{n_2\, n_4}$

a.) ขั้นที่สอง: วาดใหม่ด้วย dual symbol ให้ bubble หักล้างกัน

ดูที่ G_1: ขาเข้าทั้งสองของมันรับสายที่**ออกมาจาก bubble** (n_2 และ n_4 ออกจาก bubble ของ G_2 และ G_4) ถ้าเราวาด G_1 เป็น **negative-OR** (รูป OR ที่มี bubble ที่ขาเข้า) bubble ทั้งสองคู่จะมาชนกันพอดีแล้วหักล้างกันไป

ทำแบบเดียวกันกับ G_3 และ G_5: วาดเป็น negative-OR เพื่อให้ขาออกของมัน**ไม่มี bubble** จะได้ไปเข้าขาเข้าธรรมดาของ G_2/G_4 ได้พอดี ส่วน G_2 กับ G_4 คงรูป NAND (AND + bubble ขาออก) ไว้เหมือนเดิม



วงจร a ที่วาดใหม่ — G_3, G_5, G_1 เปลี่ยนเป็นสัญลักษณ์ negative-OR ส่วน G_2, G_4 คงเป็น NAND · วงกลมส้มสี่วงคือ bubble ที่มาชนกันเป็นคู่ (เส้นประโยงให้เห็นว่าคู่ไหนกับคู่ไหน) — หักล้างกันหมด เหลือเป็นสายตรง

③ อ่านนิพจน์ออกมาทีละบรรทัด

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$X = \overline{n_2\, n_4}$	นิยามของ NAND ที่ G_1 (จากตารางไล่เกตด้านบน)
②	$= \overline{n_2} + \overline{n_4}$	**เดอมอร์แกนสูตร 2** — ในรูปวงจรคือ “อ่าน G_1 เป็น negative-OR” นั่นเอง
③	$= \overline{\overline{AB}\, C} + \overline{\overline{DE}\, F}$	แทนค่า n_2 และ n_4 ลงไป

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
④	$= \overline{AB}\,C \;+\; \overline{DE}\,F$	**กฎขีดสองชั้น** — bubble ขาออกของ G_2 หักล้างกับ bubble ขาเข้าของ G_1 พอดี (คู่วงกลมส้มในรูป)
⑤	$= (\overline{A}+\overline{B})\,C \;+\; (\overline{D}+\overline{E})\,F$	**เดอมอร์แกนสูตร 2** กับ $\overline{AB}$ และ $\overline{DE}$ — คืออ่าน G_3, G_5 เป็น negative-OR **คำตอบข้อ a**
⑥	$= \overline{A}C + \overline{B}C + \overline{D}F + \overline{E}F$	**กฎการแจกแจง** — รูป SOP ถ้าอาจารย์ขอให้กระจาย

เช็กกลับเข้าวงจร — ไล่สายจริงหนึ่งชุด ลองป้อน $A=1,\ B=0,\ C=1,\ D=1,\ E=1,\ F=0$

สาย	คิดจาก	ค่า
n_1	$\overline{A\cdot B} = \overline{1\cdot 0} = \overline{0}$	1
n_2	$\overline{n_1\cdot C} = \overline{1\cdot 1} = \overline{1}$	0
n_3	$\overline{D\cdot E} = \overline{1\cdot 1} = \overline{1}$	0
n_4	$\overline{n_3\cdot F} = \overline{0\cdot 0} = \overline{0}$	1
X	$\overline{n_2\cdot n_4} = \overline{0\cdot 1} = \overline{0}$	1

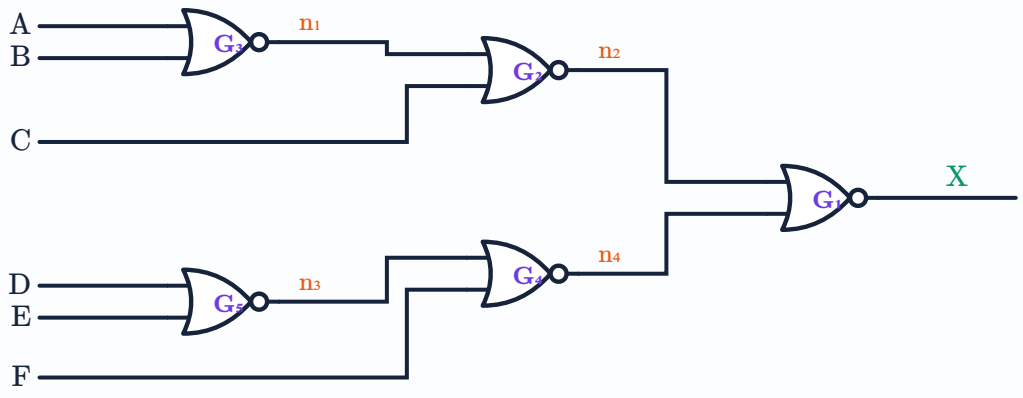
เทียบกับคำตอบ: $(\overline{A}+\overline{B})C + (\overline{D}+\overline{E})F = (0+1)\cdot 1 + (0+0)\cdot 0 = 1+0 = 1$ — **ตรงกัน** (โปรแกรมตรวจครบทั้ง 64 ชุดอินพุตแล้วว่าตรงกันทุกชุด)

④ ตอบ ข้อ 6a

$$X = (\overline{\mathbf{A}}+\overline{\mathbf{B}})\,\mathbf{C} \;+\; (\overline{\mathbf{D}}+\overline{\mathbf{E}})\,\mathbf{F}$$

กระจายแล้วได้ $X = \overline{A}C + \overline{B}C + \overline{D}F + \overline{E}F$ · การวาดใหม่: G_3, G_5, G_1 วาดเป็น negative-OR · G_2, G_4 คงเป็น NAND

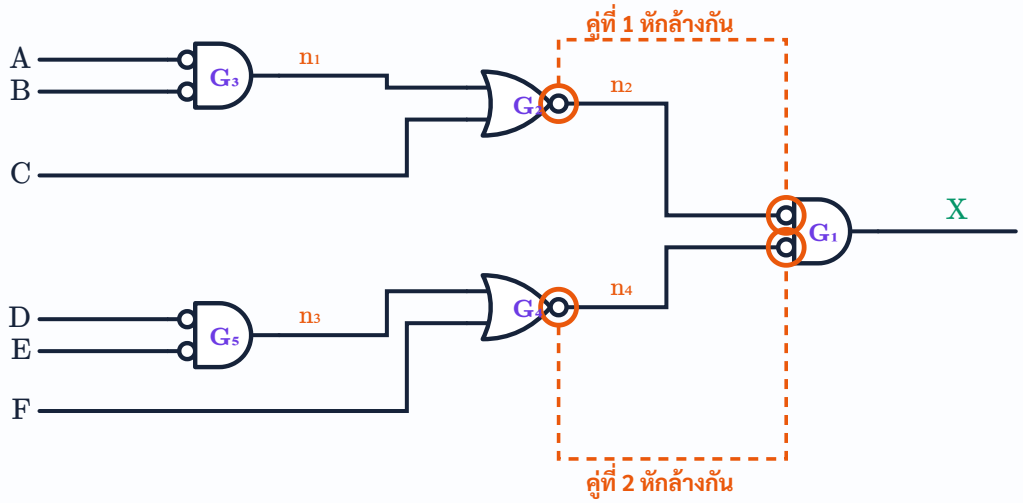
b.) วงจร NOR ล้วน — ต่อเหมือนกันทุกจุด เปลี่ยนแค่ชนิดเกต



วงจร b ตามโจทย์ — ผังการต่อสายเหมือนวง a ทุกประการ ต่างกันที่เกตทั้ง 5 ตัวเป็น NOR (รูป OR + bubble ขาออก)

สาย	ออกจากเกต	อินพุตของเกตนั้น	นิพจน์ที่สายนี้พา (นิยามของ NOR: บวกกันแล้วกลับค่า)
n_1	G_3	A, B	$n_1 = \overline{A + B}$
n_2	G_2	n_1, C	$n_2 = \overline{n_1 + C} = \overline{\overline{A + B} + C}$
n_3	G_5	D, E	$n_3 = \overline{D + E}$
n_4	G_4	n_3, F	$n_4 = \overline{n_3 + F} = \overline{\overline{D + E} + F}$
X	G_1	n_2, n_4	$X = \overline{n_2 + n_4}$

วาดใหม่ด้วยหลักเดิมเป๊ะ ๆ แต่สลับข้าง: คราวนี้ dual symbol ของ NOR คือ **negative-AND** — G_3, G_5, G_1 วาดเป็น negative-AND (รูป AND + bubble ขาเข้า) ส่วน G_2, G_4 คงเป็น NOR



วงจร b ที่วาดใหม่ — G_3, G_5, G_1 เป็น negative-AND · G_2, G_4 คงเป็น NOR · วงกลมส้มคือ bubble สองคู่ที่มาชนกันแล้วหักล้างกันไป

ขั้น	บรรทัด	บรรทัดนี้ทำอะไร · ใช้กฎชื่ออะไร
①	$X = \overline{n_2 + n_4}$	นิยามของ NOR ที่ G_1
②	$= \overline{n_2} \cdot \overline{n_4}$	**เดอมอร์แกนสูตร 1** — ในรูปวงจรคือ “อ่าน G_1 เป็น negative-AND”
③	$= (n_1 + C)(n_3 + F)$	**กฎขีดสองชั้น** — $\overline{n_2} = \overline{\overline{n_1 + C}} = n_1 + C$ (bubble ขาออกของ G_2 หักล้างกับ bubble ขาเข้าของ G_1)
④	$= (\overline{A + B} + C)(\overline{D + E} + F)$	แทนค่า n_1 และ n_3 ลงไป
⑤	$= (\overline{A}\,\overline{B} + C)(\overline{D}\,\overline{E} + F)$	**เดอมอร์แกนสูตร 1** กับ $\overline{A + B}$ และ $\overline{D + E}$ — คืออ่าน G_3, G_5 เป็น negative-AND **คำตอบข้อ b**

เช็กกลับเข้าวงจร — ไล่สายจริงชุดเดิม $A = 1,\ B = 0,\ C = 1,\ D = 1,\ E = 1,\ F = 0$

สาย	คิดจาก	ค่า
n_1	$\overline{A+B}=\overline{1+0}=\overline{1}$	0
n_2	$\overline{n_1+C}=\overline{0+1}=\overline{1}$	0
n_3	$\overline{D+E}=\overline{1+1}=\overline{1}$	0
n_4	$\overline{n_3+F}=\overline{0+0}=\overline{0}$	1
X	$\overline{n_2+n_4}=\overline{0+1}=\overline{1}$	0

เทียบกับคำตอบ: $(\overline{A}\,\overline{B}+C)(\overline{D}\,\overline{E}+F)=(0\cdot 1+1)(0\cdot 0+0)=1\cdot 0=0$ — **ตรงกัน** (โปรแกรมตรวจครบทั้ง 64 ชุดแล้ว)

④ ตอบ ข้อ 6b

$$X=(\overline{\mathbf{A}}\,\overline{\mathbf{B}}+\mathbf{C})\,(\overline{\mathbf{D}}\,\overline{\mathbf{E}}+\mathbf{F})$$

การวาดใหม่: G_3, G_5, G_1 วาดเป็น negative-AND · G_2, G_4 คงเป็น NOR

จุดที่มักพลาดข้อนี้

เปลี่ยนรูปเกตแล้ว**ลืมย้าย bubble** (หรือย้าย bubble แล้วลืมเปลี่ยนรูปเกต) — dual symbol ต้องทำสองอย่างพร้อมกันเสมอ ถ้าทำอย่างเดียววงจรจะเปลี่ยนความหมายทันที

สรุปคำตอบทั้งหมดของ Assignment #4

ข้อ 1	a) $X=\sum m(1,2,3,4,5)$, $\overline{X}=\sum m(0,6,7)$ · b) $X=\sum m(2,3,5,7)$, $\overline{X}=\sum m(0,1,4,6)$ · c) $X=\sum m(0,5,6)$, $\overline{X}=\sum m(1,2,3,4,7)$
ข้อ 2	a) SOP $\sum m(1,6)$ / POS $\prod M(0,2,3,4,5,7)$ · b) SOP $\sum m(0,4,5)$ / POS $\prod M(1,2,3,6,7)$ · c) SOP $\sum m(1,2,4)$ / POS $\prod M(0,3,5,6,7)$
ข้อ 3	a) $\sum m(2,5,6,7)$ · b) $\sum m(1,3,5,6,7,9,11,12,13,14,15)$ · c) $\sum m(3,6,7)$ · d) $\sum m(2,4,7)$
ข้อ 4	SOP $=AB+AC+BC$ · POS $=(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+C)$
ข้อ 5	SOP $=\sum m(0,1,2,4,7,8,11,13,14,15)$ · POS $=\prod M(3,5,6,9,10,12)$
ข้อ 6	a) $X=(\overline{A}+\overline{B})C+(\overline{D}+\overline{E})F$ · b) $X=(\overline{A}\,\overline{B}+C)(\overline{D}\,\overline{E}+F)$

ภาคผนวก ค · เฉลย Assignment 5 ทุกข้อ

Assignment 5 มี 5 ข้อใหญ่ และทุกข้อคือเรื่องเดียวกัน คือ **การย่อนิพจน์บูลีนด้วยแผนผังคาร์นอห์ (Karnaugh map)** ที่สอนไว้ใน **บทที่ 4 · K-Map** ของหนังสือเล่มนี้ ข้างล่างนี้ทำให้ดูทุกบรรทัด ไม่ข้ามขั้นเลยสักขั้น

อ่านตรงนี้ก่อน

ทุกข้อในภาคผนวกนี้ใช้ลำดับเดียวกัน 5 จังหวะ ถ้าจำลำดับนี้ได้ ข้อสอบข้อไหนก็ทำได้

1 **วาดผังเปล่าแล้วเติมค่า**
หัวแถว-หัวคอลัมน์ต้องเรียงแบบเกรย์โค้ด **00 · 01 · 11 · 10** เท่านั้น ห้ามเรียง 00 01 10 11 เด็ดขาด

2 **จับกลุ่มเลข 1 (สำหรับ SOP)**
ขนาดกลุ่มต้องเป็น 1, 2, 4, 8, 16 ช่องเท่านั้น · ต่อกันด้านข้างเท่านั้น ห้ามทแยง · กลุ่มใหญ่ที่สุดเท่าที่ทำได้ และจำนวนกลุ่มน้อยที่สุด · ขอบซ้ายติดขอบขวา ขอบบนติดขอบล่าง

3 **อ่านพจน์จากกลุ่ม**
ในกลุ่มนั้น ตัวแปรไหน**ค่าคงที่**ให้เขียน (คงที่ 1 เขียนตัวตรง คงที่ 0 เขียนตัวมีขีดบน) ตัวแปรไหน**เปลี่ยนค่า**ให้ตัดทิ้ง

4 **ทำ POS ด้วยวิธีเดียวกันแต่จับกลุ่ม 0**
จับกลุ่มเลข 0 จะได้ SOP ของ $\overline{X}$ แล้วใช้ De Morgan กลับด้าน → ได้ POS ของ X (ตรงนี้คือขั้นตอน 3 ข้อในสไลด์ Week 5)

5 **เช็กคำตอบกลับ**
แทนค่าทุกแถวของตารางเดิมลงในนิพจน์ที่ย่อได้ ต้องออกมาเท่ากันทุกแถว ถ้าไม่เท่าแม้แถวเดียว แปลว่าจับกลุ่มผิด

1 อ่าน K-map ที่โจทย์ให้ แล้วเขียน SOP และ POS ที่สั้นที่สุด

โจทย์ข้อ 1 (ต้นฉบับ) Assignment #5

> 1. Obtain minimum SOP and POS expressions from the following Karnaugh maps.

แปลไทย: โจทย์วาดผัง K-map มาให้แล้ว 4 ผัง (a–d) หน้าที่เราคือ **อ่านค่าจากผัง** แล้วเขียนนิพจน์ **SOP** (ผลบวกของผลคูณ เช่น $AB + C$) ที่สั้นที่สุด และ **POS** (ผลคูณของผลบวก เช่น $(A + B)(C)$) ที่สั้นที่สุด

โจทย์ซ่อนอะไรไว้: ช่องที่**ว่างเปล่า**ในผังของโจทย์ ไม่ได้แปลว่า “ไม่สนใจ” แต่แปลว่า **0** ส่วนช่องที่เขียน **X** เท่านั้นที่เป็น don’t care · และข้อนี้ต้องตอบ**สองแบบ** ทั้ง SOP และ POS ไม่ใช่แบบเดียว

วิธีที่ใช้ จับกลุ่มช่อง 1 เพื่ออ่าน SOP และจับกลุ่มช่อง 0 เพื่ออ่าน POS — วิธีนี้สอนไว้ใน **บทที่ 4 · K-Map** ของเล่มนี้

ข้อ 1(a) — ผัง 3 ตัวแปร ไม่มี don’t care

ผังของโจทย์เป็นแบบ 3 ตัวแปร หัวแถวคือ AB (เรียง 00, 01, 11, 10) หัวคอลัมน์คือ C (0 กับ 1) ช่องที่โจทย์เขียน 1 ไว้คือ $AB\,C$ = 00·1, 01·0, 01·1, 11·0, 11·1, 10·1 ส่วนอีกสองช่องปล่อยว่าง

1 แปลงช่องว่างเป็น 0 แล้วนับว่าช่องไหนเป็น minterm อะไร

แถว $AB = 00$ คอลัมน์ $C = 1$ → เลขฐานสอง 001 → m_1 · แถว 01 คอลัมน์ 0 → 010 → m_2 · แถว 01 คอลัมน์ 1 → m_3 · แถว 11 คอลัมน์ 0 → 110 → m_6 · แถว 11 คอลัมน์ 1 → m_7 · แถว 10 คอลัมน์ 1 → 101 → m_5

เขียนรวบได้ว่า $F = \sum m(1, 2, 3, 5, 6, 7)$ และช่องที่เป็น 0 คือ m_0 กับ m_4

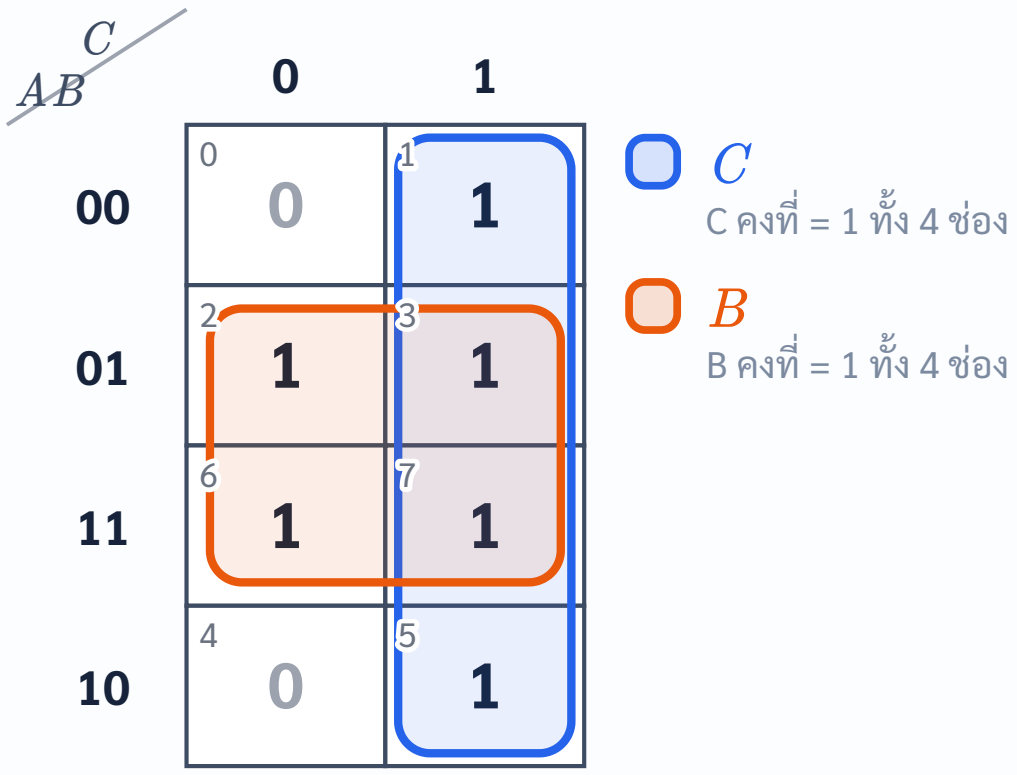
2 จับกลุ่ม 1 ให้ใหญ่ที่สุด

คอลัมน์ $C = 1$ มี 1 ครบทั้ง 4 ช่อง (m_1, m_3, m_7, m_5) → จับเป็นกลุ่ม 4 ได้ทันที · แถว $AB = 01$ กับ $AB = 11$ ก็มี 1 ครบทั้ง 4 ช่อง (m_2, m_3, m_6, m_7) → จับเป็นกลุ่ม 4 อีกกลุ่ม

3 อ่านพจน์จากแต่ละกลุ่ม

กลุ่มฟ้า 4 ช่อง: A เปลี่ยน (0,0,1,1) ตัดทิ้ง · B เปลี่ยน (0,1,1,0) ตัดทิ้ง · C คงที่ = 1 → เขียน C → **พจน์ = C**

กลุ่มส้ม 4 ช่อง: A เปลี่ยน ตัดทิ้ง · B คงที่ = 1 → เขียน $B \cdot C$ เปลี่ยน ตัดทิ้ง → **พจน์ = B**



ผัง 1(a) — จับกลุ่ม **1** เพื่ออ่าน SOP · ตัวเลขเล็กมุมซ้ายบนคือเลขประจำ minterm

4 **รวมเป็น SOP**

$F = C + B$ — ใช้ 2 พจน์ 2 literal และไม่มีทางสั้นกว่านี้แล้ว เพราะช่อง 1 มี 6 ช่อง ต้องใช้อย่างน้อย 2 กลุ่ม

5 **ทำ POS — คราวนี้จับกลุ่ม “0” แทน**

ช่อง 0 มีสองช่องคือ $m_0 = 000$ กับ $m_4 = 100$ · สองช่องนี้อยู่แถวบนสุดกับแถวล่างสุดของคอลัมน์ $C = 0$ ซึ่งในผัง K-map **ขอบบนติดกับขอบล่าง** (เพราะ AB เรียง 00 → 01 → 11 → 10 แล้ววนกลับ 00) จึงจับเป็นกลุ่ม 2 ได้

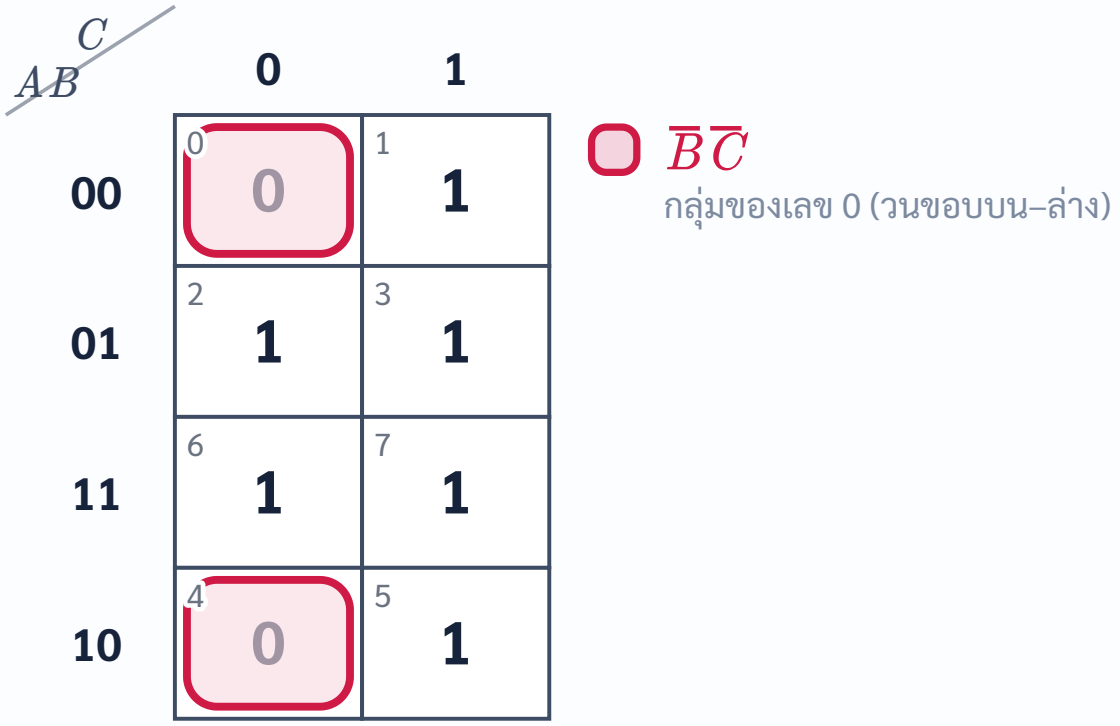
6 **อ่านพจน์ของกลุ่ม 0 → ได้ SOP ของ $\overline{F}$**

ในกลุ่มนี้ A เปลี่ยน (0→1) ตัดทิ้ง · B คงที่ = 0 → เขียน $\overline{B}$ · C คงที่ = 0 → เขียน $\overline{C}$ → $\overline{F} = \overline{B}\,\overline{C}$

7 **กลับด้านด้วย De Morgan → ได้ POS**

$F = \overline{\overline{F}} = \overline{\overline{B}\,\overline{C}}$ ใส่ขีดบนสองชั้น (involution) แล้วแตกชั้นนอก

$F = \overline{\overline{B}} + \overline{\overline{C}} = B + C$ De Morgan: ขีดคลุม AND → แตกเป็น OR ของขีดแต่ละตัว



ผัง 1(a) ใบที่สอง — คราวนี้จับกลุ่ม **0** ได้ $\overline{F} = \overline{B}\,\overline{C}$ แล้วค่อย De Morgan กลับเป็น POS

คำตอบข้อ 1(a)

$$\mathbf{F_{SOP} = B + C} \qquad \mathbf{F_{POS} = (B + C)}$$

(ข้อนี้บังเอิญหน้าตาเหมือนกัน เพราะนิพจน์มีพจน์เดียวจึงเป็นได้ทั้ง SOP และ POS)

เช็กคำตอบกลับเข้าโจทย์ — แทนค่าทั้ง 8 แถวลงใน $B + C$ แล้วเทียบกับผังเดิม

A	B	C	ผังโจทย์	$B + C$
0	0	0	0	0+0 = 0 ✓
0	0	1	1	0+1 = 1 ✓
0	1	0	1	1+0 = 1 ✓
0	1	1	1	1+1 = 1 ✓
1	0	0	0	0+0 = 0 ✓
1	0	1	1	0+1 = 1 ✓
1	1	0	1	1+0 = 1 ✓
1	1	1	1	1+1 = 1 ✓

ตรงกันครบ 8 แถว แปลว่าย่อถูก

จุดที่มักพลาดข้อนี้

เห็นช่องว่างแล้วนึกว่าเป็น don't care เลยจับกลุ่มรวมช่องว่างเข้าไปด้วย จะได้ $F = 1$ ซึ่งผิด — **ช่องว่าง = 0 เสมอ มีแต่ช่องที่เขียน X เท่านั้นที่เป็น don't care**

ข้อ 1(b) — ผัง 3 ตัวแปร ที่มี don't care (X)

ผังนี้มีตัว **X** โผล่มา 2 ช่อง X แปลว่า **"อินพุตชุดนี้ไม่มีทางเกิดขึ้นจริง จะให้ออก 0 หรือ 1 ก็ได้"** ประโยชน์ของมันคือเราเลือกให้เป็น 1 ตอนจับกลุ่ม SOP เพื่อทำกลุ่มให้ใหญ่ขึ้น และเลือกให้เป็น 0 ตอนจับกลุ่ม POS ได้ด้วย

1

ถอดค่าจากผังโจทย์ทีละช่อง

แถว $AB = 00$: ช่อง $C = 0$ เขียน X → m_0 เป็น don't care · ช่อง $C = 1$ ว่าง → $m_1 = 0$

แถว $AB = 01$: ว่างทั้งสองช่อง → $m_2 = 0$ และ $m_3 = 0$

แถว $AB = 11$: เป็น 1 ทั้งสองช่อง → $m_6 = 1$ และ $m_7 = 1$

แถว $AB = 10$: ช่อง $C = 0$ เขียน X → m_4 เป็น don't care · ช่อง $C = 1$ เป็น 1 → $m_5 = 1$

สรุป $F = \sum m(5, 6, 7) + \sum d(0, 4)$ · ช่อง 0 จริง ๆ คือ m_1, m_2, m_3

2

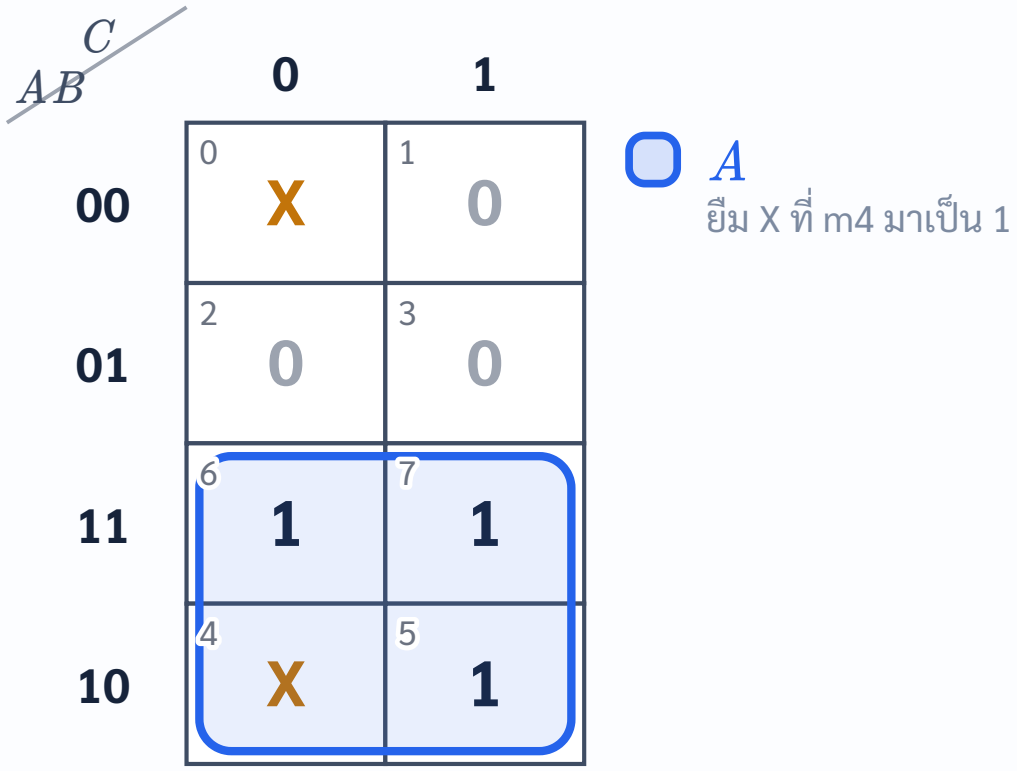
จับกลุ่ม 1 โดย "ยืม" X มาช่วย

เลข 1 อยู่ที่ m_5, m_6, m_7 ทั้งสามช่องอยู่ในครึ่งล่างของผัง (แถว $AB = 11$ กับ $AB = 10$) ซึ่งเป็นครึ่งที่ $A = 1$ ครึ่งนี้มี 4 ช่องคือ m_6, m_7, m_5 และ m_4 · m_4 เป็น X เราเลือกให้มันเป็น 1 ได้ → **จับกลุ่ม 4 ช่องได้เลย**

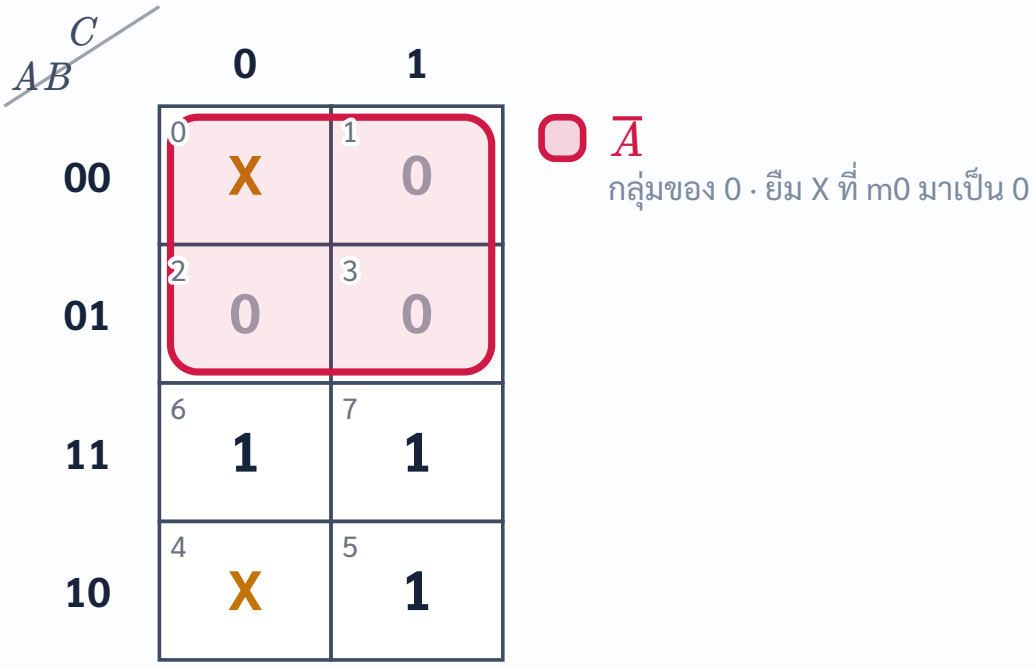
3

อ่านพจน์

ในกลุ่ม 4 ช่องนี้: A คงที่ = 1 → เขียน $A \cdot B$ เปลี่ยน (1,1,0,0) ตัดทิ้ง · C เปลี่ยน (0,1,1,0) ตัดทิ้ง → $F = A$ พจน์เดียว literal เดียว



ผัง 1(b) — SOP · ถ้าไม่ยืม X ที่ m_4 จะจับได้แค่กลุ่ม 2 สองกลุ่ม ได้คำตอบยาวกว่า



ผัง 1(b) ใบที่สอง — POS · จับกลุ่ม 0 ได้ $\overline{F} = \overline{A}$ → De Morgan → $F = A$

4

ทำ POS — จับกลุ่ม 0

เลข 0 จริง ๆ อยู่ที่ m_1, m_2, m_3 ทั้งสามอยู่ในครึ่งบน ($A = 0$) และ m_0 ซึ่งเป็น X ก็อยู่ครึ่งบนด้วย เลือกให้ m_0 เป็น 0 → จับกลุ่ม 4 ช่องได้ → A คงที่ = 0 → $\overline{F} = \overline{A}$

5 **De Morgan**

$F = \overline{\overline{A}} = A$ · เขียนเป็นรูป POS คือ (A) — ผลคูณของผลบวกที่มีวงเล็บเดียว

คำตอบข้อ 1(b)

$$\mathbf{F_{SOP} = A} \quad \mathbf{F_{POS} = (A)}$$

เช็กคำตอบกลับเข้าโจทย์ — ข้ามแถว X เพราะโจทย์บอกว่าไม่ต้องสนใจ เหลือ 6 แถวที่ต้องตรง

A	B	C	ผังโจทย์	A
0	0	1	0	0 ✓
0	1	0	0	0 ✓
0	1	1	0	0 ✓
1	0	1	1	1 ✓
1	1	0	1	1 ✓
1	1	1	1	1 ✓

จุดที่มักพลาดข้อนี้

กลัว X เลยไม่กล้าใช้ จับได้แค่ m_6, m_7 กับ m_5, m_7 ออกมาเป็น $F = AB + AC$ (2 พจน์ 4 literal) ซึ่ง**ค่าถูกแต่ไม่ minimum** ข้อสอบหักคะแนน · X มีไว้ให้ใช้ ไม่ได้มีไว้ให้กลัว

ข้อ 1(c) — ผัง 4 ตัวแปร ไม่มี don’t care

ผัง 4 ตัวแปรมี 16 ช่อง หัวแถวคือ AB หัวคอลัมน์คือ CD ทั้งคู่เรียงเกรย์โค้ด 00 · 01 · 11 · 10 วิธีอ่านเลขประจำช่อง: เอาเลขหัวแถวต่อหัวคอลัมน์ เช่น แถว 11 คอลัมน์ 10 คือ $ABCD = 1110 = m_{14}$

1 **ถอดค่าจากผังโจทย์ทีละแถว**

แถว $AB = 00$ โจทย์ใส่ 1 ที่คอลัมน์ 00 และ 10 → m_0, m_2 · แถว $AB = 01$ ใส่ 1 ที่คอลัมน์ 00 และ 10 → m_4, m_6 · แถว $AB = 11$ ใส่ 1 ครบทั้งแถว → $m_{12}, m_{13}, m_{15}, m_{14}$ · แถว $AB = 10$ ใส่ 1 ที่คอลัมน์ 00, 11, 10 → m_8, m_{11}, m_{10}

$F = \sum m(0, 2, 4, 6, 8, 10, 11, 12, 13, 14, 15)$ · ช่อง 0 คือ m_1, m_3, m_5, m_7, m_9 รวม 5 ช่อง

2 **มองหากลุ่ม 8 ช่องก่อนเสมอ (ใหญ่สุดก่อน)**

คอลัมน์ $CD = 00$ กับ $CD = 10$ คือคอลัมน์ที่ $D = 0$ ทั้งคู่ และมันอยู่ **หัว-ท้าย** ของผังซึ่งติดกันแบบวนขอบ สองคอลัมน์นี้มี 8 ช่อง และเป็น 1 **ครบทั้ง 8** → จับกลุ่ม 8 ได้ → D คงที่ = 0 ตัวเดียว → พจน์ $\overline{D}$

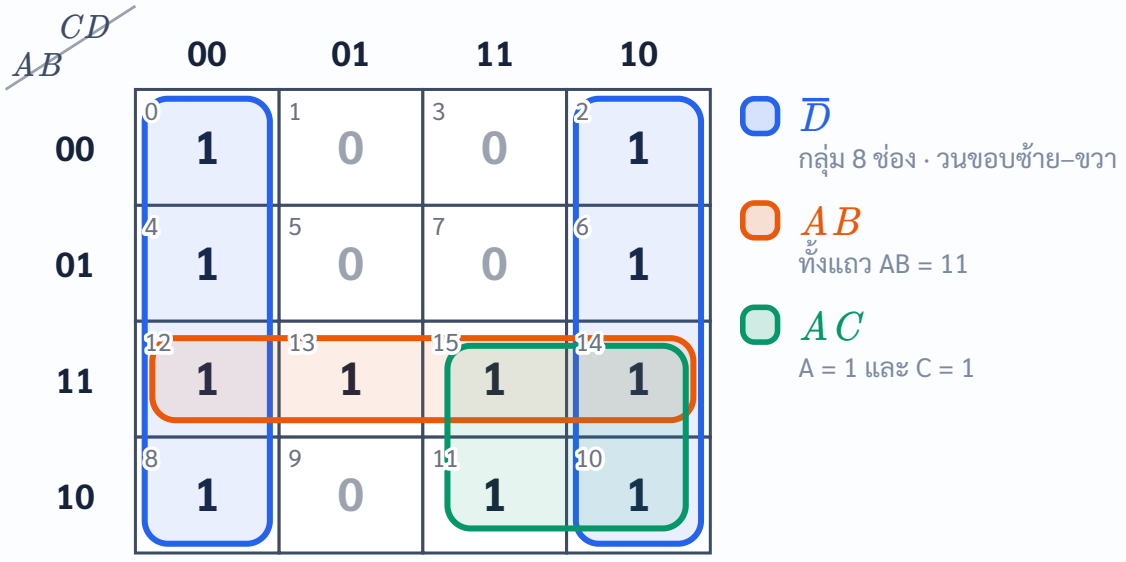
3 **เหลือ 1 ที่ยังไม่ถูกคลุม 3 ช่อง**

m_{11}, m_{13}, m_{15} (ทั้งสามมี $D = 1$) · สามช่องนี้ไม่ได้ติดกันเป็นก้อนสี่เหลี่ยม จับกลุ่มเดียวไม่ได้ ต้องใช้ 2 กลุ่ม

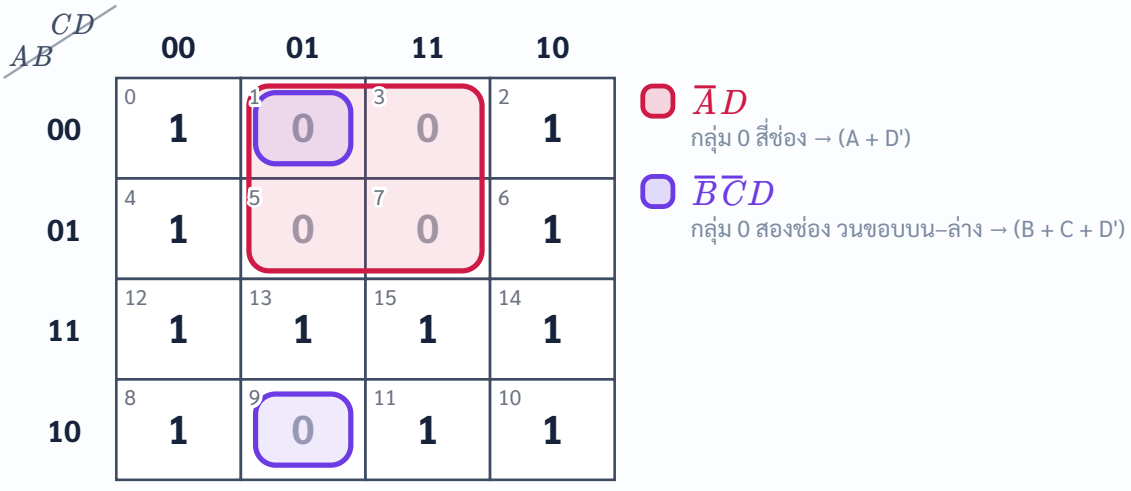
4 **ขยายทั้งสองกลุ่มให้ใหญ่ที่สุด — นี่คือจุดที่ทำให้ literal น้อยลง**

คลุม m_{13}: แทนที่จะจับคู่ $m_{13} + m_{15}$ (ได้ ABD 3 literal) ให้จับทั้งแถว $AB = 11$ คือ $m_{12}, m_{13}, m_{15}, m_{14}$ → ได้ AB แค่ 2 literal

คลุม m_{11}: จับสี่ช่อง $m_{10}, m_{11}, m_{14}, m_{15}$ → A คงที่ 1, C คงที่ 1 → ได้ AC 2 literal (และคลุม m_{15} ไปด้วย)



ผัง 1(c) — SOP · กลุ่มฟ้าแยกเป็นสองแผ่นเพราะคอลัมน์ 00 กับ 10 ติดกันแบบวนขอบ นับเป็นกลุ่มเดียว 8 ช่อง



ผัง 1(c) ใบที่สอง — POS · จับกลุ่ม **0** ได้ $\overline{F} = \overline{A}D + \overline{B}\,\overline{C}D$

5

POS ขั้นที่ 1 — จับกลุ่ม 0 ให้ใหญ่ที่สุด

ช่อง 0 คือ m_1, m_3, m_5, m_7, m_9 · สี่ช่องแรกอยู่ครึ่งบน ($A = 0$) ในคอลัมน์ 01 กับ 11 (ซึ่งคือ $D = 1$) → จับกลุ่ม 4 ได้ → A คงที่ 0, D คงที่ 1 → พจน์ของ $\overline{F}$ คือ $\overline{A}D$

เหลือ m_9 ยังไม่ถูกคลุม · $m_9 = 1001$ จับคู่กับ $m_1 = 0001$ ได้ (วนขอบบน–ล่าง) → B คงที่ 0, C คงที่ 0, D คงที่ 1 → $\overline{B}\,\overline{C}D$

6

POS ขั้นที่ 2 — De Morgan ทีละชั้น

$\overline{F} = \overline{A}D + \overline{B}\,\overline{C}D$ SOP ของส่วนกลับ ที่อ่านจากกลุ่ม 0

$F = \overline{\overline{A}D + \overline{B}\,\overline{C}D}$ ใส่ขีดบนทั้งก้อน

$F = \overline{\overline{A}D} \cdot \overline{\overline{B}\,\overline{C}D}$ De Morgan: ขีดคลุม OR → AND ของขีดแต่ละก้อน

$F = (A + \overline{D})(B + C + \overline{D})$ De Morgan อีกชั้น: ขีดคลุม AND → OR ของขีดแต่ละตัว

คำตอบข้อ 1(c)

$$\mathbf{F_{SOP} = \overline{D} + AB + AC}$$ (3 พจน์ 5 literal)

$$\mathbf{F_{POS} = (A + \overline{D})(B + C + \overline{D})}$$ (2 พจน์ 5 literal)

เช็กคำตอบกลับเข้าโจทย์ — วิธีเร็วที่สุดคือเช็กเฉพาะ 5 ช่องที่ต้องเป็น 0 ถ้า 5 ช่องนี้เป็น 0 และพจน์ทุกอันคลุมแต่ช่อง 1 ก็แปลว่าถูก

ช่อง	$ABCD$	$\overline{D}$	AB	AC	SOP รวม	ต้องได้
m_1	0001	0	0	0	0	0 ✓
m_3	0011	0	0	0	0	0 ✓
m_5	0101	0	0	0	0	0 ✓
m_7	0111	0	0	0	0	0 ✓
m_9	1001	0	0	0	0	0 ✓
m_{11}	1011	0	0	1	1	1 ✓
m_{13}	1101	0	1	0	1	1 ✓

ที่เหลืออีก 9 ช่องมี $D = 0$ ทั้งหมด จึงถูกคลุมด้วย $\overline{D}$ ได้ 1 ทันที — ครบ 16 ช่อง

จุดที่มักพลาดข้อนี้

ลืมว่าคอลัมน์ $CD = 00$ กับ $CD = 10$ **ติดกัน** เลยจับได้แค่กลุ่ม 4 สองกลุ่ม ($\overline{C}\,\overline{D}$ กับ $C\overline{D}$) แล้วตอบ $F = \overline{C}\,\overline{D} + C\overline{D} + AB + AC$ ซึ่งค่าถูกแต่ยาวเกินไป 1 พจน์

ข้อ 1(d) — ผัง 4 ตัวแปร ที่มี don’t care

1

ถอดค่าจากผังโจทย์ทีละแถว

แถว $AB = 00$: คอลัมน์ 00 เป็น 1 → m_0 · คอลัมน์ 10 เป็น X → m_2 · สองช่องกลางว่าง = 0

แถว $AB = 01$: คอลัมน์ 01 และ 11 เป็น 1 → m_5, m_7 · แถว $AB = 11$: คอลัมน์ 01 และ 11 เป็น 1 → m_{13}, m_{15}

แถว $AB = 10$: คอลัมน์ 00 เป็น X → m_8 · คอลัมน์ 10 เป็น 1 → m_{10}

$F = \sum m(0, 5, 7, 10, 13, 15) + \sum d(2, 8)$

2

สังเกตรูปทรงก่อนจับกลุ่ม

เลข 1 สี่ตัว (m_5, m_7, m_{13}, m_{15}) เกาะกันเป็นสี่เหลี่ยมกลางผังพอดี → จับกลุ่ม 4 ได้เลย ส่วน m_0 กับ m_{10} อยู่คนละมุมของผัง ต้องอาศัย X ช่วย

3

กลุ่มที่ 1 — สี่เหลี่ยมกลางผัง

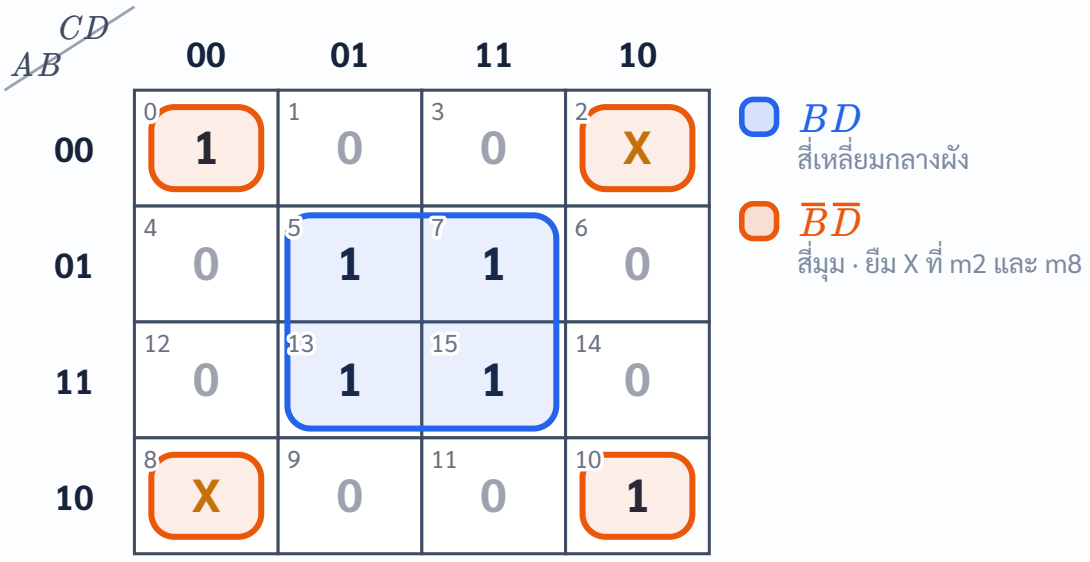
$m_5(0101), m_7(0111), m_{13}(1101), m_{15}(1111)$ · A เปลี่ยน ตัดทิ้ง · B คงที่ = 1 → B · C เปลี่ยน ตัดทิ้ง · D คงที่ = 1 → D → **พจน์** BD

4

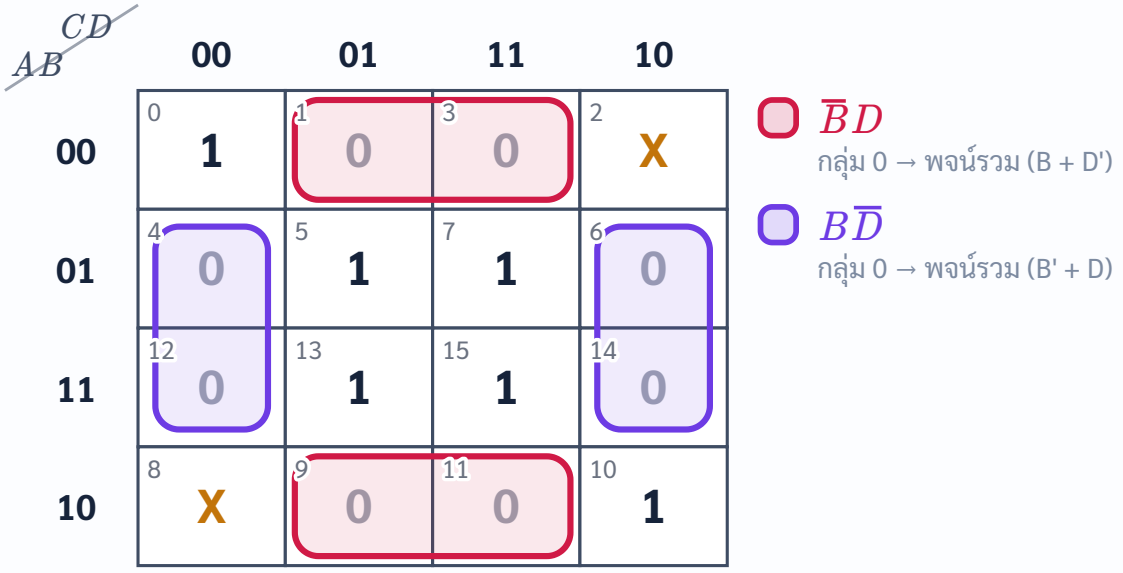
กลุ่มที่ 2 — สี่มุมของผัง (จุดที่คนพลาดมากที่สุด)

$m_0(0000)$ มุมซ้ายบน · $m_2(0010)$ มุมขวาบน (X) · $m_8(1000)$ มุมซ้ายล่าง (X) · $m_{10}(1010)$ มุมขวาล่าง ผัง K-map วนทั้งซ้าย–ขวาและบน–ล่าง **สี่มุมจึงถือว่าติดกันหมด** นับเป็นกลุ่ม 4 ช่องหนึ่งกลุ่ม

A เปลี่ยน ตัดทิ้ง · B คงที่ = 0 → $\overline{B}$ · C เปลี่ยน ตัดทิ้ง · D คงที่ = 0 → $\overline{D}$ → **พจน์** $\overline{B}\,\overline{D}$



ผัง 1(d) — SOP · กลุ่มส้มคือ “สี่มุม” ซึ่งเป็นกลุ่มเดียวกัน 4 ช่อง ไม่ใช่ 4 กลุ่มเดี่ยว



ผัง 1(d) ใบที่สอง — POS · คราวนี้ไม่ต้องยืม X เพราะกลุ่ม 0 ใหญ่พออยู่แล้ว

5

POS — จับกลุ่ม 0

กลุ่มม่วง m_1, m_3, m_9, m_{11}: B คงที่ 0, D คงที่ 1 → พจน์ของ $\overline{F}$ คือ $\overline{B}D$

กลุ่มชมพู m_4, m_6, m_{12}, m_{14}: B คงที่ 1, D คงที่ 0 → พจน์ของ $\overline{F}$ คือ $B\overline{D}$

6

De Morgan

$\overline{F} = \overline{B}D + B\overline{D}$ SOP ของส่วนกลับ

$F = \overline{\overline{B}D} \cdot \overline{B\overline{D}}$ De Morgan ชั้นนอก

$F = (B + \overline{D})(\overline{B} + D)$ De Morgan ชั้นใน

คำตอบข้อ 1(d)

$$\mathbf{F_{SOP} = BD + \overline{B}\,\overline{D}}$$ (2 พจน์ 4 literal)

$$\mathbf{F_{POS} = (B + \overline{D})(\overline{B} + D)}$$ (2 พจน์ 4 literal)

สังเกต: นี่คือ $B \odot D$ นั่นเอง (XNOR — เป็น 1 เมื่อ B กับ D เท่ากัน) · A กับ C หายไปทั้งคู่

เช็กคำตอบกลับเข้าโจทย์ — เช็กทั้ง 14 ช่องที่ไม่ใช่ X โดยดูแค่ค่า B กับ D

BD	ช่องที่มี BD นี้	$BD + \overline{B}\,\overline{D}$	ผังโจทย์
0 0	m_0 (X: m_2, m_8) m_{10}	0 + 1 = 1	1, 1 ✓
0 1	m_1, m_3, m_9, m_{11}	0 + 0 = 0	0 ทั้งสี่ ✓
1 0	m_4, m_6, m_{12}, m_{14}	0 + 0 = 0	0 ทั้งสี่ ✓
1 1	m_5, m_7, m_{13}, m_{15}	1 + 0 = 1	1 ทั้งสี่ ✓

จุดที่มักพลาดข้อนี้

มองไม่เห็นว่า **สี่มุมเป็นกลุ่มเดียวกัน** เลยจับ m_0 กับ m_{10} แยกกันเป็นกลุ่มละ 2 ได้ $F = BD + \overline{A}\,\overline{B}\,\overline{D} + A\overline{B}\,\overline{D}$ (3 พจน์ 8 literal) — ค่าถูกแต่ยาวกว่าเท่าตัว

2 จากตารางค่าความจริง → วาด K-map เอง → SOP และ POS

โจทย์ข้อ 2 (ต้นฉบับ) Assignment #5

> 2. Obtain minimum SOP and POS expressions from the following truth tables using Karnaugh maps.

แปลไทย: คราวนี้โจทย์ไม่วาดผังมาให้ ให้**ตารางค่าความจริง**มาแทน เราต้อง**วาดผังเอง**แล้วค่อยย่อ มีสองตาราง ตารางแรก 3 ตัวแปร (A, B, C) ตารางที่สอง 4 ตัวแปร (A, B, C, D)

โจทย์ซ่อนอะไรไว้: ตารางค่าความจริงเรียงแบบ**เลขฐานสองปกติ** (000, 001, 010, 011 ...) แต่ผัง K-map เรียงแบบ**เกรย์โค้ด** (00, 01, 11, 10) ถ้าลอกตัวเลขลงผังตามลำดับตารางจะผิดทันที — ต้องแปลงแถวเป็นเลข minterm ก่อนแล้วค่อยหย่อนลงช่องที่ถูก

วิธีที่ใช้ ตารางค่าความจริง → เลข minterm → หย่อนลง K-map → จับกลุ่ม — **บทที่ 4 · K-Map**

ข้อ 2 ตารางแรก — 3 ตัวแปร

ขั้นแรก เติมเลข minterm ให้ทุกแถว โดยอ่าน ABC เป็นเลขฐานสองแล้วแปลงเป็นฐานสิบ

A	B	C	X	แถวนี้คือ
0	0	0	1	$000_2 = 0 \rightarrow m_0$
0	0	1	0	$001_2 = 1 \rightarrow m_1$
0	1	0	0	$010_2 = 2 \rightarrow m_2$
0	1	1	1	$011_2 = 3 \rightarrow m_3$
1	0	0	1	$100_2 = 4 \rightarrow m_4$
1	0	1	0	$101_2 = 5 \rightarrow m_5$
1	1	0	0	$110_2 = 6 \rightarrow m_6$
1	1	1	1	$111_2 = 7 \rightarrow m_7$

แถวเขียว (เอาต์พุต 1) คือ $m_0, m_3, m_4, m_7 \rightarrow X = \sum m(0, 3, 4, 7)$ · แถวแดงคือ m_1, m_2, m_5, m_6

1

หย่อนเลขลงผัง — ดูว่าแต่ละ minterm ไปอยู่ช่องไหน

$m_0 = 000$ → แถว $AB = 00$ คอลัมน์ $C = 0$ · $m_3 = 011$ → แถว $AB = 01$ คอลัมน์ $C = 1$ · $m_4 = 100$ → แถว $AB = 10$ คอลัมน์ $C = 0$ **(ระวัง แถว 10 คือแถวล่างสุด ไม่ใช่แถวที่สาม)** · $m_7 = 111$ → แถว $AB = 11$ คอลัมน์ $C = 1$

2

จับกลุ่ม 1

m_0 กับ m_4 อยู่คอลัมน์ $C = 0$ ทั้งคู่ แถวบนสุดกับล่างสุด → ติดกันแบบวนขอบ → กลุ่ม 2 ช่อง

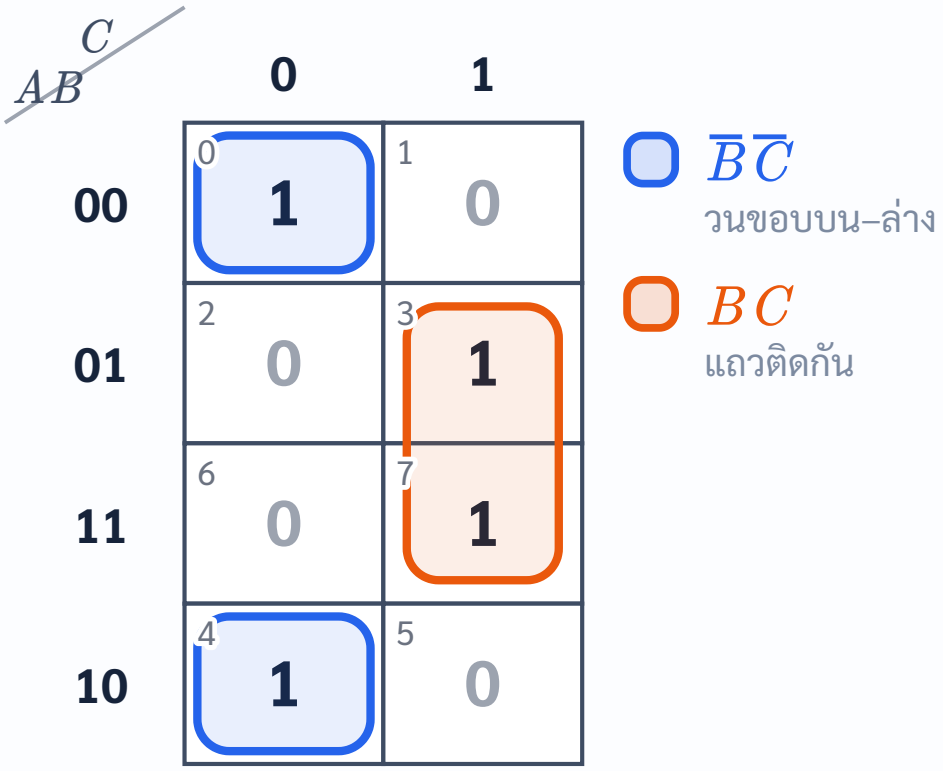
m_3 กับ m_7 อยู่คอลัมน์ $C = 1$ แถวติดกันพอดี → กลุ่ม 2 ช่อง

3

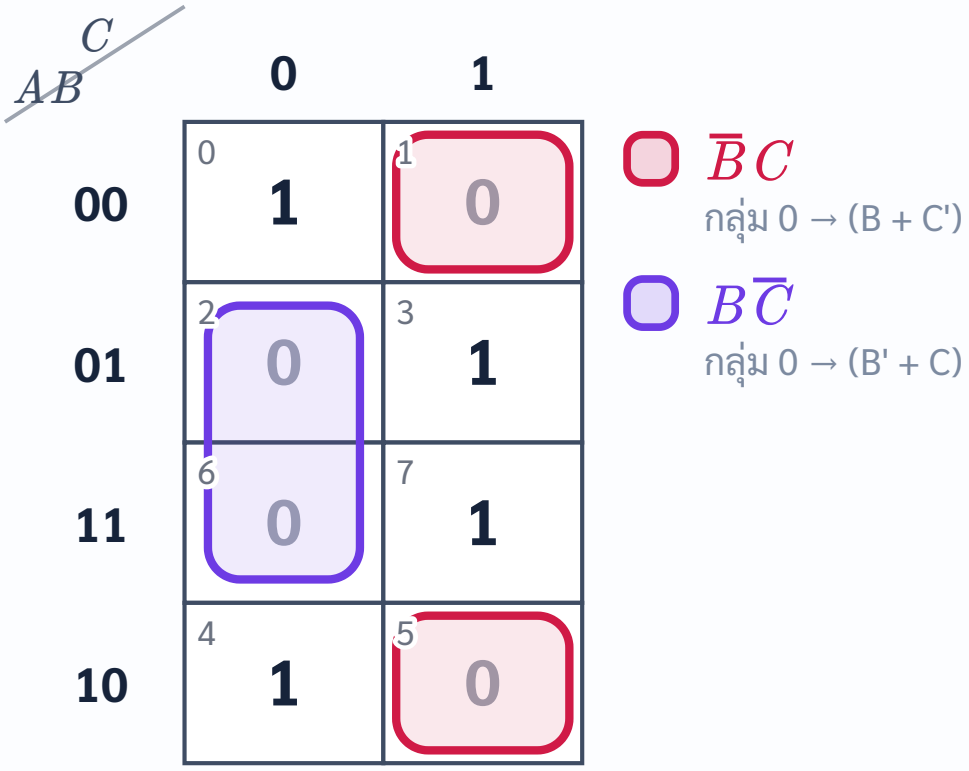
อ่านพจน์

กลุ่มฟ้า (m_0, m_4): A เปลี่ยน ตัดทิ้ง · B คงที่ 0 → $\overline{B}$ · C คงที่ 0 → $\overline{C}$ → $\overline{B}\,\overline{C}$

กลุ่มส้ม (m_3, m_7): A เปลี่ยน ตัดทิ้ง · B คงที่ 1 → B · C คงที่ 1 → C → BC



ผังข้อ 2 ตารางแรก — SOP · A หายไปจากคำตอบทั้งสองพจน์ แปลว่า A ไม่มีผลต่อเอาต์พุตเลย



ผังข้อ 2 ตารางแรก ใบที่สอง — POS · จับกลุ่ม **0**

4

POS — จับกลุ่ม 0 แล้ว De Morgan

$m_1(001)$ กับ $m_5(101)$: B คงที่ 0, C คงที่ 1 → $\overline{B}C$ · $m_2(010)$ กับ $m_6(110)$: B คงที่ 1, C คงที่ 0 → $B\overline{C}$

$\overline{X} = \overline{B}C + B\overline{C}$ SOP ของส่วนกลับ

$X = \overline{\overline{B}C} \cdot \overline{B\overline{C}} = (B + \overline{C})(\overline{B} + C)$ De Morgan สองชั้น

คำตอบข้อ 2 ตารางแรก

$$\mathbf{X_{SOP} = \overline{B}\,\overline{C} + BC \quad X_{POS} = (B + \overline{C})(\overline{B} + C)}$$

นี่คือ $B \odot C$ (XNOR) — เป็น 1 เมื่อ B เท่ากับ C

เช็กคำตอบกลับเข้าโจทย์ — แทนค่าทั้ง 8 แถว

ABC	ตารางโจทย์	$\overline{B}\,\overline{C}$	BC	SOP รวม
000	1	1	0	1 ✓
001	0	0	0	0 ✓
010	0	0	0	0 ✓
011	1	0	1	1 ✓
100	1	1	0	1 ✓
101	0	0	0	0 ✓
110	0	0	0	0 ✓
111	1	0	1	1 ✓

จุดที่มักพลาดข้อนี้

ลอกค่าจากตารางลงผังเรียงตามลำดับแถว (00, 01, **10**, 11) แทนที่จะเป็นเกรย์โค้ด (00, 01, **11**, 10) ทำให้ m_4 กับ m_6 สลับที่กัน แล้วได้คำตอบผิดทั้งข้อ — **ก่อนหย่อนค่าลงผัง ให้เขียนเลข minterm กำกับทุกแถวก่อนเสมอ**

ข้อ 2 ตารางที่สอง — 4 ตัวแปร

A	B	C	D	X	แถวนี้คือ
0	0	0	0	0	m_0
0	0	0	1	0	m_1
0	0	1	0	1	m_2
0	0	1	1	0	m_3
0	1	0	0	0	m_4
0	1	0	1	0	m_5
0	1	1	0	1	m_6
0	1	1	1	1	m_7
1	0	0	0	1	m_8
1	0	0	1	1	m_9
1	0	1	0	1	m_{10}
1	0	1	1	1	m_{11}
1	1	0	0	0	m_{12}
1	1	0	1	1	m_{13}
1	1	1	0	1	m_{14}
1	1	1	1	1	m_{15}

แถวเขียวคือ $X = \sum m(2, 6, 7, 8, 9, 10, 11, 13, 14, 15)$ · แถวแดง (ช่อง 0) คือ $m_0, m_1, m_3, m_4, m_5, m_{12}$ — มีแค่ 6 ช่อง

1

หาช่องที่ไม่มีทางเลือกก่อน — เทคนิคที่ทำให้ไม่หลง

ดูช่อง 1 ที่มีเพื่อนบ้านเป็น 1 น้อยที่สุด แล้วขยายกลุ่มของมันให้ใหญ่ที่สุด ถ้าขยายได้ทางเดียว แปลว่ากลุ่มนั้น**ต้องใช้แน่นอน**

$m_2(0010)$: เพื่อนบ้านคือ $m_0 = 0, m_3 = 0, m_6 = 1, m_{10} = 1$ → ขยายได้เป็น m_2, m_6, m_{10}, m_{14} ซึ่งคือคอลัมน์ $CD = 10$ ทั้งคอลัมน์ → **ต้องใช้พจน์** $C\overline{D}$

$m_8(1000)$: เพื่อนบ้านคือ $m_9 = 1, m_{10} = 1, m_{12} = 0, m_0 = 0$ → ขยายได้เป็นแถว $AB = 10$ ทั้งแถว → **ต้องใช้พจน์** $A\overline{B}$

$m_7(0111)$: เพื่อนบ้านคือ $m_6 = 1, m_3 = 0, m_5 = 0, m_{15} = 1$ → ขยายได้เป็น m_6, m_7, m_{14}, m_{15} → **ต้องใช้พจน์** BC

$m_{13}(1101)$: เพื่อนบ้านคือ $m_{12} = 0, m_{15} = 1, m_9 = 1, m_5 = 0$ → ขยายได้เป็น $m_9, m_{11}, m_{13}, m_{15}$ → **ต้องใช้พจน์** AD

2 **เช็กว่า 4 กลุ่มนี้คลุมครบหรือยัง**

$C\overline{D}$ คลุม 2, 6, 10, 14 · $A\overline{B}$ คลุม 8, 9, 10, 11 · BC คลุม 6, 7, 14, 15 · AD คลุม 9, 11, 13, 15

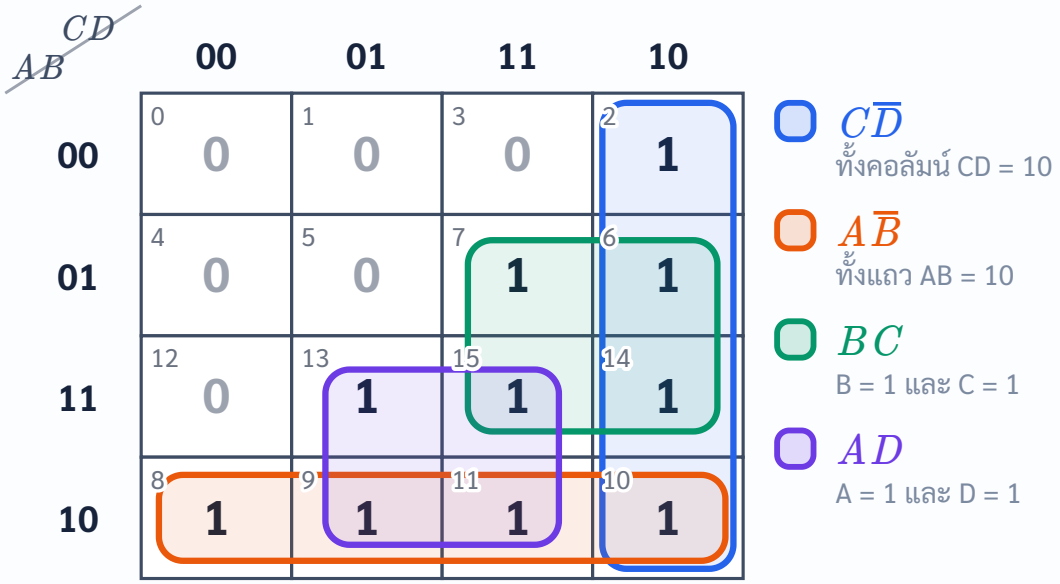
รวมกันได้ 2, 6, 7, 8, 9, 10, 11, 13, 14, 15 ซึ่งคือช่อง 1 ทั้งหมดพอดี → **ไม่ต้องเพิ่มกลุ่มอีก**

3 **อ่านพจน์จากทั้ง 4 กลุ่ม**

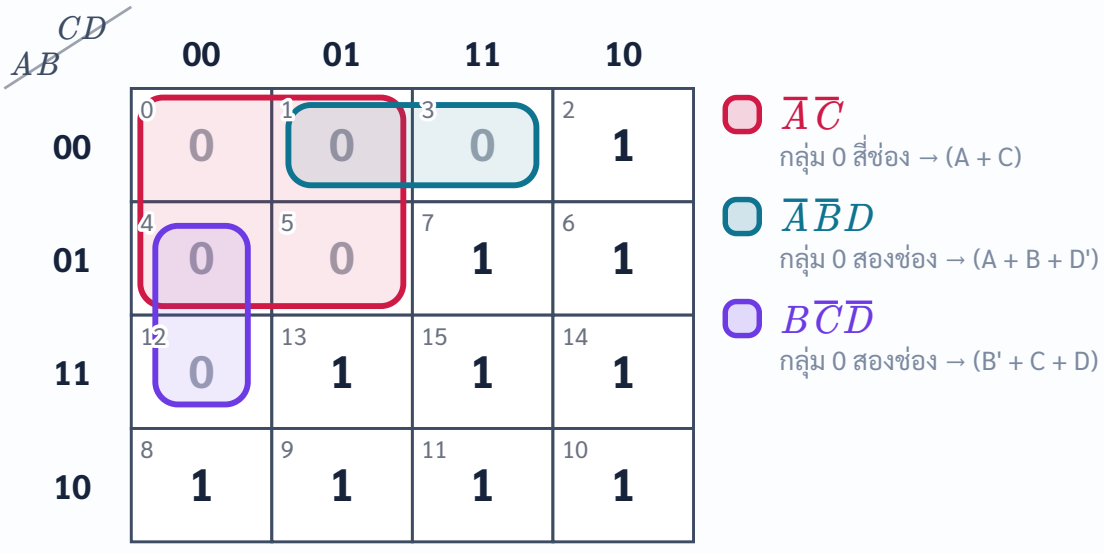
คอลัมน์ $CD = 10$: C คงที่ 1, D คงที่ 0, A กับ B เปลี่ยน → $C\overline{D}$

แถว $AB = 10$: A คงที่ 1, B คงที่ 0, C กับ D เปลี่ยน → $A\overline{B}$

m_6, m_7, m_{14}, m_{15} : B คงที่ 1, C คงที่ 1 → BC · $m_9, m_{11}, m_{13}, m_{15}$: A คงที่ 1, D คงที่ 1 → AD



ผังข้อ 2 ตารางที่สอง — SOP · ทั้ง 4 กลุ่มเป็นกลุ่มที่ต้องใช้ เพราะแต่ละกลุ่มมีช่องที่กลุ่มอื่นคลุมไม่ถึง



ผังข้อ 2 ตารางที่สอง ใบที่สอง — POS · ช่อง 0 มีแค่ 6 ช่อง การจับกลุ่ม 0 จึงง่ายกว่าจับกลุ่ม 1

4

POS — จับกลุ่ม 0 ทีละกลุ่ม

$m_3(0011)$ เพื่อนบ้านที่เป็น 0 มีแค่ m_1 → จับคู่ m_1, m_3 → A คงที่ 0, B คงที่ 0, D คงที่ 1 → $\overline{A}\,\overline{B}D$

$m_{12}(1100)$ เพื่อนบ้านที่เป็น 0 มีแค่ m_4 → จับคู่ m_4, m_{12} → B คงที่ 1, C คงที่ 0, D คงที่ 0 → $B\overline{C}\,\overline{D}$

เหลือ m_0 กับ m_5 ยังไม่ถูกคลุม → จับกลุ่ม 4 ช่อง m_0, m_1, m_4, m_5 (มุมซ้ายบนของผัง) → A คงที่ 0, C คงที่ 0 → $\overline{A}\,\overline{C}$

5

De Morgan

$\overline{X} = \overline{A}\,\overline{C} + \overline{A}\,\overline{B}D + B\overline{C}\,\overline{D}$ SOP ของส่วนกลับ ที่อ่านจากกลุ่ม 0

$X = (A + C)(A + B + \overline{D})(\overline{B} + C + D)$ De Morgan: ทุกพจน์กลายเป็นวงเล็บ ทุกตัวแปรกลับขีด

คำตอบข้อ 2 ตารางที่สอง

$$\mathbf{X_{SOP} = A\overline{B} + AD + BC + C\overline{D}}$$ (4 พจน์ 8 literal)

$$\mathbf{X_{POS} = (A + C)(A + B + \overline{D})(\overline{B} + C + D)}$$ (3 พจน์ 8 literal)

เช็กคำตอบกลับเข้าโจทย์ — เช็ก 6 แถวที่ต้องเป็น 0 ก่อน ถ้าพจน์ไหนติด 1 ในแถวพวกนี้ แปลว่าจับกลุ่มเกิน

ช่อง	$ABCD$	$A\overline{B}$	AD	BC	$C\overline{D}$	รวม	ต้องได้
m_0	0000	0	0	0	0	0	0 ✓
m_1	0001	0	0	0	0	0	0 ✓
m_3	0011	0	0	0	0	0	0 ✓
m_4	0100	0	0	0	0	0	0 ✓
m_5	0101	0	0	0	0	0	0 ✓
m_{12}	1100	0	0	0	0	0	0 ✓
m_2	0010	0	0	0	1	1	1 ✓
m_{13}	1101	0	1	0	0	1	1 ✓

ช่อง 1 ที่เหลือ ($m_6, m_7, m_8, m_9, m_{10}, m_{11}, m_{14}, m_{15}$) อยู่ในกรอบสีใดสีหนึ่งในภาพอยู่แล้ว จึงได้ 1 ทุกช่อง

จุดที่มักพลาดข้อนี้

เห็นว่า $m_{10}, m_{11}, m_{14}, m_{15}$ จับเป็น AC ได้ เลยใส่เพิ่มเข้าไปเป็นพจน์ที่ห้า ค่ายังถูกแต่**เกินมา 1 พจน์** — กฎคือกลุ่มใหญ่ที่สุด **และจำนวนกลุ่มน้อยที่สุด** ถ้าช่อง 1 ถูกคลุมครบแล้ว ห้ามเพิ่มกลุ่มอีก

3 พล็อตฟังก์ชันลง K-map จากสัญกรณ์ $\sum m$ และ $\sum d$

โจทย์ข้อ 3 (ต้นฉบับ) Assignment #5

> 3. Plot the following function on a Karnaugh map and obtain the minimum SOP expression:
> $G(W, X, Y, Z) = \sum m(3, 4, 7, 10, 11, 14) + \sum d(2, 13, 15)$

แปลไทย: โจทย์เขียนฟังก์ชันมาในรูปย่อ ให้เราพล็อตลงผัง K-map แล้วหา SOP ที่สั้นที่สุด (ข้อนี้**ไม่ต้องทำ POS**)

อ่านสัญกรณ์ให้ออกก่อน: $\sum m(\dots)$ คือรายชื่อช่องที่เอาต์พุตเป็น **1** · $\sum d(\dots)$ คือรายชื่อช่องที่เป็น **don't care (X)** · ช่องที่**ไม่ถูกเอ่ยถึงเลย** คือ 0 ทั้งหมด

โจทย์ซ่อนอะไรไว้: ตัวแปรชื่อ W, X, Y, Z ไม่ใช่ A, B, C, D โดย W เป็นบิตซ้ายสุด (MSB) และ Z เป็นบิตขวาสุด (LSB) ถ้าจำสลับ ตำแหน่งทุกช่องจะเพี้ยนหมด

วิธีที่ใช้ แปลงเลข minterm เป็นเลขฐานสอง 4 บิต → หย่อนลงช่อง → จับกลุ่ม — **บทที่ 4 · K-Map**

1 แปลงเลขทุกตัวเป็นฐานสอง 4 บิต ก่อนวาดอะไรทั้งสิ้น

ช่องที่เป็น 1: $3 = 0011 \cdot 4 = 0100 \cdot 7 = 0111 \cdot 10 = 1010 \cdot 11 = 1011 \cdot 14 = 1110$

ช่องที่เป็น X: $2 = 0010 \cdot 13 = 1101 \cdot 15 = 1111$

ช่องที่เหลือเป็น 0: 0, 1, 5, 6, 8, 9, 12

2 หย่อนลงผัง — สองบิตซ้าย (WX) บอกแถว สองบิตขวา (YZ) บอกคอลัมน์

เช่น $10 = 1010 \rightarrow WX = 10$ (แถวล่างสุด) และ $YZ = 10$ (คอลัมน์ขวาสุด) · $4 = 0100 \rightarrow WX = 01$ (แถวที่สอง) และ $YZ = 00$ (คอลัมน์ซ้ายสุด)

3 มองหาช่อง 1 ที่โดดเดี่ยวก่อน

$m_4 = 0100$ เพื่อนบ้านทั้งสี่คือ $m_{12} = 0$ (พลิก W), $m_0 = 0$ (พลิก X), $m_6 = 0$ (พลิก Y), $m_5 = 0$ (พลิก Z)

ทั้งสี่เป็น 0 หมด → m_4 **จับกลุ่มกับใครไม่ได้เลย ต้องอยู่เดี่ยว ๆ** → พจน์ $\overline{W}X\overline{Y}\,\overline{Z}$ (4 literal เลี่ยงไม่ได้)

4 จับกลุ่มที่เหลือ โดยยืม X มาช่วย

คอลัมน์ $YZ = 11$ มี $m_3 = 1, m_7 = 1, m_{15} = \text{X}, m_{11} = 1$ → เลือกให้ m_{15} เป็น 1 → จับกลุ่ม 4 ช่องทั้งคอลัมน์ → Y คงที่ 1, Z คงที่ 1 → พจน์ YZ

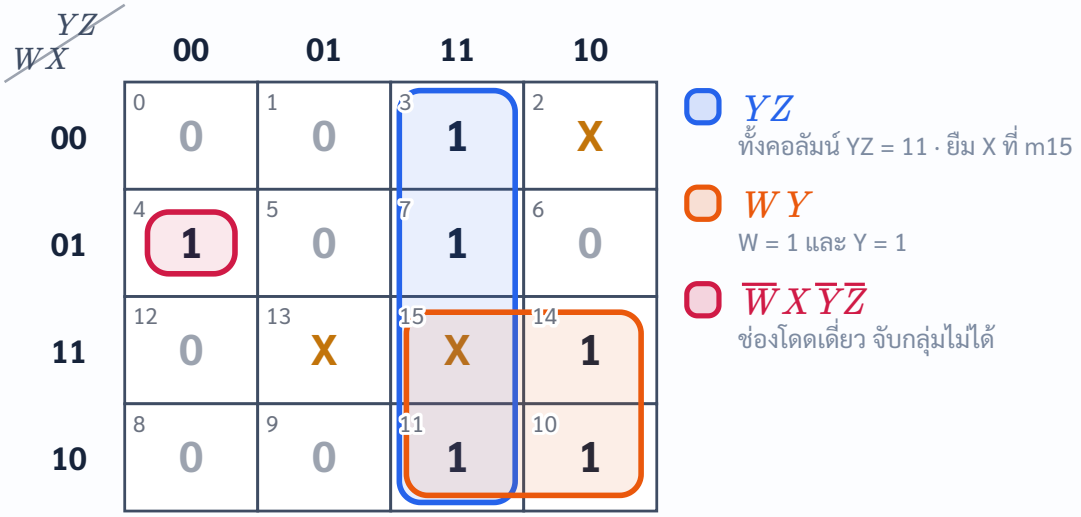
ครึ่งล่างขวาของผัง ($m_{10}, m_{11}, m_{14}, m_{15}$) → W คงที่ 1, Y คงที่ 1 → พจน์ WY — กลุ่มนี้เก็บ m_{10} กับ m_{14} ที่ยังเหลือได้พอดี

5

เช็กว่าช่อง 1 ถูกคลุมครบหรือยัง

YZ คลุม 3, 7, 11 (และ 15 ที่เป็น X) · WY คลุม 10, 11, 14 (และ 15) · $\overline{W}X\overline{Y}\,\overline{Z}$ คลุม 4

รวมได้ 3, 4, 7, 10, 11, 14 = ครบตามที่โจทย์สั่ง · ส่วน X ที่ m_2 กับ m_{13} เราเลือกไม่ใช้ ปล่อยให้เป็น 0 ไป (ใช้ก็ไม่ทำให้สั้นลง)



ผังข้อ 3 — X ที่ m_2 กับ m_{13} ปล่อยเป็น 0 เพราะเอามาใช้แล้วไม่ทำให้นิพจน์สั้นลง

คำตอบข้อ 3

$$\mathbf{G = YZ + WY + \overline{W}X\overline{Y}\,\overline{Z}}$$ (3 พจน์ 8 literal)

เช็กคำตอบกลับเข้าโจทย์ — เช็กทีละช่องว่าได้ค่าตามที่โจทย์สั่งจริง

ช่อง	$WXYZ$	YZ	WY	$\overline{W}X\overline{Y}\,\overline{Z}$	รวม	โจทย์สั่ง
m_0	0000	0	0	0	0	0 ✓
m_1	0001	0	0	0	0	0 ✓
m_3	0011	1	0	0	1	1 ✓
m_4	0100	0	0	1	1	1 ✓
m_5	0101	0	0	0	0	0 ✓
m_6	0110	0	0	0	0	0 ✓
m_7	0111	1	0	0	1	1 ✓
m_8	1000	0	0	0	0	0 ✓
m_9	1001	0	0	0	0	0 ✓
m_{10}	1010	0	1	0	1	1 ✓
m_{11}	1011	1	1	0	1	1 ✓
m_{12}	1100	0	0	0	0	0 ✓
m_{14}	1110	0	1	0	1	1 ✓

อีก 3 ช่อง (m_2, m_{13}, m_{15}) เป็น don’t care โจทย์ไม่สนใจว่าออกอะไร จึงไม่ต้องเช็ก — ได้ครบ 16 ช่อง

จุดที่มักพลาดข้อนี้

เห็น X ที่ m_2 แล้วอยากใช้ เลยจับ m_2, m_3, m_{10}, m_{11} เป็น $\overline{X}Y$ ด้วย กลายเป็น 4 พจน์ทั้งที่ 3 พจน์ก็พอ · **X ใช้เมื่อมันทำให้กลุ่มใหญ่ขึ้นหรือกลุ่มน้อยลงเท่านั้น ไม่ใช่ใช้ทุกตัวที่เห็น**

4 ย่อนิพจน์ที่ให้มาแล้ว ด้วย K-map

โจทย์ข้อ 4 (ต้นฉบับ) Assignment #5

> 4. Reduce the following expressions to the minimum SOP form using Karnaugh mapping. The number of terms and literals of the minimum SOP expressions are given in parentheses.
>
> $F = A + B + \overline{A}\,\overline{B}CD$ (3 terms, 4 literals)
> $G = WX\overline{Y} + XYZ + W\overline{X}YZ + XY\overline{Z} + W\overline{Y}$ (3 terms, 6 literals)
> $G = AB\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}\,\overline{C} + B\overline{C}D + \overline{A}D$ (2 terms, 4 literals)
> $F = \overline{P}\,\overline{Q}R + \overline{P}Q\overline{R} + \overline{P}QR + PQ\overline{R} + P\overline{Q}\,\overline{R}$ (3 terms, 6 literals)

แปลไทย: โจทย์ให้นิพจน์มาแล้ว (ไม่ใช่ผังหรือตาราง) ให้ย่อให้สั้นที่สุดในรูป SOP ด้วย K-map · ตัวเลขในวงเล็บคือ**เฉลยบางส่วน** บอกว่าคำตอบต้องมีกี่พจน์และกี่ literal — เอาไว้ตรวจตัวเองว่าจับกลุ่มครบหรือยัง

โจทย์ซ่อนอะไรไว้: นิพจน์ที่ให้มา**ไม่ใช่ standard SOP** — แต่ละพจน์มีตัวแปรไม่ครบ 4 ตัว ดังนั้นจะหย่อนลงผังตรง ๆ ไม่ได้ ต้อง**กางแต่ละพจน์ออกเป็นช่องที่มันคลุม**ก่อน

วิธีที่ใช้ กางพจน์ → หย่อนลงผัง → จับกลุ่มใหม่ — **บทที่ 4 · K-Map** · วิธีกางพจน์คือ ตัวแปรที่**ขาดหายไป**ในพจน์นั้น ให้แทนค่าได้ทั้ง 0 และ 1 เช่น A (ในระบบ 4 ตัวแปร) ขาด B, C, D ไป 3 ตัว จึงคลุม $2^3 = 8$ ช่อง

ข้อ 4.1 — $F = A + B + \overline{A}\,\overline{B}CD$

1

กางทีละพจน์ ว่าแต่ละพจน์คลุมช่องไหนบ้าง

A = $1---$ → ทุกช่องที่บิตแรกเป็น 1 → m_8 ถึง m_{15} (8 ช่อง)

B = $-1--$ → ทุกช่องที่บิตสองเป็น 1 → $m_4, m_5, m_6, m_7, m_{12}, m_{13}, m_{14}, m_{15}$ (8 ช่อง)

$\overline{A}\,\overline{B}CD$ = 0011 → ช่องเดียว → m_3

รวมช่องที่เป็น 1: m_3 ถึง m_{15} ทั้งหมด · ช่องที่เป็น 0 เหลือแค่ m_0, m_1, m_2

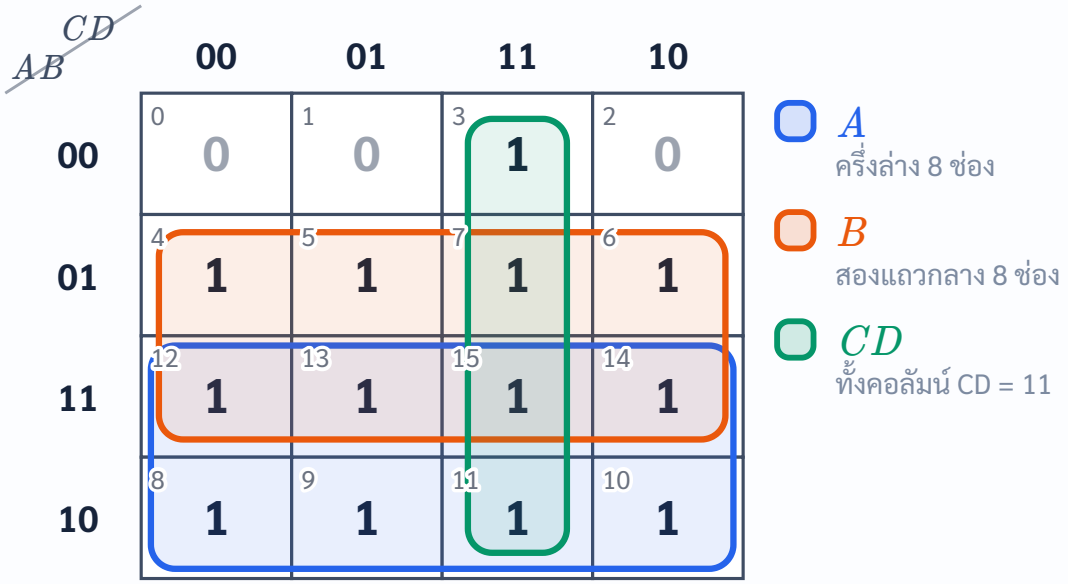
2

จับกลุ่มใหม่บนผัง — อย่าใช้กลุ่มเดิมจากนิพจน์

กลุ่ม 8 ช่องที่ 1: ครึ่งล่างของผัง (แถว $AB = 11$ กับ $AB = 10$) เป็น 1 ครบ → A คงที่ 1 → พจน์ A

กลุ่ม 8 ช่องที่ 2: แถวกลางสองแถว (แถว $AB = 01$ กับ $AB = 11$) เป็น 1 ครบ → B คงที่ 1 → พจน์ B

เหลือ m_3 ช่องเดียวที่ยังไม่ถูกคลุม · $m_3 = 0011$ เพื่อนบ้านคือ $m_7 = 1, m_{11} = 1, m_1 = 0, m_2 = 0$ → ขยายไปทางคอลัมน์ $CD = 11$ ได้ทั้งคอลัมน์ (m_3, m_7, m_{15}, m_{11} เป็น 1 หมด) → C คงที่ 1, D คงที่ 1 → พจน์ CD



ผังข้อ 4.1 — พจน์ $\overline{A}\,\overline{B}CD$ ในโจทย์ยาว 4 literal แต่พอวาดลงผังแล้วขยายได้เหลือ CD แค่ 2 literal

คำตอบข้อ 4.1

$$\mathbf{F = A + B + CD}$$ (3 พจน์ 4 literal — ตรงกับที่โจทย์บอกไว้ในวงเล็บ)

เช็กคำตอบกลับเข้าโจทย์ — เช็ก 3 ช่องที่ต้องเป็น 0 และเช็กว่าพจน์ที่ยาวลดลงแล้วยังคลุม m_3 อยู่

ช่อง	$ABCD$	A	B	CD	รวม	นิพจน์เดิม
m_0	0000	0	0	0	0	$0+0+0=0$ ✓
m_1	0001	0	0	0	0	$0+0+0=0$ ✓
m_2	0010	0	0	0	0	$0+0+0=0$ ✓
m_3	0011	0	0	1	1	$0+0+1=1$ ✓
m_4	0100	0	1	0	1	$0+1+0=1$ ✓
m_8	1000	1	0	0	1	$1+0+0=1$ ✓

ช่องที่เหลือทั้งหมดมี $A=1$ หรือ $B=1$ อย่างน้อยหนึ่งอย่าง จึงได้ 1 ทั้งในนิพจน์เดิมและนิพจน์ใหม่

จุดที่มักพลาดข้อนี้

ลอกพจน์ $\overline{A}\,\overline{B}CD$ จากโจทย์ลงคำตอบตรง ๆ ได้ 6 literal แทนที่จะเป็น 4 · **พจน์ในโจทย์ไม่ใช่กลุ่มที่ดีที่สุด** ต้องวาดลงผังแล้วขยายกลุ่มใหม่เสมอ · ถ้ารู้กฎ $X + \overline{X}Y = X + Y$ จะเห็นทางลัดว่า $(A+B) + \overline{(A+B)}CD = (A+B) + CD$

ข้อ 4.2 — $G = WX\overline{Y} + XYZ + W\overline{X}YZ + XY\overline{Z} + W\overline{Y}$

1

กางทีละพจน์ (ตัวแปรเรียง $W\ X\ Y\ Z$ จากซ้ายไปขวา)

$WX\overline{Y}$ = 110− → ขาด Z → $m_{12}(1100), m_{13}(1101)$

XYZ = −111 → ขาด W → $m_7(0111), m_{15}(1111)$

$W\overline{X}YZ$ = 1011 → ครบทุกตัว → m_{11}

$XY\overline{Z}$ = −110 → ขาด W → $m_6(0110), m_{14}(1110)$

$W\overline{Y}$ = 1−0− → ขาด X กับ Z → 4 ช่อง → $m_8(1000), m_9(1001), m_{12}(1100), m_{13}(1101)$

รวมช่องที่เป็น 1: $m_6, m_7, m_8, m_9, m_{11}, m_{12}, m_{13}, m_{14}, m_{15}$ (9 ช่อง) · ที่เหลือ 7 ช่องเป็น 0

2

หาช่องที่บังคับกลุ่ม

$m_6(0110)$: เพื่อนบ้าน $m_7 = 1, m_{14} = 1, m_4 = 0, m_2 = 0$ → ขยายได้ทางเดียวเป็น m_6, m_7, m_{14}, m_{15} → X คงที่ 1, Y คงที่ 1 → XY

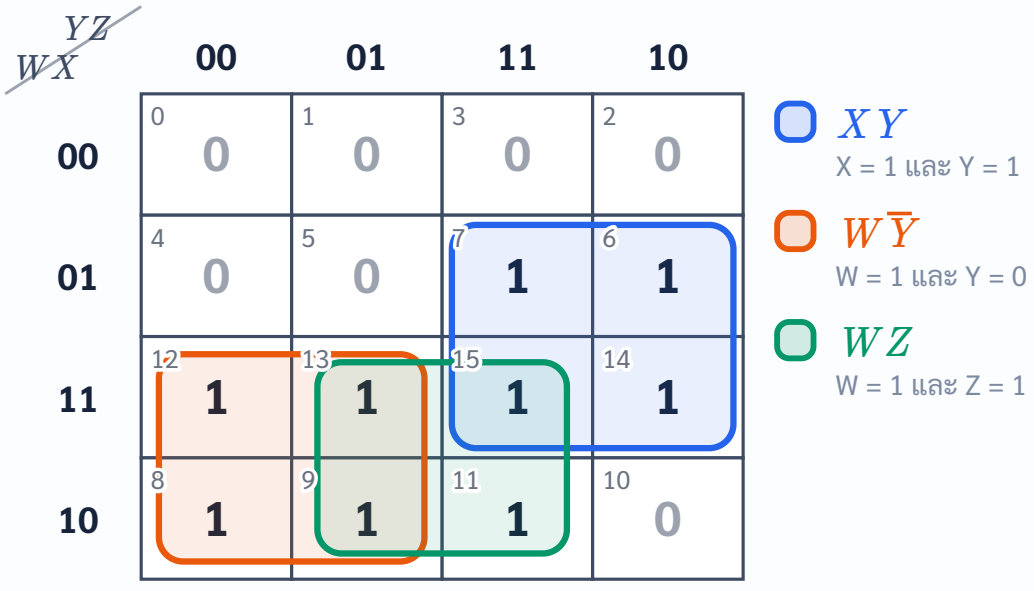
$m_8(1000)$: เพื่อนบ้าน $m_9 = 1, m_{12} = 1, m_{10} = 0, m_0 = 0$ → ขยายได้เป็น m_8, m_9, m_{12}, m_{13} → W คงที่ 1, Y คงที่ 0 → $W\overline{Y}$

$m_{11}(1011)$: เพื่อนบ้าน $m_9 = 1, m_{15} = 1, m_{10} = 0, m_3 = 0$ → ขยายได้เป็น $m_9, m_{11}, m_{13}, m_{15}$ → W คงที่ 1, Z คงที่ 1 → WZ

3

เช็กความครบ

XY คลุม 6, 7, 14, 15 · $W\overline{Y}$ คลุม 8, 9, 12, 13 · WZ คลุม 9, 11, 13, 15 → รวมได้ครบทั้ง 9 ช่อง → พอแล้ว



ผังข้อ 4.2 — โจทย์ให้มา 5 พจน์ ย่อเหลือ 3 พจน์ · m_{10} เป็น 0 จึงจับ WY ทั้งครึ่งไม่ได้

คำตอบข้อ 4.2

$$\mathbf{G = XY + W\overline{Y} + WZ}$$ (3 พจน์ 6 literal — ตรงกับวงเล็บในโจทย์)

เช็กคำตอบกลับเข้าโจทย์ — เช็กช่องที่ต้องเป็น 0 ทั้ง 7 ช่อง เพราะถ้าย่อเกิน ข้อผิดจะโผล่ตรงนี้

ช่อง	$WXYZ$	XY	$W\overline{Y}$	WZ	รวม	ต้องได้
m_0	0000	0	0	0	0	0 ✓
m_1	0001	0	0	0	0	0 ✓
m_2	0010	0	0	0	0	0 ✓
m_3	0011	0	0	0	0	0 ✓
m_4	0100	0	0	0	0	0 ✓
m_5	0101	0	0	0	0	0 ✓
m_{10}	1010	0	0	0	0	0 ✓
m_{11}	1011	0	0	1	1	1 ✓

อีก 8 ช่องที่เป็น 1 ($m_6, m_7, m_8, m_9, m_{12}, m_{13}, m_{14}, m_{15}$) อยู่ในกรอบสีใดสีหนึ่งของภาพ จึงได้ 1 ครบ

จุดที่มักพลาดข้อนี้

เผลอจับครึ่งล่างของผัง ($W = 1$ ทั้ง 8 ช่อง) เป็นกลุ่มเดียวแล้วตอบ $G = W + XY$ · ลืมดูว่า m_{10} **เป็น 0** กลุ่ม 8 ช่องจึงใช้ไม่ได้ · ก่อนตีกรอบใหญ่ ให้ไล่นับให้ครบทุกช่องในกรอบว่าเป็น 1 จริงทุกช่อง

ข้อ 4.3 — $G = AB\overline{C} + \overline{A}\,\overline{C}D + A\overline{B}\,\overline{C} + B\overline{C}D + \overline{A}D$

1

กางทีละพจน์ (ตัวแปรเรียง $A\,B\,C\,D$)

$AB\overline{C}$ = $110-$ → m_{12}, m_{13}

$\overline{A}\,\overline{C}D$ = $0-01$ → ขาด B → $m_1(0001), m_5(0101)$

$A\overline{B}\,\overline{C}$ = $100-$ → m_8, m_9

$B\overline{C}D$ = -101 → ขาด A → $m_5(0101), m_{13}(1101)$

$\overline{A}D$ = $0--1$ → ขาด B กับ C → m_1, m_3, m_5, m_7

รวมช่องที่เป็น 1: $m_1, m_3, m_5, m_7, m_8, m_9, m_{12}, m_{13}$ (8 ช่อง)

2 **มองรูปทรงบนผัง**

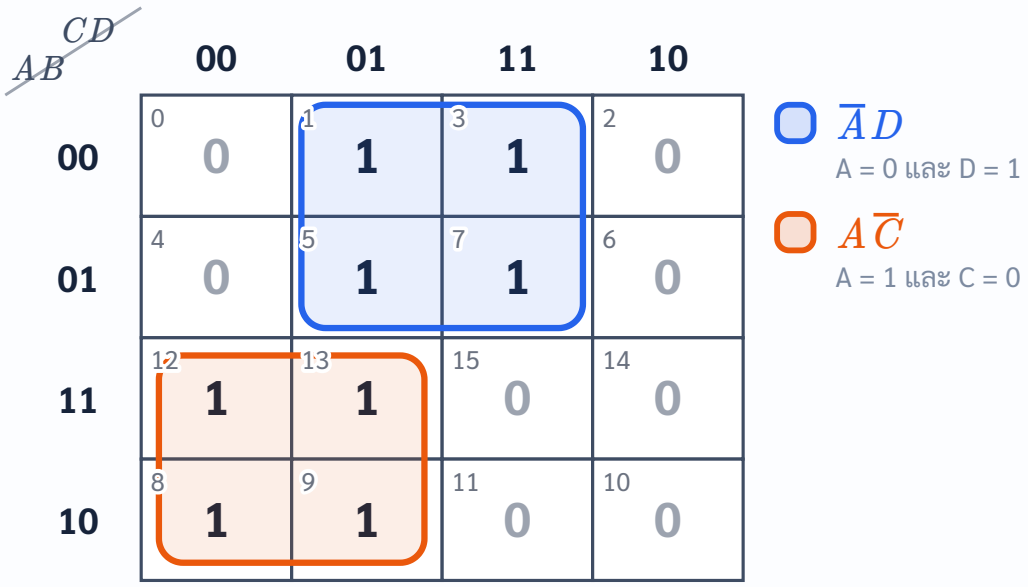
m_1, m_3, m_5, m_7 อยู่ครึ่งบนของผัง ($A = 0$) ในคอลัมน์ $CD = 01$ กับ $CD = 11$ ซึ่งเป็นคอลัมน์ที่ $D = 1$ ทั้งคู่ → เป็นสี่เหลี่ยม 4 ช่องพอดี

m_8, m_9, m_{12}, m_{13} อยู่ครึ่งล่าง ($A = 1$) ในคอลัมน์ $CD = 00$ กับ $CD = 01$ ซึ่งเป็นคอลัมน์ที่ $C = 0$ ทั้งคู่ → เป็นสี่เหลี่ยม 4 ช่องอีกอัน

3 **อ่านพจน์**

กลุ่มฟ้า: A คงที่ 0 → $\overline{A}$ · B เปลี่ยน ตัดทิ้ง · C เปลี่ยน ตัดทิ้ง · D คงที่ 1 → D → $\overline{A}D$

กลุ่มส้ม: A คงที่ 1 → A · B เปลี่ยน ตัดทิ้ง · C คงที่ 0 → $\overline{C}$ · D เปลี่ยน ตัดทิ้ง → $A\overline{C}$



ผังข้อ 4.3 — 5 พจน์ในโจทย์ยุบเหลือ 2 กลุ่มสวย ๆ เพราะหลายพจน์ทับกันอยู่แล้ว

คำตอบข้อ 4.3

$$\mathbf{G = \overline{A}D + A\overline{C}}$$ (2 พจน์ 4 literal — ตรงกับวงเล็บในโจทย์)

เช็กคำตอบกลับเข้าโจทย์ — เช็กครบทั้ง 16 ช่อง โดยแบ่งเป็นครึ่งบน ($A = 0$) กับครึ่งล่าง ($A = 1$)

ช่อง	$ABCD$	$\overline{A}D$	$A\overline{C}$	รวม	นิพจน์เดิม
m_0	0000	0	0	0	0 ✓
m_1	0001	1	0	1	1 ✓
m_2	0010	0	0	0	0 ✓
m_3	0011	1	0	1	1 ✓
m_4	0100	0	0	0	0 ✓
m_5	0101	1	0	1	1 ✓
m_6	0110	0	0	0	0 ✓
m_7	0111	1	0	1	1 ✓
m_8	1000	0	1	1	1 ✓
m_9	1001	0	1	1	1 ✓
m_{10}	1010	0	0	0	0 ✓
m_{11}	1011	0	0	0	0 ✓
m_{12}	1100	0	1	1	1 ✓
m_{13}	1101	0	1	1	1 ✓
m_{14}	1110	0	0	0	0 ✓
m_{15}	1111	0	0	0	0 ✓

จุดที่มักพลาดข้อนี้

พจน์ $B\overline{C}D$ ในโจทย์คลุม m_5 กับ m_{13} ซึ่ง**ถูกพจน์อื่นคลุมไว้หมดแล้ว** (m_5 อยู่ใน $\overline{A}D$, m_{13} อยู่ใน $A\overline{C}$) · คนที่ย่อด้วยพีชคณิตมักลืมตัดพจน์ซ้ำแบบนี้ทิ้ง แต่ K-map จะเห็นทันทีเพราะไม่มีช่องใหม่ให้คลุม

ข้อ 4.4 — $F = \overline{P}\,\overline{Q}R + \overline{P}Q\overline{R} + \overline{P}QR + PQ\overline{R} + P\overline{Q}\,\overline{R}$

ข้อนี้มี 3 ตัวแปร (P, Q, R) และทุกพจน์มีตัวแปรครบ 3 ตัวอยู่แล้ว (เป็น standard SOP) จึงไม่ต้องกาง — อ่านเป็นเลข minterm ได้ตรง ๆ โดยใช้กฎ **ตัวไม่มีขีด = 1, ตัวมีขีด = 0**

1 **แปลงแต่ละพจน์เป็นเลข minterm**

$\overline{P}\,\overline{Q}R \to P=0, Q=0, R=1 \to 001_2 = 1 \to m_1$

$\overline{P}Q\overline{R} \to 010_2 = 2 \to m_2$ · $\overline{P}QR \to 011_2 = 3 \to m_3$

$PQ\overline{R} \to 110_2 = 6 \to m_6$ · $P\overline{Q}\,\overline{R} \to 100_2 = 4 \to m_4$

$F = \sum m(1,2,3,4,6)$ · ช่องที่เป็น 0 คือ $m_0(000), m_5(101), m_7(111)$

2

หาช่องที่บังคับกลุ่ม

$m_1(001)$: เพื่อนบ้าน $m_0 = 0, m_3 = 1, m_5 = 0$ → จับได้กับ m_3 ทางเดียว → กลุ่ม m_1, m_3 → P คงที่ 0, R คงที่ 1 → $\overline{P}R$ **ต้องใช้**

$m_4(100)$: เพื่อนบ้าน $m_0 = 0, m_5 = 0, m_6 = 1$ → จับได้กับ m_6 ทางเดียว → กลุ่ม m_4, m_6 → P คงที่ 1, R คงที่ 0 → $P\overline{R}$ **ต้องใช้**

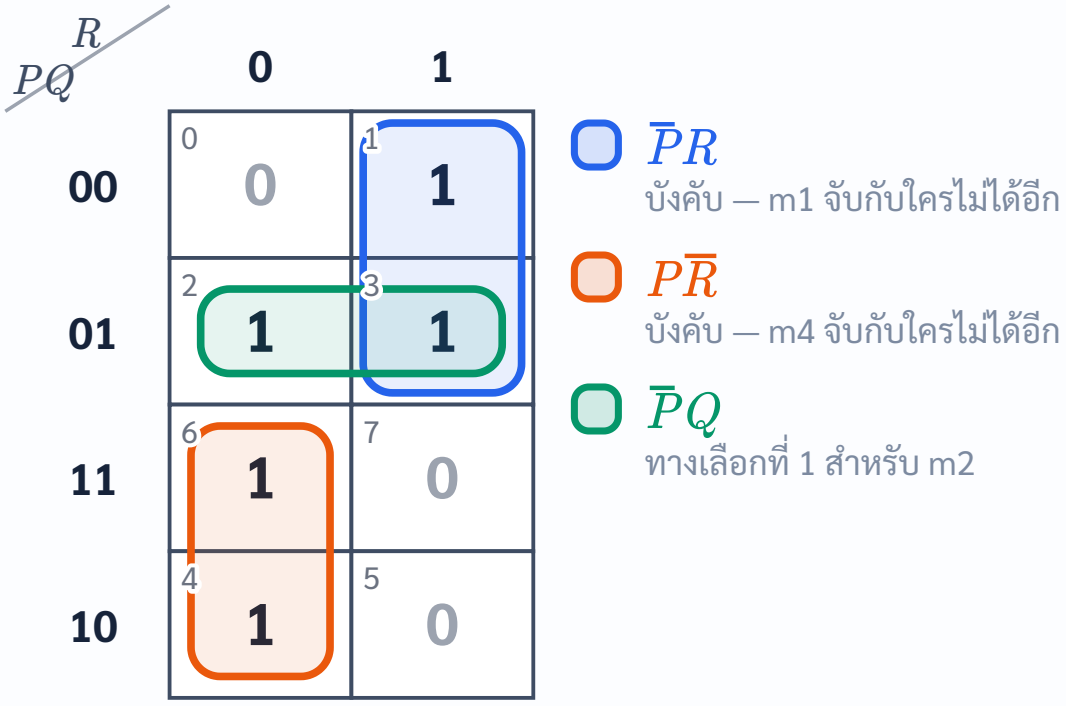
3

เหลือ m_2 ช่องเดียว — และตรงนี้มีสองทางเลือกที่ดีเท่ากัน

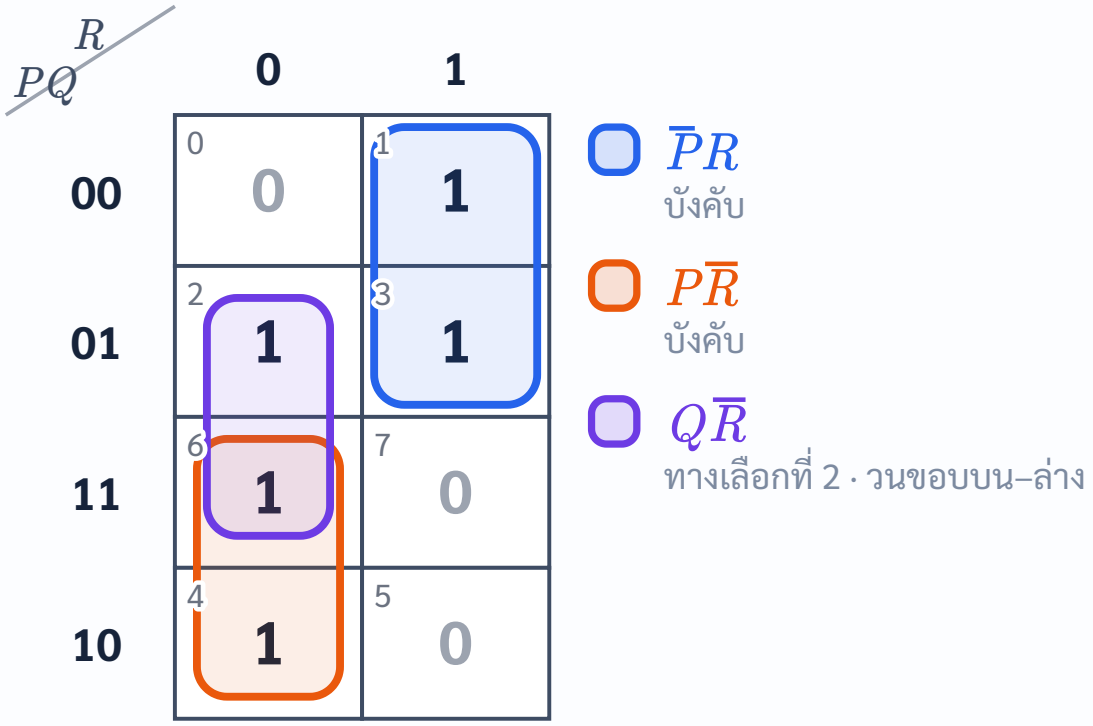
ทางที่ 1: จับ m_2 กับ m_3 → P คงที่ 0, Q คงที่ 1 → $\overline{P}Q$

ทางที่ 2: จับ m_2 กับ m_6 (ขอบบนติดขอบล่าง) → Q คงที่ 1, R คงที่ 0 → $Q\overline{R}$

สองทางนี้ได้ 3 พจน์ 6 literal เท่ากันเป๊ะ → **ข้อนี้มีคำตอบถูก 2 แบบ ตอบแบบไหนก็ได้**



ผังข้อ 4.4 แบบที่ 1 — คลุม m_2 ด้วย $\overline{P}Q$



ผังข้อ 4.4 แบบที่ 2 — คลุม m_2 ด้วย $Q\overline{R}$ · สั้นเท่ากันทุกประการ

คำตอบข้อ 4.4 (ถูกทั้งสองแบบ)

$$\mathbf{F = \overline{P}R + P\overline{R} + \overline{P}Q} \quad \text{หรือ} \quad \mathbf{F = \overline{P}R + P\overline{R} + Q\overline{R}}$$

ทั้งคู่ 3 พจน์ 6 literal — ตรงกับวงเล็บในโจทย์

เช็กคำตอบกลับเข้าโจทย์ — เช็กทั้ง 8 แถว ทั้งสองแบบพร้อมกัน

PQR	$\overline{P}R$	$P\overline{R}$	$\overline{P}Q$	แบบ 1	$Q\overline{R}$	แบบ 2	ต้องได้
000	0	0	0	0	0	0	0 ✓
001	1	0	0	1	0	1	1 ✓
010	0	0	1	1	1	1	1 ✓
011	1	0	1	1	0	1	1 ✓
100	0	1	0	1	0	1	1 ✓
101	0	0	0	0	0	0	0 ✓
110	0	1	0	1	1	1	1 ✓
111	0	0	0	0	0	0	0 ✓

จุดที่มักพลาดข้อนี้

จับทั้ง $\overline{P}Q$ และ $Q\overline{R}$ ใส่ไปพร้อมกัน ได้ 4 พจน์ ทั้งที่ 3 พจน์ก็คลุมครบแล้ว · **เมื่อมีทางเลือกที่ดีเท่ากันหลายทาง ให้เลือกแค่ทางเดียว** · และอย่าตกใจถ้าคำตอบของเราไม่เหมือนเพื่อน ให้ดูที่จำนวนพจน์กับจำนวน literal ว่าเท่ากันไหม

5 หา *ทุก* คำตอบ SOP ที่สั้นที่สุด

โจทย์ข้อ 5 (ต้นฉบับ) Assignment #5

> 5. For the following expression, find all minimum SOP solutions:
> $F(A,B,C,D) = \sum m(0,2,3,5,7,8,10,11,12,13,14,15)$ (4 solutions)

แปลไทย: ข้อนี้ไม่ได้ขอคำตอบเดียว แต่ขอ **ทุกคำตอบที่สั้นเท่ากัน** — โจทย์บอกใบ้ว่ามี 4 คำตอบ

โจทย์ซ่อนอะไรไว้: คำว่า “all minimum solutions” แปลว่าต้องหาให้ครบ ไม่ใช่หาเจอหนึ่งแล้วจบ วิธีคิดคือ **แยกกลุ่มที่บังคับ (essential) ออกจากกลุ่มที่เลือกได้** แล้วนับว่าทางเลือกมีกี่ทาง

วิธีที่ใช้ หากลุ่มบังคับก่อน แล้วค่อยแจกแจงทางเลือกที่เหลือ — **บทที่ 4 · K-Map**

1 พล็อตลงผัง — ช่อง 0 มีแค่ 4 ช่อง จำง่ายกว่าไล่ช่อง 1

โจทย์ให้ 1 มา 12 ช่อง ช่องที่**ไม่อยู่ในรายชื่อ**คือ m_1, m_4, m_6, m_9 → สี่ช่องนี้เป็น 0

$m_1 = 0001 \cdot m_4 = 0100 \cdot m_6 = 0110 \cdot m_9 = 1001$

2 หากลุ่มบังคับ — ดูช่อง 1 ที่ขยายได้ทางเดียว

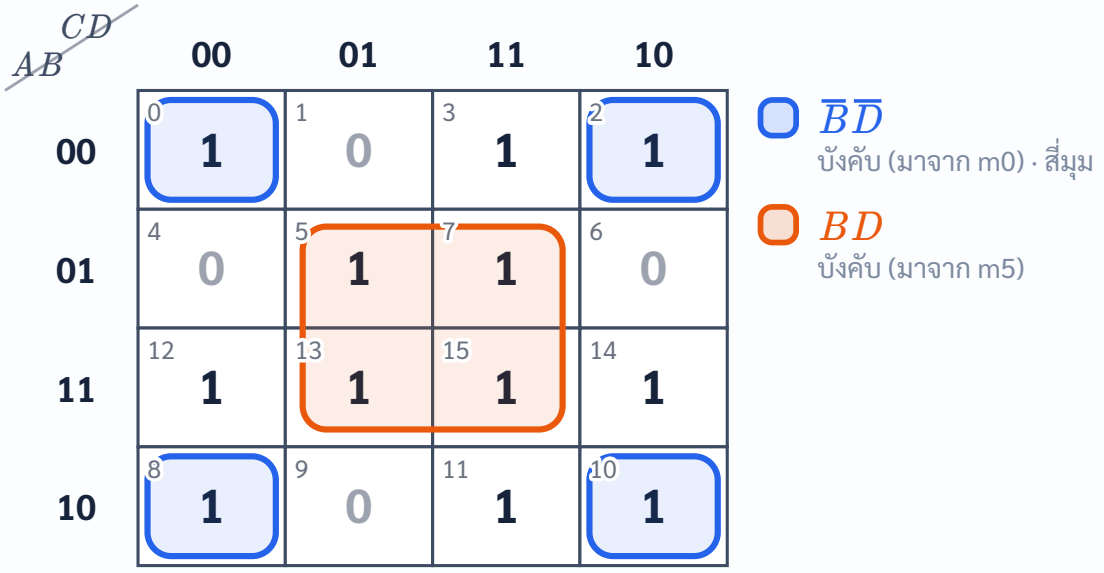
$m_0(0000)$: เพื่อนบ้าน $m_1 = 0, m_2 = 1, m_4 = 0, m_8 = 1$ → ขยายได้เฉพาะทาง m_2 กับ m_8 → กลุ่มใหญ่สุดคือสี่มุม m_0, m_2, m_8, m_{10} → B คงที่ 0, D คงที่ 0 → $\overline{B}\,\overline{D}$ **บังคับใช้**

$m_5(0101)$: เพื่อนบ้าน $m_4 = 0, m_7 = 1, m_1 = 0, m_{13} = 1$ → ขยายได้เฉพาะทาง m_7 กับ m_{13} → กลุ่มใหญ่สุดคือ m_5, m_7, m_{13}, m_{15} → B คงที่ 1, D คงที่ 1 → BD **บังคับใช้**

3 เช็กว่าเหลือช่องไหนบ้าง

$\overline{B}\,\overline{D}$ คลุม 0, 2, 8, 10 · BD คลุม 5, 7, 13, 15 → รวม 8 ช่อง

ช่อง 1 มี 12 ช่อง → ยังเหลือ $m_3, m_{11}, m_{12}, m_{14}$ ที่ต้องหากลุ่มมาคลุม



ผังข้อ 5 ขั้นที่ 1 — สองกลุ่มบังคับ · ช่องที่ยังไม่ถูกกรอบคลุมคือ $m_3, m_{11}, m_{12}, m_{14}$

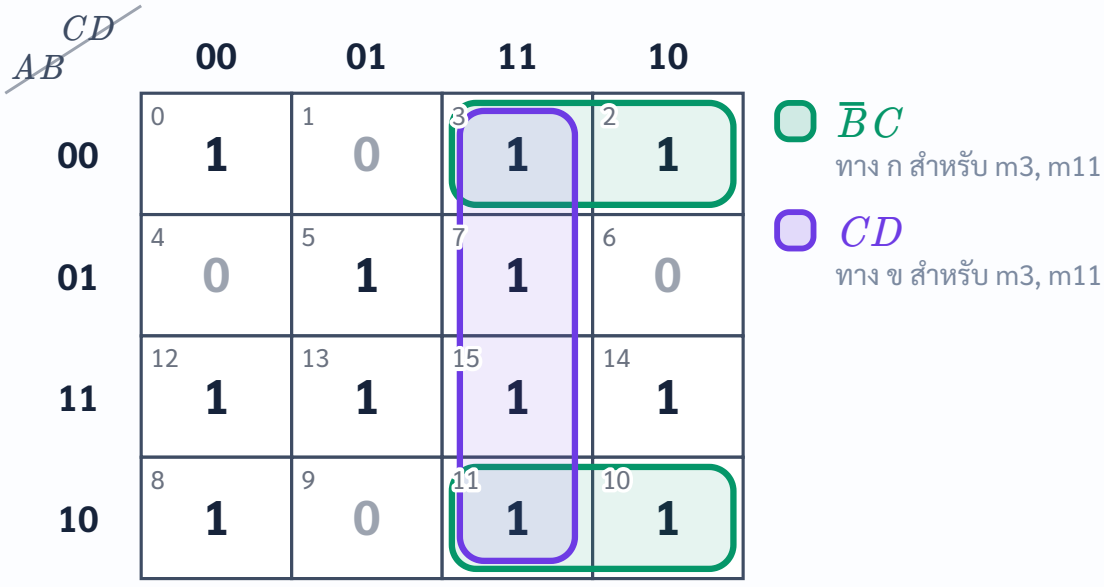
4

คลุม m_3 กับ m_{11} — มี 2 ทาง

ทาง ก: m_2, m_3, m_{10}, m_{11} → B คงที่ 0, C คงที่ 1 → $\overline{B}C$

ทาง ข: m_3, m_7, m_{11}, m_{15} (ทั้งคอลัมน์ $CD = 11$) → C คงที่ 1, D คงที่ 1 → CD

ทั้งสองทางเป็นกลุ่ม 4 ช่อง 2 literal เท่ากัน → เลือกทางไหนก็สั้นเท่ากัน



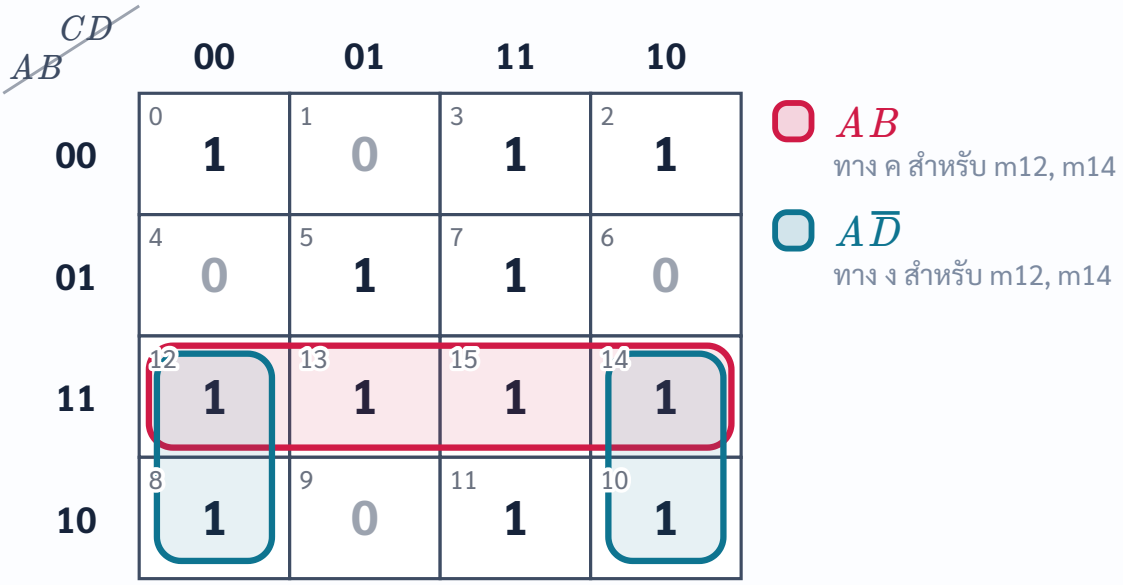
ผังข้อ 5 ขั้นที่ 2 — สองทางเลือกที่คลุม m_3 กับ m_{11} ได้เหมือนกัน

5

คลุม m_{12} กับ m_{14} — มี 2 ทางเช่นกัน

ทาง ค: $m_{12}, m_{13}, m_{15}, m_{14}$ (ทั้งแถว $AB = 11$) → A คงที่ 1, B คงที่ 1 → AB

ทาง ง: $m_8, m_{10}, m_{12}, m_{14}$ → A คงที่ 1, D คงที่ 0 → $A\overline{D}$



ผังข้อ 5 ขั้นที่ 3 — สองทางเลือกที่คลุม m_{12} กับ m_{14} ได้เหมือนกัน

6

คูณทางเลือกเข้าด้วยกัน

ทางเลือกสำหรับ m_3, m_{11} มี 2 ทาง และทางเลือกสำหรับ m_{12}, m_{14} มี 2 ทาง → จับคู่ได้ $2 \times 2 = 4$ แบบ ซึ่งตรงกับที่โจทย์บอกว่ามี 4 solutions พอดี

ทุกแบบมี 4 พจน์ พจน์ละ 2 literal → 8 literal เท่ากันหมด

คำตอบข้อ 5 — ครบทั้ง 4 แบบ

แบบที่ 1 $\mathbf{F = \overline{B}\,\overline{D} + BD + \overline{B}C + AB}$

แบบที่ 2 $\mathbf{F = \overline{B}\,\overline{D} + BD + \overline{B}C + A\overline{D}}$

แบบที่ 3 $\mathbf{F = \overline{B}\,\overline{D} + BD + CD + AB}$

แบบที่ 4 $\mathbf{F = \overline{B}\,\overline{D} + BD + CD + A\overline{D}}$

ทุกแบบ 4 พจน์ 8 literal

เช็กคำตอบกลับเข้าโจทย์ — เช็กว่าแต่ละแบบให้ 0 ครบทั้ง 4 ช่องที่ต้องเป็น 0 และให้ 1 ที่ $m_3, m_{11}, m_{12}, m_{14}$

ช่อง	$ABCD$	แบบ 1	แบบ 2	แบบ 3	แบบ 4	ต้องได้
m_1	0001	0	0	0	0	0 ✓
m_4	0100	0	0	0	0	0 ✓
m_6	0110	0	0	0	0	0 ✓
m_9	1001	0	0	0	0	0 ✓
m_3	0011	1	1	1	1	1 ✓
m_{11}	1011	1	1	1	1	1 ✓
m_{12}	1100	1	1	1	1	1 ✓
m_{14}	1110	1	1	1	1	1 ✓

ตัวอย่างการไล่หนึ่งช่องให้เห็นเต็ม ๆ — ที่ $m_9 = 1001$ ($A=1, B=0, C=0, D=1$) ในแบบที่ 1: $\overline{B}\,\overline{D} = 1 \cdot 0 = 0 \cdot BD = 0 \cdot 1 = 0 \cdot \overline{B}C = 1 \cdot 0 = 0 \cdot AB = 1 \cdot 0 = 0$ → รวม $0+0+0+0=0$ ถูกต้อง

ส่วนช่อง 1 ที่เหลือ ($m_0, m_2, m_5, m_7, m_8, m_{10}, m_{13}, m_{15}$) ถูกคลุมด้วยกลุ่มบังคับสองกลุ่มอยู่แล้วทุกแบบ

จุดที่มักพลาดข้อนี้

หาเจอแบบเดียวแล้วหยุด ได้คะแนนแค่ 1 ใน 4 · เทคนิคคือ **ทำเครื่องหมายกลุ่มบังคับก่อนเสมอ** แล้วดูว่าช่องที่เหลือมีกี่จุดตัดสินใจ ถ้ามี 2 จุด จุดละ 2 ทาง คำตอบจะมี $2 \times 2 = 4$ แบบ

สรุป Assignment 5 หนึ่งบรรทัด

ทุกข้อใช้ขั้นตอนเดียวกัน: แปลงโจทย์ให้เป็นช่องบนผังก่อน → จับกลุ่ม 1 ให้ใหญ่ที่สุดและน้อยกลุ่มที่สุดได้ SOP → จับกลุ่ม 0 แล้ว De Morgan ได้ POS → แทนค่ากลับทุกแถวเพื่อตรวจ

ภาคผนวก ง · เฉลย Assignment 6 ทุกข้อ

Assignment 6 มี 4 ข้อใหญ่ ทั้งหมดอยู่ในเรื่อง **วงจรผสมที่เอาไปใช้งานจริง** ซึ่งสอนไว้ใน **บทที่ 5 · วงจรผสม** ของเล่มนี้ — ข้อ 1 กับ 2 เป็นวงจรบวก ข้อ 3 เป็นตัวถอดรหัสไปจอ 7 ส่วน ข้อ 4 เป็นมัลติเพล็กเซอร์กับดีมัลติเพล็กเซอร์

1 เติมตารางค่าความจริงของ full-adder แล้วบวกเลข 4 บิต

โจทย์ข้อ 1 (ต้นฉบับ) Assignment #6

> 1. Complete the truth table below for each stage of a 4-bit parallel adder.
> Then, find the sum and output carry for the addition of the following two 4-bit numbers if the input carry (C_{in}) is 0:
> $A_3A_2A_1A_0 = 1100$ and $B_3B_2B_1B_0 = 1100$

แปลไทย: ตารางที่โจทย์ให้มามีคอลัมน์อินพุต 3 คอลัมน์ (C_{n-1}, A_n, B_n) เติมมาครบ 8 แถวแล้ว แต่คอลัมน์เอาต์พุต 2 คอลัมน์ (Σ_n กับ C_n) ว่างอยู่ ให้เราเติม · จากนั้นเอาตารางนั้นไปบวกเลข $1100 + 1100$ โดยตัวทดเข้า $= 0$

โจทย์ซ่อนอะไรไว้: ตารางนี้คือ **full-adder หนึ่งตัว** ไม่ใช่ทั้งวงจร 4 บิต · วงจร 4 บิตคือ full-adder 4 ตัวต่อกัน โดย C_n ของหลักหนึ่งกลายเป็น C_{n-1} ของหลักถัดไป · และต้องตอบ**ทั้งผลบวกและตัวทดออก** (C_{out}) ไม่ใช่แค่ผลบวก

วิธีที่ใช้ full-adder และ parallel adder — **บทที่ 5 · วงจรผสม**

1 เข้าใจก่อนว่าตารางนี้ถามอะไร

full-adder รับเลข 3 ตัว คือบิตของ A บิตของ B และตัวทดที่วิ่งมาจากหลักขวา (C_{n-1}) แล้วบวกกันหมด · ผลบวกที่ได้มีค่าได้ตั้งแต่ 0 ถึง 3 ซึ่งเขียนเป็นเลขฐานสอง 2 บิต → **บิตขวาคือ Σ_n บิตซ้ายคือ C_n**

2 เติมทีละแถว โดย “นับจำนวน 1 ในสามคอลัมน์อินพุต”

นับได้ 0 ตัว → ผลรวม 0 → เขียนฐานสอง 00 → $\Sigma = 0, C = 0$

นับได้ 1 ตัว → ผลรวม 1 → 01 → $\Sigma = 1, C = 0$

นับได้ 2 ตัว → ผลรวม 2 → 10 → $\Sigma = 0, C = 1$

นับได้ 3 ตัว → ผลรวม 3 → 11 → $\Sigma = 1, C = 1$

ตารางที่เติมครบแล้ว (ช่องสองคอลัมน์ขวาคือคำตอบของข้อนี้)

C_{n-1}	A_n	B_n	Σ_n	C_n	นับ 1 ได้กี่ตัว / ผลรวมฐานสอง
0	0	0	0	0	0 ตัว → $0 = 00_2$
0	0	1	1	0	1 ตัว → $1 = 01_2$
0	1	0	1	0	1 ตัว → $1 = 01_2$
0	1	1	0	1	2 ตัว → $2 = 10_2$
1	0	0	1	0	1 ตัว → $1 = 01_2$
1	0	1	0	1	2 ตัว → $2 = 10_2$
1	1	0	0	1	2 ตัว → $2 = 10_2$
1	1	1	1	1	3 ตัว → $3 = 11_2$

อ่านจากตารางเป็นนิพจน์ได้ว่า $\Sigma_n = A_n \oplus B_n \oplus C_{n-1}$ (เป็น 1 เมื่อจำนวน 1 เป็นเลขคี่) และ $C_n = A_nB_n + (A_n \oplus B_n)C_{n-1}$ (เป็น 1 เมื่อมี 1 ตั้งแต่สองตัวขึ้นไป)

ตอนที่สอง — เอาตารางไปบวก $1100 + 1100$ โดย $C_{in} = 0$

วงจรบวกแบบขนาน 4 บิต คือ full-adder 4 ตัวต่อกันเป็นแถว · **เริ่มที่หลักขวาสุด (หลักที่ 0) เสมอ** เพราะตัวทดต้องวิ่งจากขวาไปซ้าย ทำหลักซ้ายก่อนไม่ได้เพราะยังไม่รู้ตัวทดที่จะเข้ามา

0 **หลักที่ 0 (ขวาสุด)**

$A_0 = 0, B_0 = 0, C_{-1} = C_{in} = 0$ → นับ 1 ได้ 0 ตัว → เปิดตารางแถว 000 → $\Sigma_0 = 0, C_0 = 0$

1 **หลักที่ 1**

$A_1 = 0, B_1 = 0$, ตัวทดเข้า $= C_0 = 0$ → นับ 1 ได้ 0 ตัว → แถว 000 → $\Sigma_1 = 0, C_1 = 0$

2 **หลักที่ 2**

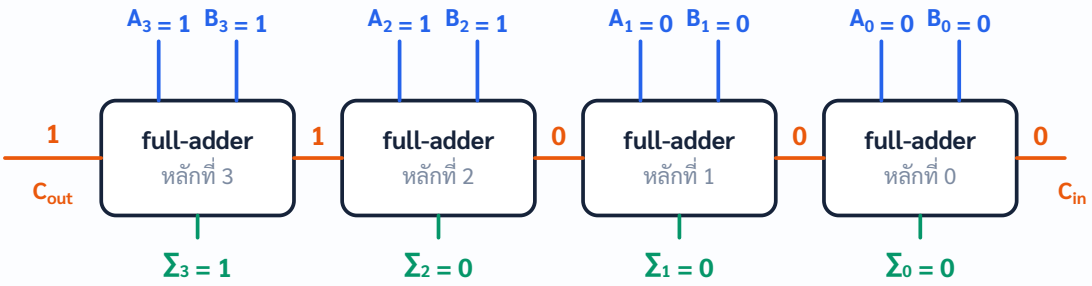
$A_2 = 1, B_2 = 1$, ตัวทดเข้า $= C_1 = 0$ → นับ 1 ได้ 2 ตัว → เปิดตารางแถว $C_{n-1}A_nB_n = 011$ → $\Sigma_2 = 0, C_2 = 1$

ตรงนี้คือจุดที่ตัวทดเกิดขึ้น: $1 + 1 = 10_2$ เขียน 0 ทดไป 1

3 **หลักที่ 3 (ซ้ายสุด)**

$A_3 = 1, B_3 = 1$, ตัวทดเข้า $= C_2 = 1$ → นับ 1 ได้ 3 ตัว → เปิดตารางแถว 111 → $\Sigma_3 = 1, C_3 = 1$

C_3 คือตัวทดที่ล้นออกนอกวงจร เรียกว่า C_{out} → $C_{out} = 1$



วงจรบวกขนาน 4 บิตพร้อมค่าจริงทุกสาย — สีส้มคือตัวทดที่วิ่งจากขวาไปซ้าย สีเขียวคือผลบวกแต่ละหลัก

เขียนแบบตั้งหลักให้เห็นตัวทด (เทียบกับวิธีบวกมือที่เคยทำ)

บวกฐานสอง

```
 1 1  0  0  ← ตัวทด
   1 1 0 0

+  1 1 0 0
-------------

1 1 0 0 0
```

คำตอบข้อ 1

ตารางเติมครบแล้วด้านบน · $\mathbf{\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = 1000}$ และ $\mathbf{C_{out} = 1}$

อ่านรวมทั้งหมดเป็นเลข 5 บิตคือ 11000_2

เช็กคำตอบกลับเข้าโจทย์ — แปลงเป็นฐานสิบแล้วบวกมือ

ตรวจด้วยฐานสิบ

$A = 1100_2 = 8 + 4 + 0 + 0 = 12$

$B = 1100_2 = 8 + 4 + 0 + 0 = 12$

$C_{in} = 0$

รวม $= 12 + 12 + 0 = 24$

คำตอบที่ได้ $= C_{out}\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = 11000_2 = 16 + 8 + 0 + 0 + 0 = 24$ ✓

ตรงกัน แปลว่าเติมตารางและไล่ตัวทดถูกต้อง

จุดที่มักพลาดข้อนี้

ตอบแค่ $\Sigma = 1000$ แล้วจบ **ลืม** C_{out} — ซึ่งทำให้คำตอบกลายเป็น 8 แทนที่จะเป็น 24 ผิดไปครึ่งหนึ่ง · เลข 4 บิตบวกกันได้มากสุด $15 + 15 + 1 = 31$ ซึ่งต้องใช้ 5 บิต ดังนั้น**ต้องมี** C_{out} **ในคำตอบเสมอ**

2 บวกด้วยวงจร look-ahead carry adder 4 ชั้น

โจทย์ข้อ 2 (ต้นฉบับ) Assignment #6

> 2. Add A = 1001 and B = 0101 with C_0 = 1 using a 4-stage look-ahead carry adder. Write the look-ahead expression for each carry and evaluate numerically. Also compute the sum bits and the output carry C_{out}.

แปลไทย: บวก $A = 1001$ กับ $B = 0101$ โดยตัวทดเข้า $C_0 = 1$ ด้วยวงจรแบบ look-ahead carry 4 ชั้น · ต้อง**เขียนสูตร look-ahead ของตัวทดทุกตัว** แล้วแทนค่าเป็นตัวเลข · และหาบิตผลบวกกับ C_{out} ด้วย

โจทย์ซ่อนอะไรไว้: โจทย์ไม่ได้ให้บวกเฉย ๆ แต่ให้**เขียนสูตรออกมาก่อน** — ถ้าตอบแค่ตัวเลขจะได้คะแนนไม่เต็ม · และ C_0 ในโจทย์คือ**ตัวทดที่ป้อนเข้าวงจร** (ตัวเดียวกับที่สไลด์เรียกว่า C_{in}) ไม่ใช่ตัวทดที่ออกจากชั้นที่ 0

วิธีที่ใช้ look-ahead carry adder — **บทที่ 5 · วงจรผสม**

ทบทวนสูตรก่อนลงมือ

วงจรบวกแบบธรรมดา (ripple carry) ช้าเพราะหลักที่ 3 ต้องรอตัวทดจากหลัก 2 ซึ่งรอหลัก 1 ซึ่งรอหลัก 0 · look-ahead แก้ด้วยการ**คำนวณตัวทดทุกตัวพร้อมกันจากอินพุตตรง ๆ**

แต่ละหลักมีสองอย่างที่ต้องหาก่อน

$$C_{g_i} = A_i B_i \text{ (generated carry — หลักนี้สร้างตัวทดเองได้)}$$

$$C_{p_i} = A_i + B_i \text{ (propagated carry — หลักนี้ส่งตัวทดที่ได้รับมาต่อไปได้)}$$

แล้วตัวทดออกของแต่ละชั้นคือ

$$C_{out_0} = C_{g_0} + C_{p_0} C_{in}$$

$$C_{out_1} = C_{g_1} + C_{p_1} C_{g_0} + C_{p_1} C_{p_0} C_{in}$$

$$C_{out_2} = C_{g_2} + C_{p_2} C_{g_1} + C_{p_2} C_{p_1} C_{g_0} + C_{p_2} C_{p_1} C_{p_0} C_{in}$$

$$C_{out_3} = C_{g_3} + C_{p_3} C_{g_2} + C_{p_3} C_{p_2} C_{g_1} + C_{p_3} C_{p_2} C_{p_1} C_{g_0} + C_{p_3} C_{p_2} C_{p_1} C_{p_0} C_{in}$$

และบิตผลบวกยังใช้สูตรเดิมของ full-adder คือ $\Sigma_i = A_i \oplus B_i \oplus C_{i-1}$ โดย C_{i-1} คือตัวทดที่**เข้า**หลักนั้น

1 **แยกบิตให้ถูกหลักก่อน (บิตขวาสุดคือหลักที่ 0)**

$A = 1001 \rightarrow A_3 = 1, A_2 = 0, A_1 = 0, A_0 = 1$

$B = 0101 \rightarrow B_3 = 0, B_2 = 1, B_1 = 0, B_0 = 1$ · $C_{in} = C_0 = 1$

2 **หา C_g และ C_p ของทุกหลัก — ทำได้พร้อมกันทั้ง 4 หลักเพราะใช้แค่อินพุต**

หลัก 0: $C_{g_0} = A_0 B_0 = 1 \cdot 1 = \mathbf{1}$ · $C_{p_0} = A_0 + B_0 = 1 + 1 = \mathbf{1}$

หลัก 1: $C_{g_1} = 0 \cdot 0 = \mathbf{0}$ · $C_{p_1} = 0 + 0 = \mathbf{0}$

หลัก 2: $C_{g_2} = 0 \cdot 1 = \mathbf{0}$ · $C_{p_2} = 0 + 1 = \mathbf{1}$

หลัก 3: $C_{g_3} = 1 \cdot 0 = \mathbf{0}$ · $C_{p_3} = 1 + 0 = \mathbf{1}$

หลัก i	A_i	B_i	$C_{g_i} = A_i B_i$	$C_{p_i} = A_i + B_i$
0	1	1	**1**	**1**
1	0	0	**0**	**0**
2	0	1	**0**	**1**
3	1	0	**0**	**1**

3 **แทนค่าลงสูตรตัวทดชั้นที่ 0**

$C_{out_0} = C_{g_0} + C_{p_0} C_{in}$ เขียนสูตรก่อน

$C_{out_0} = 1 + (1)(1)$ แทนค่า

$C_{out_0} = 1 + 1 = \mathbf{1}$ OR ของ 1 กับ 1 คือ 1

4 **ชั้นที่ 1**

$C_{out_1} = C_{g_1} + C_{p_1} C_{g_0} + C_{p_1} C_{p_0} C_{in}$ สูตร

$C_{out_1} = 0 + (0)(1) + (0)(1)(1)$ แทนค่า

$C_{out_1} = 0 + 0 + 0 = \mathbf{0}$ $C_{p_1} = 0$ จึงดับสองพจน์หลังทั้งคู่

5 **ชั้นที่ 2**

$C_{out_2} = C_{g_2} + C_{p_2} C_{g_1} + C_{p_2} C_{p_1} C_{g_0} + C_{p_2} C_{p_1} C_{p_0} C_{in}$ สูตร

$C_{out_2} = 0 + (1)(0) + (1)(0)(1) + (1)(0)(1)(1)$ แทนค่า

$C_{out_2} = 0 + 0 + 0 + 0 = \mathbf{0}$ ทุกพจน์มี $C_{g_2} = 0$ หรือ $C_{p_1} = 0$ คูณอยู่

6

ชั้นที่ 3 — พจน์ยาวที่สุด แต่หลักการเดิม

$C_{out_3} = C_{g_3} + C_{p_3}C_{g_2} + C_{p_3}C_{p_2}C_{g_1} + C_{p_3}C_{p_2}C_{p_1}C_{g_0} + C_{p_3}C_{p_2}C_{p_1}C_{p_0}C_{in}$ สูตร

$C_{out_3} = 0 + (1)(0) + (1)(1)(0) + (1)(1)(0)(1) + (1)(1)(0)(1)(1)$ แทนค่า

$C_{out_3} = 0 + 0 + 0 + 0 + 0 = \mathbf{0}$ ตัวทดที่หลัก 0 สร้างไว้ วิ่งต่อไม่ได้ เพราะหลัก 1 มี $C_{p_1} = 0$ ขวางอยู่

7

หาบิตผลบวก — ใช้ตัวทดที่ เข้า แต่ละหลัก

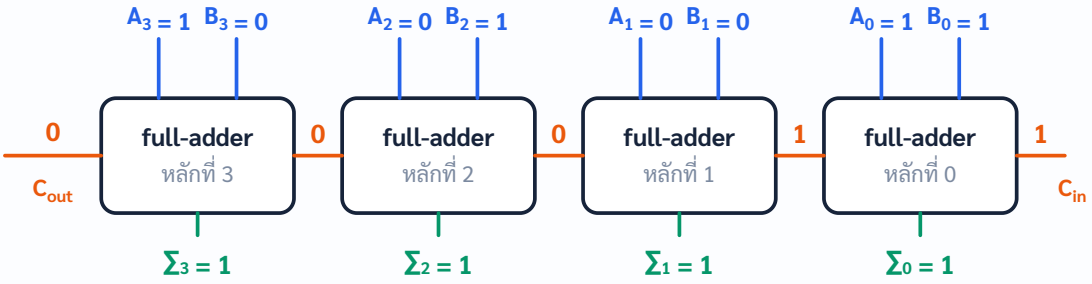
ตัวทดเข้าหลัก 0 คือ $C_{in} = 1 \rightarrow \Sigma_0 = A_0 \oplus B_0 \oplus C_{in} = 1 \oplus 1 \oplus 1 = \mathbf{1}$ (จำนวน 1 เป็นเลขคี่)

ตัวทดเข้าหลัก 1 คือ $C_{out_0} = 1 \rightarrow \Sigma_1 = 0 \oplus 0 \oplus 1 = \mathbf{1}$

ตัวทดเข้าหลัก 2 คือ $C_{out_1} = 0 \rightarrow \Sigma_2 = 0 \oplus 1 \oplus 0 = \mathbf{1}$

ตัวทดเข้าหลัก 3 คือ $C_{out_2} = 0 \rightarrow \Sigma_3 = 1 \oplus 0 \oplus 0 = \mathbf{1}$

ตัวทดออกจากวงจรคือ $C_{out} = C_{out_3} = \mathbf{0}$



ค่าทุกเส้นของการบวกข้อ 2 — สังเกตว่าตัวทดที่หลัก 0 สร้างขึ้น (สีส้ม 1) ตายที่หลัก 1 ไปต่อไม่ได้

คำตอบข้อ 2

$$\mathbf{C_{out_0} = 1 \quad C_{out_1} = 0 \quad C_{out_2} = 0 \quad C_{out_3} = 0}$$

$$\mathbf{\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = 1111 \text{ และ } C_{out} = 0}$$

เช็กคำตอบกลับเข้าโจทย์ — แปลงเป็นฐานสิบ

ตรวจด้วยฐานสิบ

$A = 1001_2 = 8 + 0 + 0 + 1 = 9$

$B = 0101_2 = 0 + 4 + 0 + 1 = 5$

$C_0 = 1$

รวม $= 9 + 5 + 1 = 15$

คำตอบที่ได้ $= C_{out}\Sigma_3\Sigma_2\Sigma_1\Sigma_0 = 01111_2 = 0 + 8 + 4 + 2 + 1 = 15$ ✓

ตรงกัน · และสังเกตว่า look-ahead ให้**ผลลัพธ์เท่ากับ ripple carry ทุกประการ** ต่างกันแค่ความเร็ว ไม่ใช่ค่าที่ได้

จุดที่มักพลาดข้อนี้

ใช้ $C_p = A \oplus B$ (XOR) แทน $C_p = A + B$ (OR) · วิชานี้ใช้นิยาม $C_p = A + B$ ตามสไลด์ · อีกจุดคือลืมว่า Σ_i ใช้ตัวทดที่**เข้า**หลักนั้น ไม่ใช่ตัวทดที่**ออก**จากหลักนั้น เช่น Σ_1 ใช้ C_{out_0} ไม่ใช่ C_{out_1}

3 ตัวถอดรหัส BCD → จอ 7 ส่วน (74LS48)

โจทย์ข้อ 3 (ต้นฉบับ) Assignment #6

> 3. Complete the following truth table for an active high BCD-to-7-segment decoder (74LS48). Using Karnaugh mapping, obtain the minimum SOP and POS expressions for LED *a-d*.

แปลไทย: เติมตารางค่าความจริงของตัวถอดรหัส BCD ไปจอ 7 ส่วนชนิด **active high** (เบอร์ 74LS48) แล้วใช้ K-map หานิพจน์ SOP และ POS ที่สั้นที่สุด **เฉพาะหลอด a, b, c, d** (หลอด e, f, g เติมตารางให้ครบแต่ไม่ต้องหานิพจน์)

โจทย์ซ่อนอะไรไว้ 3 อย่าง:

① **active high** แปลว่า **1 = หลอดติด** · ถ้าเป็นเบอร์ 74LS47 (active low) จะกลับกันคือ 0 = ติด — ข้อสอบชอบสลับเบอร์

② อินพุตเป็น **BCD** ซึ่งใช้แค่ 0–9 · รหัส 1010 ถึง 1111 (10–15) **ไม่มีทางเกิดขึ้น** โจทย์จึงเติม X ให้แล้วทั้ง 6 แถว — เอาไปใช้ย่อได้เต็มที่

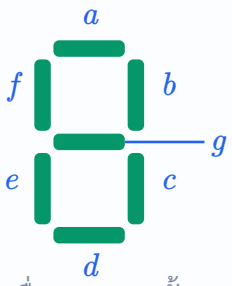
③ แถวแรก (0000) โจทย์เติมให้แล้วเป็น 1111110 นั่นคือ**ตัวอย่างให้ดู**ว่าต้องอ่านลำดับหลอดเป็น $a\,b\,c\,d\,e\,f\,g$

วิธีที่ใช้ ตัวถอดรหัส BCD-to-7-segment — **บทที่ 5 · วงจรผสม** · การย่อใช้ K-map จาก **บทที่ 4 · K-Map**

1

รู้จักหลอดทั้ง 7 ดวงก่อน

จอ 7 ส่วนคือหลอด LED 7 ดวงเรียงเป็นรูปเลข 8 · ชื่อของมันเริ่มที่ขีดบนสุดคือ a แล้ววนตามเข็มนาฬิกาเป็น b, c, d, e, f และขีดกลางคือ g



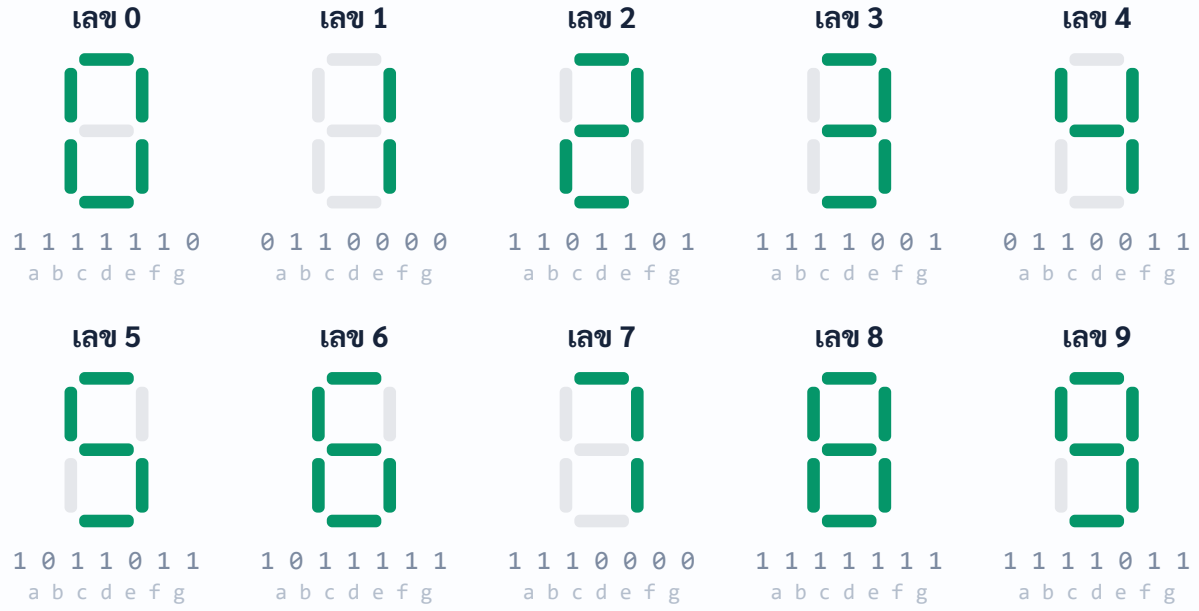
ชื่อของหลอดทั้ง 7 ดวง

ผังชื่อหลอดของจอ 7 ส่วน — เรียง a ถึง g ตามนี้เสมอ

2

วาดเลข 0–9 ออกมาดูว่าหลอดไหนติดบ้าง — นี่คือที่มาของทุกค่าในตาราง

ไม่ต้องท่องจำ · วาดรูปเลขแต่ละตัวบนจอ 7 ส่วนแล้วอ่านว่าหลอดไหนติด (1) หลอดไหนดับ (0) ตามลำดับ $a\,b\,c\,d\,e\,f\,g$



หน้าตาเลข 0–9 บนจอ 7 ส่วนตามแบบที่วิชานี้ใช้ · สีเขียว = หลอดติด (1) สีเทา = หลอดดับ (0)

จุดที่ต้องดูให้ดี

เลข **6** ในแบบของวิชานี้ **หลอด a ติด** และเลข **9 หลอด d ติด** · ไอซีบางรุ่นในโลกจริงเขียนเลข 6 โดยไม่ติด a และเลข 9 โดยไม่ติด d ซึ่งจะได้คำตอบคนละแบบ — ยึดตามรูปในสไลด์ของวิชานี้

ตารางค่าความจริงที่เติมครบแล้ว — แถวเขียวคือเลข BCD ที่ใช้จริง แถวเทาคือรหัสที่เป็นไปไม่ได้ (don’t care)

เลข	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

อ่านเป็นสัญกรณ์ $\sum m$ ได้ดังนี้ (ทุกหลอดมี $\sum d(10,11,12,13,14,15)$ เหมือนกันหมด)

หลอด	ช่องที่เป็น 1	ช่องที่เป็น 0
a	$\sum m(0,2,3,5,6,7,8,9)$	**1,4**
b	$\sum m(0,1,2,3,4,7,8,9)$	**5,6**
c	$\sum m(0,1,3,4,5,6,7,8,9)$	**2**
d	$\sum m(0,2,3,5,6,8,9)$	**1,4,7**
e	$\sum m(0,2,6,8)$	**1,3,4,5,7,9**
f	$\sum m(0,4,5,6,8,9)$	**1,2,3,7**
g	$\sum m(2,3,4,5,6,8,9)$	**0,1,7**

หลอด a — $a = \sum m(0,2,3,5,6,7,8,9) + \sum d(10,\ldots,15)$

ผัง K-map ของหลอดนี้ใช้หัวแถว D_3D_2 และหัวคอลัมน์ D_1D_0 · ช่อง 0 มีแค่สองช่องคือ m_1 (เลข 1) และ m_4 (เลข 4) เพราะเลข 1 กับเลข 4 เป็นสองตัวเดียวที่ไม่มีขีดบนหัว

1

จับกลุ่ม 8 ช่องก่อน (ใหญ่สุดที่เป็นไปได้)

ครึ่งล่างของผัง (แถว $D_3D_2 = 11$ กับ 10) คือช่อง m_8 ถึง m_{15} · m_8, m_9 เป็น 1 ส่วน m_{10}–m_{15} เป็น X → เลือกให้ X เป็น 1 ทั้งหมด → ได้กลุ่ม 8 ช่อง → D_3 คงที่ 1 → พจน์ D_3

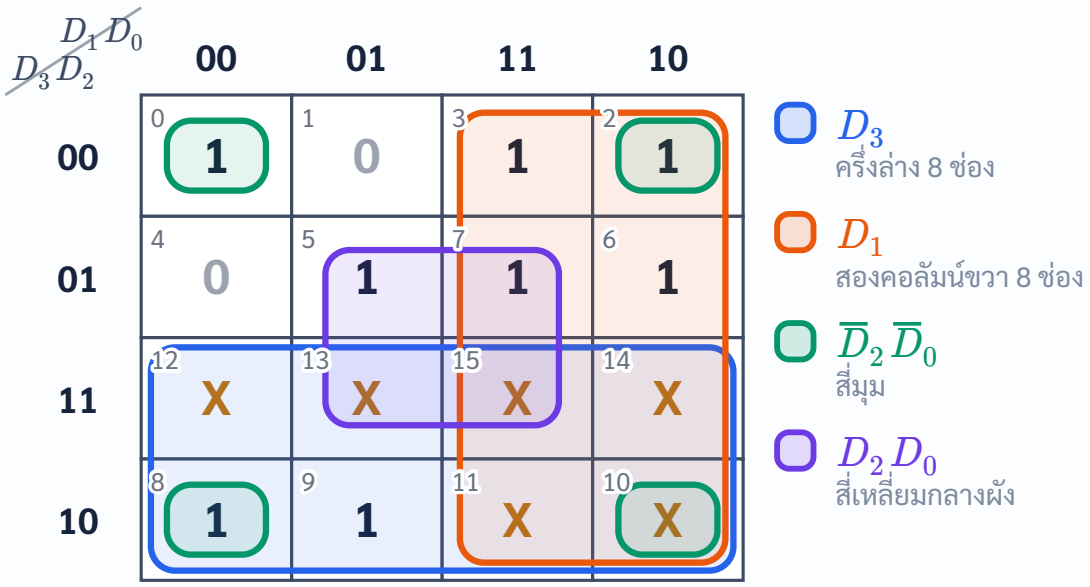
สองคอลัมน์ขวา ($D_1D_0 = 11$ กับ 10) คือช่องที่ $D_1 = 1$ ได้แก่ $m_2, m_3, m_6, m_7, m_{10}, m_{11}, m_{14}, m_{15}$ → สี่ตัวแรกเป็น 1 สี่ตัวหลังเป็น X → กลุ่ม 8 ช่อง → พจน์ D_1

2

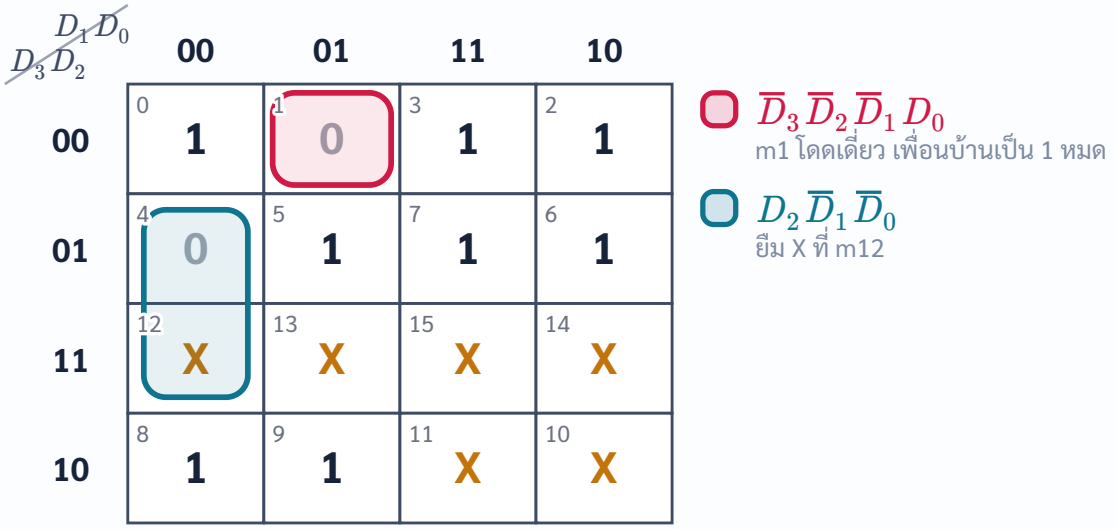
เหลือ m_0 กับ m_5 ที่ยังไม่ถูกคลุม

$m_0 = 0000$: จับสี่มุมของผัง m_0, m_2, m_8, m_{10} ได้ แต่จะดีกว่าถ้าดูว่า $D_2 = 0$ และ $D_0 = 0$ ในทั้งสี่มุม → พจน์ $\overline{D_2}\,\overline{D_0}$

$m_5 = 0101$: จับกลุ่ม m_5, m_7, m_{13}, m_{15} (m_{13}, m_{15} เป็น X) → D_2 คงที่ 1, D_0 คงที่ 1 → พจน์ D_2D_0



ผังหลอด a — SOP · X ทั้ง 6 ตัวถูกเอามาใช้เป็น 1 หมด ทำให้กลุ่มใหญ่ขึ้นมาก



ผังหลอด a ใบที่สอง — POS · จับกลุ่ม **0** ได้ $\overline{a} = \overline{D_3}\,\overline{D_2}\,\overline{D_1}D_0 + D_2\overline{D_1}\,\overline{D_0}$

3

POS — จับกลุ่ม 0

$m_1 = 0001$ เพื่อนบ้านคือ $m_0 = 1, m_3 = 1, m_5 = 1, m_9 = 1$ → ไม่มีเพื่อนที่เป็น 0 เลย → **ต้องอยู่เดี่ยว** → พจน์ของ $\overline{a}$ คือ $\overline{D_3}\,\overline{D_2}\,\overline{D_1}D_0$

$m_4 = 0100$ จับคู่กับ $m_{12} = 1100$ (เป็น X เลือกให้เป็น 0 ได้) → D_2 คงที่ 1, D_1 คงที่ 0, D_0 คงที่ 0 → $D_2\overline{D_1}\,\overline{D_0}$

$\overline{a} = \overline{D_3}\,\overline{D_2}\,\overline{D_1}D_0 + D_2\overline{D_1}\,\overline{D_0}$ SOP ของส่วนกลับ

$a = (D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)$ De Morgan สองชั้น

คำตอบหลอด a

$$\text{SOP: } \mathbf{a = D_3 + D_1 + \overline{D_2}\,\overline{D_0} + D_2D_0}$$

$$\text{POS: } \mathbf{a = (D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)}$$

เช็กคำตอบกลับเข้าโจทย์ — เช็กเฉพาะ 10 แถวที่เป็น BCD จริง (แถว X ไม่ต้องเช็ก)

เลข	$D_3D_2D_1D_0$	D_3	D_1	$\overline{D_2}\,\overline{D_0}$	D_2D_0	SOP	ต้องได้
0	0000	0	0	1	0	1	1 ✓
1	0001	0	0	0	0	0	0 ✓
2	0010	0	1	1	0	1	1 ✓
3	0011	0	1	0	0	1	1 ✓
4	0100	0	0	0	0	0	0 ✓
5	0101	0	0	0	1	1	1 ✓
6	0110	0	1	0	0	1	1 ✓
7	0111	0	1	0	1	1	1 ✓
8	1000	1	0	1	0	1	1 ✓
9	1001	1	0	0	0	1	1 ✓

จุดที่มักพลาดหลอด a

ไม่ใช้ X เลย จะได้ $a = D_3\overline{D_2}\,\overline{D_1} + D_1 + \overline{D_2}\,\overline{D_0} + D_2D_0$ ซึ่งยาวกว่าโดยไม่จำเป็น · รหัส 1010–1111 **เกิดขึ้นไม่ได้จริง ๆ** ในระบบ BCD จะให้มันออกอะไรก็ได้ ไม่มีใครเห็น

หลอด b — $b = \sum m(0,1,2,3,4,7,8,9) + \sum d(10,\ldots,15)$

หลอด b คือขีดขวาบน · ดับแค่ตอนแสดงเลข **5** กับ **6** เท่านั้น (เลขสองตัวนี้ขีดขวาบนหายไป) → ช่อง 0 คือ m_5 กับ m_6

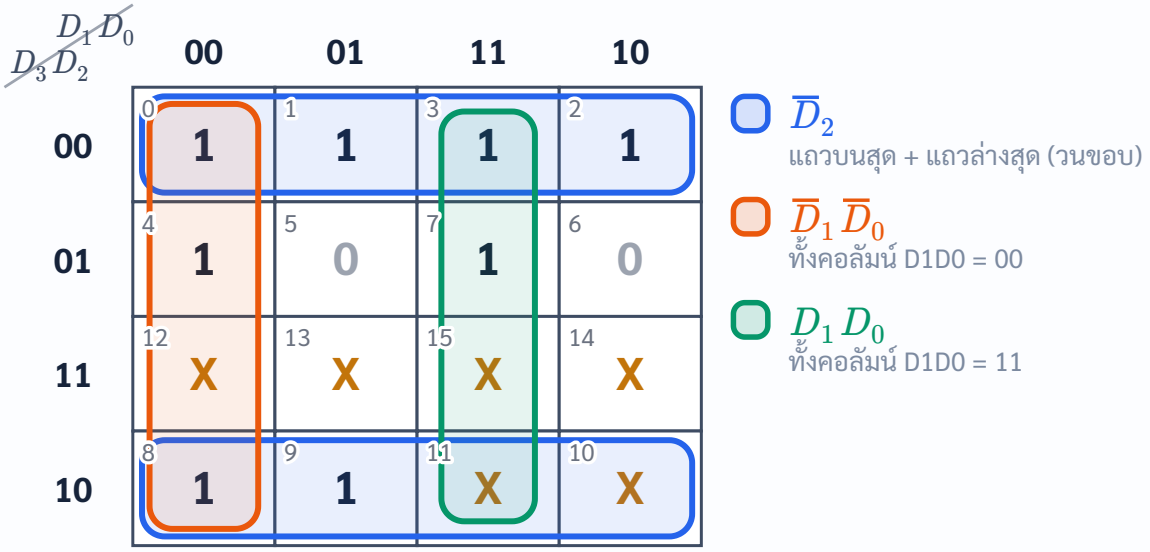
1 **จับกลุ่ม 8 ช่องก่อน**

แถว $D_3D_2 = 00$ กับ $D_3D_2 = 10$ คือแถวที่ $D_2 = 0$ (แถวบนสุดกับล่างสุด ติดกันแบบวนขอบ) · ช่องในนั้นคือ m_0, m_1, m_3, m_2 และ m_8, m_9, m_{11}, m_{10} → เป็น 1 หรือ X ทั้งหมด → กลุ่ม 8 → พจน์ $\overline{D_2}$

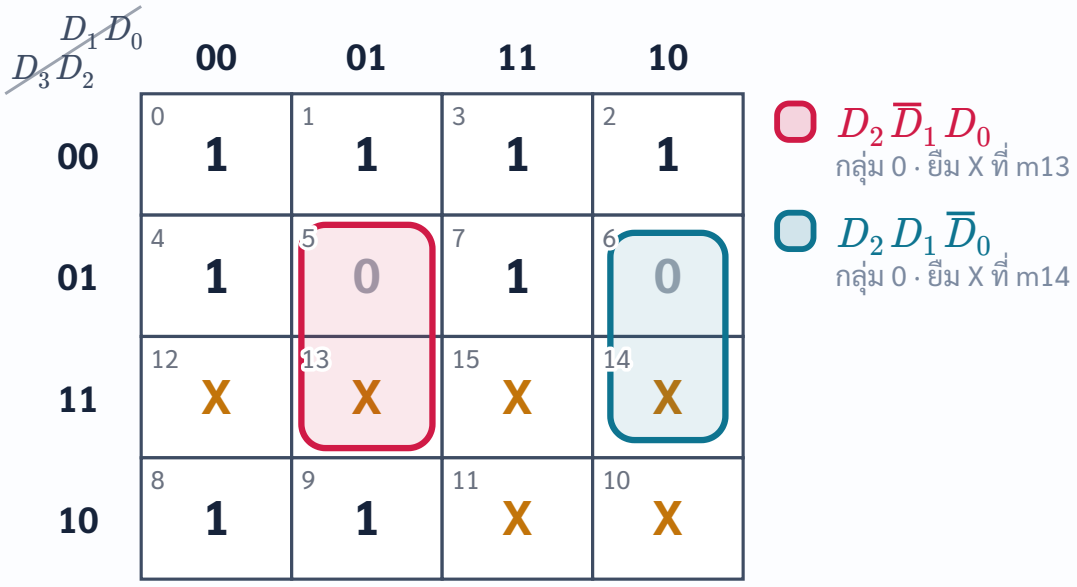
2 **เหลือ m_4 กับ m_7 ที่ยังไม่ถูกคลุม**

$m_4 = 0100$: จับทั้งคอลัมน์ $D_1D_0 = 00$ (m_0, m_4, m_{12}, m_8) → D_1 คงที่ 0, D_0 คงที่ 0 → พจน์ $\overline{D_1}\,\overline{D_0}$

$m_7 = 0111$: จับทั้งคอลัมน์ $D_1D_0 = 11$ (m_3, m_7, m_{15}, m_{11}) → D_1 คงที่ 1, D_0 คงที่ 1 → พจน์ D_1D_0



ผังหลอด b — SOP



ผังหลอด b ใบที่สอง — POS · ช่อง 0 มีแค่สองช่อง (m_5 กับ m_6) แต่ละช่องจับคู่กับ X ข้างล่างได้

3

POS — จับกลุ่ม 0 แล้ว De Morgan

$m_5(0101)$ จับกับ $m_{13}(1101)$ ซึ่งเป็น X → $D_2=1, D_1=0, D_0=1$ คงที่ → $D_2\overline{D_1}D_0$

$m_6(0110)$ จับกับ $m_{14}(1110)$ ซึ่งเป็น X → $D_2=1, D_1=1, D_0=0$ คงที่ → $D_2D_1\overline{D_0}$

$\overline{b} = D_2\overline{D_1}D_0 + D_2D_1\overline{D_0}$ SOP ของส่วนกลับ

$b = (\overline{D_2} + D_1 + \overline{D_0})(\overline{D_2} + \overline{D_1} + D_0)$ De Morgan สองชั้น

คำตอบหลอด b

$$\text{SOP: } \mathbf{b = \overline{D_2} + \overline{D_1}\,\overline{D_0} + D_1 D_0}$$

$$\text{POS: } \mathbf{b = (\overline{D_2} + D_1 + \overline{D_0})(\overline{D_2} + \overline{D_1} + D_0)}$$

เช็กคำตอบกลับเข้าโจทย์ — เช็ก 10 แถว BCD

เลข	$D_3D_2D_1D_0$	$\overline{D_2}$	$\overline{D_1}\,\overline{D_0}$	D_1D_0	SOP	ต้องได้
0	0000	1	1	0	1	1 ✓
1	0001	1	0	0	1	1 ✓
2	0010	1	0	0	1	1 ✓
3	0011	1	0	1	1	1 ✓
4	0100	0	1	0	1	1 ✓
5	0101	0	0	0	0	0 ✓
6	0110	0	0	0	0	0 ✓
7	0111	0	0	1	1	1 ✓
8	1000	1	1	0	1	1 ✓
9	1001	1	0	0	1	1 ✓

จุดที่มักพลาดหลอด b

ลืมว่าแถว $D_3D_2 = 00$ กับ $D_3D_2 = 10$ **ติดกัน** (เพราะเรียงเกรย์โค้ด 00 · 01 · 11 · 10 แล้ววน) เลยจับ $\overline{D_2}$ ไม่ออก ได้คำตอบยาวเป็น 4–5 พจน์

หลอด c — $c = \sum m(0,1,3,4,5,6,7,8,9) + \sum d(10,\ldots,15)$

หลอด c คือขีดขวาล่าง · ดับแค่ตอนแสดงเลข **2** ตัวเดียว → ช่อง 0 มีช่องเดียวคือ m_2 · ข้อนี้จึงย่อได้สั้นที่สุดในบรรดาหลอดทั้งหมด

1

เมื่อมี 0 ช่องเดียว ให้ถามว่า “ต้องปิดกั้นเฉพาะช่องนั้น” ด้วยกลุ่มอะไรบ้าง

$m_2 = 0010$ คือ $D_3 = 0, D_2 = 0, D_1 = 1, D_0 = 0$ · กลุ่ม 8 ช่องใด ๆ ที่**ไม่มี** m_2 อยู่ ใช้ได้หมด

กลุ่ม 8 ช่องที่ $D_2 = 1$ (สองแถวกลาง) → ไม่มี m_2 → พจน์ D_2

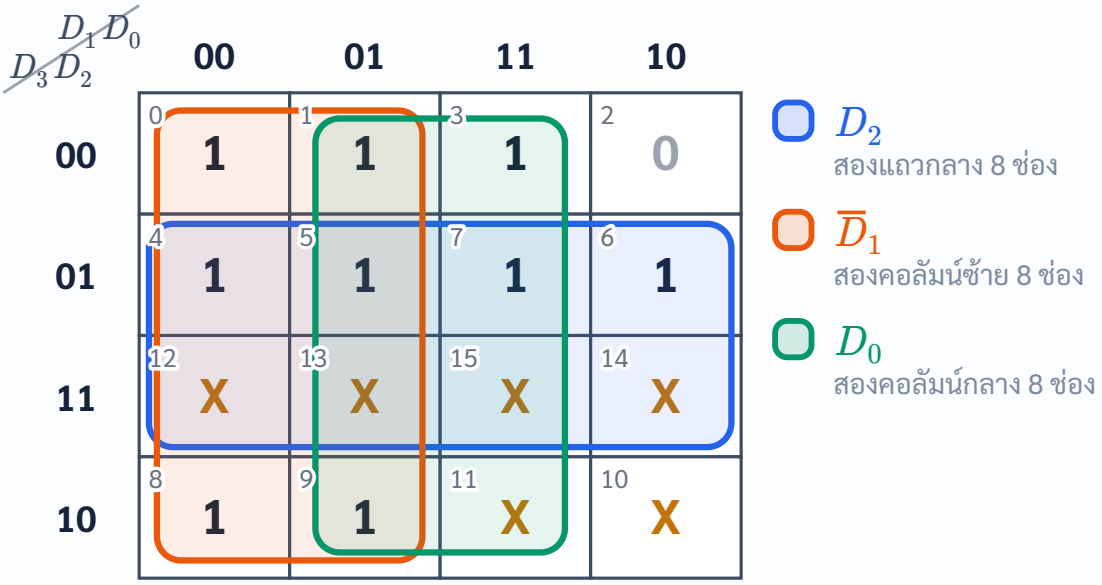
กลุ่ม 8 ช่องที่ $D_1 = 0$ (สองคอลัมน์ซ้าย) → ไม่มี m_2 → พจน์ $\overline{D_1}$

กลุ่ม 8 ช่องที่ $D_0 = 1$ (สองคอลัมน์กลาง) → ไม่มี m_2 → พจน์ D_0

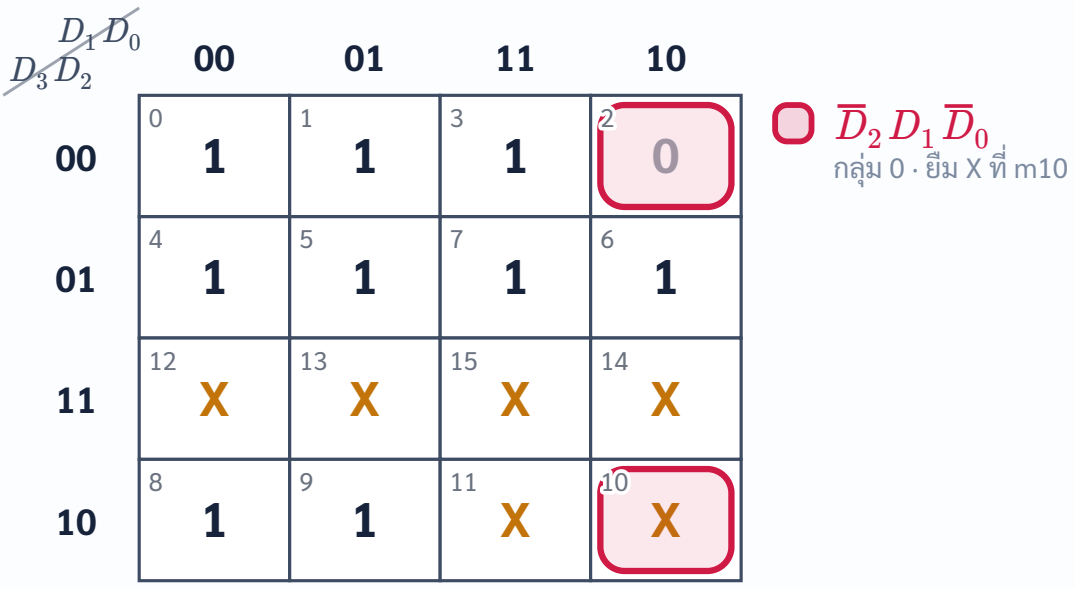
2 **เช็กว่าสามกลุ่มนี้คลุมช่อง 1 ครบไหม**

ช่อง 1 ที่ต้องคลุมคือ $m_0, m_1, m_3, m_4, m_5, m_6, m_7, m_8, m_9$

D_2 คลุม 4, 5, 6, 7 · $\overline{D_1}$ คลุม 0, 1, 4, 5, 8, 9 · D_0 คลุม 1, 3, 5, 7, 9 → รวมแล้วครบทั้ง 9 ช่อง



ผังหลอด c — SOP · ทั้งสามกลุ่มเป็นกลุ่ม 8 ช่อง จึงเหลือพจน์ละ 1 literal



ผังหลอด c ใบที่สอง — POS · ช่อง 0 มีช่องเดียว จับคู่กับ X ที่ m_{10} ได้กลุ่ม 2 ช่อง

3

POS — จับกลุ่ม 0 แล้ว De Morgan

$m_2(0010)$ จับกับ $m_{10}(1010)$ ซึ่งเป็น X → D_3 เปลี่ยน ตัดทิ้ง · D_2 คงที่ 0 · D_1 คงที่ 1 · D_0 คงที่ 0 → $\bar{c} = \overline{D_2}D_1\overline{D_0}$

$c = \overline{\overline{D_2}D_1\overline{D_0}} = D_2 + \overline{D_1} + D_0$ De Morgan ชั้นเดียวจบ เพราะ $\bar{c}$ มีพจน์เดียว

สังเกตว่า POS ที่ได้**หน้าตาเหมือน SOP เป๊ะ** — เพราะนิพจน์นี้เป็นผลบวกของ literal เดี่ยว ๆ ซึ่งนับเป็นทั้ง SOP (พจน์คูณสามพจน์ พจน์ละตัวเดียว) และ POS (พจน์บวกพจน์เดียว)

คำตอบหลอด c

$$\text{SOP: } \mathbf{c = D_2 + \overline{D_1} + D_0}$$

$$\text{POS: } \mathbf{c = (D_2 + \overline{D_1} + D_0)}$$

เช็กคำตอบกลับเข้าโจทย์ — ดูว่ามีแถวเดียวเท่านั้นที่ทั้งสามพจน์เป็น 0 พร้อมกัน

เลข	$D_3D_2D_1D_0$	D_2	$\overline{D_1}$	D_0	SOP	ต้องได้
0	0000	0	1	0	1	1 ✓
1	0001	0	1	1	1	1 ✓
2	0010	0	0	0	0	0 ✓
3	0011	0	0	1	1	1 ✓
4	0100	1	1	0	1	1 ✓
5	0101	1	1	1	1	1 ✓
6	0110	1	0	0	1	1 ✓
7	0111	1	0	1	1	1 ✓
8	1000	0	1	0	1	1 ✓
9	1001	0	1	1	1	1 ✓

จุดที่มักพลาดหลอด c

เห็นช่อง 1 เต็มผังแล้วรีบตอบ $c = 1$ · ต้องดูให้ดีว่ามี 0 ซ่อนอยู่ที่ m_2 หนึ่งช่อง · **นับช่อง 0 ให้ครบก่อนเสมอ** เพราะช่อง 0 ช่องเดียวก็เปลี่ยนคำตอบทั้งข้อ

หลอด d — $d = \sum m(0,2,3,5,6,8,9) + \sum d(10,\ldots,15)$

หลอด d คือขีดล่างสุด · ดับตอนแสดงเลข **1, 4** และ **7** → ช่อง 0 คือ m_1, m_4, m_7 · ข้อนี้ยาวที่สุดในสี่หลอด เพราะช่อง 0 กระจายตัวจนจับกลุ่ม 1 ให้ใหญ่ได้ยาก

1

กลุ่ม 8 ช่องมีอันเดียว

ครึ่งล่างของผัง ($D_3 = 1$) คือ m_8, m_9 (เป็น 1) และ m_{10} ถึง m_{15} (เป็น X) → ไม่มี 0 ปนเลย → กลุ่ม 8 → พจน์ D_3

กลุ่ม 8 อื่น ๆ ใช้ไม่ได้ เพราะต้องมี m_1 หรือ m_4 หรือ m_7 ปนเข้ามาเสมอ

2

เหลือช่อง 1 ครึ่งบนที่ต้องคลุม คือ m_0, m_2, m_3, m_5, m_6

$m_0(0000)$: จับสี่มุม m_0, m_2, m_8, m_{10} → D_2 คงที่ 0, D_0 คงที่ 0 → พจน์ $\overline{D_2}\,\overline{D_0}$

$m_3(0011)$: จับ m_2, m_3, m_{10}, m_{11} → D_2 คงที่ 0, D_1 คงที่ 1 → พจน์ $\overline{D_2}D_1$

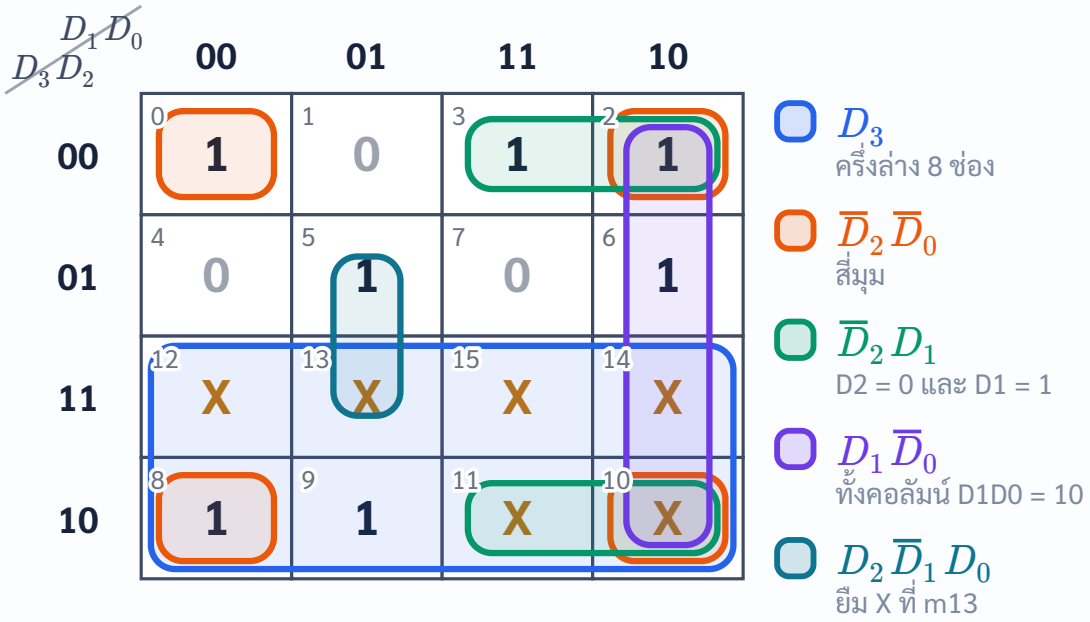
$m_6(0110)$: จับทั้งคอลัมน์ $D_1D_0 = 10$ คือ m_2, m_6, m_{14}, m_{10} → D_1 คงที่ 1, D_0 คงที่ 0 → พจน์ $D_1\overline{D_0}$

$m_5(0101)$: เพื่อนบ้านสามด้านเป็น 0 หมด (m_4, m_7, m_1) เหลือทางเดียวคือจับคู่กับ m_{13} ซึ่งเป็น X → D_2 คงที่ 1, D_1 คงที่ 0, D_0 คงที่ 1 → พจน์ $D_2\overline{D_1}D_0$ (3 literal เลี่ยงไม่ได้)

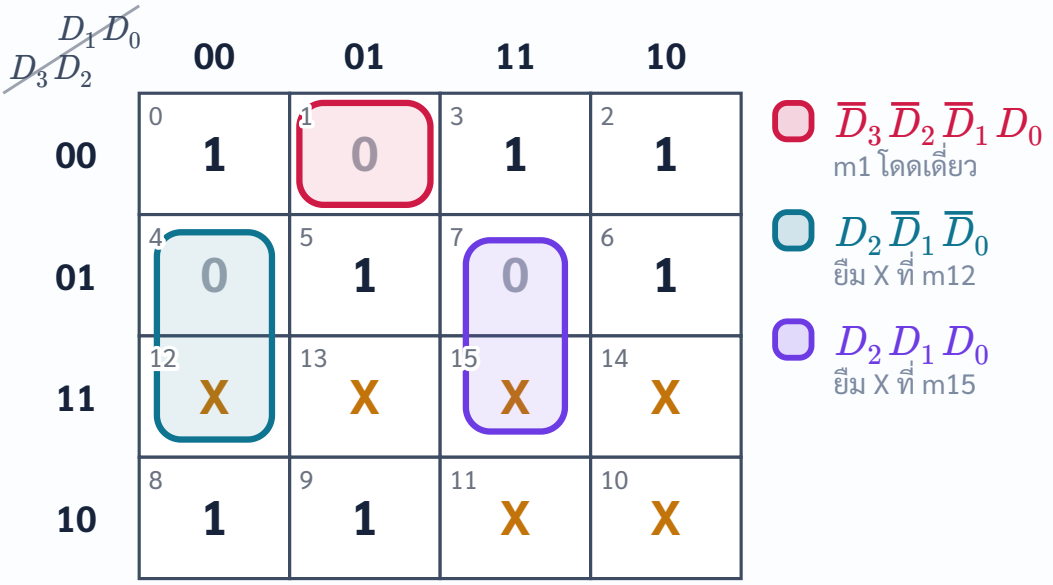
3

รวมแล้วได้ 5 พจน์ เช็กว่าคลุมครบ

D_3 คลุม 8, 9 · $\overline{D_2}\,\overline{D_0}$ คลุม 0, 2 · $\overline{D_2}D_1$ คลุม 2, 3 · $D_1\overline{D_0}$ คลุม 2, 6 · $D_2\overline{D_1}D_0$ คลุม 5 → รวมได้ 0, 2, 3, 5, 6, 8, 9 ครบพอดี



ผังหลอด d — SOP · ต้องใช้ถึง 5 กลุ่ม เพราะ 0 สามช่องวางขวางกลุ่มใหญ่ไว้หมด



ผังหลอด d ใบที่สอง — POS · จับกลุ่ม 0 ได้ 3 กลุ่ม

4

POS — จับกลุ่ม 0

$m_1(0001)$: เพื่อนบ้านคือ $m_0=1, m_3=1, m_5=1, m_9=1$ → ไม่มีเพื่อนที่เป็น 0 → อยู่เดี่ยว → $\overline{D_3}\,\overline{D_2}\,\overline{D_1}D_0$

$m_4(0100)$: จับกับ m_{12} ซึ่งเป็น X → D_2 คงที่ 1, D_1 คงที่ 0, D_0 คงที่ 0 → $D_2\overline{D_1}\,\overline{D_0}$

$m_7(0111)$: จับกับ m_{15} ซึ่งเป็น X → D_2 คงที่ 1, D_1 คงที่ 1, D_0 คงที่ 1 → $D_2D_1D_0$

$\overline{d} = \overline{D_3}\,\overline{D_2}\,\overline{D_1}D_0 + D_2\overline{D_1}\,\overline{D_0} + D_2D_1D_0$ SOP ของส่วนกลับ

$d = (D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)(\overline{D_2} + \overline{D_1} + \overline{D_0})$ De Morgan สองชั้น

คำตอบหลอด d

SOP: $\mathbf{d = D_3 + \overline{D_2}D_1 + \overline{D_2}\,\overline{D_0} + D_2\overline{D_1}D_0 + D_1\overline{D_0}}$

POS: $\mathbf{d = (D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)(\overline{D_2} + \overline{D_1} + \overline{D_0})}$

เช็กคำตอบกลับเข้าโจทย์ — ไล่ทั้ง 10 แถว BCD

เลข	$D_3D_2D_1D_0$	D_3	$\overline{D_2}D_1$	$\overline{D_2}\,\overline{D_0}$	$D_2\overline{D_1}D_0$	$D_1\overline{D_0}$	SOP	ต้องได้
0	0000	0	0	1	0	0	1	1 ✓
1	0001	0	0	0	0	0	0	0 ✓
2	0010	0	1	1	0	1	1	1 ✓
3	0011	0	1	0	0	0	1	1 ✓
4	0100	0	0	0	0	0	0	0 ✓
5	0101	0	0	0	1	0	1	1 ✓
6	0110	0	0	0	0	1	1	1 ✓
7	0111	0	0	0	0	0	0	0 ✓
8	1000	1	0	1	0	0	1	1 ✓
9	1001	1	0	0	0	0	1	1 ✓

จุดที่มักพลาดหลอด d

เผลอจับ m_5 รวมกับ m_7 เพราะสองช่องนี้ติดกันบนผัง — แต่ m_7 เป็น **0** (เลข 7 ไม่มีขีดล่าง) · กลุ่มที่มี 0 ปนอยู่แม้ช่องเดียวก็ใช้ไม่ได้ · ก่อนตีกรอบ ให้ไล่อ่านค่าทุกช่องในกรอบก่อนเสมอ

สรุปคำตอบข้อ 3 — เติมลงตาราง Boolean Expression ท้ายโจทย์

LED	รูป	Boolean Expression
a	SOP	$D_3 + D_1 + \overline{D_2}\,\overline{D_0} + D_2D_0$
	POS	$(D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)$
b	SOP	$\overline{D_2} + \overline{D_1}\,\overline{D_0} + D_1D_0$
	POS	$(\overline{D_2} + D_1 + \overline{D_0})(\overline{D_2} + \overline{D_1} + D_0)$
c	SOP	$D_2 + \overline{D_1} + D_0$
	POS	$(D_2 + \overline{D_1} + D_0)$
d	SOP	$D_3 + \overline{D_2}D_1 + \overline{D_2}\,\overline{D_0} + D_2\overline{D_1}D_0 + D_1\overline{D_0}$
	POS	$(D_3 + D_2 + D_1 + \overline{D_0})(\overline{D_2} + D_1 + D_0)(\overline{D_2} + \overline{D_1} + \overline{D_0})$

หมายเหตุเรื่องหลอด e, f, g

โจทย์สั่งให้หานิพจน์เฉพาะหลอด a ถึง d เท่านั้น แต่**ตารางค่าความจริงต้องเติมครบทั้ง 7 หลอด** ซึ่งเติมไว้ให้แล้วด้านบน · ถ้าอาจารย์ถามเพิ่มในห้องสอบ ใช้วิธีเดียวกันเป๊ะ คือพล็อต $\sum m$ ของหลอดนั้นลงผังแล้วจับกลุ่ม

4 อ่านวงจรแล้วสร้างตารางค่าความจริงเอง

โจทย์ข้อ 4 (ต้นฉบับ) Assignment #6

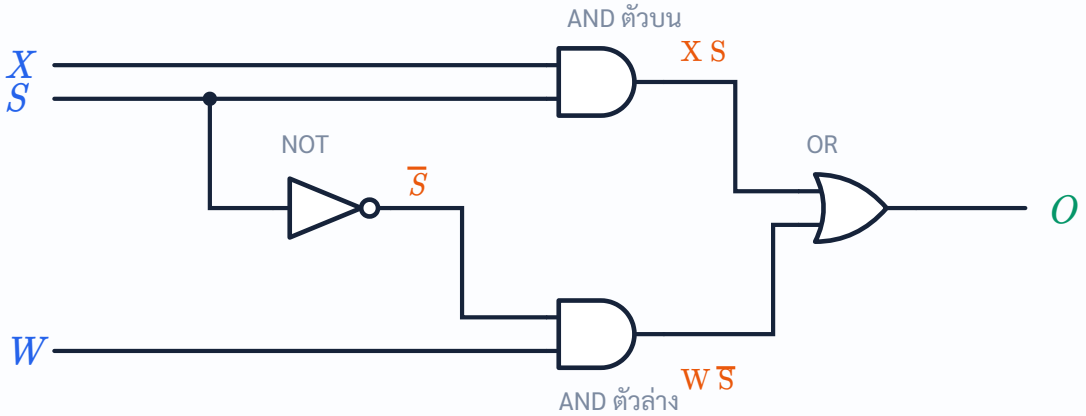
> 4. Derive the truth tables for the following circuits.
> a. 2-to-1 MUX b. 1-to-2 DEMUX

แปลไทย: โจทย์วาดวงจรมาให้ 2 วง พร้อมตารางเปล่า ให้เรา**ไล่สัญญาณจากซ้ายไปขวา** แล้วเติมคอลัมน์เอาต์พุตให้ครบทุกแถว

โจทย์ซ่อนอะไรไว้: ตารางของข้อ (a) เรียงคอลัมน์เป็น S, W, X ไม่ใช่ X, W, S ตามที่วาดในรูป — ถ้าเติมค่าตามลำดับในรูปจะสลับคอลัมน์กันหมด · และข้อ (b) มีเอาต์พุต **สองตัว** (O_1 กับ O_0) ต้องเติมสองคอลัมน์

วิธีที่ใช้ วิเคราะห์วงจรผสม (ไล่นิพจน์ทีละเกต) + มัลติเพล็กเซอร์/ดีมัลติเพล็กเซอร์ — **บทที่ 5 · วงจรผสม**

ข้อ 4(a) — 2-to-1 MUX



วงจร 2-to-1 MUX ของโจทย์ วาดใหม่ด้วยสัญลักษณ์เกตมาตรฐาน · จุดทึบคือจุดที่สาย S แยกออกเป็นสองทาง

1 **ไล่สัญญาณทีละเกต จากซ้ายไปขวา**

เกต NOT: รับ S → ออก $\overline{S}$

AND ตัวบน: รับ X กับ S → ออก $X \cdot S$

AND ตัวล่าง: รับ W กับ $\overline{S}$ → ออก $W \cdot \overline{S}$

OR: รับผลจาก AND ทั้งสองตัว → ออก $O = XS + W\overline{S}$

2 **อ่านความหมายของนิพจน์ก่อนเติมตาราง**

ถ้า $S = 0 \rightarrow XS = 0$ เสมอ และ $W\overline{S} = W \cdot 1 = W \rightarrow O = W$

ถ้า $S = 1 \rightarrow XS = X$ และ $W\overline{S} = W \cdot 0 = 0 \rightarrow O = X$

นี่คือหน้าที่ของ MUX พอดี: S เป็น**สวิตช์เลือกทาง** ว่าจะให้ W หรือ X ผ่านออกไป

3 **เติมตามลำดับคอลัมน์ที่โจทย์ให้ คือ S, W, X**

แทนค่าทีละแถวลงใน $O = XS + W\overline{S}$ ไม่ต้องใช้ทางลัด จะได้ไม่พลาด

S	W	X	$X \cdot S$	$W \cdot \overline{S}$	O	คิดยังไง
0	0	0	$0 \cdot 0 = 0$	$0 \cdot 1 = 0$	**0**	$S = 0 \rightarrow$ ส่ง $W = 0$ ออก
0	0	1	$1 \cdot 0 = 0$	$0 \cdot 1 = 0$	**0**	$S = 0 \rightarrow$ ไม่สน X ส่ง $W = 0$
0	1	0	$0 \cdot 0 = 0$	$1 \cdot 1 = 1$	**1**	$S = 0 \rightarrow$ ส่ง $W = 1$ ออก
0	1	1	$1 \cdot 0 = 0$	$1 \cdot 1 = 1$	**1**	$S = 0 \rightarrow$ ส่ง $W = 1$ ออก
1	0	0	$0 \cdot 1 = 0$	$0 \cdot 0 = 0$	**0**	$S = 1 \rightarrow$ ส่ง $X = 0$ ออก
1	0	1	$1 \cdot 1 = 1$	$0 \cdot 0 = 0$	**1**	$S = 1 \rightarrow$ ส่ง $X = 1$ ออก
1	1	0	$0 \cdot 1 = 0$	$1 \cdot 0 = 0$	**0**	$S = 1 \rightarrow$ ไม่สน W ส่ง $X = 0$
1	1	1	$1 \cdot 1 = 1$	$1 \cdot 0 = 0$	**1**	$S = 1 \rightarrow$ ส่ง $X = 1$ ออก

คำตอบข้อ 4(a)

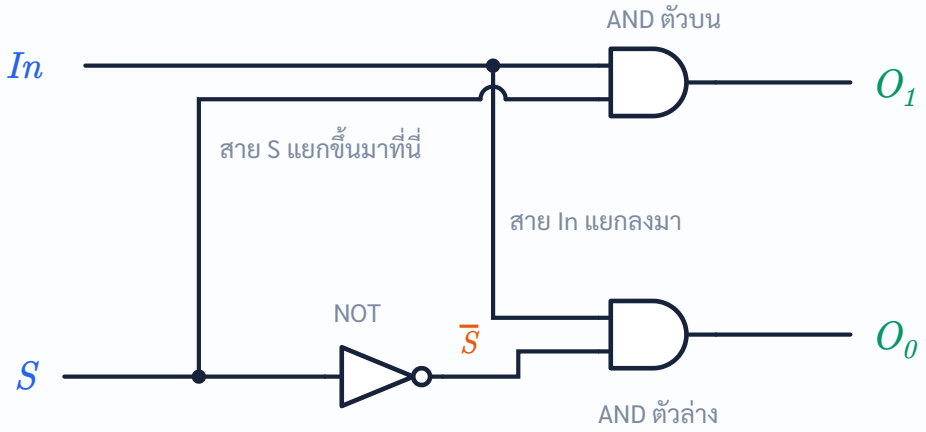
$\mathbf{O = XS + W\overline{S}}$ · คอลัมน์ O เรียงจากบนลงล่างคือ **0, 0, 1, 1, 0, 1, 0, 1**

เช็กคำตอบกลับเข้าโจทย์ — เช็กด้วยความหมายของวงจร: สี่แถวบน ($S = 0$) คอลัมน์ O ต้องเท่ากับคอลัมน์ W เป๊ะ → $0, 0, 1, 1$ ตรงกับ $W = 0, 0, 1, 1$ ✓ · สี่แถวล่าง ($S = 1$) คอลัมน์ O ต้องเท่ากับคอลัมน์ X เป๊ะ → $0, 1, 0, 1$ ตรงกับ $X = 0, 1, 0, 1$ ✓

จุดที่มักพลาดข้อนี้

ดูรูปแล้วนึกว่าเกต NOT ต่อกับ X (เพราะเส้นแยกลงมาอยู่ใต้ X) · จริง ๆ จุดแยกอยู่บน**สาย** S · วิธีกันพลาดคือ**ไล่นิ้วตามเส้นจากป้ายชื่อไปจนถึงขาเกต** อย่าเดาจากตำแหน่งบนลงล่าง

ข้อ 4(b) — 1-to-2 DEMUX



วงจร 1-to-2 DEMUX ของโจทย์ · ส่วนโค้งเล็กตรงที่สองเส้นตัดกันแปลว่า **ข้ามกัน ไม่ได้ต่อถึงกัน** — จุดที่ต่อกันจริงจะมีจุดทึบ

1

ไล่สายทีละเส้นก่อน ว่าอะไรเข้าขาไหน

สาย In วิ่งตรงเข้าขาบนของ AND ตัวบน และแยกลงมาเข้าขาบนของ AND ตัวล่าง

สาย S แยกเป็นสองทาง: ทางหนึ่งวิ่งขึ้นไปเข้าขาล่างของ AND ตัวบน · อีกทางเข้า NOT แล้วออกมาเป็น $\overline{S}$ เข้าขาล่างของ AND ตัวล่าง

2

เขียนนิพจน์ของเอาต์พุตทั้งสอง

AND ตัวบน: $O_1 = In \cdot S$

AND ตัวล่าง: $O_0 = In \cdot \overline{S}$

3

อ่านความหมายก่อนเติมตาราง

S เป็นตัวเลือกว่าจะ**ส่งข้อมูลไปออกทางไหน** · ถ้า $S = 0$ ข้อมูลไปออกที่ O_0 (และ O_1 เป็น 0 เสมอ) · ถ้า $S = 1$ ข้อมูลไปออกที่ O_1

นี่ตรงข้ามกับ MUX ในข้อ (a): MUX คือหลายทางเข้า → ทางออกเดียว · DEMUX คือทางเข้าเดียว → หลายทางออก

S	In	$O_1 = In \cdot S$	$O_0 = In \cdot \overline{S}$	คิดยังไง
0	0	$0 \cdot 0 = 0$	$0 \cdot 1 = 0$	ไม่มีข้อมูลเข้า ออกทั้งสองทางเป็น 0
0	1	$1 \cdot 0 = 0$	$1 \cdot 1 = 1$	$S = 0 \rightarrow$ ข้อมูลไปออกทาง O_0
1	0	$0 \cdot 1 = 0$	$0 \cdot 0 = 0$	ไม่มีข้อมูลเข้า ออกทั้งสองทางเป็น 0
1	1	$1 \cdot 1 = 1$	$1 \cdot 0 = 0$	$S = 1 \rightarrow$ ข้อมูลไปออกทาง O_1

คำตอบข้อ 4(b)

$$\mathbf{O_1 = In \cdot S} \quad \mathbf{O_0 = In \cdot \overline{S}}$$

คอลัมน์ O_1 จากบนลงล่าง = **0, 0, 0, 1** · คอลัมน์ O_0 = **0, 1, 0, 0**

เช็กคำตอบกลับเข้าโจทย์ — วงจรนี้ต้องมีสมบัติว่า O_1 **กับ** O_0 **เป็น 1 พร้อมกันไม่ได้** (เพราะข้อมูลไปได้ทางเดียว) · ดูตาราง: แถวที่ $O_1 = 1$ คือแถวสุดท้าย ซึ่ง $O_0 = 0$ · แถวที่ $O_0 = 1$ คือแถวที่สอง ซึ่ง $O_1 = 0$ ✓ · และ $O_1 + O_0$ ต้องเท่ากับ In เสมอ: $0, 1, 0, 1$ ตรงกับคอลัมน์ In ✓

จุดที่มักพลาดข้อนี้

เห็นเส้นสองเส้นตัดกันกลางรูปแล้วนึกว่าต่อถึงกัน เลยคิดว่า $\overline{S}$ ไปเข้า AND ตัวบนด้วย · **เส้นตัดกันเฉย ๆ ไม่ได้ต่อกัน · ต่อกันเมื่อมีจุดทึบเท่านั้น**

สรุป Assignment 6 หนึ่งบรรทัด

วงจรผสมทุกตัวในชุดนี้อ่านได้ด้วยวิธีเดียวกัน: ไล่สัญญาณจากซ้ายไปขวาทีละเกต เขียนนิพจน์ที่ขาออกของทุกเกต แล้วแทนค่าอินพุตทุกชุดลงไป — ส่วนตัวถอดรหัส 7 ส่วนเพิ่มอีกขั้นเดียวคือเอาตารางที่ได้ไปย่อด้วย K-map

หน้าสรุปสูตรและขั้นตอน

ทั้งขอบเขตสอบ อยู่ในไม่กี่หน้านี้

วิธีใช้หน้านี้

หน้านี้**ไม่ได้สอน**ครับ เป็นที่ทวนของที่เรียนไปแล้วเท่านั้น ถ้าอ่านบรรทัดไหนแล้วนึกภาพไม่ออกว่ามันมาจากไหน **แปลว่ายังไม่ควรอ่านหน้านี้** ให้กลับไปอ่านบทนั้นก่อน ส่วนถ้าเหลือเวลา 15 นาทีสุดท้ายก่อนเข้าห้องสอบ ให้กวาดตาหน้านี้อย่างเดียวพอ

1 ระบบเลขฐานและการแปลงฐาน

จะทำอะไร	ทำยังไง	อ่านคำตอบทางไหน
ฐาน r → ฐานสิบ	เอาเลขแต่ละหลักคูณน้ำหนักของหลักนั้นแล้วบวกกัน · น้ำหนัก $= r^{k}$ โดย k นับจากจุดออกไป	หน้าจุด $k = 0, 1, 2, \ldots$ · หลังจุด $k = -1, -2, \ldots$
ฐานสิบ → ฐาน r (ส่วนหน้าจุด)	**หารสั้นด้วย r ซ้ำ ๆ** เก็บเศษไว้ทุกรอบ จนผลหารเป็น 0	**อ่านเศษจากล่างขึ้นบน**
ฐานสิบ → ฐาน r (ส่วนหลังจุด)	**คูณด้วย r ซ้ำ ๆ** เก็บเลขหน้าจุดที่งอกออกมาทุกรอบ แล้วทิ้งไว้แต่เศษ	**อ่านจากบนลงล่าง** — คนละทิศกับส่วนหน้าจุด
ฐานสอง ↔ ฐานแปด	จับกลุ่มละ **3 บิต** เริ่มจากจุดออกไปทั้งสองทาง เติม 0 ให้ครบกลุ่ม	กลุ่มละ 1 หลักฐานแปด · **ไม่ต้องแวะฐานสิบ**
ฐานสอง ↔ ฐานสิบหก	จับกลุ่มละ **4 บิต** วิธีเดียวกัน	กลุ่มละ 1 หลักฐานสิบหก
ฐานแปด ↔ ฐานสิบหก	**ต้องแวะฐานสองเสมอ** — 3 บิตกับ 4 บิตหารกันไม่ลงตัว	แตกเป็นบิตก่อน แล้วค่อยจับกลุ่มใหม่

ฐานสิบหก `A=10 B=11 C=12 D=13 E=14 F=15`

ตรวจตัวเอง เลขฐาน r ห้ามมีหลักที่ $\geq r$ · เห็นเลข 8 ในฐานแปด = ผิดแน่

ตรวจคำตอบ แปลงกลับเป็นฐานสิบ ต้องได้เลขเดิม · บังคับทำทุกข้อ

2 คอมพลีเมนต์ เลขมีเครื่องหมาย และ overflow

ชื่อ	ทำยังไง	หมายเหตุที่ข้อสอบชอบถาม
1's complement	กลับทุกบิต 0↔1	ของ n บิต: $1\text{'s}(X) = (2^n - 1) - X$
2's complement	กลับทุกบิตแล้ว **+1** · หรือวิธีลัด: คัดลอกจากขวามาจนถึงเลข 1 ตัวแรก**รวมตัวนั้นด้วย** แล้วกลับบิตที่เหลือทั้งหมด	ของ n บิต: $2\text{'s}(X) = 2^n - X$ · ทำ 2's ซ้ำสองครั้งได้เลขเดิม
Sign-magnitude	บิตซ้ายสุด = เครื่องหมาย (0 บวก 1 ลบ) · บิตที่เหลืออ่านเป็นขนาดตรง ๆ	มีศูนย์สองตัว (+0 และ −0)
1's complement form	เลขบวกเขียนปกติ · เลขลบ = กลับทุกบิตของเลขบวกนั้น	มีศูนย์สองตัวเหมือนกัน · ต้องบวกตัวทดวนกลับ (end-around carry)
2's complement form	เลขบวกเขียนปกติ · เลขลบ = 2's complement ของเลขบวกนั้น	**มีศูนย์ตัวเดียว** · เป็นแบบที่คอมพิวเตอร์จริงใช้

พิสัยของเลข n บิต	สูตร	ตัวอย่าง $n = 8$
Sign-magnitude	$-(2^{n-1} - 1)$ ถึง $+(2^{n-1} - 1)$	−127 ถึง +127
1's complement	$-(2^{n-1} - 1)$ ถึง $+(2^{n-1} - 1)$	−127 ถึง +127
2's complement	-2^{n-1} ถึง $+(2^{n-1} - 1)$	**−128 ถึง +127** (ฝั่งลบเกินมา 1)
ไม่มีเครื่องหมาย	0 ถึง $2^n - 1$	0 ถึง 255

วิธีเช็ก overflow ของการบวกแบบ 2's complement

กฎที่ใช้ได้เสมอ: เกิด overflow **ก็ต่อเมื่อ** ตัวตั้งกับตัวบวก**เครื่องหมายเดียวกัน** แต่ผลลัพธ์**เครื่องหมายตรงข้าม** · ถ้าเครื่องหมายของตัวตั้งกับตัวบวกต่างกันอยู่แล้ว **ไม่มีทาง overflow**

กฎที่เทียบเท่ากัน: ตัวทดที่**เข้า**หลักเครื่องหมาย ไม่เท่ากับ ตัวทดที่**ออก**จากหลักเครื่องหมาย · **ตัวทดที่ล้นออกมาเฉย ๆ ไม่ใช่ overflow** — คนละเรื่องกัน อย่าสับสน

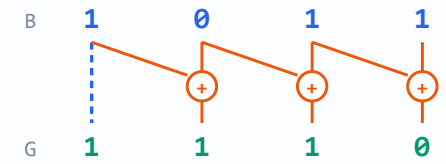
3 รหัส: BCD · Gray · parity · CRC · Hamming

BCD — เก็บเลขฐานสิบทีละหลัก

วิธีเข้ารหัส	หลักฐานสิบ 1 หลัก → 4 บิต · เช่น $57_{10} = \underset{5}{0101}\ \underset{7}{0111}$ **ไม่ใช่** 111001_2
รหัสที่ใช้ไม่ได้	**1010 · 1011 · 1100 · 1101 · 1110 · 1111** (คือ 10–15) รวม 6 รหัส — เห็นในโจทย์เมื่อไรแปลว่าไม่ใช่ BCD
บวกแบบ BCD	บวกทีละ 4 บิตก่อน · ถ้ากลุ่มไหน**เกิน 9 หรือมีตัวทดออก** ให้ **บวก 0110 (6) เพิ่ม**ที่กลุ่มนั้น แล้วทดไปกลุ่มถัดไป

Gray code — เลขติดกันต่างกันบิตเดียว

ไบนารี → เกรย์	บิตซ้ายสุด**คัดลอกมาเลย** · บิตที่เหลือ = **XOR ของบิตคู่ที่ติดกันในไบนารี** $G_{n-1} = B_{n-1} \cdot G_i = B_{i+1} \oplus B_i$
เกรย์ → ไบนารี	บิตซ้ายสุด**คัดลอกมาเลย** · บิตถัดไป = **บิตไบนารีที่เพิ่งได้ XOR กับบิตเกรย์ตัวนั้น** $B_{n-1} = G_{n-1} \cdot B_i = B_{i+1} \oplus G_i$



B = 1011 (ไบนารี) ให้ G = 1110 (เกรย์) · ทำย้อนกลับได้เลขเดิม

Parity — รู้ว่าผิด แต่ไม่รู้ว่าผิดตรงไหน

Even parity	เติมบิตพาริตีให้**จำนวนเลข 1 ทั้งชุด (รวมบิตพาริตี) เป็นเลขคู่** · บิตพาริตี = XOR ของบิตข้อมูลทั้งหมด
Odd parity	เติมให้**จำนวนเลข 1 ทั้งชุดเป็นเลขคี่** · บิตพาริตี = ส่วนกลับของ XOR ของบิตข้อมูลทั้งหมด
ตรวจฝั่งรับ	นับเลข 1 ใหม่ · ผิดข้อตกลง = มีบิตผิด **จำนวนคี่บิต** · **ผิดพร้อมกัน 2 บิตจะจับไม่ได้**

CRC — หารด้วยพหุนาม เหลือศูนย์คือผ่าน

1 **นับดีกรีของตัวหาร**

ตัวหาร (generator) ยาว k บิต → ดีกรี $r = k - 1$ · เช่น 1101 ยาว 4 บิต ดีกรีเป็น 3

2 **เติมศูนย์ต่อท้ายข้อมูล r ตัว**

ข้อมูล 1011 กับ $r = 3$ กลายเป็น 1011 000

3 **หารแบบ mod-2 — ใช้ XOR ทุกขั้น ไม่มีการยืม**

วางตัวหารชิดบิต 1 ซ้ายสุดที่เหลือ แล้ว XOR ลงไป ทำซ้ำจนความยาวที่เหลือสั้นกว่าตัวหาร

4 **เศษ r บิตคือ CRC — เอาไปแทนที่ศูนย์ที่เติมไว้**

ส่งออกไปเป็น **ข้อมูลเดิม + เศษ** · ฝั่งรับหารทั้งชุดด้วยตัวหารเดิม **เหลือเศษ 0 = ผ่าน**

Hamming — บอกได้ว่าบิตไหนผิด และแก้ได้

ขั้นตอน	รายละเอียด
หาจำนวนบิตพาริตี	ข้อมูล d บิต ต้องมีพาริตี p บิตที่น้อยที่สุดที่ทำให้ $2^p \geq d + p + 1$
วางตำแหน่ง	บิตพาริตีอยู่ตำแหน่ง**กำลังสอง** คือ 1, 2, 4, 8, ... (นับจากซ้าย เริ่มที่ 1) · ที่เหลือใส่ข้อมูลเรียงไป
แต่ละตัวคุมใคร	p_1 คุมตำแหน่ง 1,3,5,7,9,11,... · p_2 คุม 2,3,6,7,10,11,... · p_4 คุม 4,5,6,7,12,13,14,15 · p_8 คุม 8–15 กฎจำ: ตำแหน่งไหนที่**เลขตำแหน่งเขียนเป็นฐานสองแล้วมีบิตตรงกับ p ตัวนั้น** ก็ถูกคุม
ตั้งค่าพาริตี	ตั้งให้แต่ละกลุ่มมีจำนวนเลข 1 **เป็นคู่** (even parity — เป็นแบบที่วิชานี้ใช้)
ฝั่งรับหาบิตผิด	ตรวจทุกกลุ่มใหม่ · กลุ่มไหนไม่ผ่าน เอา**เลขตำแหน่งของพาริตีกลุ่มนั้นมาบวกกัน** ผลบวกคือ**ตำแหน่งบิตที่ผิด** · ได้ 0 = ไม่มีบิตผิด

4 เกต 7 ตัว — สัญลักษณ์ สูตร และตารางเดียวจบ



สัญลักษณ์แบบ distinctive shape ตามตำรา Floyd · จุดกลมเล็ก ๆ ที่ปลาย = กลับค่า · เครื่องหมาย (+) ในวงคือ $\oplus$

A	B	NOT A $\overline{A}$	AND AB	OR $A+B$	NAND $\overline{AB}$	NOR $\overline{A+B}$	XOR $A \oplus B$	XNOR $\overline{A \oplus B}$
0	0	1	0	0	1	1	0	1
0	1	1	0	1	1	0	1	0
1	0	0	0	1	1	0	1	0
1	1	0	1	1	0	0	0	1

คอลัมน์ NOT A ใช้เฉพาะค่าของ A · อีก 6 คอลัมน์ใช้ทั้ง A และ B · **แถวล่างสุด (A=1, B=1) คือแถวที่แยกเกตออกจากกันได้ไวที่สุด**

อ่านเป็นประโยคเดียวต่อหนึ่งเกต

AND	ออก 1 เมื่อ**ขาเข้าเป็น 1 ทุกขา** · มี 0 ขาเดียวก็ตก	**NAND**	ตรงข้ามกับ AND ทุกช่อง · ออก 0 ช่องเดียว คือตอนสูงครบ
OR	ออก 0 เมื่อ**ขาเข้าเป็น 0 ทุกขา** · มี 1 ขาเดียวก็ขึ้น	**NOR**	ตรงข้ามกับ OR ทุกช่อง · ออก 1 ช่องเดียวคือตอนต่ำครบ
XOR	ออก 1 เมื่อขาเข้า**ต่างกัน** · 2 ขา = "ไม่เท่ากัน"	**XNOR**	ออก 1 เมื่อขาเข้า**เหมือนกัน** · ใช้เป็นตัวเทียบ "เท่ากันไหม"
NOT	ออก 1 เมื่อขาเข้าเป็น 0 · ขาเดียวเท่านั้น · ต่อสองตัวติดกันแล้วค่าเดิมกลับมา ($\overline{\overline{A}} = A$)		

Opposing logic และ Dual symbols

Opposing logic — คนละเกต ค่าตรงข้ามกันทุกแถว

AND ↔ NAND · OR ↔ NOR · XOR ↔ XNOR
เอาตารางมาทาบกัน **ต้องไม่มีแถวไหนเหมือนกันเลย**

Dual symbols — เกตตัวเดียวกัน วาดได้สองหน้าตา

NAND ≡ negative-OR · **NOR ≡ negative-AND**
ตารางค่าความจริง**เหมือนกันทุกแถว** เพราะเป็นเดอมอร์แกนที่วาดเป็นรูป



NAND = negative-OR



NOR = negative-AND

$\overline{AB} = \overline{A} + \overline{B}$ (ซ้าย) และ $\overline{A + B} = \overline{A}\,\overline{B}$ (ขวา) — เดอมอร์แกนที่วาดออกมาเป็นสัญลักษณ์

Universal gates — NAND ตัวเดียว หรือ NOR ตัวเดียว สร้างได้ทุกเกต



NOT — มัดขาเข้าเข้าด้วยกัน · 1 ตัว

AND — NAND แล้วกลับค่า · 2 ตัว

OR — กลับค่าขาเข้าทั้งสองก่อน แล้ว NAND · 3 ตัว

ตัวที่สามคือเดอมอร์แกนตรง ๆ: $\overline{\overline{A} \cdot \overline{B}} = A + B$ — กลับค่าขาเข้าก่อนแล้วค่อย NAND จึงได้ OR

จะสร้าง	ใช้ NAND อย่างเดียว	ใช้ NOR อย่างเดียว
NOT	มัดขาเข้าสองขาเข้าด้วยกัน — **1 ตัว**	มัดขาเข้าสองขาเข้าด้วยกัน — **1 ตัว**
AND	NAND แล้วต่อ NAND-ตัวกลับค่า — **2 ตัว**	กลับค่าขาเข้าทั้งสองก่อน แล้ว NOR — **3 ตัว**
OR	กลับค่าขาเข้าทั้งสองก่อน แล้ว NAND — **3 ตัว**	NOR แล้วต่อ NOR-ตัวกลับค่า — **2 ตัว**
NAND	ตัวมันเอง — **1 ตัว**	สร้าง AND ก่อน แล้วกลับค่า — **4 ตัว**
NOR	สร้าง OR ก่อน แล้วกลับค่า — **4 ตัว**	ตัวมันเอง — **1 ตัว**

กฎจำ: **เกตที่อยู่กลุ่มเดียวกับตัวตั้งต้นใช้ 2 ตัว เกตที่ข้ามกลุ่มใช้ 3 ตัว** (NAND อยู่กลุ่ม AND · NOR อยู่กลุ่ม OR) · ตัวที่ต้องกลับค่าเพิ่มอีกชั้นก็บวกอีก 1

Bubble-to-bubble — กฎที่ทำให้วงจรอ่านง่ายขึ้นทันที

จุดกลมต่อจุดกลม = หักล้างกัน

จุดกลมเล็ก ๆ (bubble) ที่ปลายขาของเกต แปลว่า **"กลับค่าตรงนี้"** ถ้าขาออกที่มีจุดกลม ต่อเข้ากับขาเข้าที่มีจุดกลม **กลับค่าสองครั้ง = ไม่ได้กลับเลย** เราจึง**อ่านผ่านจุดต่อนั้นได้ตรง ๆ** เหมือนไม่มีอะไรกั้น

nonbubble-to-nonbubble ก็เหมือนกัน — ไม่มีจุดกลมทั้งสองฝั่ง ก็ไม่มีการกลับค่า อ่านตรงได้เช่นกัน

ที่ต้องระวังคือแบบผสม — จุดกลมฝั่งเดียว แปลว่า**ยังเหลือการกลับค่าอยู่หนึ่งครั้ง** ลืมนับเมื่อไรคำตอบกลับด้านทั้งข้อ · วิธีใช้งานจริง: เปลี่ยนเกตบางตัวเป็น**สัญลักษณ์คู่**ของมัน เพื่อให้จุดต่อกลายเป็นจุดกลม-ต่อ-จุดกลม แล้ววงจรจะอ่านออกเป็นสูตรได้ในบรรทัดเดียว

5 พีชคณิตบูลีน · De Morgan · Consensus

ชื่อกฎ	รูปแบบ OR	รูปแบบ AND
Commutative	$A + B = B + A$	$AB = BA$
Associative	$A + (B + C) = (A + B) + C$	$A(BC) = (AB)C$
Distributive	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Identity	$A + 0 = A$	$A \cdot 1 = A$
Null	$A + 1 = 1$	$A \cdot 0 = 0$
Idempotent	$A + A = A$	$A \cdot A = A$
Complementary	$A + \overline{A} = 1$	$A\,\overline{A} = 0$
Involution	$\overline{\overline{A}} = A$ — ขีดบนสองชั้นหักล้างกัน	
Absorption	$A + AB = A$	$A(A + B) = A$
Absorption (แบบมีขีด)	$A + \overline{A}B = A + B$	$A(\overline{A} + B) = AB$

De Morgan — "Break the bar and change the sign!"

$$\overline{XY} = \overline{X} + \overline{Y} \quad \overline{X + Y} = \overline{X}\,\overline{Y}$$

ขั้นตอนที่ใช้ได้กับนิพจน์ซ้อนกันกี่ชั้นก็ได้: หักขีดบน**ชั้นนอกสุดก่อน**ทีละชั้น ทุกครั้งที่หักขีด ให้**สลับเครื่องหมาย** (คูณกลายเป็นบวก · บวกกลายเป็นคูณ) แล้ว**แจกขีดที่หักลงไปให้ทุกก้อนข้างใน**

หยุดเมื่อไม่มีขีดบนอันไหนคลุมตัวแปรเกินหนึ่งตัวแล้ว · ขยายได้เกินสองตัว: $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$ และ $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$

Consensus Theorem — ตัดพจน์ที่ไม่ได้ทำอะไร

$$AB + \overline{A}C + BC = AB + \overline{A}C$$

แบบ SOP — พจน์ BC คือพจน์ส่วนเกิน

$$(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$$

แบบ POS — พจน์ $(B + C)$ คือพจน์ส่วนเกิน

วิธีหาพจน์ส่วนเกินในข้อสอบ: ① มองหาสองพจน์ที่มีตัวแปรตัวหนึ่งโผล่ทั้งแบบมีขีดและไม่มีขีด (A กับ $\overline{A}$) ② เอาตัวแปรที่เหลือของสองพจน์นั้นมาคูณกัน ได้ BC ③ ถ้าผลคูณนั้น**มีอยู่จริงในนิพจน์** ให้**ตัดทิ้งได้เลย**

ทำไมตัดได้: ถ้า $A = 1$ พจน์ BC จะเป็น 1 ได้ก็ต้องมี $B = 1$ ซึ่งตอนนั้น AB เป็น 1 อยู่แล้ว · ถ้า $A = 0$ พจน์ BC จะเป็น 1 ได้ก็ต้องมี $C = 1$ ซึ่งตอนนั้น $\overline{A}C$ เป็น 1 อยู่แล้ว — **ไม่ว่า A จะเป็นอะไร BC ก็ไม่เคยเป็นคนเดียวที่ทำให้ผลเป็น 1** จึงตัดทิ้งได้

6 Minterm · Maxterm · SOP · POS

ศัพท์	นิยาม	ตัวอย่าง 3 ตัวแปร A, B, C
Literal	ตัวแปรหนึ่งตัว จะมีขีดบนหรือไม่ก็ได้	$A \cdot \overline{B}$ — นับเป็นคนละ literal
Product term	literal หลายตัว**คูณ**กัน	$A\overline{B} \cdot \overline{A}BC$
Sum term	literal หลายตัว**บวก**กัน	$A + \overline{B} \cdot \overline{A} + B + C$
Minterm m_i	product term ที่มี**ตัวแปรครบทุกตัว** · เป็น 1 ได้**แถวเดียว**คือแถว i กติกา: บิตของแถวเป็น **1** → **ตัวแปรไม่มีขีด** · บิตเป็น **0** → **ตัวแปรมีขีด**	แถว $ABC = 101$ ได้ $m_5 = A\overline{B}C$
Maxterm M_i	sum term ที่มี**ตัวแปรครบทุกตัว** · เป็น 0 ได้**แถวเดียว**คือแถว i กติกา**กลับกันกับ minterm**: บิตเป็น **1** → **ตัวแปรมีขีด** · บิตเป็น **0** → **ตัวแปรไม่มีขีด**	แถว $ABC = 101$ ได้ $M_5 = \overline{A} + B + \overline{C}$
ความสัมพันธ์	$M_i = \overline{m_i}$ — เอาเดอมอร์แกนใส่มินเทอมแล้วได้แม็กซ์เทอมเลขเดียวกันพอดี	

ตาราง → SOP (Sum of Products)

มองเฉพาะ**แถวที่ผลออกเป็น 1**
แถวหนึ่ง = มินเทอมหนึ่งตัว แล้ว**บวกกันทั้งหมด**
$F = \sum m($**เลขแถวที่ออก** $1)$

จำ: ออก 1 → เขียนตรงตัว (1 คือไม่มีขีด)

ตาราง → POS (Product of Sums)

มองเฉพาะ**แถวที่ผลออกเป็น 0**
แถวหนึ่ง = แม็กซ์เทอมหนึ่งตัว แล้ว**คูณกันทั้งหมด**
$F = \prod M($**เลขแถวที่ออก** $0)$

จำ: ออก 0 → เขียน**กลับค่า** (1 คือมีขีด)

เช็กตัวเองได้ทันที

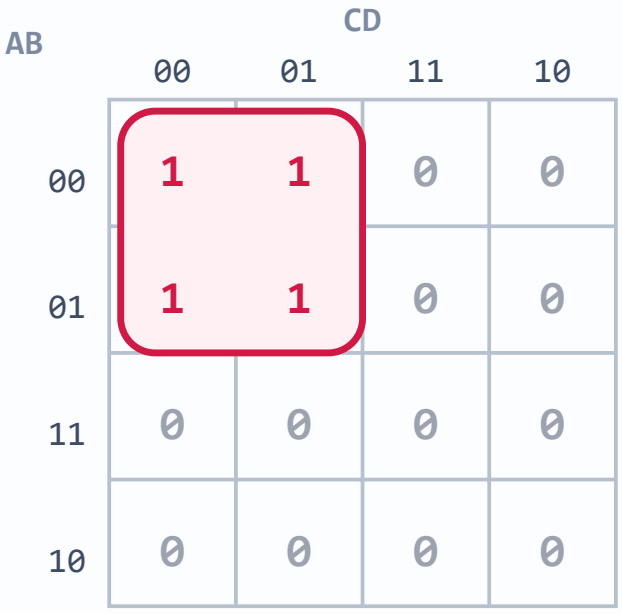
เลขแถวที่ใช้ใน $\sum m$ กับเลขแถวที่ใช้ใน $\prod M$ ต้อง**ไม่ซ้ำกันเลยสักตัว** และเอามารวมกันต้องได้**ครบทุกแถวพอดี** · ตัวแปร n ตัว แถวทั้งหมด 2^n แถว เพราะฉะนั้น **จำนวนพจน์ของ SOP + จำนวนพจน์ของ POS** $= 2^n$ **เสมอ** — ถ้าไม่เท่า แปลว่าตกแถวไปแถวหนึ่ง

ทำให้เป็นรูปมาตรฐาน (standard form) เมื่อพจน์ยังไม่ครบตัวแปร

SOP: พจน์ไหนขาดตัวแปร X ให้**คูณด้วย** $(X + \overline{X})$ ซึ่งมีค่าเท่ากับ 1 จึงไม่เปลี่ยนค่า แล้วกระจายออก
เช่น $A\overline{B}$ ในโดเมน A, B, C กลายเป็น $A\overline{B}(C + \overline{C}) = A\overline{B}C + A\overline{B}\,\overline{C}$

POS: พจน์ไหนขาดตัวแปร X ให้**บวกด้วย** $X\overline{X}$ ซึ่งมีค่าเท่ากับ 0 แล้วใช้กฎ distributive กระจาย
เช่น $(A + \overline{B})$ กลายเป็น $(A + \overline{B} + C\overline{C}) = (A + \overline{B} + C)(A + \overline{B} + \overline{C})$

7 Karnaugh Map — กฎจับกลุ่มและประโยคกล



กลุ่ม 4 ช่อง

มองซ้าย: A ไม่เปลี่ยน เป็น 0 ตลอด

มองซ้าย: B เปลี่ยน 0 เป็น 1 — ตัดทิ้ง

มองบน: C ไม่เปลี่ยน เป็น 0 ตลอด

มองบน: D เปลี่ยน 0 เป็น 1 — ตัดทิ้ง

เหลือตัวที่ไม่เปลี่ยน 2 ตัว

F = A'C'

(ทั้งคู่เป็น 0 จึงใส่ขีดบนทั้งคู่)

หัวตารางเรียงแบบเกรย์ 00 01 11 10 — ช่องที่ติดกันต่างกันบิตเดียวเสมอ

$F = \sum m(0,1,4,5)$ · กลุ่ม 4 ช่องตัดตัวแปรทิ้งได้ 2 ตัว เหลือ $F = \overline{A}\,\overline{C}$

ประโยคกลของอาจารย์

"look left, look up, look for variable that do not change."

มองซ้าย (หัวแถว) · มองบน (หัวคอลัมน์) · หาตัวแปรที่ไม่เปลี่ยนค่า

ตัวแปรที่**ไม่เปลี่ยนค่า**ภายในกลุ่ม ให้**เก็บไว้** (เป็น 1 เขียนตรงตัว · เป็น 0 เขียนใส่ขีดบน) · ตัวแปรที่**เปลี่ยนค่า**ภายในกลุ่ม ให้**ตัดทิ้ง** เพราะมันไม่มีผลกับคำตอบ

กฎจับกลุ่ม	รายละเอียด
ขนาดกลุ่ม	ต้องเป็น**กำลังของสอง**เท่านั้น: 1 · 2 · 4 · 8 · 16 — ไม่มีกลุ่ม 3 ช่องหรือ 6 ช่อง
รูปทรง	ต้องเป็น**สี่เหลี่ยมมุมฉาก**ที่ติดกัน — ห้ามเป็นตัว L หรือทแยงมุม
ต่อขอบได้	ขอบ**ซ้ายกับขวา**ติดกัน · ขอบ**บนกับล่าง**ติดกัน · **สี่มุม**ของตาราง 4 ตัวแปรจับเป็นกลุ่มเดียวกันได้
ซ้อนทับกันได้	ช่องเดียวกันอยู่ได้หลายกลุ่ม · ไม่ผิดกติกา และมักทำให้คำตอบสั้นลง
ใหญ่ไว้ก่อน	จับ**กลุ่มใหญ่ที่สุดเท่าที่ทำได้** · กลุ่ม 2^k ช่อง ตัดตัวแปรทิ้งได้ k ตัว
น้อยกลุ่มไว้ก่อน	ใช้**จำนวนกลุ่มน้อยที่สุด**ที่ยังคลุมเลข 1 ได้ครบทุกช่อง · 1 กลุ่ม = 1 พจน์ในคำตอบ
ห้ามมีช่องตกหล่น	เลข 1 ทุกช่อง**ต้องอยู่ในอย่างน้อยหนึ่งกลุ่ม**

กฎจับกลุ่ม	รายละเอียด
SOP หรือ POS	**จับเลข 1 ได้ SOP** (พจน์คูณบวกกัน) · **จับเลข 0 ได้ POS** (พจน์บวกคูณกัน และ**กลับค่าตัวแปรทุกตัว**ตอนอ่าน)
Don't care (X)	ถือเป็น 1 **เมื่อมันช่วยให้กลุ่มใหญ่ขึ้น** · ถ้าไม่ช่วย ให้ถือเป็น 0 แล้วไม่ต้องคลุม · **ห้ามจับกลุ่มที่มีแต่ X ล้วน**

กับดัก K-Map ที่เจอบ่อย

① เขียนหัวตารางเป็น 00 01 **10 11** — ผิด ต้องเป็น **00 01 11 10** ไม่งั้นช่องที่ติดกันจะต่างกันสองบิต จับกลุ่มแล้วได้คำตอบผิด

② ลืมว่าขอบต่อกัน ทำให้พลาดกลุ่มใหญ่ แล้วได้คำตอบที่ถูกแต่**ไม่สั้นที่สุด** ซึ่งโจทย์ที่ถามหา minimum จะไม่ให้คะแนนเต็ม

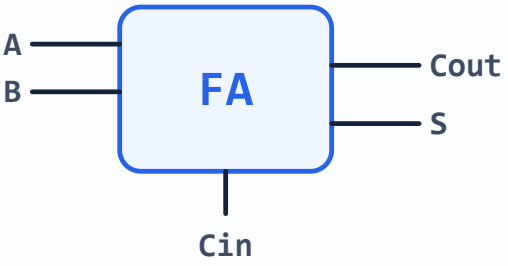
③ อ่าน POS แล้วลืมกลับค่าตัวแปร — จับเลข 0 ต้องอ่านกลับด้าน

8 วงจรผสมทุกตัวในขอบเขตสอบ พร้อมสมการประจำตัว

ทุกสมการในหน้านี้ตรวจด้วยโปรแกรมแล้วว่าตรงกับตารางค่าความจริงของวงจรนั้นทุกแถว (ดูไฟล์ `check_front.py`) · เครื่องหมาย $\oplus$ คือ XOR · $\odot$ คือ XNOR · ขีดบนคือกลับค่า

วงจรบวก — Adders

วงจร	สมการ
Half adder เข้า A, B · ออก S, C_{out}	$S = A \oplus B$ $C_{out} = AB$ ไม่มีตัวทดเข้า จึงบวกได้แค่หลักขวาสุด
Full adder เข้า A, B, C_{in} · ออก S, C_{out}	$S = A \oplus B \oplus C_{in}$ $C_{out} = AB + C_{in}(A \oplus B)$ เขียนแบบเทียบเท่า: $C_{out} = AB + AC_{in} + BC_{in}$ (เสียงข้างมาก)
Parallel binary adder ripple carry	เอา full adder มาต่อกัน n ตัว · C_{out} ของหลักหนึ่ง กลายเป็น C_{in} ของหลักถัดไปทางซ้าย C_{in} ของหลักขวาสุดตั้งเป็น 0 · **ช้าเพราะหลักซ้ายต้องรอหลักขวาเสร็จก่อน**
Look-ahead carry มองตัวทดล่วงหน้า	**carry generation** $G = AB$ **carry propagation** $P = A + B$ **output carry** $C_{out} = G + P\,C_{in}$ $G = 1$ = หลักนี้**สร้างตัวทดเอง** ไม่ต้องรอใคร · $P = 1$ = หลักนี้**ส่งตัวทดที่รับมาผ่านไป**



full adder รับ 3 บิต ออก 2 บิต

วงจรลบ — Subtractors

วงจร	สมการ
Half subtractor เข้า A, B · ออก D, B_{out}	$D = A \oplus B$ (ผลต่าง — สูตรเดียวกับวงจรบวกเป๊ะ) $B_{out} = \overline{A}B$ **ตรงนี้ต่างจากวงจรบวก** — ยืมเมื่อตัวตั้งเป็น 0 แต่ตัวลบเป็น 1
Full subtractor เข้า A, B, B_{in} · ออก D, B_{out}	$D = A \oplus B \oplus B_{in}$ $B_{out} = \overline{A}B + \overline{A}B_{in} + B\,B_{in}$ เขียนแบบเทียบเท่า: $B_{out} = \overline{A}B + \overline{(A \oplus B)}\,B_{in}$

กับดักวงจรลบ

ช่อง**ผลต่าง**ของวงจรลบ เหมือนช่อง**ผลบวก**ของวงจรบวกทุกแถว (ใช้ XOR เหมือนกัน) คนจึงเผลอคิดว่าตัวยืมก็เหมือนตัวทด — **ไม่เหมือน**ครับ ตัวทดคือ AB (สมมาตร สลับ A กับ B ได้) แต่ตัวยืมคือ $\overline{A}B$ (**ไม่สมมาตร** สลับแล้วผิดทันที) เพราะการลบสนใจว่า**ใครเป็นตัวตั้ง ใครเป็นตัวลบ**

วงจรเปรียบเทียบ — Comparator

ถามว่าอะไร	สมการ (เทียบ 1 บิต)
$A = B$ **ไหม**	$\overline{A \oplus B}$ (คือ XNOR) — ออก 1 เมื่อสองบิตเหมือนกัน
$A > B$ **ไหม**	$A\overline{B}$ — ออก 1 เมื่อ A เป็น 1 และ B เป็น 0
$A < B$ **ไหม**	$\overline{A}B$ — ออก 1 เมื่อ A เป็น 0 และ B เป็น 1
เทียบเท่ากันหลายบิต	เอา XNOR ของทุกหลักมา **AND** กันหมด: $(A_1 \odot B_1)(A_0 \odot B_0)$
มากกว่า หลายบิต	ไล่จาก**บิตซ้ายสุดลงมา**: $A > B$ เมื่อ $A_1\overline{B_1} + (A_1 \odot B_1)A_0\overline{B_0}$ อ่านว่า "ชนะที่บิตซ้าย" หรือ "บิตซ้ายเสมอกัน แล้วชนะที่บิตถัดมา"

Encoder — สายเข้าหลายเส้น ออกมาเป็นเลข

วงจร	สมการขาออก
8-to-3 encoder เข้า $D_0..D_7$ · ออก $A_2A_1A_0$	$A_2 = D_4 + D_5 + D_6 + D_7$ $A_1 = D_2 + D_3 + D_6 + D_7$ $A_0 = D_1 + D_3 + D_5 + D_7$ กฎอ่าน: A_k เป็น OR ของทุกเส้นที่**เลขประจำเส้นมีบิตที่ k เป็น 1**
Decimal-to-BCD encoder ปุ่ม 0–9 · ออก $A_3A_2A_1A_0$	$A_3 = D_8 + D_9$ $A_2 = D_4 + D_5 + D_6 + D_7$ $A_1 = D_2 + D_3 + D_6 + D_7$ $A_0 = D_1 + D_3 + D_5 + D_7 + D_9$ ไม่มีสมการของ D_0 เพราะกดปุ่ม 0 แล้วขาออกเป็น 0000 อยู่แล้ว

Decoder — รับเลขเข้า ปลุกสายออกหนึ่งเส้น

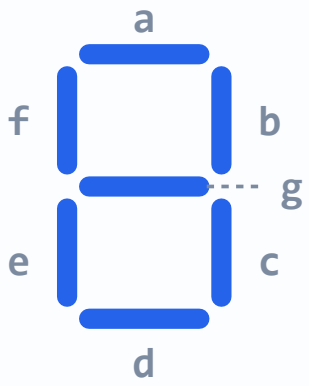
วงจร	สมการขาออก
3-to-8 decoder เข้า $A_2A_1A_0$ · ออก $Y_0..Y_7$	Y_i **คือมินเทอมตัวที่** i **พอดี** $Y_0 = \overline{A_2}\,\overline{A_1}\,\overline{A_0}$ · $Y_5 = A_2\overline{A_1}A_0$ · $Y_7 = A_2A_1A_0$ ทุกจังหวะมีสายออก**ติดหนึ่งเส้นเท่านั้น**
BCD-to-decimal decoder เข้า 4 บิต · ออก $Y_0..Y_9$	Y_i คือมินเทอมตัวที่ i ของ 4 บิต เฉพาะ $i = 0..9$ รหัส 1010–1111 ไม่มีสายออกเส้นไหนติดเลย เพราะเป็นรหัสที่ BCD ไม่ใช้



ออกติดทีละหนึ่งเส้น

7-Segment LED display และ BCD-to-7-Segment decoder

ขีด	ติดที่ตัวเลขอะไรบ้าง (เขียนเป็น $\sum m$)
a บนสุด	$\sum m(0,2,3,5,6,7,8,9)$ — ดับที่ 1 กับ 4
b ขวาบน	$\sum m(0,1,2,3,4,7,8,9)$ — ดับที่ 5 กับ 6
c ขวาล่าง	$\sum m(0,1,3,4,5,6,7,8,9)$ — ดับที่ 2 ตัวเดียว
d ล่างสุด	$\sum m(0,2,3,5,6,8,9)$
e ซ้ายล่าง	$\sum m(0,2,6,8)$ — ติดน้อยที่สุด
f ซ้ายบน	$\sum m(0,4,5,6,8,9)$
g กลาง	$\sum m(2,3,4,5,6,8,9)$ — ดับที่ 0, 1, 7



ชื่อขีดมาตรฐาน a–g

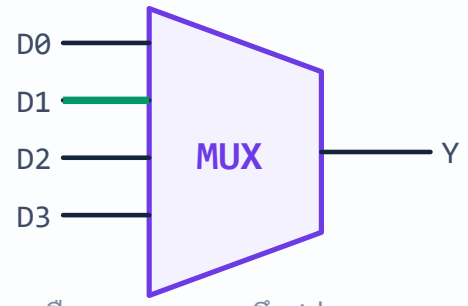
วิธีหาสมการของแต่ละขีด: เอาเลขในวงเล็บไปลง K-Map 4 ตัวแปร โดยให้ 1010–1111 เป็น **don't care** (เพราะ BCD ไม่ใช้) แล้วจับกลุ่ม — จะได้สมการที่สั้นกว่าการรวมมินเทอมตรง ๆ มาก

Code converter — ไบนารี ↔ เกรย์

ทิศทาง	สมการ · โครงวงจร
ไบนารี → เกรย์	$G_3 = B_3 \cdot G_2 = B_3 \oplus B_2 \cdot G_1 = B_2 \oplus B_1 \cdot G_0 = B_1 \oplus B_0$ ใช้ XOR $n-1$ ตัว · **ทุกตัวรับขาเข้าจากไบนารีโดยตรง จึงทำงานพร้อมกันได้หมด**
เกรย์ → ไบนารี	$B_3 = G_3 \cdot B_2 = B_3 \oplus G_2 \cdot B_1 = B_2 \oplus G_1 \cdot B_0 = B_1 \oplus G_0$ ใช้ XOR $n-1$ ตัวเหมือนกัน แต่**ต่อกันเป็นลูกโซ่** — ตัวถัดไปต้องรอผลของตัวก่อนหน้า

MUX และ DEMUX

วงจร	สมการ
Multiplexer 4-to-1 เข้า $D_0..D_3$ · เลือกด้วย S_1S_0 · ออก Y เส้นเดียว	$Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$ อ่านว่า "มินเทอมของขาเลือก คูณกับข้อมูลช่องนั้น แล้วบวกกันหมด" — ขาเลือกเป็นตัวปลดล็อกทีละช่อง
Demultiplexer 1-to-4 เข้า D เส้นเดียว · เลือกด้วย S_1S_0 · ออก $Y_0..Y_3$	$Y_0 = \overline{S_1}\,\overline{S_0}D \cdot Y_1 = \overline{S_1}S_0D$ $Y_2 = S_1\overline{S_0}D \cdot Y_3 = S_1S_0D$ เหมือน decoder ที่เอา D มาคูณเพิ่มทุกเส้น — ถ้า $D = 0$ สายออกดับหมดทุกเส้น



ขาเลือก S1S0 = 01 จึงปล่อย D1 ออก

เส้นทางลัดที่ใช้ได้กับวงจรผสมทุกตัว

ถ้าจำสมการของวงจรไหนไม่ได้ ให้**สร้างตารางค่าความจริงของมันขึ้นมาเองจากคำนิยาม** แล้วอ่านออกมาเป็น SOP ตามหัวข้อ 6 ครับ — ช้ากว่าแต่**ไม่มีทางผิด** ตัวอย่าง: จำสมการ full adder ไม่ได้ ก็เขียนตาราง 8 แถวว่าบวกเลขสามบิตได้เท่าไร แล้วอ่านคอลัมน์ S กับ C_{out} ออกมาเป็น SOP ก็ได้สมการกลับมาเอง

ปิดท้าย

ทุกตัวเลข ทุกตารางค่าความจริง และทุกสมการในหน้านี้ ผ่านการตรวจด้วยโปรแกรม `check_front.py` แบบไล่ทุกกรณี (brute force) ก่อนพิมพ์ลงเล่ม ถ้าเจอที่ไหนไม่ตรงกับที่อาจารย์สอน ให้ยึดของอาจารย์ไว้ก่อน แล้วค่อยมาดูว่าต่างกันเพราะนิยามคนละแบบหรือเปล่าครับ