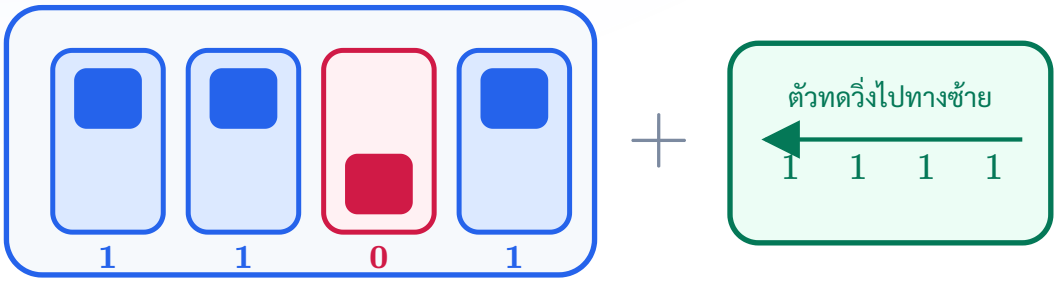
02 EP

เลขคณิตฐานสอง และเลขที่มีเครื่องหมาย

บวก ลบ คูณ หารยาว (เห็นตัวทด-ตัวยืมทุกหลัก) · 1's complement · 2's complement · sign-magnitude / 1's / 2's · ปัญหา -0 · พิสัยของเลข n บิต



สวิตช์มีแค่ปิดกับเปิด — เลขคณิตทั้งหมดต้องทำได้จากตรงนี้ครับ

พี่สอนน้องแบบไม่ต้องมีพื้นฐานมาก่อนเลยครับ · อ้างอิงสไลด์ EES271 Week 1 หน้า 18–25 และ 35–46 + Assignment #1 ข้อ 8–15

ตอนนี้เราจะเดินไปทางไหนกัน

ก่อนเริ่ม พี่ขอกางแผนที่ให้น้องดูก่อนครับ ว่า EP นี้มีอะไรบ้าง จะได้ไม่หลงทาง

1	**ทำไมต้องเอา 0 กับ 1 มาบวกลบกัน**	อุ่นเครื่อง
2	**บวกเลขฐานสอง — กฎแค่ 4 บรรทัด**	พื้นฐานสุด
3	**ลบเลขฐานสอง — เรื่องของการยืม**	จุดพลาดเยอะ
4	**คูณเลขฐานสอง — ลอกกับใส่ศูนย์เท่านั้น**	ออกสอบชัวร์
5	**หารยาวฐานสอง — ถามคำถามเดียวซ้ำ ๆ**	ออกสอบชัวร์
6	**คอมพลีเมนต์ — 1's และ 2's complement**	หัวใจของ EP
7	**เลขมีเครื่องหมาย — เก็บได้สามหน้าตา**	หัวใจของ EP
8	**อ่านค่าฐานสิบกลับ และพิสัยของเลข n บิต**	จุดตาย
9	**เฉลย Assignment #1 ข้อ 8–11 (คอมพลีเมนต์ + เลขมีเครื่องหมาย)**	เฉลยเต็ม
10	**เฉลย Assignment #1 ข้อ 12–13 (บวก และ ลบ)**	เฉลยเต็ม
11	**เฉลย Assignment #1 ข้อ 14–15 (คูณ และ หาร)**	เฉลยเต็ม
12	**กับดักรวม + สรุปหน้าเดียว + เช็คลิสต์ก่อนสอบ**	ทบทวน

ชีทนี้ใช้ยังไง

I DO

พี่ทำให้ดูทีละขั้น ทุกขั้นมีรูปในช่องซ้ายและมีเหตุผลว่า **ทำไมถึงทำแบบนั้น** ไม่ใช่แค่บอกว่าทำอะไรครับ

WE DO

ทำด้วยกัน — มีช่องว่างให้เติม แล้วเฉลย**ทุกช่องพร้อมเหตุผล**อยู่ใต้ข้อทันที ไม่ต้องพลิกไปท้ายเล่มครับ

YOU DO

น้องทำเอง แล้วเฉลยละเอียดอยู่ติดกันเลย — ลองปิดไว้ก่อนแล้วค่อยเปิดเทียบนะครับ

ส่วนที่สำคัญที่สุดของ EP นี้

ตอนที่ **9 ถึง 11** คือ**เฉลย Assignment #1 ข้อ 8–15 ครบทั้ง 19 ข้อย่อย**ครับ ทุกข้อย่อยมีภาพไล่ขั้นตอนของตัวเอง ไม่มีข้อไหนเขียนว่า “ทำเหมือนข้อก่อนหน้า” และทุกข้อมีการตรวจคำตอบด้วยวิธีที่สอง (แปลงเป็นฐานสิบทั้งสองฝั่ง) กำกับไว้

ทุกข้อจะเดินครบ **6 ขั้น** เหมือนกันหมด: ① โจทย์ต้นฉบับและสิ่งที่โจทย์ซ่อนไว้ → ② เตรียมก่อนลงมือ → ③ ไล่ทีละขั้นพร้อมภาพ → ④ คำตอบ → ⑤ ตรวจด้วยวิธีที่สอง → ⑥ กับดักของข้อนี้

มาจาก EP01

EP01 สอนให้น้อง**อ่านและแปลงเลขฐาน**ได้แล้วครับ — positional numbering, ฐาน 2 / 8 / 16, การหารสั้น เก็บเศษ, การจับกลุ่ม 3 บิตกับ 4 บิต

EP นี้จะเอาเลขฐานสองที่น้องแปลงเป็นแล้ว มาบวกลบคูณหารกันจริง ๆ และหาทางเก็บเลขติดลบลงในบิตที่มีแค่ 0 กับ 1 ครับ

1 ทำไมต้องเอา 0 กับ 1 มาบวกลบกัน

ตะขอเกี่ยว

ตอนนี้น้องอ่านและแปลงเลขฐานสองได้แล้วครับ เห็น 1101_2 ก็รู้ทันทีว่าเป็น 13_{10} เห็น 45_{10} ก็กดหารสั้นออกมาเป็น 101101_2 ได้

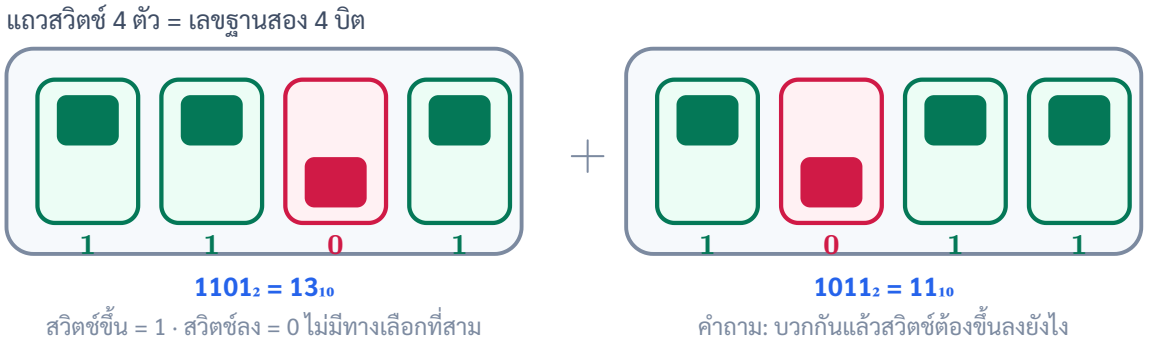
แต่สมมติว่าถ้าน้องเจอโจทย์แบบนี้ละครับ — **“เอาเลขฐานสองสองตัวมาบวกลบคูณหารกัน”** น้องจะทำยังไงครับ จะแปลงกลับเป็นฐานสิบ บวกในฐานสิบ แล้วแปลงกลับมาอีกทีไหมครับ (ทำได้ แต่ในห้องสอบมันช้าเกินไป และคอมพิวเตอร์ก็ไม่ได้ทำแบบนั้น)

แล้วอีกเรื่องที่หนักกว่า — **เลขติดลบเก็บยังไงในเมื่อมีแค่ 0 กับ 1** ครับ ในกระดาษเราขีดเครื่องหมายลบข้างหน้าได้ แต่ในสายไฟมันมีแค่ไฟติดกับไฟดับ ไม่มี “ขีดเล็ก ๆ” ให้ใช้ครับ ตอนนี้เราจะแก้ปัญหาสองเรื่องนี้ให้จบ

ของใกล้ตัว: แถวสวิตช์ไฟบนผนัง

ลองนึกถึงแถวสวิตช์ไฟ 4 ตัวบนผนังห้องครับ สวิตช์แต่ละตัวมีแค่**ปิดกับเปิด** ไม่มีท่าที่สาม — นี่คือทั้งหมดที่คอมพิวเตอร์มีเลยครับ

คำถามคือ ถ้าพี่อยากให้แถวสวิตช์นี้ “จำเลข 13” แล้ว “บวก 11 เข้าไป” ให้กลายเป็น 24 มันต้องมีกติกาที่**ทำได้ด้วยการเปิด-ปิดสวิตช์ล้วน ๆ** ครับ ห้ามใช้อย่างอื่นเลย



ทุกอย่างใน EP นี้ต้องทำได้ด้วยการเปิด-ปิดสวิตช์เท่านั้นครับ นี่คือกติกาข้อเดียวของวงจรดิจิทัล

EP นี้จะปลดล็อกอะไรให้น้อง

พอจบ EP นี้ น้องจะ**บวก ลบ คูณ หาร**เลขฐานสองได้ด้วยมือ และ**เก็บเลขติดลบ**ลงในบิตล้วน ๆ ได้ทั้งสามแบบที่ข้อสอบใช้ครับ

และที่สำคัญกว่านั้น — เรื่องนี้คือ**ฐานของวงจรบวกเลข (adder)** ที่น้องจะได้ต่อจริงด้วยลอจิกเกตใน EP ถัด ๆ ไปครับ ถ้าตรงนี้ไม่แน่น ตอนต่อวงจรจะงงหนักมาก

แผนที่ของ EP นี้

ครึ่งแรก — เลขคณิต

บวก · ลบ · คูณ · หารยาว
ทุกวิธีเขียนแบบตั้งหลักเหมือนฐานสิบเป๊ะ ต่างกันแค่**"เลขทดขึ้นเร็วกว่า"** เพราะฐานสองมีแค่ 0 กับ 1 ครับ

ครึ่งหลัง — เลขติดลบ

1's complement · 2's complement · สามรูปแบบของเลขมีเครื่องหมาย · การอ่านค่ากลับ · พิสัยของเลข n บิต

ศัพท์สอบชุดแรก (เจอในโจทย์ทุกข้อของ EP นี้)

English	แปลว่า	ข้อสอบเขียนมาแบบนี้
carry	ตัวทด (ที่ทดขึ้นหลักซ้าย)	1 + 1 = 0 with a carry of 1
borrow	ตัวยืม (ยืมจากหลักซ้าย)	0 − 1 with a borrow of 1
LSB / MSB	บิตขวาสุด / บิตซ้ายสุด	from right (LSB) to left (MSB)
partial product	แถวย่อยตอนคูณ	Form one partial product for each multiplier bit
quotient / remainder	ผลหาร / เศษ	write 1 in the quotient and subtract
complement	ค่าเติมเต็ม (ตัวกลับ)	Determine the 1's complement of ...
signed number	เลขที่มีเครื่องหมาย	Express −45 as an 8-bit signed number
sign bit	บิตเครื่องหมาย (บิตซ้ายสุด)	0 sign bit = positive number

วิธีใช้ชีทนี้

ทุกหัวข้อจะเดินเป็น **I DO** (พี่ทำให้ดู) → **WE DO** (ทำด้วยกัน) → **YOU DO** (น้องทำเอง แล้วเฉลยอยู่ใต้ข้อทันที) ครับ ส่วนท้ายเล่มเป็น**เฉลย Assignment #1 ข้อ 8–15 ทุกข้อย่อย** แบบไล่ทุกหลักไม่ข้ามเลยครับ

2 บวกเลขฐานสอง — กฎแค่ 4 บรรทัด

ตะขอเกี่ยว

สมมติว่าถ้าน้องต้องบวก $7 + 6$ ในฐานสิบ น้องทำได้ในหัวเลยใช่ไหมครับ แต่ถ้าโจทย์เขียนมาว่า $111_2 + 110_2$ ละครับ — **ห้ามแปลงเป็นฐานสิบก่อน** เพราะข้อสอบต้องการให้เห็นการทดทีละหลัก ตรงนี้แหละที่เราจะฝึกกันครับ

ของใกล้ตัว: มาตรวัดที่มีแค่สองเลข

ในฐานสิบ เลขประจำหลักไล่ 0,1,2,...,9 พอครบ 9 แล้วเติมอีกหนึ่ง มัน**หมุนกลับเป็น 0 แล้วทดขึ้นหลักซ้าย 1** ครับ

ฐานสองก็เหมือนกันเป๊ะ ต่างกันแค่ว่า**หลักหนึ่งมีแค่ 0 กับ 1** เพราะฉะนั้นพอถึง 1 แล้วเติมอีกหนึ่ง มันก็**หมุนกลับเป็น 0 แล้วทดขึ้นหลักซ้าย 1** ทันที — ทดเร็วกว่าฐานสิบมากครับ

ฐานสิบ — หมุนครบ 10 ค่อยทด

$\ldots 7 \quad 8 \quad 9 \quad \mathbf{10}$ เติม 1 เข้าไปที่ 9 → กลับเป็น 0 แล้วทด 1

ฐานสอง — หมุนครบ 2 ก็ทดเลย

$0 \quad 1 \quad \mathbf{10}$ เติม 1 เข้าไปที่ 1 → กลับเป็น 0 แล้วทด 1 ทันที นี่คือที่มาของ 1+1 = 10 ครับ

"10" ในฐานสองอ่านว่า **หนึ่ง-ศูนย์** ไม่ใช่ "สิบ" นะครับ มันมีค่าเท่ากับ 2 ในฐานสิบ

DEFINITION · Binary addition rules

0 + 0 = 0 (Sum of 0 with a carry of 0)

0 + 1 = 1 (Sum of 1 with a carry of 0)

1 + 0 = 1 (Sum of 1 with a carry of 0)

1 + 1 = 10 (Sum of 0 with a carry of 1)

พี่แปลว่า: มีแค่กรณีเดียวที่ต้องทด คือ**ตอนที่ 1 เจอ 1** ครับ นอกนั้นเขียนลงไปตรง ๆ ได้เลย — และถ้ามีตัวทดจากหลักขวามาด้วย ก็จะเกิดกรณีที่ห้าคือ $1 + 1 + 1 = 11$ (เขียน 1 ทดอีก 1) ครับ สไลด์เขียนไว้ในบรรทัดคำอธิบายพอดี

$0 + 0 = 0$	ไม่ทด
$0 + 1 = 1$	ไม่ทด
$1 + 0 = 1$	ไม่ทด
$1 + 1 = 10$	**เขียน 0 ทด 1**

$1 + 1 + 1 = 11$

กรณีนี้เกิดเมื่อหลักขวาส่งตัวทดขึ้นมา พร้อมกับที่หลักนี้มี 1 บวก 1 อยู่แล้ว

เขียน 1 แล้วทดขึ้นอีก 1

ตัวทดจะไม่มีทางเกิน 1 เด็ดขาดครับ

ทั้ง EP นี้ ตัวทดของการบวกฐานสอง**มีได้แค่ 0 หรือ 1 เท่านั้น** ถ้าน้องทดได้ 2 แปลว่าคิดผิดแล้วครับ

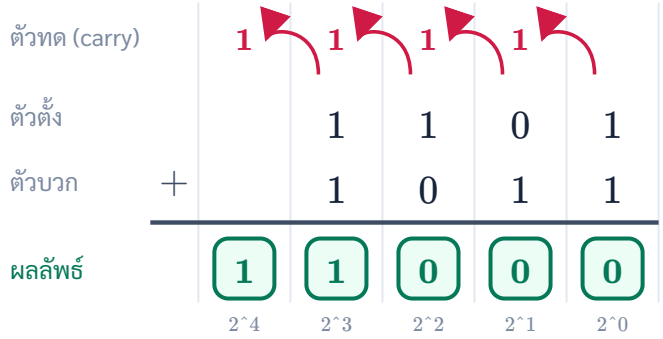
ทางลัด · จำแค่ประโยคเดียว

“นับจำนวน 1 ในหลักนั้น (รวมตัวทดด้วย) — ได้เลขคี่ให้เขียน 1 ได้เลขคู่ให้เขียน 0 แล้วถ้านับได้ตั้งแต่ 2 ขึ้นไปให้ทด 1”

ลองเช็คดูครับ: $1 + 1 + 1$ นับ 1 ได้ 3 ตัว (คี่ → เขียน 1) และ $3 \geq 2$ (→ ทด 1) ได้ 11 ถูกเป๊ะ · $1 + 1 + 0$ นับได้ 2 ตัว (คู่ → เขียน 0) และ ≥ 2 (→ ทด 1) ได้ 10 ถูกเป๊ะครับ

ไล่ให้ดูทุกหลัก — 1101 + 1011

พี่จะไล่ให้ดูก่อนหนึ่งข้อครับ ขอให้น้องดู**เลขสีแดงบนหัว** ให้ดี ๆ นั่นคือตัวทดที่หลักขวาส่งขึ้นมาให้



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 13 + 11 = 24

1101 + 1011 (แบบเขียนลงกระดาษสอบ)

```
  1 1 1 1
    1 1 0 1
+   1 0 1 1
-----------
1 1 0 0 0
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	1	2	**0**	**1**
2^1	1	0	1	2	**0**	**1**

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^2	1	1	0	2	**0**	**1**
2^3	1	1	1	3	**1**	**1**

กับดัก

คนพลาดเพราะลืมตัวทดตัวสุดท้ายมากที่สุดครับ — ในข้อนี้หลักซ้ายสุดคือ $1+1+1=11$ ตัวทดตัวสุดท้ายต้องออกไปเป็น**บิตใหม่ทางซ้าย** ทำให้คำตอบยาว 5 บิตคือ 11000_2 ไม่ใช่ 4 บิต

ถ้าลืมตัวนี้จะได้ $1000_2 = 8$ ซึ่งผิดไป 16 เลยครับ (คำตอบจริงคือ 24)

I DO พี่ทำโจทย์จากสไลด์ให้ดูครับ

ตัวอย่างที่พี่ทำ — สไลด์ EES271-01 · Binary Addition Example

> Add 1011 and 0110.

แปล: จงบวก 1011_2 กับ 0110_2 ครับ — โจทย์เขียนเลขให้ยาวเท่ากันมาแล้ว 4 บิต ทำให้ตั้งหลักได้เลย

1

$$\begin{array}{r} 1\,0\,1\,1 \\ 0\,1\,1\,0 \end{array}$$ เริ่ม

ตั้งหลักให้ตรงกัน แล้วเริ่มจากขวาสุด

เพราะตัวทดวิ่งจาก**ขวาไปซ้าย**เสมอครับ ถ้าเริ่มจากซ้ายจะไม่รู้ว่าต้องบวกตัวทดเท่าไร

2

$1+0=1$

เขียน 1 · ไม่มีตัวทด

หลัก 2^0 : $1+0=1$

เข้ากฎบรรทัดที่สาม เขียน 1 ลงไปตรง ๆ เพราะไม่มี 1 สองตัวมาชนกันครับ

3

$1+1=10$

เขียน 0 · ทด 1 ขึ้นไป

หลัก 2^1 : $1+1=10$

เข้ากฎบรรทัดที่สี่ครับ เขียน 0 ไว้ที่หลักนี้ แล้ว**ยก 1 ไปฝากหลักซ้าย** เพราะหลักนี้เก็บได้แค่ 0 กับ 1

4

$1+0+1=10$

เขียน 0 · ทด 1 อีกรอบ

หลัก 2^2 : ตัวทด 1 + บิต 0 + บิต 1

อย่าลืมตัวทดจากขั้นที่แล้วนะครับ นับ 1 ได้ 2 ตัว → คู่ เขียน 0 · และ ≥ 2 → ทด 1

5

$1 + 1 + 0 = 10$

เขียน 0 · ทดออกนอกกรอบ

หลัก 2^3 : ตัวทด 1 + บิต 1 + บิต 0

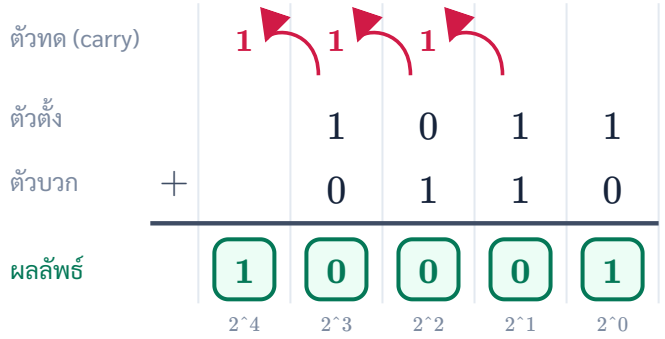
ได้ 0 ทด 1 อีกครั้ง คราวนี้**ไม่มีหลักซ้ายให้ทดแล้ว** ตัวทดตัวนี้จึงกลายเป็นบิตที่ห้าครับ

6

1 0 0 0 1

อ่านผลลัพธ์จากซ้ายไปขวา

ได้ 10001_2 ครับ — ตรวจฐานสิบ: $11 + 6 = 17$ และ $10001_2 = 16 + 1 = 17$ ตรงกันพอดี



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 11 + 6 = 17

1011 + 0110 (แบบเขียนลงกระดาษสอบ)

```
  1 1 1
    1 0 1 1
+   0 1 1 0
-----------
  1 0 0 0 1
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	0	1	**1**	**0**
2^1	0	1	1	2	**0**	**1**
2^2	1	0	1	2	**0**	**1**
2^3	1	1	0	2	**0**	**1**

คำตอบ I DO

ขั้น 1 — ตั้งหลักให้บิตขวาสุดตรงกัน เพราะบิตขวาสุดคือหลัก 2^0 ทั้งคู่ ถ้าตั้งเหลื่อมกันน้ำหนักจะผิดทันทีครับ

ขั้น 2 — ไล่จากขวาไปซ้าย ทีละหลัก ได้ตัวทดตามลำดับ $0, 1, 1, 1$ (ดูแถวสีแดงในภาพ) เพราะทุกครั้งที่หลักหนึ่งมี 1 ตั้งแต่สองตัวขึ้นไป มันต้องส่งขึ้นหลักซ้าย

ขั้น 3 — ตัวทดตัวสุดท้ายกลายเป็นบิตใหม่ เพราะไม่มีหลักซ้ายรองรับแล้ว ผลลัพธ์จึงยาว 5 บิต

ตอบ: $1011_2 + 0110_2 = 10001_2$

“1011 + 0110 = 10001 (11 + 6 = 17 in decimal).”

WE DO ทำด้วยกันครับ — สไลด์ทิ้งข้อนี้ไว้ให้ลอง

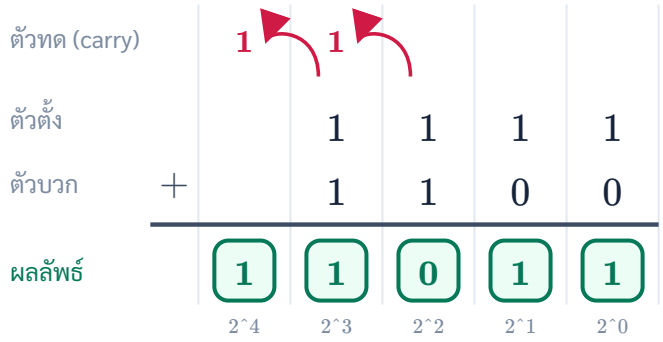
ทำด้วยกัน สไลด์ EES271-01 · “Example: Add 1111 and 1100”

> Add 1111 and 1100.

ข้อนี้พิเศษตรงที่ตัวตั้งเป็น 1 ทุกบิต ลองเดาก่อนไหมครับว่าตัวทดจะวิ่งกี่รอบ

เติมช่องว่างไปพร้อมพี่ครับ (คิดในใจก่อนแล้วค่อยดูเฉลยข้างล่าง):

หลัก	ตัวทดที่รับมา	บิตบน	บิตล่าง	นับ 1 ได้กี่ตัว	เขียน	ทด
2^0	0	1	0	.....	.....	.....
2^1	.....	1	0	.....	.....	.....
2^2	.....	1	1	.....	.....	.....
2^3	.....	1	1	.....	.....	.....



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 15 + 12 = 27

1111 + 1100 (แบบเขียนลงกระดาษสอบ)

```
  1 1
    1 1 1 1
+   1 1 0 0
-----------
  1 1 0 1 1
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	0	1	**1**	**0**

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^1	0	1	0	1	**1**	**0**
2^2	0	1	1	2	**0**	**1**
2^3	1	1	1	3	**1**	**1**

เฉลย WE DO — เติมทีละช่องพร้อมเหตุผล

หลัก 2^0 : ตัวทดเข้า 0 · บิต $1 + 0$ → นับ 1 ได้ **1 ตัว** (คี่) → **เขียน 1** · ไม่ถึง 2 → **ทด 0** เพราะยังไม่มี 1 สองตัวมาชนกันครับ

หลัก 2^1 : ตัวทดเข้า **0** (มาจากหลักก่อนหน้า) · บิต $1 + 0$ → นับได้ **1 ตัว** → **เขียน 1** · **ทด 0**

หลัก 2^2 : ตัวทดเข้า **0** · บิต $1 + 1$ → นับได้ **2 ตัว** (คู่) → **เขียน 0** · ถึง 2 แล้ว → **ทด 1** นี่คือจุดแรกที่ตัวทดเกิดขึ้นครับ

หลัก 2^3 : ตัวทดเข้า **1** · บิต $1 + 1$ → นับได้ **3 ตัว** (คี่) → **เขียน 1** · เกิน 2 → **ทด 1** ตัวทดตัวนี้ไม่มีหลักซ้ายรองรับ จึงออกไปเป็นบิตที่ห้า

ตอบ: $1111_2 + 1100_2 = 11011_2$ — ตรวจฐานสิบ $15 + 12 = 27$ และ $11011_2 = 16 + 8 + 2 + 1 = 27$ ตรงกันครับ

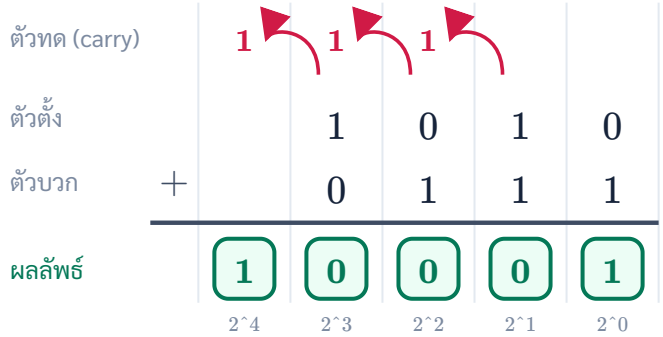
“1111 + 1100 = 11011 (15 + 12 = 27).”

YOU DO ลองเองครับ ปิดเฉลยไว้ก่อน

Y2.1

> Add the binary numbers 1010 and 0111. Show the carry into every column.

บวก $1010_2 + 0111_2$ แล้วเขียนตัวทดของทุกหลักออกมาด้วยครับ



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 10 + 7 = 17

1010 + 0111 (แบบเขียนลงกระดาษสอบ)

```
      1 1 1
        1 0 1 0
    +   0 1 1 1
  -------------
    1 0 0 0 1
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	0	1	1	**1**	**0**
2^1	0	1	1	2	**0**	**1**
2^2	1	0	1	2	**0**	**1**
2^3	1	1	0	2	**0**	**1**

เฉลย Y2.1

ขั้น 1 — หลัก 2^0: $0 + 1 = 1$ ไม่มีตัวทดเข้า เขียน 1 ทด 0 เพราะมี 1 ตัวเดียวครับ

ขั้น 2 — หลัก 2^1: $1 + 1 = 10$ เขียน 0 ทด 1 เพราะ 1 ชน 1 พอดี

ขั้น 3 — หลัก 2^2: ตัวทด 1 + 0 + 1 = 10 เขียน 0 ทด 1 อีกครั้ง ตรงนี้คนลืมตัวทดบ่อยแล้วตอบ 1

ขั้น 4 — หลัก 2^3: ตัวทด 1 + 1 + 0 = 10 เขียน 0 ทด 1 ตัวทดตัวสุดท้ายกลายเป็นบิตที่ห้า

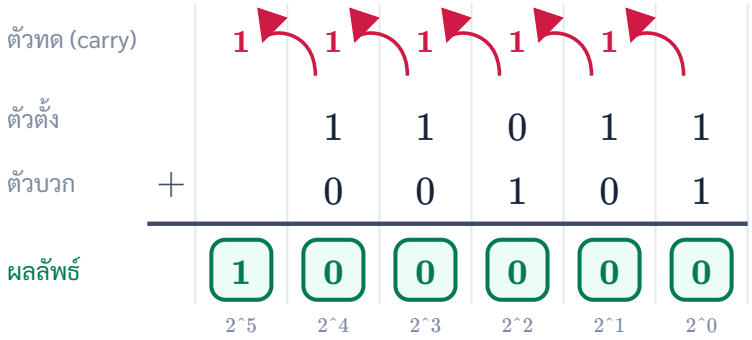
ตอบ: $1010_2 + 0111_2 = 10001_2$ — ตรวจ $10 + 7 = 17 = 10001_2$ ครับ

“1010 + 0111 = 10001; the carries into the columns are 0, 1, 1, 1 from LSB to MSB.”

Y2.2

> Add the binary numbers 11011 and 101. Write the answer in 6 bits.

ระวังครับ ข้อนี้เลขสองตัว**ยาวไม่เท่ากัน** ต้องเติมศูนย์ให้ครบก่อนตั้งหลัก



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 27 + 5 = 32

11011 + 00101 (แบบเขียนลงกระดาษสอบ)

```
  1 1 1 1 1
    1 1 0 1 1
+   0 0 1 0 1
-----------------
  1 0 0 0 0 0
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	1	2	**0**	**1**
2^1	1	1	0	2	**0**	**1**
2^2	1	0	1	2	**0**	**1**
2^3	1	1	0	2	**0**	**1**
2^4	1	1	0	2	**0**	**1**

เฉลย Y2.2

ขั้น 1 — เติมศูนย์นำหน้าให้ยาวเท่ากันก่อน 101_2 กลายเป็น 00101_2 เพราะการเติม 0 ข้างหน้า**ไม่เปลี่ยนค่า** แต่ทำให้ตั้งหลักไม่เหลื่อมกันครับ

ขั้น 2 — หลัก 2^0: $1 + 1 = 10$ เขียน 0 ทด 1

ขั้น 3 — หลัก 2^1: ตัวทด 1 + 1 + 0 = 10 เขียน 0 ทด 1

ขั้น 4 — หลัก 2^2: ตัวทด 1 + 0 + 1 = 10 เขียน 0 ทด 1

ขั้น 5 — หลัก 2^3: ตัวทด 1 + 1 + 0 = 10 เขียน 0 ทด 1 — ตัวทดวิ่งติดกันสี่รอบเลยครับ

ขั้น 6 — หลัก 2^4: ตัวทด 1 + 1 + 0 = 10 เขียน 0 ทด 1 → ตัวทดสุดท้ายเป็นบิตที่หก

ตอบ: $11011_2 + 101_2 = 100000_2$ — ตรวจ $27 + 5 = 32$ และ $100000_2 = 32$ ตรงกันครับ

“11011 + 101 = 100000 (27 + 5 = 32).”

เทียบของเก่า · EP01

สังเกตไหมครับว่า $100000_2 = 32 = 2^5$ พอดี — ตรงกับที่ EP01 สอนว่า**บิต 1 ตัวเดียวที่หลัก 2^k มีค่า 2^k** เพราะฉะนั้นถ้าคำตอบออกมาเป็น 1 ตามด้วย 0 ล้วน น้องอ่านค่าได้ทันทีโดยไม่ต้องบวกน้ำหนักทีละหลักเลยครับ

3 ลบเลขฐานสอง — เรื่องของการยืม

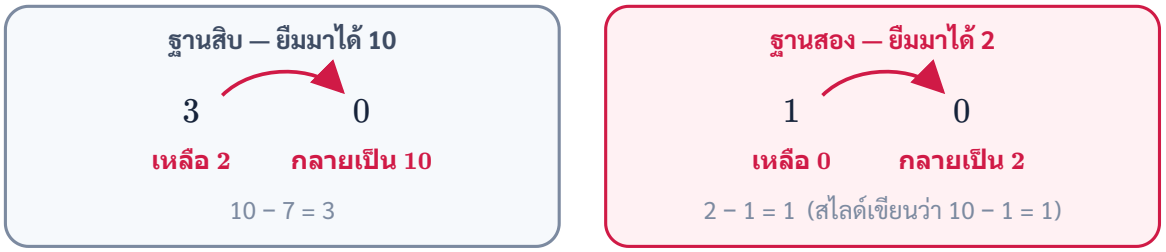
ตะขอเกี่ยว

ตอนที่แล้วน้องบวกเลขฐานสองได้แล้ว และเห็นว่าตัวทดวิ่งจากขวาไปซ้ายใช่ไหมครับ สมมติว่าถ้าโจทย์กลับด้านเป็น $1101_2 - 0110_2$ ละครับ — หลักที่เจอ $0 - 1$ จะทำยังไง ในเมื่อฐานสองไม่มีเลข -1 ให้เขียนลงในหลักครับ

ของใกล้ตัว: ยืมเหรียญจากลิ้นชักข้าง ๆ

ในฐานสิบ ถ้าน้องเจอ $30 - 7$ น้องก็ยืม 1 จากหลักสิบมาเป็น 10 หน่วยใช่ไหมครับ หลักสิบเลยเหลือ 2 แล้วหลักหน่วยกลายเป็น 10 พอลบ 7 ได้ 3

ฐานสองก็เหมือนกันเป๊ะ ต่างกันแค่ว่า**ยืมมาแล้วได้ 2 ไม่ใช่ 10** ครับ เพราะหลักซ้ายมีน้ำหนักเป็นสองเท่าของหลักขวาเสมอ



สไลด์เขียน $10 - 1 = 1$ นะครับ ตัว “10” ตรงนั้นคือ**เลขฐานสองที่มีค่า 2** ไม่ใช่สิบ

DEFINITION · Binary subtraction rules

0 − 0 = 0

1 − 1 = 0

1 − 0 = 1

10 − 1 = 1 (0 − 1 with a borrow of 1)

พี่แปลว่า: มีแค่กรณีเดียวที่ต้องยืม คือ**ตอนที่ 0 ต้องลบ 1** ครับ พอยืมแล้ว 0 จะกลายเป็น 2 (เขียนเป็นฐานสองว่า 10) แล้ว $2 - 1 = 1$ **และหลักซ้ายที่ให้ยืมจะถูกหักไป 1 ทันที** — จุดนี้แหละที่คนลืมกันมากที่สุดครับ

กับดักที่ 1 · ลืมหักหลักที่ให้ยืม

การยืมมี**สองผล**เสมอครับ ① หลักที่ยืมได้ +2 ② **หลักที่ให้ยืมโดนหัก −1** คนส่วนใหญ่ทำข้อ ① แล้วลืม ② ทำให้คำตอบใหญ่เกินจริง

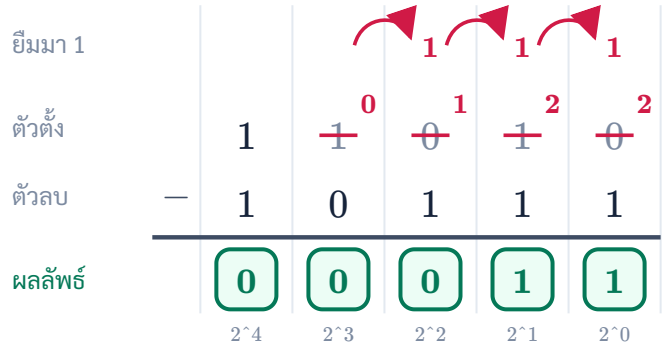
ในชีทนี้พี่จะเขียน**ค่าที่ใช้จริง**เป็นตัวเลขสีแดงกำกับไว้บนบิตทุกตัวที่โดนแก้ค่า น้องจะได้ไม่มีทางลืมครับ

กับดักที่ 2 · ยืมต่อกันเป็นทอด ๆ

ถ้าหลักซ้ายที่จะให้ยืมเป็น **0** อยู่แล้ว มันก็ต้องไปยืมจากหลักซ้ายของมันอีกทีครับ เกิดเป็น**ลูกโซ่การยืม** เช่น $10000_2 - 1_2$ ต้องยืมติดกันถึง 4 รอบ (เดี๋ยวเจอในข้อ Y3.2 ครับ)

ไล่ให้ดูทุกหลัก — 11010 − 10111

ข้อนี้พี่เลือกมาเพราะมีการยืมติดกันหลายหลัก น้องดู**เลข 1 สีแดงบนหัว** ว่าหลักไหนยืมบ้าง และดู**เลขสีแดงข้างบิต** ว่าหลังยืมแล้วบิตนั้นเหลือค่าเท่าไรครับ



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 26 − 23 = 3

11010 − 10111 (แบบเขียนลงกระดาษสอบ)

```
      1 1 1
    1 1 0 1 0
  - 1 0 1 1 1
-------------
    0 0 0 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	ยืมมา +2	—	**2**	1	2 − 1 = **1**
2^1	1	ยืมมา +2	จ่ายไป −1	**2**	1	2 − 1 = **1**
2^2	0	ยืมมา +2	จ่ายไป −1	**1**	1	1 − 1 = **0**
2^3	1	—	จ่ายไป −1	**0**	0	0 − 0 = **0**
2^4	1	—	—	**1**	1	1 − 1 = **0**

I DO พี่ทำโจทย์จากสไลด์ให้ดูครับ

ตัวอย่างที่พี่ทำ — สไลด์ EES271-01 · Binary Subtraction Example

> Subtract 0110 from 1101.

แปล: เอา 0110_2 **ไปลบออกจาก** 1101_2 ครับ — ระวังลำดับ ภาษาอังกฤษ “subtract A from B” แปลว่า $B - A$ ไม่ใช่ $A - B$

1

$$\begin{array}{rr} & 1\,1\,0\,1 \\ - & 0\,1\,1\,0 \end{array}$$

อ่านโจทย์ให้ถูกก่อน แล้วตั้งหลัก

“Subtract 0110 from 1101” = $1101 - 0110$ ครับ **ตัวที่อยู่หลังคำว่า from คือตัวตั้ง** เสมอ

2

$1 - 0 = 1$

ไม่ต้องยืม

หลัก 2^0 : $1 - 0 = 1$

เข้ากฎบรรทัดที่สาม ลบได้ตรง ๆ เพราะตัวตั้งใหญ่กว่าตัวลบครับ

3

$0 - 1 \rightarrow$ ยืม

$2 - 1 = 1$

หลัก 2^2 โดนหัก −1

หลัก 2^1 : $0 - 1$ ต้องยืม

ยืมจากหลัก 2^2 มา 1 → หลักนี้กลายเป็น 2 แล้ว $2 - 1 = 1$ ครับ **และหลัก 2^2 ที่เดิมเป็น 0 จะติดหนี้ทันที**

4

0 ให้ยืมไป $\rightarrow -1$

ยืมต่อ: $2 - 1 = 1$

1 − 1 = 0

หลัก 2^2 : ต้องยืมต่ออีกทอด

หลักนี้เป็น 0 อยู่แล้วแต่ต้องจ่ายให้หลักขวา จึงต้องไปยืมจากหลัก 2^3 ได้ $0 + 2 - 1 = 1$ แล้วเอาไปลบตัวลบ 1 เหลือ **0** ครับ

5

$1 - 1 - 0 = 0$

หลักนี้ให้ยืมไป 1 เหลือ 0

หลัก 2^3 : ให้ยืมไปแล้วเหลือ 0

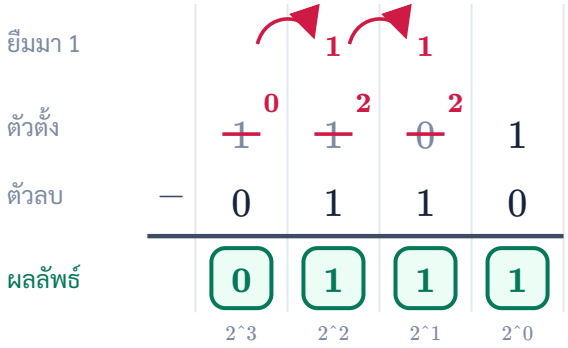
เดิมเป็น 1 แต่ให้หลัก 2^2 ยืมไป 1 เหลือ 0 แล้วลบตัวลบ 0 ได้ **0** ครับ

6

0 1 1 1

อ่านผลลัพธ์

ได้ $0111_2 = 7$ ครับ ตรวจฐานสิบ $13 - 6 = 7$ ตรงกันพอดี



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 13 − 6 = 7

1101 − 0110 (แบบเขียนลงกระดาษสอบ)

```
    1 1
  1 1 0 1
- 0 1 1 0
---------
  0 1 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	1	—	—	**1**	0	1 − 0 = **1**
2^1	0	**ยืมมา +2**	—	**2**	1	2 − 1 = **1**
2^2	1	**ยืมมา +2**	**จ่ายไป −1**	**2**	1	2 − 1 = **1**
2^3	1	—	**จ่ายไป −1**	**0**	0	0 − 0 = **0**

คำตอบ I DO

ขั้น 1 — แปลโจทย์ให้ถูกทิศ “subtract 0110 from 1101” คือ $1101 - 0110$ เพราะถ้าสลับทิศจะได้ค่าติดลบ ซึ่งวิธีตั้งหลักธรรมดายังทำไม่ได้ครับ

ขั้น 2 — ไล่ทีละหลักจากขวา หลัก 2^0 ลบได้ตรง ๆ · หลัก 2^1 ต้องยืม · หลัก 2^2 เป็น 0 แต่ต้องจ่ายให้หลักขวา จึงต้อง**ยืมต่ออีกทอด** · หลัก 2^3 ถูกหักไป 1

ขั้น 3 — ตรวจด้วยการบวกกลับ $0111 + 0110 = 1101$ ✓ ตรงกับตัวตั้งเดิม แปลว่าลบถูกครับ

ตอบ: $1101_2 - 0110_2 = 0111_2$

“1101 − 0110 = 0111 (13 − 6 = 7).”

WE DO ทำด้วยกันครับ

ทำด้วยกัน สไลด์ EES271-01 · “Example: Subtract 100 from 111”

> Subtract 100 from 111.

ข้อนี้สไลด์ทิ้งไว้ให้ลองครับ ลองเดาก่อนไหมว่าต้องยืมกี่ครั้ง

		2^2	2^1	2^0
ยืมมา 1				
ตัวตั้ง		1	1	1
ตัวลบ	−	1	0	0
ผลลัพธ์		**0**	**1**	**1**

ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 7 − 4 = 3

111 − 100 (แบบเขียนลงกระดาษสอบ)

```
  1 1 1
- 1 0 0
-------
  0 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	1	—	—	**1**	0	1 − 0 = **1**
2^1	1	—	—	**1**	0	1 − 0 = **1**
2^2	1	—	—	**1**	1	1 − 1 = **0**

เฉลย WE DO — เติมทีละช่องพร้อมเหตุผล

หลัก 2^0 : $1 - 0 = 1$ · **ไม่ต้องยืม** เพราะตัวตั้งเป็น 1 ตัวลบเป็น 0 เข้ากฎบรรทัดที่สามพอดี

หลัก 2^1 : $1 - 0 = 1$ · **ไม่ต้องยืม** ด้วยเหตุผลเดียวกัน

หลัก 2^2 : $1 - 1 = 0$ · **ไม่ต้องยืม** เข้ากฎบรรทัดที่สอง

สรุป: ข้อนี้ไม่มีการยืมเลยแม้แต่ครั้งเดียว เพราะทุกหลักของตัวตั้งมากกว่าหรือเท่ากับตัวลบอยู่แล้วครับ ข้อแบบนี้คือของฟรี อย่าเผลอไปยืมเองนะครับ

ตอบ: $111_2 - 100_2 = 011_2 = 11_2$ — ตรวจ $7 - 4 = 3$ และ $11_2 = 3$ ตรงกันครับ

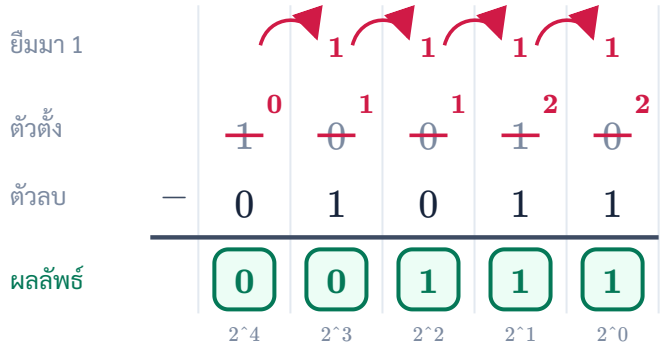
“111 − 100 = 011 (7 − 4 = 3); no borrow is needed in any column.”

YOU DO ลองเองครับ

Y3.1

> Perform the binary subtraction 10010 − 1011. Mark every column that needs a borrow.

ทำ $10010_2 - 1011_2$ แล้วชี้ว่าหลักไหนต้องยืมบ้างครับ



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 18 − 11 = 7

10010 − 01011 (แบบเขียนลงกระดาษสอบ)

```
    1 1 1 1
  1 0 0 1 0
- 0 1 0 1 1
-----------
  0 0 1 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	ยืมมา +2	—	**2**	1	2 − 1 = **1**
2^1	1	ยืมมา +2	จ่ายไป −1	**2**	1	2 − 1 = **1**
2^2	0	ยืมมา +2	จ่ายไป −1	**1**	0	1 − 0 = **1**
2^3	0	ยืมมา +2	จ่ายไป −1	**1**	1	1 − 1 = **0**
2^4	1	—	จ่ายไป −1	**0**	0	0 − 0 = **0**

เฉลย Y3.1

ขั้น 1 — เติมศูนย์ให้ยาวเท่ากัน $1011_2 \rightarrow 01011_2$ เพื่อให้ตั้งหลักได้ 5 หลักเท่ากันครับ

ขั้น 2 — หลัก 2^0: $0 - 1$ ต้องยืม ได้ $2 - 1 = 1$ และหลัก 2^1 ติดหนี้

ขั้น 3 — หลัก 2^1: ตัวตั้งเป็น 1 แต่จ่ายไป 1 เหลือ 0 แล้วต้องลบ 1 → ต้องยืมอีก ได้ $0 + 2 - 1 = 1$ และหลัก 2^2 ติดหนี้

ขั้น 4 — หลัก 2^2: ตัวตั้งเป็น 0 จ่ายไป 1 → ต้องยืมอีก ได้ $0 + 2 - 1 = 1$ แล้วลบตัวลบ 0 เหลือ **1**

ขั้น 5 — หลัก 2^3: ตัวตั้งเป็น 0 จ่ายไป 1 → ต้องยืมอีก ได้ $0 + 2 - 1 = 1$ แล้วลบตัวลบ 1 เหลือ **0**

ขั้น 6 — หลัก 2^4: ตัวตั้งเป็น 1 จ่ายไป 1 เหลือ 0 ลบตัวลบ 0 ได้ **0**

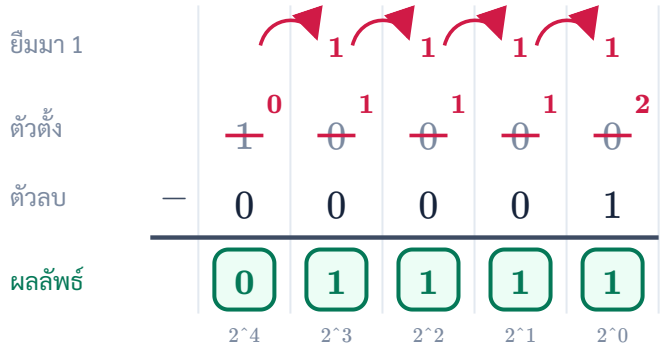
ตอบ: $10010_2 - 1011_2 = 00111_2 = 111_2$ — ตรวจ $18 - 11 = 7$ ครับ หลักที่ต้องยืมคือ $2^0, 2^1, 2^2, 2^3$ (สี่หลักติด)

"10010 − 1011 = 111 (18 − 11 = 7); borrows occur in the four lowest columns."

Y3.2

> Perform the binary subtraction 10000 − 1.

ข้อนี้คือลูกโซ่การยืมที่ยาวที่สุดครับ ลองดูว่าเกิดอะไรขึ้น



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 16 − 1 = 15

10000 − 00001 (แบบเขียนลงกระดาษสอบ)

```
    1 1 1 1
  1 0 0 0 0
- 0 0 0 0 1
-----------
  0 1 1 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	**ยืมมา +2**	—	**2**	1	2 − 1 = **1**

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^1	0	ยืมมา +2	จ่ายไป −1	**1**	0	1 − 0 = **1**
2^2	0	ยืมมา +2	จ่ายไป −1	**1**	0	1 − 0 = **1**
2^3	0	ยืมมา +2	จ่ายไป −1	**1**	0	1 − 0 = **1**
2^4	1	—	จ่ายไป −1	**0**	0	0 − 0 = **0**

เฉลย Y3.2

ขั้น 1 — หลัก 2^0: $0 - 1$ **ต้องยืม** แต่หลัก 2^1 ก็เป็น 0 ยืมไม่ได้ทันที ต้องไล่ไปทางซ้ายจนกว่าจะเจอ 1 ครับ

ขั้น 2 — หลัก $2^1, 2^2, 2^3$ เป็น 0 หมด แต่ละหลัก**ยืมจากซ้ายมา 2 แล้วจ่ายให้ขวาไป 1** เหลือค่าใช้จริงเป็น 1 ทั้งสามหลัก ลบตัวลบ 0 ก็ได้ 1

ขั้น 3 — หลัก 2^4 เป็น 1 จ่ายให้ยืมไป 1 เหลือ 0 ลูกโซ่จบตรงนี้ครับ

ขั้น 4 — อ่านคำตอบ ได้ $01111_2 = 1111_2 = 15$

ตอบ: $10000_2 - 1_2 = 1111_2$ — ตรวจ $16 - 1 = 15$ ครับ
ทางลัดจำไว้เลย: $2^n - 1$ จะได้เลข 1 เรียงกัน n ตัวเสมอ
“10000 − 1 = 1111 (16 − 1 = 15); the borrow ripples through four columns.”

ทางลัดตรวจงานลบ

เอาคำตอบไปบวกกับตัวลบ ต้องได้ตัวตั้งเดิมเป๊ะครับ — วิธีนี้ใช้ได้ทุกข้อและใช้เวลาแค่ไม่กี่วินาที เพราะน้องบวกฐานสองเป็นแล้วจากตอนที่ 2 ถ้าบวกกลับแล้วไม่ตรง แปลว่ามีหลักไหนสักหลักที่ลืมหักตัวยืมครับ

4 คูณเลขฐานสอง — ลอกกับใส่ศูนย์เท่านั้น

ตะขอเกี่ยว

สองตอนที่แล้วน้องบวกและลบฐานสองได้แล้วครับ สมมติว่าถ้าโจทย์ให้ $1011_2 \times 101_2$ ละครับ — น้องต้องท่องสูตรคูณฐานสองไหมครับ ข่าวดีคือ**ไม่ต้องท่องอะไรเลย** เพราะสูตรคูณฐานสองมีแค่ 4 บรรทัด และผลลัพธ์เป็นได้แค่ 0 กับ 1 ครับ

ของใกล้ตัว: สวิตช์ประตูกับหลอดไฟ

คิดว่าตัวคูณคือ**สวิตช์** และตัวตั้งคือ**ของที่จะปล่อยผ่าน** ครับ — สวิตช์เปิด (บิตเป็น 1) ของก็ผ่านไปทั้งชุด · สวิตช์ปิด (บิตเป็น 0) ก็ไม่มีอะไรผ่านเลย ได้ศูนย์ล้วน

เพราะฉะนั้นการคูณฐานสองจึงไม่มีการ “คิดเลข” เลยครับ มีแค่ **ลอกลงมา** กับ **ใส่ศูนย์** เท่านั้น งานคิดจริง ๆ อยู่ตอนบวกแถวย่อยรวมกันตอนท้าย ซึ่งน้องทำเป็นแล้วจากตอนที่ 2 ครับ

DEFINITION · Binary multiplication rules

0 × 0 = 0 · 0 × 1 = 0 · 1 × 0 = 0 · 1 × 1 = 1

สไลด์สรุปวิธีทำไว้สามบรรทัด พี่แปลให้ครับ:

① **Form one partial product for each multiplier bit (LSB first)** — ทำแถวย่อยหนึ่งแถวต่อหนึ่งบิตของตัวคูณ เริ่มจากบิตขวาสุด

② **a copy of the multiplicand if the bit is 1, or all zeros if the bit is 0** — บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ล้วน

③ **Shift each partial product one column to the left, then add all the partial products** — เลื่อนแถวย่อยไปทางซ้ายทีละหนึ่งหลัก แล้วบวกทุกแถวเข้าด้วยกัน

บิตตัวคูณเป็น 1 = สวิตช์เปิด

1 0 1 1
|
1 0 1 1

ลอกตัวตั้งลงมาทั้งชุด

บิตตัวคูณเป็น 0 = สวิตช์ปิด

1 0 1 1
|
0 0 0 0

ใส่ศูนย์ล้วน ไม่มีอะไรผ่าน

ทั้งการคูณมีแค่สองท่านี้ครับ ไม่มีท่าที่สาม

ทางลัด · คูณด้วยกำลังของสอง = เลื่อนซ้าย

$\times 2$ คือ**เติม 0 ข้างท้าย 1 ตัว** · $\times 4$ คือเติม 0 สองตัว · $\times 8$ คือเติม 0 สามตัว ครับ เหมือนฐานสิบที่ $\times 10$ แล้วเติมศูนย์ท้ายเลย

เช่น $1011_2 \times 100_2 = 101100_2$ (คือ $11 \times 4 = 44$) — ตอบได้ทันทีโดยไม่ต้องตั้งคูณเลยครับ ข้อสอบชอบแอบใส่มาข้อหนึ่ง

I DO พี่ทำโจทย์จากสไลด์ให้ดูครับ

ตัวอย่างที่พี่ทำ — สไลด์ EES271-01 · Binary Multiplication Example

> Multiply 1011 by 101.

แปล: จงคูณ 1011_2 ด้วย 101_2 ครับ — ตัวคูณมี 3 บิต แปลว่าจะมี**แถวย่อย 3 แถว**พอดี

1	ตัวคูณมี 3 บิต **จะได้ 3 แถวย่อย**	**นับก่อนว่าจะมีกี่แถวย่อย** ทำแบบนี้ก่อนเสมอครับ จะได้รู้ว่าเขียนครบหรือยัง — **จำนวนแถวย่อย = จำนวนบิตของตัวคูณ**
2	บิตขวาสุดของ 101 = 1 **1 0 1 1**	**แถวที่ 1 — บิต 2^0 ของตัวคูณเป็น 1** ลอกตัวตั้งลงมาตรง ๆ ไม่ต้องเลื่อน เพราะเป็นหลัก 2^0 ครับ
3	บิตกลางของ 101 = 0 **0 0 0 0 0**	**แถวที่ 2 — บิต 2^1 เป็น 0** ใส่ศูนย์ล้วน แล้ว**เลื่อนซ้าย 1 หลัก** ครับ ห้ามข้ามแถวนี้เด็ดขาด เพราะมันกำหนดตำแหน่งของแถวถัดไป
4	บิตซ้ายสุดของ 101 = 1 **1 0 1 1 0 0**	**แถวที่ 3 — บิต 2^2 เป็น 1** ลอกตัวตั้งลงมาแล้ว**เลื่อนซ้าย 2 หลัก** (เติม 0 ท้าย 2 ตัว) เพราะเป็นหลัก 2^2 ครับ
5	**1 1 0 1 1 1**	**บวกทุกแถวย่อยเข้าด้วยกัน** ใช้กติกาบวกจากตอนที่ 2 เลยครับ ได้ $110111_2 = 55$ · ตรวจ $11 \times 5 = 55$ ตรงกันพอดี

บิตที่ 0 ของตัวคูณ = 1			1	0	1	1
บิตที่ 1 ของตัวคูณ = 0		0	0	0	0	0
บิตที่ 2 ของตัวคูณ = 1	1	0	1	1	0	0
บวกทุกแถวรวมกัน	**1**	**1**	**0**	**1**	**1**	**1**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 11 × 5 = 55

1011 × 101 (แบบเขียนลงกระดาษสอบ)

```
        1 0 1 1
    x     1 0 1
  -----------------
        1 0 1 1
      0 0 0 0 0
    1 0 1 1 0 0
  -----------------
    1 1 0 1 1 1
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	1	**1011**	0 หลัก	11
2^1	0	**00000**	1 หลัก	0
2^2	1	**101100**	2 หลัก	44
บวกทุกแถวย่อยเข้าด้วยกัน				**55**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
      1
    0 0 1 0 1 1
+   1 0 1 1 0 0
  -----------------
  0 1 1 0 1 1 1
```

คำตอบ I DO

ขั้น 1 — นับบิตตัวคูณก่อน 101_2 มี 3 บิต จึงต้องมีแถวย่อย 3 แถวพอดี ถ้าเขียนได้ 2 แถวแปลว่าลืมแถวศูนย์ครับ

ขั้น 2 — สร้างแถวย่อยจากขวาไปซ้าย บิต 1 → ลอก 1011 · บิต 0 → ใส่ 0000 · บิต 1 → ลอก 1011 พร้อมเลื่อนซ้าย 0, 1, 2 หลักตามลำดับ เพราะแต่ละบิตมีน้ำหนัก $2^0, 2^1, 2^2$

ขั้น 3 — บวกทุกแถวย่อย แถวศูนย์บวกแล้วไม่เปลี่ยนอะไร เหลือแค่ $1011 + 101100 = 110111$

ตอบ: $1011_2 \times 101_2 = 110111_2$ — ตรวจ $11 \times 5 = 55 = 110111_2$ ครับ

“1011 x 101 = 110111 (11 x 5 = 55).”

WE DO ทำด้วยกันครับ

ทำด้วยกัน สไลด์ EES271-01 · “Example: Multiply 1101 * 1010”

> Multiply 1101 by 1010.

คราวนี้ตัวคูณมี 4 บิต และมีศูนย์ 2 ตัว ลองเดาก่อนว่าจะเหลือแถวที่ต้องบวกจริงกี่แถวครับ

บิตที่ 0 ของตัวคูณ = 0				0	0	0	0
บิตที่ 1 ของตัวคูณ = 1			1	1	0	1	0
บิตที่ 2 ของตัวคูณ = 0		0	0	0	0	0	0
บิตที่ 3 ของตัวคูณ = 1	1	1	0	1	0	0	0
บวกทุกแถวรวมกัน	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 13 × 10 = 130

1101 × 1010 (แบบเขียนลงกระดาษสอบ)

```
          1 1 0 1
    x     1 0 1 0
-----------------
          0 0 0 0
        1 1 0 1 0
      0 0 0 0 0 0
    1 1 0 1 0 0 0
-----------------
  1 0 0 0 0 0 1 0
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	0	**0000**	0 หลัก	0

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^1	1	**11010**	1 หลัก	26
2^2	0	**000000**	2 หลัก	0
2^3	1	**1101000**	3 หลัก	104
บวกทุกแถวย่อยเข้าด้วยกัน				**130**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
  1 1 1 1
    0 0 1 1 0 1 0
+   1 1 0 1 0 0 0
  ---------------
  1 0 0 0 0 0 1 0
```

เฉลย WE DO — ทีละแถวพร้อมเหตุผล

แถวที่ 1 (บิต 2^0 = 0) → ใส่ศูนย์ล้วน **0000** ไม่เลื่อน — เพราะบิตเป็น 0 สวิตช์ปิด ไม่มีอะไรผ่านครับ

แถวที่ 2 (บิต 2^1 = 1) → ลอก **1101** แล้วเลื่อนซ้าย 1 หลัก ได้ **11010** — เพราะบิตนี้มีน้ำหนัก 2

แถวที่ 3 (บิต 2^2 = 0) → ใส่ศูนย์ แล้วเลื่อนซ้าย 2 หลัก ได้ **000000**

แถวที่ 4 (บิต 2^3 = 1) → ลอก **1101** แล้วเลื่อนซ้าย 3 หลัก ได้ **1101000** — เพราะบิตนี้มีน้ำหนัก 8

บวกรวม เหลือแค่สองแถวที่ไม่เป็นศูนย์: $11010 + 1101000 = 10000010$

ตอบ: $1101_2 \times 1010_2 = 10000010_2$ — ตรวจ $13 \times 10 = 130$ และ $10000010_2 = 128 + 2 = 130$ ครับ

“1101 × 1010 = 10000010 (13 × 10 = 130).”

YOU DO ลองเองครับ

Y4.1

> Perform the binary multiplication 111 × 11. Show every partial product.

คูณ $111_2 \times 11_2$ แล้วโชว์แถวย่อยทุกแถวครับ

บิตที่ 0 ของตัวคูณ = 1		1	1	1
บิตที่ 1 ของตัวคูณ = 1	1	1	1	0
บวกทุกแถวรวมกัน	**1**	**0**	**1**	**0**	**1**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 7 × 3 = 21

111 × 11 (แบบเขียนลงกระดาษสอบ)

```
      1 1 1
  x     1 1
-----------
      1 1 1
    1 1 1 0
-----------
  1 0 1 0 1
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	1	**111**	0 หลัก	7
2^1	1	**1110**	1 หลัก	14
บวกทุกแถวย่อยเข้าด้วยกัน				**21**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
  1 1 1
    0 1 1 1
+   1 1 1 0
-----------
  1 0 1 0 1
```

เฉลย Y4.1

ขั้น 1 — ตัวคูณ 11_2 **มี 2 บิต** จึงต้องได้แถวย่อย 2 แถวครับ

ขั้น 2 — แถวที่ 1 (บิต 2^0 **= 1)** ลอก **111** ลงมา ไม่เลื่อน

ขั้น 3 — แถวที่ 2 (บิต 2^1 **= 1)** ลอก **111** แล้วเลื่อนซ้าย 1 หลัก ได้ **1110**

ขั้น 4 — บวกสองแถว $111 + 1110$: หลัก 2^0 ได้ 1 · หลัก 2^1 $1 + 1$ เขียน 0 ทด 1 · หลัก 2^2 ตัวทด 1 + 1 + 1 เขียน 1 ทด 1 · หลัก 2^3 ตัวทด 1 + 1 เขียน 0 ทด 1 → ได้ **10101**

ตอบ: $111_2 \times 11_2 = 10101_2$ — ตรวจ $7 \times 3 = 21$ และ $10101_2 = 16 + 4 + 1 = 21$ ครับ

“111 × 11 = 10101 (7 × 3 = 21).”

Y4.2

> Perform the binary multiplication 1010 × 110.

ข้อนี้มีศูนย์ทั้งในตัวตั้งและตัวคูณ ระวังอย่าลืมแถวศูนย์นะครับ

บิตที่ 0 ของตัวคูณ = 0			0	0	0	0
บิตที่ 1 ของตัวคูณ = 1		1	0	1	0	0
บิตที่ 2 ของตัวคูณ = 1	1	0	1	0	0	0
บวกทุกแถวรวมกัน	**1**	**1**	**1**	**1**	**0**	**0**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 10 × 6 = 60

1010 × 110 (แบบเขียนลงกระดาษสอบ)

```
          1 0 1 0
    x       1 1 0
  ---------------
          0 0 0 0
        1 0 1 0 0
      1 0 1 0 0 0
  ---------------
      1 1 1 1 0 0
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	0	**0000**	0 หลัก	0
2^1	1	**10100**	1 หลัก	20
2^2	1	**101000**	2 หลัก	40
บวกทุกแถวย่อยเข้าด้วยกัน				**60**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
    0 1 0 1 0 0
+   1 0 1 0 0 0
-------------------
  0 1 1 1 1 0 0
```

เฉลย Y4.2

ขั้น 1 — ตัวคูณ 110_2 **มี 3 บิต** → ต้องได้แถวย่อย 3 แถว

ขั้น 2 — แถวที่ 1 (บิต 2^0 **= 0)** ใส่ศูนย์ล้วน **0000** — แถวนี้บวกไปก็ไม่เปลี่ยนค่า แต่**ห้ามลืมนับ** ไม่งั้นแถวถัดไปจะเลื่อนผิดตำแหน่งครับ

ขั้น 3 — แถวที่ 2 (บิต 2^1 **= 1)** ลอก **1010** เลื่อนซ้าย 1 หลัก ได้ **10100**

ขั้น 4 — แถวที่ 3 (บิต 2^2 **= 1)** ลอก **1010** เลื่อนซ้าย 2 หลัก ได้ **101000**

ขั้น 5 — บวกรวม $10100 + 101000 = 111100$

ตอบ: $1010_2 \times 110_2 = 111100_2$ — ตรวจ $10 \times 6 = 60$ และ $111100_2 = 32 + 16 + 8 + 4 = 60$ ครับ *“1010 × 110 = 111100 (10 × 6 = 60).”*

กับดักของการคูณ

① **ลืมเลื่อนแถว** — แถวที่ k (นับจาก 0) ต้องเลื่อนซ้าย k หลักเสมอ ถ้าเขียนชิดขวาหมดจะได้คำตอบเล็กเกินจริง

② **ข้ามแถวที่เป็นศูนย์** — บางคนเห็นว่าเป็นศูนย์เลยไม่เขียน แล้วเลื่อนแถวถัดไปผิดไปหนึ่งหลัก วิธีกันคือ **เขียนให้ครบทุกแถวเสมอ** ครับ

③ **ตรวจความยาวคำตอบ** — ตัวตั้ง m บิต คูณตัวคูณ n บิต คำตอบจะยาว **ไม่เกิน** $m + n$ **บิต** ถ้าน้องได้ยาวกว่านั้น แปลว่าบวกแถวย่อยผิดแน่นอนครับ

5 หารยาวฐานสอง — ถามคำถามเดียวซ้ำ ๆ

ตะขอเกี่ยว

การหารยาวในฐานสิบน่ากลัวตรงที่ต้อง**เดาว่าใส่ได้กี่ครั้ง** (จะใส่ 6 หรือ 7 ดี) ใช่ไหมครับ สมมติว่าถ้าเป็นฐานสองละครับ — ข่าวดีมากคือ**คำตอบมีแค่ 0 กับ 1** เพราะฉะนั้นทุกรอบเราถามคำถามเดียวเท่านั้น: **“ลบตัวหารออกได้ไหม”** ได้ใส่ 1 ไม่ได้ใส่ 0 จบครับ

ของใกล้ตัว: หยอดเหรียญเข้าเครื่อง

ลองนึกว่าตัวหารคือ**ราคาของ 1 ชิ้น** แล้วเราค่อย ๆ หยอดเหรียญเข้าไปทีละเหรียญ (คือดึงบิตลงมาทีละบิต) ครับ ทุกครั้งที่หยอด เราถามว่า “เงินในเครื่องพอซื้อของชิ้นหนึ่งหรือยัง”

พอ — ก็**ซื้อ (ลบออก) แล้วจดว่าซื้อได้ 1 ชิ้น** · ยังไม่พอ — ก็**จด 0 แล้วหยอดเหรียญต่อ** ครับ พอหยอดครบทุกเหรียญ เงินที่เหลือในเครื่องคือ**เศษ (remainder)** นั่นเอง

DEFINITION · Binary division (จากสไลด์)

Compare the divisor with the leading bits of the dividend: if the divisor fits (is less than or equal), write 1 in the quotient and subtract; otherwise write 0.

Bring down the next bit of the dividend and repeat until all bits of the dividend are used. Whatever is left at the end is the remainder.

พี่แปลว่า: เอาตัวหารไปเทียบกับบิตหัวแถวของตัวตั้ง — **ถ้าลบออกได้ ใส่ 1 ในผลหารแล้วลบจริง** ถ้าลบไม่ได้ **ใส่ 0** แล้วดึงบิตถัดไปลงมา ทำซ้ำจนบิตหมด สิ่งที่เหลือคือเศษครับ

ทางลัด · หารด้วยกำลังของสอง = เลื่อนขวา

$\div 2$ คือ**ตัดเลขท้าย 1 ตัวทิ้ง** (ตัวที่ตัดทิ้งกลายเป็นเศษ) · $\div 4$ คือตัดท้าย 2 ตัว ครับ

เช่น $1100_2 \div 100_2$ ตัดท้าย 2 ตัวได้ $11_2 = 3$ เศษ 0 ทันที — ตรงกับที่หารยาวได้เป๊ะ แต่เร็วกว่ามาก ใช้เช็คคำตอบได้ดีครับ

I DO พี่ทำโจทย์จากสไลด์ให้ดูครับ

ตัวอย่างที่พี่ทำ — สไลด์ EES271-01 · Binary Division Example

> Divide 1111 by 101.

แปล: จงหาร 1111_2 ด้วย 101_2 ครับ — ตัวหารเป็น 3 บิต แปลว่า**สองรอบแรกจะลบไม่ได้แน่นอน**เพราะบิตที่ดึงลงมายังน้อยกว่า

1	มีอยู่ 1 (=1) $1 < 101 \rightarrow$ ใส่ 0	**รอบที่ 1 — ดึงบิตแรกลงมา ได้** $1_2 = 1$ ตัวหารคือ $101_2 = 5$ ครับ $1 < 5$ จึงลบไม่ได้ **ใส่ 0 ในผลหาร** แล้วดึงบิตถัดไป
2	มีอยู่ 11 (=3) $3 < 5 \rightarrow$ ใส่ 0	**รอบที่ 2 — ดึงลงมาอีกบิต ได้** $11_2 = 3$ ยังน้อยกว่า 5 อยู่ครับ **ใส่ 0 อีกตัว** — ศูนย์สองตัวหน้านี้จะถูกตัดทิ้งตอนเขียนคำตอบ
3	มีอยู่ 111 (=7) $7 \geq 5 \rightarrow$ ใส่ 1 $111 - 101 = 10$	**รอบที่ 3 — ได้** $111_2 = 7$ **ลบได้แล้ว** **ใส่ 1 ในผลหาร แล้วลบจริง** ครับ $111 - 101 = 10_2 = 2$ เหลือ 2 ไว้ทำรอบต่อไป
4	ดึง 1 ลงมา → 101 (=5) $5 \geq 5 \rightarrow$ ใส่ 1 $101 - 101 = 0$	**รอบที่ 4 — ดึงบิตสุดท้าย ได้** $101_2 = 5$ **เท่ากันพอดีก็นับว่าลบได้ครับ** (สไลด์เขียนว่า “is less than or equal”) ใส่ 1 แล้วลบเหลือ 0
5	ผลหาร 0011 → 11 เศษ 0	**อ่านผลหาร แล้วตัดศูนย์นำหน้าทิ้ง** ได้ $11_2 = 3$ เศษ 0 ครับ ตรวจฐานสิบ $15 \div 5 = 3$ พอดี

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 101 (=5) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	1	11 (=3)	ยังลบไม่ได้	**0**	11
3	1	111 (=7)	ลบได้	**1**	10
4	1	101 (=5)	ลบได้	**1**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 15 ÷ 5 = 3 เศษ 0

1111 ÷ 101 (หารยาวแบบเขียนลงกระดาษสอบ)

```
                1 1
              -------
      1 0 1 ) 1 1 1 1
            - 1 0 1
            -------
              1 0 1
            - 1 0 1
            -------
                  0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
11 x 101 = 1111
1111 + 0 = 1111
ฐานสิบ: 3 x 5 + 0 = 15
```

คำตอบ I DO

ขั้น 1 — เขียนตัวหารไว้นอกวงเล็บ ตัวตั้งไว้ใน แล้วไล่บิตจากซ้ายมาขวา เพราะการหารต้องเริ่มจากหลักที่มีน้ำหนักมากที่สุดก่อนครับ (ตรงข้ามกับบวก-ลบ-คูณ)

ขั้น 2 — ทุกรอบถามคำถามเดียว “ลบได้ไหม” ได้บิตผลหาร $0, 0, 1, 1$ ตามลำดับ

ขั้น 3 — ตัดศูนย์นำหน้าออก $0011_2 \rightarrow 11_2$ เพราะศูนย์นำหน้าไม่มีค่า

ตอบ: $1111_2 \div 101_2 = 11_2$ **เศษ** 0 — ตรวจ $11_2 \times 101_2 = 1111_2$ ✓ และฐานสิบ $3 \times 5 = 15$ ครับ

“1111 ÷ 101 = 11 with remainder 0 (15 ÷ 5 = 3).”

WE DO ทำด้วยกันครับ

ทำด้วยกัน สไลด์ EES271-01 · “Example: Divide 1100 by 100”

> Divide 1100 by 100.

ข้อนี้ตัวหารเป็นกำลังของสองพอดี ลองใช้ทางลัด “ตัดท้าย” เช็คคำตอบดูด้วยนะครับ

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 100 (=4) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	1	11 (=3)	ยังลบไม่ได้	**0**	11
3	0	110 (=6)	ลบได้	**1**	10
4	0	100 (=4)	ลบได้	**1**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 12 ÷ 4 = 3 เศษ 0

1100 ÷ 100 (หารยาวแบบเขียนลงกระดาษสอบ)

```
                1 1
              -------
      1 0 0 ) 1 1 0 0
            - 1 0 0
            -------
              1 0 0
            - 1 0 0
            -------
                  0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
11 x 100 = 1100
1100 + 0 = 1100
ฐานสิบ: 3 x 4 + 0 = 12
```

เฉลย WE DO — ทีละรอบพร้อมเหตุผล

รอบที่ 1 : ดึงบิตแรกลงมาได้ $1_2 = 1$ · ตัวหาร $100_2 = 4 \cdot 1 < 4$ → **ใส่ 0** เพราะยังไม่มีอะไรให้ลบครับ

รอบที่ 2 : ดึงลงมาอีก ได้ $11_2 = 3 \cdot 3 < 4$ → **ใส่ 0**

รอบที่ 3 : ดึงลงมาอีก ได้ $110_2 = 6 \cdot 6 \geq 4 \rightarrow$ **ใส่ 1 แล้วลบ** $110 - 100 = 10_2 = 2$

รอบที่ 4 : ดึงบิตสุดท้ายลงมา ได้ $100_2 = 4 \cdot 4 \geq 4 \rightarrow$ **ใส่ 1 แล้วลบ** เหลือ 0

อ่านผลหาร $0011_2 \rightarrow 11_2 = 3$ เศษ 0

ตอบ: $1100_2 \div 100_2 = 11_2$ **เศษ** 0
เช็คด้วยทางลัด: $100_2 = 2^2$ เพราะฉะนั้นแค่ตัดเลขท้าย 2 ตัวของ 1100 ทิ้ง เหลือ 11 ตรงกันเป๊ะครับ
“1100 ÷ 100 = 11 with remainder 0 (12 ÷ 4 = 3).”

YOU DO ลองเองครับ

Y5.1

> Divide 10110 by 11. Give the quotient and the remainder.

ข้อนี้**หารไม่ลงตัว** ครับ ต้องตอบเศษด้วย

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 11 (=3) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	0	10 (=2)	ยังลบไม่ได้	**0**	10
3	1	101 (=5)	ลบได้	**1**	10
4	1	101 (=5)	ลบได้	**1**	10
5	0	100 (=4)	ลบได้	**1**	1

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 22 ÷ 3 = 7 เศษ 1

10110 ÷ 11 (หารยาวแบบเขียนลงกระดาษสอบ)

```
              1 1 1
           ---------
   1 1 ) 1 0 1 1 0
         - 1 1
         -------
           1 0 1
           - 1 1
           -------
             1 0 0
             - 1 1
             -------
                 1
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
111 x 11 = 10101
10101 + 1 = 10110
ฐานสิบ: 7 x 3 + 1 = 22
```

เฉลย Y5.1

รอบที่ 1 : มีอยู่ $1_2 = 1 \cdot$ ตัวหาร $11_2 = 3 \cdot 1 < 3 \rightarrow$ **ใส่ 0**

รอบที่ 2 : ได้ $10_2 = 2 \cdot 2 < 3 \rightarrow$ **ใส่ 0**

รอบที่ 3 : ได้ $101_2 = 5 \cdot 5 \geq 3 \rightarrow$ **ใส่ 1** แล้วลบ $101 - 11 = 10_2 = 2$

รอบที่ 4 : ดึง 1 ลงมาได้ $101_2 = 5 \cdot 5 \geq 3 \rightarrow$ **ใส่ 1** แล้วลบเหลือ $10_2 = 2$

รอบที่ 5 : ดึง 0 ลงมาได้ $100_2 = 4 \cdot 4 \geq 3 \rightarrow$ **ใส่ 1** แล้วลบเหลือ $1_2 = 1$

บิตหมดแล้ว สิ่งที่เหลือคือเศษ = 1_2

ตอบ: $10110_2 \div 11_2 = 111_2$ **เศษ** 1_2 — ตรวจ $111_2 \times 11_2 + 1_2 = 10101_2 + 1_2 = 10110_2$ ✓ และฐานสิบ $7 \times 3 + 1 = 22$ ครับ

“10110 ÷ 11 = 111 remainder 1 (22 ÷ 3 = 7 r 1).”

Y5.2

> Divide 110110 by 110.

ตัวหาร 3 บิต ตัวตั้ง 6 บิต ลองไล่ให้ครบทุกรอบนะครับ

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 110 (=6) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	1	11 (=3)	ยังลบไม่ได้	**0**	11
3	0	110 (=6)	ลบได้	**1**	0
4	1	1 (=1)	ยังลบไม่ได้	**0**	1
5	1	11 (=3)	ยังลบไม่ได้	**0**	11
6	0	110 (=6)	ลบได้	**1**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 54 ÷ 6 = 9 เศษ 0

110110 ÷ 110 (หารยาวแบบเขียนลงกระดาษสอบ)

```
              1 0 0 1
           -----------
   1 1 0 ) 1 1 0 1 1 0
           - 1 1 0
           -------
                 1 1 0
               - 1 1 0
               -------
                     0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
1001 x 110 = 110110
110110 + 0 = 110110
ฐานสิบ: 9 x 6 + 0 = 54
```

เฉลย Y5.2

รอบที่ 1 : มีอยู่ $1_2 = 1$ · ตัวหาร $110_2 = 6$ · ลบไม่ได้ → **ใส่ 0**

รอบที่ 2 : ได้ $11_2 = 3$ · ยังน้อยกว่า 6 → **ใส่ 0**

รอบที่ 3 : ได้ $110_2 = 6$ · เท่ากันพอดีก็ลบได้ → **ใส่ 1** แล้วลบเหลือ 0

รอบที่ 4 : ดึง 1 ลงมาได้ $1_2 = 1$ · ลบไม่ได้ → **ใส่ 0**

รอบที่ 5 : ดึง 1 ลงมาได้ $11_2 = 3$ · ลบไม่ได้ → **ใส่ 0**

รอบที่ 6 : ดึง 0 ลงมาได้ $110_2 = 6$ · ลบได้ → **ใส่ 1** แล้วลบเหลือ 0

ผลหารดิบคือ 001001_2 ตัดศูนย์นำหน้าเหลือ $1001_2 = 9$ เศษ 0

ตอบ: $110110_2 \div 110_2 = 1001_2$ **เศษ** 0 — ตรวจ $54 \div 6 = 9$ ครับ

"110110 ÷ 110 = 1001 with remainder 0 (54 ÷ 6 = 9)."

กับดักของการหาร

① **ลืมใส่ 0 ในรอบที่ลบไม่ได้** — ถ้าไม่ใส่ ผลหารจะสั้นไปหนึ่งบิตและค่าผิดไปครึ่งหนึ่งเลยครับ ให้ท่องว่า **"ทุกบิตของตัวตั้ง = ผลหารหนึ่งบิตเสมอ"**

② **เท่ากันพอดีคิดว่าลบไม่ได้** — สไลด์เขียน "less than or equal" ชัดเจน เท่ากันพอดี**ต้องใส่ 1** ครับ

③ **ลืมตอบเศษ** — ถ้าโจทย์ไม่ได้บอกว่าหารลงตัว ให้ตอบทั้งผลหารและเศษเสมอครับ

เทียบของเก่า · ตอนที่ 2 และ 3

การหารยาวไม่ใช่ทักษะใหม่เลยครับ มันคือ**การลบซ้ำ ๆ** ที่น้องเพิ่งฝึกไปในตอนที่ 3 และการตรวจคำตอบก็คือ**การคูณกับการบวก** จากตอนที่ 2 และ 4 — ครึ่งแรกของ EP นี้จบแล้วครับ ต่อไปเราจะไปเรื่องเลขติดลบกัน

6 คอมพลีเมนต์ — 1's และ 2's complement

ตะขอเกี่ยว

ครึ่งแรกของ EP นี้น้องบวกลบคูณหารฐานสองได้หมดแล้วครับ แต่ทั้งหมดนั้นเป็น**เลขบวกล้วน ๆ**

สมมติว่าถ้าคอมพิวเตอร์ต้องเก็บค่า **−45** ละครับ — ในสายไฟไม่มี “ขีดลบ” ให้ใช้ มีแค่ไฟติดกับไฟดับเท่านั้น เขาแก้ปัญหานี้ด้วยของที่ชื่อว่า **complement** ครับ ตอนนี้เราจะทำความรู้จักมันก่อน แล้วตอนหน้าค่อยเอาไปใช้เก็บเลขติดลบจริง ๆ

ของใกล้ตัว: สวิตช์ที่กลับด้านพร้อมกันทั้งแถว

ลองนึกถึงแถวสวิตช์ไฟ 8 ตัวอีกครั้งครับ ถ้าพี่เดินไป**สับกลับด้านทุกตัวพร้อมกัน** — ตัวที่เปิดกลายเป็นปิด ตัวที่ปิดกลายเป็นเปิด — สิ่งที่ได้ก็คือ **1's complement** ครับ ง่ายแค่นั้นจริง ๆ

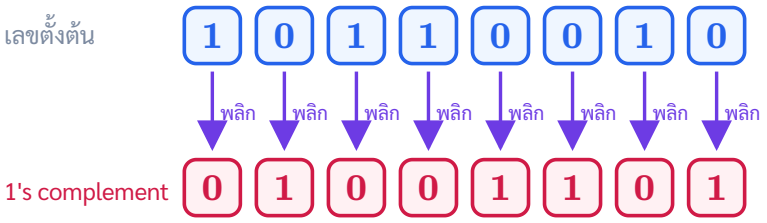
ส่วน **2's complement** คือทำแบบเดิมแล้ว**เติมอีกหนึ่ง**เข้าไปที่สวิตช์ขวาสุด แค่นั้นเองครับ

DEFINITION · 1's complement

The 1's complement of a binary number is found by changing all 1s to 0s and all 0s to 1s.

พี่แปลว่า: **พลิกทุกบิต** ครับ 0 เป็น 1 และ 1 เป็น 0 — ไม่มีการบวก ไม่มีการทด ไม่มีอะไรทั้งสิ้น สไลด์เรียกอีกชื่อว่า `radix-minus-one form` และบอกว่าใช้กันในยุคแรก ๆ ของคอมพิวเตอร์ครับ

ตัวอย่างจากสไลด์ — 1's complement ของ 1011 0010



1's complement คือ “พลิกทุกบิต” 0 เป็น 1 และ 1 เป็น 0 ครับ ไม่มีการบวกอะไรทั้งสิ้น · ตรวจ: 10110010 + 01001101 = 11111111 (ได้ 1 ทุกหลักเสมอ)

ที่มา · ทำไมพลิกบิตแล้วเรียกว่า “เติมเต็ม”

คำว่า complement แปลว่า “ตัวเติมเต็ม” ครับ ลองสังเกตว่า **เลขตั้งต้น + 1's complement ของมัน = 1 ทุกหลักเสมอ**

เช่น $10110010 + 01001101 = 11111111$ ครับ — และ 11111111_2 ก็คือ $2^8 - 1 = 255$ พอดี เพราะฉะนั้นสูตรจริงของมันคือ $1\text{'s comp}(N) = (2^n - 1) - N$ สำหรับเลข n บิต ตรงกับชื่อ radix-minus-one ที่สไลด์เขียนไว้เป๊ะครับ

DEFINITION · 2's complement

2's complement = (1's complement) + 1

An alternative method of finding the 2's complement is as follows:

1. Start at the right with the LSB and write the bits as they are up to and including the first 1.

2. Take the 1's complements of the remaining bits.

พี่แปลวิธีที่สองว่า: **ไล่จากขวาสุดเข้ามา ลอกบิตลงมาเหมือนเดิมไปเรื่อย ๆ จนกว่าจะเจอ 1 ตัวแรก (รวมตัว 1 นั้นด้วย) แล้วบิตที่เหลือทางซ้ายให้พลิกทั้งหมด** ครับ — วิธีนี้เร็วกว่ามากเพราะไม่ต้องบวกเลย ในห้องสอบพี่แนะนำวิธีนี้

ตัวอย่างจากสไลด์ — 2's complement ของ 0100 1000 (สองวิธีต้องได้เท่ากัน)



2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 01001000 + 10111000 = 100000000 (ล้นออกไปนอกกรอบ 8 บิต แล้วเหลือศูนย์หมด)

ทางลัด · ทำไมวิธีที่ 2 ถึงได้ผลเหมือนวิธีที่ 1

ลองคิดตามนะครับ เวลาเราพลิกทุกบิตแล้ว**บวก 1** ตัวทดจะวิ่งจากขวาไปเรื่อย ๆ **ตราบใดที่เจอบิต 1** (เพราะ $1 + 1$ ทดต่อ) และจะ**หยุดทันทีที่เจอบิต 0** ตัวแรก

บิต 0 ตัวแรกในเลขที่พลิกแล้ว ก็คือ**บิต 1 ตัวแรกจากขวาในเลขตั้งต้น** นั่นเองครับ เพราะฉะนั้นบิตตั้งแต่ตรงนั้นลงไปทางขวาจึงกลับมาเหมือนเดิมเป๊ะ ส่วนที่อยู่ซ้ายมือของมันตัวทดไปไม่ถึง เลยค้างเป็นบิตที่พลิกแล้ว — นี่คือเหตุผลว่าทำไมทางลัดถึงใช้ได้เสมอครับ ไม่ใช่เรื่องบังเอิญ

แปลงกลับเป็นเลขจริง (uncomplemented form)

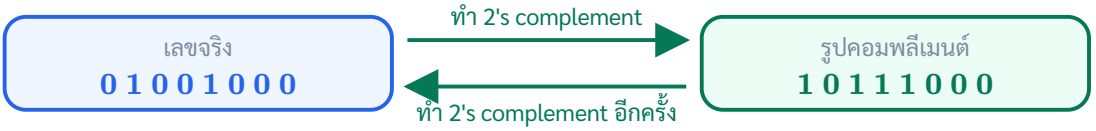
สไลด์บอกวิธีกลับไว้สั้นมากครับ พี่ขยายให้:

กลับจาก 1's complement

For 1's complement: reverse all the bits.
พลิกทุกบิตอีกรอบ ก็ได้เลขเดิมกลับมาครับ เพราะพลิกสองครั้งเท่ากับไม่พลิก

กลับจาก 2's complement

Take the 1's complement of the 2's complement number and add 1 to the LSB.
ทำ 2's complement ซ้ำอีกรอบ ก็ได้เลขเดิมครับ — วิธีเดิมเป๊ะ ไม่ต้องจำสูตรใหม่



ทำสองครั้งแล้วกลับมาที่เดิมเสมอ — จำแค่วิธีเดียวก็ไปทั้งสองทางได้ครับ

คุณสมบัติสำคัญ: $\text{2's comp}(\text{2's comp}(N)) = N$ เสมอครับ

กับดักใหญ่ที่สุดของเรื่องนี้

ต้องรู้ก่อนว่าใช้กี่บิตครับ — คอมพลีเมนต์ของ 110 แบบ 3 บิต กับแบบ 8 บิต**ไม่เหมือนกันเลย**

3 บิต: 1's comp ของ 110	**001**	พลิกแค่ 3 ตัวที่มีอยู่
8 บิต: 1's comp ของ 00000110	**11111001**	ต้องเติมศูนย์นำหน้าให้ครบ 8 ก่อนพลิก

กติกาในข้อสอบ: ถ้าโจทย์ไม่ได้บอกจำนวนบิต ให้ใช้**ความยาวที่โจทย์เขียนมาเป๊ะ ๆ** (เช่น Assignment ข้อ 8a เขียนมาว่า 110 ก็ตอบ 3 บิต) แต่ถ้าโจทย์บอกว่า “as an 8-bit number” ต้องเติมให้ครบ 8 เสมอครับ

I DO พี่ทำให้ดูครับ

ตัวอย่างที่พี่ทำ — พี่แต่งเอง (ปูทางไป Assignment ข้อ 8–9)

> Determine the 1's complement and the 2's complement of 10110010.

หาทั้ง 1's และ 2's complement ของเลขตัวเดียวกันครับ จะได้เห็นว่าต่างกันแค่ 1

1 — นับบิต: 8 บิต
คำตอบต้องยาว 8 บิตเท่ากัน

นับจำนวนบิตก่อนเสมอ
เพราะคอมพลีเมนต์**ต้องยาวเท่าเดิมเป๊ะ** ห้ามตัดศูนย์นำหน้าทิ้งครับ

2

$$10110010$$
$$01001101$$

1's complement — พลิกทุกบิต
ไล่ทีละตัวเลยครับ 1→0, 0→1, 1→0, 1→0, 0→1, 0→1, 1→0, 0→1 ได้ **01001101**

3

$$01001101$$
$$+\,1$$
$$01001110$$

2's complement วิธีที่ 1 — บวก 1 ต่อ
บิตขวาสุดเป็น 1 บวก 1 ได้ 0 ทด 1 · บิตถัดไปเป็น 0 บวกตัวทด 1 ได้ 1 หยุด → **01001110**

4

$$10110010$$
$$01001110$$

2's complement วิธีที่ 2 — ทางลัด
ไล่จากขวา: บิตแรกเป็น 0 ลอกลงมา · บิตที่สองเป็น 1 **ลอกลงมาแล้วหยุด** · ที่เหลือ 6 บิตทางซ้ายพลิกหมด ได้ **01001110** เท่ากันเป๊ะครับ

คำตอบ I DO

ขั้น 1 — ยืนยันความยาว 8 บิต คำตอบทั้งสองต้องยาว 8 บิตเท่ากัน เพราะคอมพลีเมนต์ไม่เปลี่ยนจำนวนบิตครับ

ขั้น 2 — 1's complement พลิกทุกบิตได้ **01001101** ตรวจง่าย ๆ ด้วยการบวกกลับ: $10110010 + 01001101 = 11111111$ ได้ 1 ทุกหลัก ✓

ขั้น 3 — 2's complement บวก 1 ต่อจาก 1's complement ได้ **01001110** ตรวจ: $10110010 + 01001110 = 100000000$ คือ 1 ตามด้วยศูนย์ 8 ตัว ซึ่งล้นออกนอกกรอบ 8 บิตพอดี ✓

ตอบ: 1's complement = 01001101 · **2's complement =** 01001110

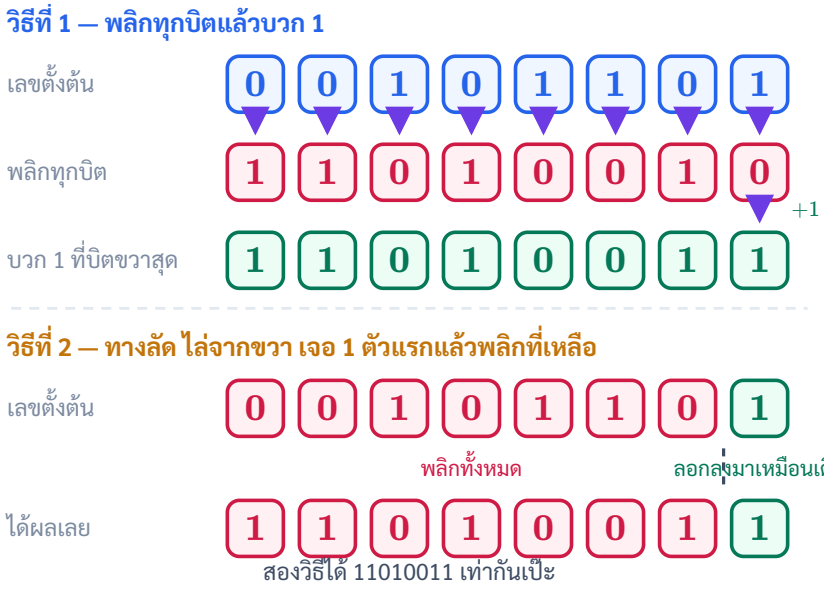
“The 1's complement of 10110010 is 01001101 and its 2's complement is 01001110.”

WE DO ทำด้วยกันครับ

ทำด้วยกัน พี่แต่งเอง (เลขตัวนี้จะกลับมาใน Assignment ข้อ 10)

> Determine the 2's complement of 00101101 using both methods.

$00101101_2 = 45$ ครับ ลองทำทั้งสองวิธีแล้วเทียบกันดู



2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 00101101 + 11010011 = 100000000 (ล้นออกไปนอกกรอบ 8 บิต แล้วเหลือศูนย์หมด)

เฉลย WE DO — เติมทีละช่องพร้อมเหตุผล

วิธีที่ 1 ขั้นแรก — พลิกทุกบิต $00101101 \rightarrow$ **11010010** เพราะกฎ 1's complement คือสลับ 0 กับ 1 ทุกตัว โดยไม่สนใจตำแหน่ง

วิธีที่ 1 ขั้นสอง — บวก 1 ที่บิตขวาสุด $11010010 + 1$: บิตขวาสุดเป็น 0 บวก 1 ได้ 1 **ไม่มีตัวทด หยุดทันที** → **11010011**

วิธีที่ 2 — ทางลัด ไล่จากขวา: บิตแรกเป็น **1** เลย **ลอกลงมาแล้วหยุดทันที** ที่เหลือ 7 บิตทางซ้าย (0010110) พลิกทั้งหมดได้ 1101001 → ต่อกันเป็น **11010011**

สองวิธีตรงกัน ✓ และตรวจอีกชั้น: $00101101 + 11010011 = 100000000$ ล้นออกนอก 8 บิตพอดี แปลว่าทั้งสองตัวรวมกันได้ 2^8 ซึ่งเป็นนิยามของ 2's complement เป๊ะครับ

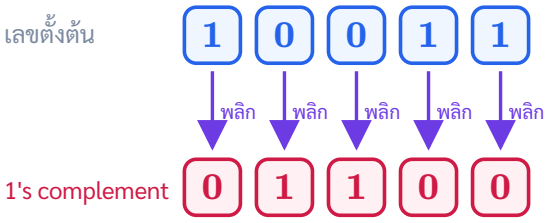
ตอบ: 2's complement ของ 00101101 **คือ** 11010011
“The 2's complement of 00101101 is 11010011.”

YOU DO ลองเองครับ

Y6.1

> Determine the 1's complement of 10011.

ข้อนี้ 5 บิตนะครับ คำตอบต้องยาว 5 บิตเท่ากัน



1's complement คือ “พลิกทุกบิต” 0 เป็น 1 และ 1 เป็น 0 ครับ ไม่มีการบวกอะไรทั้งสิ้น · ตรวจ: 10011 + 01100 = 11111 (ได้ 1 ทุกหลักเสมอ)

เฉลย Y6.1

ขั้น 1 — นับบิต 10011 มี 5 บิต ห้ามเติมและห้ามตัดครับ เพราะโจทย์ไม่ได้สั่งให้ทำเป็น 8 บิต

ขั้น 2 — พลิกทีละตัวจากซ้ายไปขวา $1 \to 0 \cdot 0 \to 1 \cdot 0 \to 1 \cdot 1 \to 0 \cdot 1 \to 0$

ขั้น 3 — ตรวจด้วยการบวกกลับ $10011 + 01100 = 11111$ ได้ 1 ครบทุกหลัก ✓ แปลว่าพลิกถูกทุกตัวครับ

ตอบ: 1's complement ของ 10011 **คือ** 01100

“The 1's complement of 10011 is 01100.”

Y6.2

> Determine the 2's complement of 11000 using the shortcut method.

ใช้ทางลัด “ไล่จากขวา เจอ 1 ตัวแรกแล้วพลิกที่เหลือ” ครับ ข้อนี้มีศูนย์ท้ายสามตัว



2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 11000 + 01000 = 100000 (ล้นออกไปนอกกรอบ 5 บิตแล้วเหลือศูนย์หมด)

เฉลย Y6.2

ขั้น 1 — ไล่จากขวาสุดเข้ามา เจอ $0, 0, 0$ สามตัว → **ลอกลงมาเหมือนเดิมทั้งสามตัว** เพราะกฎบอกว่าให้เขียนบิตตามเดิมไปจนกว่าจะถึง 1 ตัวแรกครับ

ขั้น 2 — เจอ 1 ตัวแรกที่หลัก 2^3 → **ลอกลงมาด้วย แล้วหยุดตรงนี้** (กฎบอกว่า “up to and including the first 1”)

ขั้น 3 — บิตที่เหลือทางซ้ายมีตัวเดียวคือ 1 → พลิกเป็น 0

ขั้น 4 — ตรวจด้วยวิธีปกติ 1's complement ของ 11000 คือ 00111 บวก 1 ได้ 01000 ✓ ตรงกัน

ตอบ: 2's complement ของ 11000 **คือ** 01000 — ตรวจอีกชั้น $11000 + 01000 = 100000$ ล้นออกนอก 5 บิตพอดีครับ

“The 2's complement of 11000 is 01000.”

ศัพท์สอบของตอนนี้

English	แปลว่า	ข้อสอบเขียนมาแบบนี้
1's complement	พลิกทุกบิต	changing all 1s to 0s and all 0s to 1s

English	แปลว่า	ข้อสอบเขียนมาแบบนี้
2's complement	พลิกทุกบิตแล้วบวก 1	obtained by adding 1 to the LSB of the 1's complement
radix-minus-one form	อีกชื่อของ 1's complement	1's complement form (or “radix-minus-one”)
uncomplemented / true form	เลขจริงก่อนพลิก	convert back to the true (uncomplemented) binary form
LSB	บิตขวาสุด	add 1 to the least significant bit

7 เลขมีเครื่องหมาย — เก็บได้สามหน้าตา

ตะขอเกี่ยว

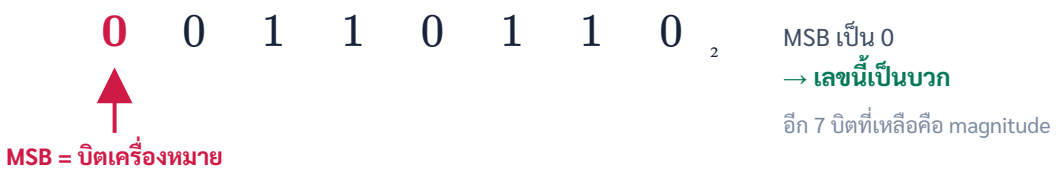
ตอนที่แล้วน้องทำ 1's และ 2's complement เป็นแล้วครับ แต่มันยังเป็นแค่ “การพลิกบิต” เฉย ๆ สมมติว่าถ้าข้อสอบสั่งว่า **“Express −45 as an 8-bit number”** ละครับ — น้องจะรู้ได้ยังไงว่าต้องพลิกหรือไม่พลิก และบิตไหนคือเครื่องหมายลบ ตอนนี้เราจะประกอบทุกอย่างเข้าด้วยกันครับ

DEFINITION · Locating the Sign Bit

The most significant bit (MSB) in a signed binary number is the sign bit, which tells you whether the number is positive or negative.

0 sign bit = positive number · 1 sign bit = negative number

พี่แปลว่า: **บิตซ้ายสุดคือบิตเครื่องหมาย** ครับ เห็น 0 แปลว่าบวก เห็น 1 แปลว่าลบ — กติกานี้ใช้เหมือนกันทั้งสามรูปแบบ เพราะฉะนั้นสิ่งแรกที่ต้องทำเสมอเวลาเจอเลขมีเครื่องหมายคือ **ดูบิตซ้ายสุดก่อน**



ตัวอย่างจากสไลด์ครับ — 00110110_2 มี MSB เป็น 0 จึงเป็นเลขบวกทั้งสามรูปแบบ

DEFINITION · สามรูปแบบ (จากสไลด์ Signed Numbers)

1.) Sign-Magnitude Form — The MSB is the sign bit and the remaining bits are the magnitude bits. The magnitude bits are in true (uncomplemented) binary for both positive and negative numbers.
พี่แปล: บิตซ้ายสุดบอกเครื่องหมาย ที่เหลือ**เขียนขนาดตรง ๆ ไม่พลิกเลย** ทั้งบวกและลบ

2.) 1's Complement Form — Positive numbers are represented the same way as the positive sign-magnitude numbers. Negative numbers are the 1's complements of the corresponding positive numbers.
พี่แปล: **เลขบวกเขียนเหมือนเดิม** ส่วน**เลขลบให้เอาเลขบวกมาพลิกทุกบิต**

3.) 2's Complement Form — Positive numbers are represented the same way as in the sign-magnitude and 1's complement forms. Negative numbers are the 2's complements of the corresponding positive numbers.
พี่แปล: **เลขบวกเขียนเหมือนเดิมอีกเช่นกัน** ส่วน**เลขลบให้เอาเลขบวกมาทำ 2's complement**

ทางลัด · สังเกตจุดร่วมสามข้อ

① **เลขบวกเขียนเหมือนกันหมดทั้งสามแบบ** — เจอ MSB เป็น 0 ตอบได้เลยโดยไม่ต้องถามว่าใช้รูปแบบไหน
② **ทั้งสามแบบเริ่มจากเลขบวกเสมอ** — เจอโจทย์เลขลบ ให้เขียนเลขบวกก่อนทุกครั้ง แล้วค่อยแปลง
③ **ทั้งสามแบบต้องระบุจำนวนบิตก่อน** — เพราะการพลิกบิตขึ้นกับความยาว

ภาพเทียบสามแบบด้วยเลขตัวเดียวกัน — -45 แบบ 8 บิต

นี่คือโจทย์ Assignment ข้อ 10 เป๊ะ ๆ ครับ พี่เอามาเทียบให้เห็นทั้งสามช่องพร้อมกันก่อน เดี๋ยวท้ายเล่มจะมีเฉลยเต็มอีกรอบ



เลขตัวเดียวกัน -45 แบบ 8 บิต เขียนได้สามหน้าตา ข้อสอบจะบอกว่าให้ใช้แบบไหน ถ้าไม่บอกให้ตอบทั้งสามครับ

1	45 → เลขฐานสอง 8 บิต 0 0 1 0 1 1 0 1	**ขั้นร่วมของทั้งสามแบบ — เขียนค่าบวกก่อน** $45 = 32 + 8 + 4 + 1 = 00101101_2$ ครับ **ทุกวิธีเริ่มจากตรงนี้เหมือนกันหมด**
2	0 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1	**sign-magnitude — เปลี่ยนแค่บิตซ้ายสุด** อีก 7 บิตคง 0101101 ไว้เหมือนเดิมเป๊ะ ได้ **10101101** ครับ ง่ายที่สุดในสามแบบ
3	0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 0	**1's complement — พลิกทุกบิต** พลิกทั้ง 8 ตัวรวมบิตเครื่องหมายด้วยครับ ได้ **11010010** — สังเกตว่า MSB กลายเป็น 1 เองอัตโนมัติ
4	1 1 0 1 0 0 1 0 + 1 1 1 0 1 0 0 1 1	**2's complement — บวก 1 ต่อจาก 1's** บิตขวาสุดเป็น 0 บวก 1 ได้ 1 ไม่มีตัวทด ได้ **11010011** ครับ

กับดัก · อย่าเผลอเซ็ตบิตเครื่องหมายซ้ำ

ใน 1's และ 2's complement เรา**ไม่ต้องไปตั้ง MSB เป็น 1 เอง**ครับ — การพลิกบิตทำให้ MSB กลายเป็น 1 อยู่แล้วโดยอัตโนมัติ (เพราะเลขบวกที่อยู่ในพิสัยจะมี MSB เป็น 0 เสมอ)

ถ้าน้องพลิกแล้วยังไปเปลี่ยน MSB อีกรอบ จะได้ 01010010 ซึ่ง**กลายเป็นเลขบวก 82** ผิดคนละเรื่องเลยครับ ส่วน sign-magnitude ต่างหากที่ต้องเปลี่ยน MSB ด้วยมือ

อีกตัวอย่างจากสไลด์ — -39 แบบ 8 บิต

เริ่มจากค่าบวก 39 = 00100111 ทั้งสามวิธี



เลขตัวเดียวกัน -39 แบบ 8 บิต เขียนได้สามหน้าตา ข้อสอบจะบอกว่าให้ใช้แบบไหน ถ้าไม่บอกให้ตอบทั้งสามครับ

ไล่ให้ดูว่า −39 ได้มายังไง

$39 = 32 + 4 + 2 + 1 = 00100111_2$ ครับ

- **sign-magnitude**: เปลี่ยน MSB เป็น 1 → **10100111**
- **1's complement**: พลิกทุกบิต → **11011000**
- **2's complement**: บวก 1 ต่อ → **11011001** (บิตขวาสุดเป็น 0 บวก 1 ได้ 1 ไม่มีตัวทด)

ปัญหาใหญ่ที่ทำให้โลกจริงเลือก 2's complement — ศูนย์ซ้ำสองตัว

สไลด์ตั้งหัวข้อว่า `Note for 1's complement` แล้วเขียนว่า **“there is the existence of positive and negative zeros — possible to have two numbers that both mean zero”** ครับ พี่จะกางให้ดูด้วยเลข 4 บิต



รหัส 1000 ที่ sign-magnitude ทิ้งไว้เป็น −0 เปล่าประโยชน์ 2's complement เอาไปใช้เก็บ −8 แทนครับ

สไลด์อธิบายไว้ว่า

“+0 = 0000, −0 = 10000 (but in 4 bits, overflow is ignored) → 0000. Thus, both +0 and −0 result in 0000.”

พี่ขยายให้ครับ: ถ้าเราหา 2's complement ของ 0000 จะได้ 1's complement $= 1111$ บวก 1 ได้ 10000 ซึ่งมี 5 บิต — แต่กรอบเรามีแค่ 4 บิต บิตที่ล้นออกไปจึงถูกทิ้ง เหลือ 0000 พอดี **แปลว่า “ลบศูนย์” กับ “บวกศูนย์” เป็นรหัสเดียวกันใน 2's complement** จึงไม่มีปัญหาศูนย์ซ้ำครับ

แล้วทำไมโลกจริงถึงเลือก 2's complement

เหตุผลที่ 1 — ศูนย์มีตัวเดียว

ไม่ต้องเขียนโปรแกรมเช็คสองกรณีว่า “นี่ศูนย์หรือเปล่า” และไม่เสียรหัสไปฟรี ๆ หนึ่งตัวครับ

เหตุผลที่ 2 — เก็บได้มากกว่าหนึ่งค่า

รหัสที่เคยเป็น −0 ถูกเอาไปใช้เก็บค่าลบสุดขอบ 8 บิตจึงเก็บได้ถึง **−128** แทนที่จะหยุดแค่ −127 ครับ

เหตุผลที่ 3 — บวกได้ด้วยวงจรเดิม

$A - B$ ทำได้โดย**เอา 2's complement ของ B ไปบวก** ไม่ต้องสร้างวงจรลบแยกอีกชุด ประหยัดเกตมหาศาลครับ

พิสูจน์เหตุผลที่ 3 ให้ดูเลย

ลองบวก 10 กับ -10 แบบ 8 บิต 2's complement ด้วยกติกาบวกธรรมดาจากตอนที่ 2 ครับ:

$10 = 00001010_2$ · $-10 = 11110110_2$ (2's complement ของ 00001010)

$00001010 + 11110110 = \mathbf{1}00000000$ — ตัวทดตัวสุดท้ายล้นออกนอกกรอบ 8 บิตแล้วถูกทิ้ง เหลือ **00000000 = 0** ถูกต้องพอดีครับ

นี่คือความมหัศจรรย์ของ 2's complement — **วงจรบวกตัวเดียวทำได้ทั้งบวกและลบ** เดี๋ยว EP03 จะใช้เรื่องนี้เต็ม ๆ ครับ

WE DO ทำด้วยกันครับ

ทำด้วยกัน พี่แต่งเอง

> Express the decimal number −20 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

ทำตามลำดับสามขั้นเหมือนเดิมครับ — เขียนค่าบวกก่อน แล้วค่อยแปลงทีละแบบ

เริ่มจากค่าบวก 20 = 00010100 ทั้งสามวิธี



เลขตัวเดียวกัน -20 แบบ 8 บิต เขียนได้สามหน้าตา ข้อสอบจะบอกว่าให้ใช้แบบไหน ถ้าไม่บอกให้ตอบทั้งสามครับ

เฉลย WE DO — ทีละขั้นพร้อมเหตุผล

ขั้น 1 — เขียน 20 เป็นฐานสอง 8 บิตก่อน $20 = 16 + 4 = 10100_2$ เติมศูนย์นำหน้าให้ครบ 8 บิตได้ **00010100** — ต้องเติมให้ครบก่อนพลิกเสมอครับ ไม่งั้นจำนวนบิตจะผิด

ขั้น 2 — sign-magnitude เปลี่ยนเฉพาะ MSB จาก 0 เป็น 1 ที่เหลือคงเดิม → **10010100** เพราะนิยามบอกว่า magnitude bits อยู่ในรูป true binary ทั้งบวกและลบ

ขั้น 3 — 1's complement พลิกทั้ง 8 บิตของ 00010100 → **11101011** เพราะนิยามบอกว่าเลขลบคือ 1's complement ของเลขบวกที่คู่กัน

ขั้น 4 — 2's complement บวก 1 ให้ 11101011 : บิตขวาสุดเป็น 1 บวก 1 ได้ 0 ทด 1 · บิตถัดไปเป็น 1 บวกตัวทดได้ 0 ทด 1 · บิตถัดไปเป็น 0 บวกตัวทดได้ 1 หยุด → **11101100**

ขั้น 5 — ตรวจด้วยทางลัด ไล่จากขวาของ 00010100 เจอ 0 แล้วเจอ 1 ตัวแรกที่หลัก 2^2 → ลอก 100 ลงมา แล้วพลิก 00010 เป็น 11101 ได้ **11101100** ✓ ตรงกันครับ

ตอบ: sign-magnitude $= 10010100$ · **1's complement** $= 11101011$ · **2's complement** $= 11101100$
“−20 is 10010100 (sign-magnitude), 11101011 (1's complement), and 11101100 (2's complement).”

YOU DO ลองเองครับ

Y7.1

> Express −13 as an 8-bit number in the three signed forms.

เลข 13 น้องคุ้นแล้วจากตอนที่ 2 ครับ ลองไล่สามขั้นดู

เริ่มจากค่าบวก 13 = 00001101 ทั้งสามวิธี



เลขตัวเดียวกัน -13 แบบ 8 บิต เขียนได้สามหน้าตา ข้อสอบจะบอกว่าให้ใช้แบบไหน ถ้าไม่บอกให้ตอบทั้งสามครับ

เฉลย Y7.1

ขั้น 1 — เขียนค่าบวกก่อน $13 = 8 + 4 + 1 = 1101_2$ เติมศูนย์ครบ 8 บิตได้ **00001101**

ขั้น 2 — sign-magnitude เปลี่ยน MSB เป็น 1 ที่เหลือคงเดิม → **10001101**

ขั้น 3 — 1's complement พลิกทุกบิต → **11110010**

ขั้น 4 — 2's complement บวก 1 ให้ 11110010 บิตขวาสุดเป็น 0 บวก 1 ได้ 1 ไม่มีตัวทด → **11110011**

ขั้น 5 — ตรวจ $00001101 + 11110011 = 100000000$ ล้นออกนอกกรอบ 8 บิตแล้วเหลือศูนย์หมด ✓ แปลว่า $13 + (-13) = 0$ จริงครับ

ตอบ: $10001101 \cdot 11110010 \cdot 11110011$

“−13 = 10001101 (sign-magnitude), 11110010 (1's complement), 11110011 (2's complement).”

Y7.2

> Express +45 as an 8-bit number in the three signed forms. Explain what you notice.

คราวนี้เป็นเลข**บวก**ครับ ลองสังเกตว่าสามแบบต่างกันไหม

เริ่มจากค่าบวก 45 = 00101101 ทั้งสามวิธี



เลขบวก 45 แบบ 8 บิต — ทั้งสามรูปแบบให้บิตชุดเดียวกันเป๊ะ ความต่างเกิดเฉพาะกับเลขลบเท่านั้นครับ

เฉลย Y7.2

ขั้น 1 — เขียน 45 เป็นฐานสอง 8 บิต $45 = 32 + 8 + 4 + 1 = 00101101_2$

ขั้น 2 — ตรวจเครื่องหมาย เลขเป็นบวก แปลว่า MSB ต้องเป็น 0 — ซึ่ง 00101101 มี MSB เป็น 0 อยู่แล้ว ✓

ขั้น 3 — สรุปทั้งสามแบบ นิยามทั้งสามข้อบอกตรงกันว่า **เลขบวกเขียนเหมือนกันหมด** (sign-magnitude เขียนขนาดตรง ๆ · 1's complement บอกว่าเลขบวกเขียนเหมือน sign-magnitude · 2's complement บอกว่าเลขบวกเขียนเหมือนอีกสองแบบ) จึงได้ 00101101 ทั้งสามช่อง

ตอบ: ทั้งสามรูปแบบได้ 00101101 **เหมือนกันหมด — สิ่งที่สังเกตได้คือความต่างของสามรูปแบบเกิดเฉพาะกับเลขลบเท่านั้น**ครับ เพราะฉะนั้นถ้าข้อสอบให้เลขบวก น้องตอบครั้งเดียวได้ทั้งสามช่องเลย

“+45 = 00101101 in all three forms; the forms differ only for negative numbers.”

8 อ่านค่าฐานสิบกลับ และพิสัยของเลข n บิต

ตะขอเกี่ยว

ตอนที่แล้วน้องเดินจาก**ฐานสิบไปเป็นบิต**ได้แล้วครับ สมมติว่าถ้าข้อสอบเดินย้อนทางล่ะครับ — โยนบิตมาให้ชุดหนึ่ง เช่น 10010100_2 แล้วถามว่า “ค่าฐานสิบเท่าไร ถ้าเป็น sign-magnitude / 1's complement / 2's complement” น้องจะได้**คำตอบไม่เท่ากันถึงสามค่า** จากบิตชุดเดียวกัน — ตอนนี้เราจะทำให้ชัดครับ

DEFINITION · The Decimal Value of Signed Numbers (จากสไลด์)

1.) Sign-magnitude form — Summing the weights in all the magnitude bit positions. Sign is determined by examination of the sign bit (MSB).
พี่แปล: **บวกน้ำหนักเฉพาะบิต magnitude** (ไม่รวม MSB) แล้วดู MSB ว่าจะติดลบไหม

2.) 1's complement form — For negative numbers: assigning a negative value to the weight of the sign bit, and summing the weights in all bit positions and adding 1 to the result.
พี่แปล: ให้ **MSB มีน้ำหนักติดลบ** แล้วบวกน้ำหนักทุกหลัก จากนั้น**บวก 1 ท้ายสุด**

3.) 2's complement form — For negative numbers: assigning a negative value to the weight of the sign bit, and summing the weights in all bit positions.
พี่แปล: ให้ **MSB มีน้ำหนักติดลบ** แล้วบวกน้ำหนักทุกหลัก **ไม่ต้องบวก 1**

ทางลัดที่พี่แนะนำให้ใช้ในห้องสอบ

วิธีน้ำหนักติดลบของสไลด์ถูกครับ แต่คิดเลขเยอะ พี่ขอเสนอวิธีที่พลาดยากกว่า:

ขั้นที่ 1 — ดู MSB ก่อน ถ้าเป็น 0 → อ่านตรง ๆ เหมือนเลขฐานสองธรรมดา จบเลย ทั้งสามแบบให้ค่าเดียวกัน
ขั้นที่ 2 — ถ้า MSB เป็น 1 → เป็นเลขลบแน่นอน แล้ว**แปลงกลับ**ตามรูปแบบ:

- sign-magnitude → ตัด MSB ทิ้ง อ่าน 7 บิตที่เหลือ แล้วใส่เครื่องหมายลบ
- 1's complement → **พลิกทุกบิต** อ่านค่า แล้วใส่เครื่องหมายลบ
- 2's complement → **ทำ 2's complement** อ่านค่า แล้วใส่เครื่องหมายลบ

ทั้งสองวิธีให้คำตอบเดียวกันเป๊ะครับ พี่จะเขียนทั้งสองวิธีกำกับไว้ในทุกกล่องเฉลย น้องเลือกใช้อันที่ถนัดได้เลย

ตัวอย่างที่ 1 จากสไลด์ — 00110110_2 (MSB เป็น 0)



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 54

1's complement

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 54

2's complement

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 54

อ่านผลลัพธ์

MSB เป็น 0 แปลว่าเป็นเลขบวก เพราะฉะนั้น**ทั้งสามรูปแบบให้ค่าเท่ากันหมด**ครับ บวกน้ำหนักได้ $32 + 16 + 4 + 2 = \mathbf{54}$ — ข้อแบบนี้ตอบครั้งเดียวได้สามช่องเลย ประหยัดเวลามาก

ตัวอย่างที่ 2 จากสไลด์ — 10110110_2 (MSB เป็น 1)



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 1 แปลว่าลบ · เอา 7 บิตที่เหลือ **0110110** มาอ่านตรง ๆ ได้ 54

= -54

1's complement

MSB = 1 แปลว่าลบ · พลิกกลับได้ **01001001** = 73

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้ว**บวก 1** ท้ายสุด: $-128 + 54 + 1 = -73$

= -73

2's complement

MSB = 1 แปลว่าลบ · ทำ 2's complement กลับได้ **01001010** = 74

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้วบวกน้ำหนักทุกหลักตรง ๆ: $-128 + 54 = -74$

= -74

เทียบให้เห็นว่าทำไมได้สามค่าไม่เท่ากัน

บิตชุดเดียวกันแท้ ๆ แต่ให้ค่าต่างกันถึงสามค่าครับ เพราะเรา**ตกลงกติกาการอ่านไม่เหมือนกัน**:

sign-magnitude	**−54**	ตัด MSB ทิ้ง เหลือ $0110110_2 = 54$ แล้วใส่ลบ
1's complement	**−73**	พลิกทุกบิตได้ $01001001_2 = 73$ แล้วใส่ลบ
2's complement	**−74**	ทำ 2's complement ได้ $01001010_2 = 74$ แล้วใส่ลบ

สังเกต: ค่าของ 1's complement กับ 2's complement ต่างกันอยู่ **1 เสมอ** ครับ (เพราะสองรูปแบบนี้ต่างกันแค่บวก 1) — ใช้เช็คงานได้ดีมาก ถ้าน้องได้สองค่านี้ห่างกันไม่ใช่ 1 แปลว่าคิดผิดแน่นอน

ตารางเทียบครบทุกรหัส 4 บิต (Conversion Table จากสไลด์)

ตารางนี้พี่แนะนำให้จำภาพรวมไว้ครับ มันตอบทุกคำถามของ EP นี้ในหน้าเดียว

Decimal	2's Complement	1's Complement	Signed Magnitude
−8	**1000**	—	—
−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	**1111 หรือ 0000**	**1000 หรือ 0000**
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

อ่านตารางนี้ให้เป็น 3 อย่างครับ: ① แถวบวกเหมือนกันหมดทั้งสามคอลัมน์ ② แถว 0 ของ 1's complement กับ signed magnitude มี**สองรหัส** — นี่คือปัญหา −0 ที่เราคุยกันไปแล้ว ③ มีแค่ 2's complement เท่านั้นที่มีค่า **−8** ส่วนอีกสองแบบเขียนขีดไว้เพราะเก็บไม่ได้

พิสัยของเลข n บิต — ทำไมฝั่งลบมีมากกว่าฝั่งบวกหนึ่งตัว

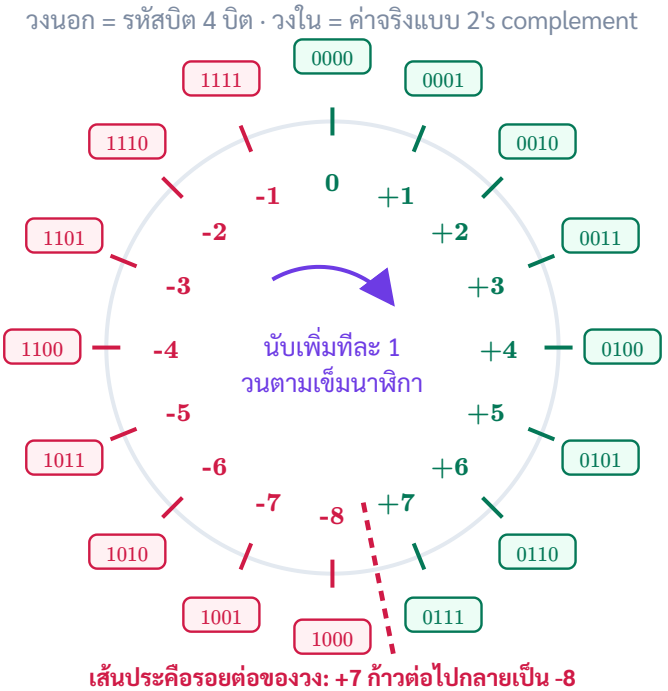
สูตรจากสไลด์

$$\text{Total combinations} = 2^n$$

$$\text{Range} = -(2^{n-1}) \text{ to } +(2^{n-1} - 1)$$

Note that there is one less positive number than there are negative numbers because zero is represented as a positive number (all zeros).
พี่แปล: ฝั่งบวกมีน้อยกว่าฝั่งลบอยู่หนึ่งตัว เพราะ**ศูนย์ไปกินโควตาฝั่งบวก** (รหัส 00000000 มี MSB เป็น 0) ครับ

คำอธิบายที่พี่ชอบที่สุดคือมองเป็น**มาตรวัดระยะทางรถ** (odometer) ครับ — รหัสบิตไม่ได้เรียงเป็นเส้นตรงแต่เรียงเป็นวงกลม พอนับเลยขอบบนก็หมุนกลับมาโผล่ที่ขอบล่างทันที



ง +7) · ฝั่งแดง 8 ค่า (−1 ถึง -8) — ฝั่งลบมากกว่าหนึ่งตัวเพราะศูนย์ไ

เลข 4 บิตแบบ 2's complement ไม่ได้เรียงเป็นเส้นตรง แต่เป็น**วงกลม**ครับ นับเพิ่มไปเรื่อย ๆ จนถึง +7 แล้วก้าวต่ออีกหนึ่งจะกระโดดไป -8 ทันที เหมือนมาตรวัดระยะทางรถที่หมุนครบรอบ

อ่านวงกลมนี้ให้ออก

① **ฝั่งเขียวมี 8 ค่า** คือ 0 ถึง $+7$ — สังเกตว่า**ศูนย์อยู่ฝั่งเขียว** เพราะรหัส 0000 มี MSB เป็น 0 นับเป็นเลขบวก
② **ฝั่งแดงมี 8 ค่า** คือ -1 ถึง -8 — ฝั่งนี้ได้ค่าจริง 8 ค่าเต็ม ๆ ไม่ต้องแบ่งให้ศูนย์
③ **รวมกัน 16 ค่า** ตรงกับ $2^4 = 16$ พอดี
④ ที่ฝั่งลบ “มากกว่า” ฝั่งบวกหนึ่งตัว จึงเป็นเพราะ**ฝั่งบวกต้องเสียที่ให้ศูนย์ไปหนึ่งช่อง**ครับ ไม่ใช่เรื่องบังเอิญ

ตัวอย่างจากสไลด์ — หาพิสัยของ 4, 8 และ 16 บิต

n	**จำนวนรหัสทั้งหมด** 2^n	**ค่าน้อยสุด** $-(2^{n-1})$	**ค่ามากสุด** $+(2^{n-1}-1)$
4	16	$-(2^3) = -8$	$+(2^3-1) = +7$
8	256	$-(2^7) = -128$	$+(2^7-1) = +127$
16	65536	$-(2^{15}) = -32768$	$+(2^{15}-1) = +32767$

สไลด์ยังตั้งคำถามต่อว่า **“How many bits are needed to store −100 in 2's complement?”** ครับ — วิธีคิดคือไล่ขึ้นทีละบิตจนกว่าจะครอบคลุม: 7 บิตได้ -64 ถึง $+63$ ซึ่ง**ยังเก็บ −100 ไม่ได้** · 8 บิตได้ -128 ถึง $+127$ ซึ่ง**เก็บได้** → **ตอบ 8 บิต**

I DO พี่ทำให้ดูครับ

ตัวอย่างที่พี่ทำ — พี่แต่งเอง (ปูทางไป Assignment ข้อ 11)

> Determine the decimal value of the 8-bit number 11101100 assuming sign-magnitude, 1's complement, and 2's complement forms.

บิตชุดนี้คือ -20 แบบ 2's complement ที่เราทำไว้ตอนที่ 7 ครับ ลองอ่านย้อนกลับดู



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 1 แปลว่าลบ · เอา 7 บิตที่เหลือ **1101100** มาอ่านตรง ๆ ได้ 108

= -108

1's complement

MSB = 1 แปลว่าลบ · พลิกกลับได้ **00010011** = 19

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้ว**บวก 1** ท้ายสุด: $-128 + 108 + 1 = -19$

= -19

2's complement

MSB = 1 แปลว่าลบ · ทำ 2's complement กลับได้ **00010100** = 20

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้วบวกน้ำหนักทุกหลักตรง ๆ: $-128 + 108 = -20$

= -20

คำตอบ I DO

ขั้น 1 — ดู MSB ก่อนเสมอ MSB $= 1$ แปลว่า**เป็นเลขลบทั้งสามรูปแบบ** เพราะฉะนั้นคำตอบทุกช่องต้องติดลบ ถ้าน้องได้ค่าบวกออกมาแปลว่าพลาดแน่นอนครับ

ขั้น 2 — sign-magnitude ตัด MSB ทิ้ง เหลือ magnitude $1101100_2 = 64 + 32 + 8 + 4 = 108$ → **−108**

(วิธีสไลด์: บวกน้ำหนักเฉพาะ magnitude bits ได้ 108 แล้วดู MSB ว่าเป็นลบ — ได้เท่ากันครับ)

ขั้น 3 — 1's complement พลิกทุกบิตได้ $00010011_2 = 16 + 2 + 1 = 19$ → **−19**

(วิธีสไลด์: $-128 + 64 + 32 + 8 + 4 = -20$ แล้วบวก 1 ได้ -19 ✓)

ขั้น 4 — 2's complement ทำ 2's complement ได้ $00010100_2 = 16 + 4 = 20$ → **−20**

(วิธีสไลด์: $-128 + 64 + 32 + 8 + 4 = -20$ ✓ ไม่ต้องบวก 1)

ขั้น 5 — เช็คว่าสองค่าห่างกัน 1 -19 กับ -20 ห่างกัน 1 พอดี ✓ ถูกต้องครับ

ตอบ: sign-magnitude $= -108$ · **1's complement** $= -19$ · **2's complement** $= -20$

“11101100 is −108 (sign-magnitude), −19 (1's complement), −20 (2's complement).”

WE DO ทำด้วยกันครับ

ทำด้วยกัน พี่แต่งเอง

> Find the range of 6-bit signed numbers in 2's complement form, and state how many bits are needed to store −40.

ใช้สูตรจากสไลด์ตรง ๆ เลยครับ แล้วค่อยไล่หาจำนวนบิตที่พอดี

เฉลย WE DO — ทีละช่องพร้อมเหตุผล

ขั้น 1 — จำนวนรหัสทั้งหมด $2^6 = \mathbf{64}$ รหัส เพราะแต่ละบิตเลือกได้ 2 แบบ และมี 6 บิต

ขั้น 2 — ค่าน้อยสุด $-(2^{6-1}) = -(2^5) = \mathbf{-32}$ ตามสูตร $-(2^{n-1})$

ขั้น 3 — ค่ามากสุด $+(2^5 - 1) = \mathbf{+31}$ — ที่ต้องลบ 1 เพราะ**ศูนย์กินโควตาฝั่งบวกไปหนึ่งช่อง**

ขั้น 4 — ตรวจจำนวนช่อง ฝั่งลบ 32 ค่า (-1 ถึง -32) + ฝั่งบวกกับศูนย์ 32 ค่า (0 ถึง $+31$) รวม 64 ตรงกับ 2^6 ✓

ขั้น 5 — จะเก็บ -40 **ต้องใช้กี่บิต** ไล่ขึ้นทีละบิตครับ:

6 บิต → -32 ถึง $+31$ **ไม่พอ** (-40 เลยขอบล่าง)

7 บิต → -64 ถึง $+63$ **พอแล้ว**

ตอบ: พิสัยของ 6 บิตคือ -32 **ถึง** $+31$ **(64 ค่า) · ต้องใช้ 7 บิตจึงจะเก็บ** -40 **ได้**

"A 6-bit 2's complement number ranges from −32 to +31; storing −40 requires 7 bits."

YOU DO ลองเองครับ

Y8.1

> Determine the decimal value of 11111111 assuming sign-magnitude, 1's complement, and 2's complement forms.

ข้อนี้เป็น 1 ทุกบิตครับ เป็นข้อที่โผล่ในข้อสอบบ่อยมาก



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 1 แปลว่าลบ · เอา 7 บิตที่เหลือ **1111111** มาอ่านตรง ๆ ได้ 127

= -127

1's complement

MSB = 1 แปลว่าลบ · พลิกกลับได้ **00000000** = 0

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้ว**บวก 1** ท้ายสุด: $-128 + 127 + 1 = 0$

= 0

2's complement

MSB = 1 แปลว่าลบ · ทำ 2's complement กลับได้ **00000001** = 1

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้วบวกน้ำหนักทุกหลักตรง ๆ: $-128 + 127 = -1$

= -1

เฉลย Y8.1

ขั้น 1 — MSB เป็น 1 → เลขลบทั้งสามแบบครับ

ขั้น 2 — sign-magnitude ตัด MSB ทิ้ง เหลือ $1111111_2 = 127$ → **−127** ซึ่งเป็นค่าที่**น้อยที่สุด**ที่ sign-magnitude 8 บิตเก็บได้พอดี

ขั้น 3 — 1's complement พลิกทุกบิตได้ $00000000_2 = 0$ → **−0** นี่คือ**ศูนย์ตัวที่สอง**ที่เราคุยกันไปแล้วครับ ค่ามันคือศูนย์ แต่รหัสไม่ใช่ 00000000

ขั้น 4 — 2's complement ทำ 2's complement: พลิกได้ 00000000 บวก 1 ได้ $00000001_2 = 1$ → **−1**

ขั้น 5 — เช็ค ค่าของ 1's (0) กับ 2's (-1) ห่างกัน 1 พอดี ✓

ตอบ: sign-magnitude $= -127$ **· 1's complement** $= -0$ **(คือ 0) · 2's complement** $= -1$

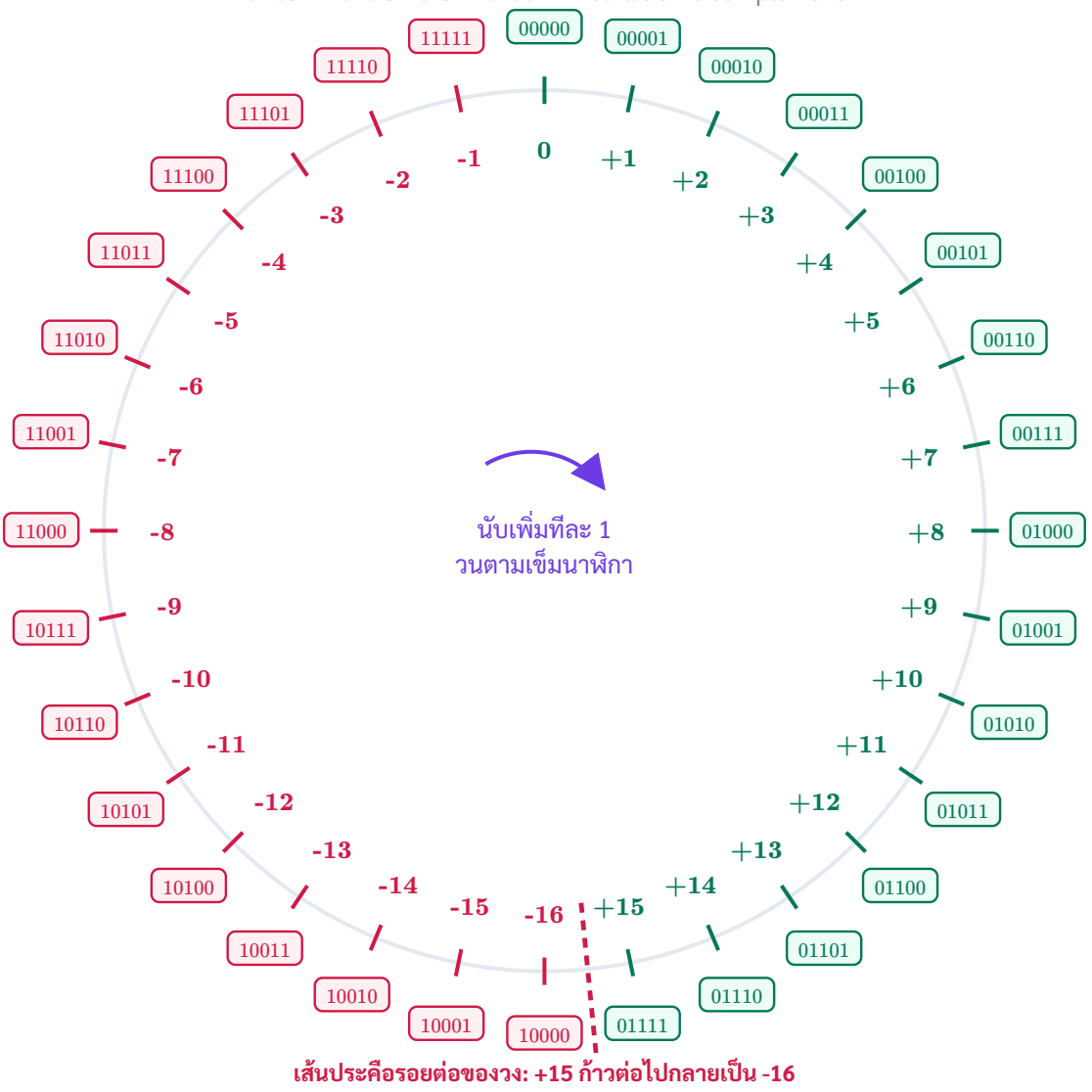
“11111111 is −127 (sign-magnitude), −0 (1's complement), and −1 (2's complement).”

Y8.2

> What is the range of a 5-bit signed number in 2's complement form? How many different values can it represent?

ใช้สูตรเดิมครับ อย่าลืมว่าจำนวนค่าทั้งหมดคือ 2^n ไม่ใช่ผลต่างของขอบสองข้าง

วงนอก = รหัสบิต 5 บิต · วงใน = ค่าจริงแบบ 2's complement



ฝั่งเขียว 16 ค่า (0 ถึง +15) · ฝั่งแดง 16 ค่า (−1 ถึง -16) — ฝั่งลบมากกว่าหนึ่งตัวเพราะศูนย์ไปนับรวมอยู่ฝั่งบวก

เลข 5 บิตแบบ 2's complement ไม่ได้เรียงเป็นเส้นตรง แต่เป็น**วงกลม**ครับ นับเพิ่มไปเรื่อย ๆ จนถึง +15 แล้วก้าวต่ออีกหนึ่งจะกระโดดไป -16 ทันที เหมือนมาตรวัดระยะทางรถที่หมุนครบรอบ

เฉลย Y8.2

ขั้น 1 — จำนวนรหัสทั้งหมด $2^5 = 32$ รหัส เพราะ 5 บิต แต่ละบิตเลือกได้ 2 แบบ

ขั้น 2 — ค่าน้อยสุด แทน $n = 5$ ลงในสูตร $-(2^{n-1})$ ได้ $-(2^4) = -16$

ขั้น 3 — ค่ามากสุด $+(2^4 - 1) = +15$ ที่ลบ 1 เพราะศูนย์ไปนับอยู่ฝั่งบวก

ขั้น 4 — ตรวจด้วยการนับจริง ฝั่งลบ -1 ถึง -16 มี 16 ค่า · ฝั่งบวกกับศูนย์ 0 ถึง $+15$ มี 16 ค่า รวม 32 ค่า ✓ ตรงกับ 2^5 พอดี — ดูวงกลม odometer ข้างบนประกอบได้เลยครับ

ตอบ: พิสัยคือ -16 ถึง $+15$ และเก็บได้ทั้งหมด 32 ค่า

“A 5-bit 2's complement number ranges from −16 to +15 and can represent 32 different values.”

ศัพท์สอบของตอนนี้

English	แปลว่า	ข้อสอบเขียนมาแบบนี้
magnitude bits	บิตที่เก็บขนาด (ไม่รวม MSB)	summing the weights in all the magnitude bit positions
weight	น้ำหนักประจำหลัก	assigning a negative value to the weight of the sign bit
range	พิสัย (ค่าต่ำสุดถึงสูงสุด)	the range of values for n-bit numbers is ...
byte	8 บิต	8-bit numbers or byte
combinations	จำนวนรหัสที่เป็นไปได้	the number of different combinations of n bits is 2^n

STOP

หน้าเฉลย Assignment #1

ตั้งแต่หน้านี้ไปคือเฉลยข้อ 8–15 ของ Assignment #1 ครับ ถ้าน้องยังไม่ได้ลงมือทำเอง พี่ขอให้กลับไปทำก่อนนะครับ การอ่านเฉลยแล้วรู้สึกว่า “อ๋อ ง่ายจัง” มันคนละเรื่องกับการทำได้เองในห้องสอบ ตรงนี้เป็นคะแนน Assignment 25% ของวิชานี้ด้วยครับ

9 เฉลย Assignment #1 ข้อ 8–11

ทุกข้อย่อยพี่จะไล่ให้ครบ 6 ชั้นเหมือนกันหมดครับ: **① โจทย์และสิ่งที่โจทย์ซ่อนไว้** → **② เตรียมก่อนลงมือ** → **③ ไล่ทีละขั้นพร้อมภาพ** → **④ คำตอบ** → **⑤ ตรวจด้วยวิธีที่สอง** → **⑥ กับดักของข้อนี้**

ข้อ 8 (a)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 8 (a)

> Determine the 1's complement of the following binary numbers: a. 110

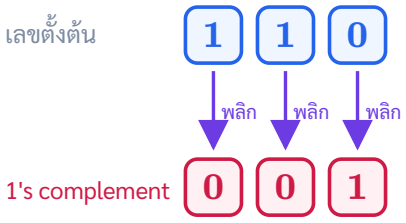
แปล: จงหา 1's complement ของ 110_2 ครับ

โจทย์ซ่อนอะไรไว้: โจทย์**ไม่ได้บอกจำนวนบิต** — เขียนมา 3 บิตก็ต้องตอบ 3 บิต ห้ามเผลอขยายเป็น 8 บิตเอง ไม่งั้นได้ 11111001 ซึ่งไม่ตรงกับที่โจทย์ถามครับ

② เตรียมก่อนลงมือ

ใช้กฎ 1's complement ตรง ๆ ครับ: **changing all 1s to 0s and all 0s to 1s** — เลือกวิธีนี้เพราะมันมีวิธีเดียว ไม่มีการบวก ไม่มีตัวทด แค่พลิกบิตทีละตัวจนครบ

③ ไล่ทีละขั้นพร้อมภาพ



1's complement คือ “พลิกทุกบิต” 0 เป็น 1 และ 1 เป็น 0 ครับ ไม่มีการบวกอะไรทั้งสิ้น · ตรวจ: 110 + 001 = 111 (ได้ 1 ทุกหลักเสมอ)

ไล่ทีละบิตจากซ้ายไปขวาครับ — บิตที่ 1 เป็น **1** พลิกเป็น **0** · บิตที่ 2 เป็น **1** พลิกเป็น **0** · บิตที่ 3 เป็น **0** พลิกเป็น **1** ครบ 3 บิตพอดี

④ คำตอบ

1's complement ของ 110_2 คือ $\mathbf{001_2}$

“The 1's complement of 110 is 001.”

⑤ ตรวจด้วยวิธีที่สอง

บวกกลับต้องได้ 1 ทุกหลัก: $110 + 001 = 111$ ✓ เพราะนิยาม radix-minus-one บอกว่า $N + \text{1's comp}(N) = 2^n - 1$ และ $2^3 - 1 = 7 = 111_2$ พอดีครับ

ตรวจฐานสิบ: $110_2 = 6 \cdot 001_2 = 1 \cdot 6 + 1 = 7 = 2^3 - 1$ ✓

⑥ กับดักของข้อนี้

ตัดศูนย์นำหน้าทิ้ง — หลายคนได้ 001 แล้วเขียนคำตอบเป็น 1 ครับ **ผิด** เพราะคอมพลีเมนต์ต้องยาวเท่าเลขตั้งต้นเสมอ ต้องเขียน 001 ให้ครบ 3 บิต

หัวใจของข้อนี้: จำนวนบิตของคำตอบต้องเท่ากับจำนวนบิตของโจทย์เป๊ะ

ข้อ 8 (b)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 8 (b)

> Determine the 1's complement of the following binary numbers: b. 11010111

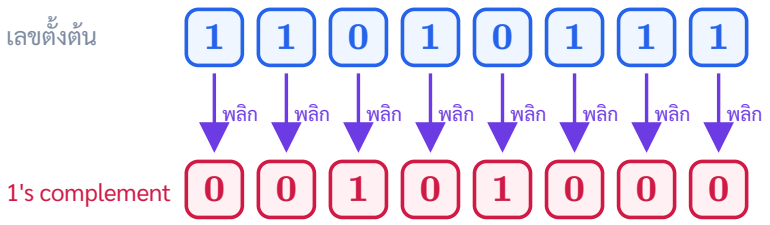
แปล: จงหา 1's complement ของ 11010111_2 ครับ

โจทย์ซ่อนอะไรไว้: คราวนี้ยาว 8 บิต ต้องพลิกให้ครบทั้ง 8 ตัว **รวมบิตซ้ายสุดด้วย** — คนมักลืมพลิก MSB เพราะคิดว่ามันเป็น “บิตเครื่องหมาย” แต่ข้อนี้โจทย์ไม่ได้พูดถึงเครื่องหมายเลยครับ

② เตรียมก่อนลงมือ

ใช้กฎเดียวกับข้อ (a) ครับ แต่พี่แนะนำให้**แบ่งเป็นสองกลุ่ม กลุ่มละ 4 บิต** (1101 กับ 0111) แล้วพลิกทีละกลุ่ม จะไม่หลงตำแหน่งเวลามีบิตเยอะ

③ ไล่ทีละขั้นพร้อมภาพ



1's complement คือ “พลิกทุกบิต” 0 เป็น 1 และ 1 เป็น 0 ครับ ไม่มีการบวกอะไรทั้งสิ้น · ตรวจ: 11010111 + 00101000 = 11111111 (ได้ 1 ทุกหลักเสมอ)

ตำแหน่งบิต	1	2	3	4	5	6	7	8
เลขตั้งต้น	1	1	0	1	0	1	1	1
พลิกแล้ว	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**

④ คำตอบ

1's complement ของ 11010111_2 คือ $\mathbf{00101000_2}$

“The 1's complement of 11010111 is 00101000.”

⑤ ตรวจด้วยวิธีที่สอง

บวกกลับต้องได้ 1 ทุกหลัก: $11010111 + 00101000 = 11111111$ ✓

ตรวจฐานสิบ: $11010111_2 = 128 + 64 + 16 + 4 + 2 + 1 = 215 \cdot 00101000_2 = 32 + 8 = 40 \cdot 215 + 40 = 255 = 2^8 - 1$ ✓ ตรงกับสูตร radix-minus-one พอดีครับ

⑥ กับดักของข้อนี้

พลิกไม่ครบ 8 ตัว — เวลาบิตเยอะ ๆ คนชอบพลิกได้ 6-7 ตัวแล้วนึกว่าครบ ถ้าลืมพลิกบิตสุดท้ายจะได้ 00101001 ซึ่งบวกกลับแล้วได้ 11111110 ไม่ใช่ 1 ทุกหลัก **ทดสอบด้วยการบวกกลับทุกครั้งจึงจับผิดได้ทันที** ครับ

หัวใจของข้อนี้: 1's complement ไม่สนใจว่าบิตไหนเป็นบิตเครื่องหมาย พลิกให้หมดทุกตัว

ข้อ 9 (a)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 9 (a)

> Determine the 2's complement of the following binary numbers: a. 111

แปล: จงหา 2's complement ของ 111_2 ครับ

โจทย์ซ่อนอะไรไว้: ข้อนี้เป็น **1 ทุกบิต** พลิกแล้วจะได้ 0 ทุกบิต แล้วต้องบวก 1 — เป็นเคสที่**ทางลัดใช้ง่ายที่สุด** เพราะบิตขวาสุดเป็น 1 อยู่แล้ว

② เตรียมก่อนลงมือ

พี่จะทำ**ทั้งสองวิธี**ให้ดูครับ — วิธีที่ 1 (พลิกแล้วบวก 1) เพื่อให้เห็นตามนิยาม และวิธีที่ 2 (ทางลัดไล่จากขวา) เพื่อให้ใช้จริงตอนสอบ ผลต้องตรงกัน ถ้าไม่ตรงแปลว่ามีอันหนึ่งผิด

③ ไล่ทีละขั้นพร้อมภาพ



2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 111 + 001 = 1000 (ล้นออกไปนอกกรอบ 3 บิตแล้วเหลือศูนย์หมด)

วิธีที่ 1: พลิก $111 \rightarrow 000$ แล้วบวก 1 → บิตขวาสุด $0 + 1 = 1$ ไม่มีตัวทด ได้ 001
วิธีที่ 2: ไล่จากขวา เจอ **1** เป็นบิตแรกเลย → ลอกลงมาแล้วหยุดทันที ที่เหลือ 11 พลิกเป็น 00 ได้ 001

④ คำตอบ

2's complement ของ 111_2 คือ 001_2

“The 2's complement of 111 is 001.”

⑤ ตรวจด้วยวิธีที่สอง

บวกกลับต้องล้นออกนอกกรอบพอดี: $111 + 001 = 1000$ ซึ่งเป็น 1 ตามด้วยศูนย์ 3 ตัว บิตที่ล้นออกไปถูกทิ้ง เหลือ 000 ✓ เพราะนิยามคือ $N + 2\text{'s comp}(N) = 2^n$ และ $2^3 = 8 = 1000_2$ พอดีครับ

ตรวจฐานสิบ: $111_2 = 7 \cdot 001_2 = 1 \cdot 7 + 1 = 8 = 2^3$ ✓

⑥ กับดักของข้อนี้

บวก 1 แล้วเผลอทดต่อ — บางคนพลิกได้ 000 แล้วบวก 1 ไปเขียนเป็น 100 (ใส่ 1 ผิดหลัก) ครับ ต้องจำว่า **บวก 1 ที่ LSB คือบิตขวาสุดเท่านั้น** คำตอบคือ 001 ไม่ใช่ 100

หัวใจของข้อนี้: 2's complement ของเลขที่เป็น 1 ล้วน จะได้ 1 โดด ๆ ที่ตำแหน่งขวาสุดเสมอ

ข้อ 9 (b)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 9 (b)

> Determine the 2's complement of the following binary numbers: b. 1101

แปล: จงหา 2's complement ของ 1101_2 ครับ

โจทย์ซ่อนอะไรไว้: 4 บิตพอดี และบิตขวาสุดเป็น 1 อีกเช่นกัน แปลว่าทางลัดจะจบเร็วมาก แต่ระวังตอนพลิก 3 บิตที่เหลือให้ครบครับ

② เตรียมก่อนลงมือ

ทำสองวิธีเทียบกันเหมือนข้อ (a) ครับ — และข้อนี้พี่จะโชว์ให้เห็นว่า **ทางลัดคือการ “ลอกบิตท้ายจนถึง 1 ตัวแรก แล้วพลิกที่เหลือ”** ตรงตามที่สไลด์เขียนไว้เป๊ะ

③ ไล่ทีละขั้นพร้อมภาพ



2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 1101 + 0011 = 10000 (ล้นออกไปนอกกรอบ 4 บิตแล้วเหลือศูนย์หมด)

วิธีที่ 1: พลิก $1101 \to 0010$ แล้วบวก $1 \to$ บิตขวาสุด $0 + 1 = 1$ ไม่มีตัวทด ได้ 0011

วิธีที่ 2: ไล่จากขวา เจอ **1** ทันที → ลอกลงมาแล้วหยุด · ที่เหลือ 110 พลิกเป็น 001 ต่อกันได้ 0011

④ คำตอบ

2's complement ของ 1101_2 คือ $\mathbf{0011_2}$

“The 2's complement of 1101 is 0011.”

⑤ ตรวจด้วยวิธีที่สอง

บวกกลับ: $1101 + 0011 = 10000$ ล้นออกนอกกรอบ 4 บิตพอดี เหลือ 0000 ✓

ตรวจฐานสิบ: $1101_2 = 13 \cdot 0011_2 = 3 \cdot 13 + 3 = 16 = 2^4$ ✓ — และถ้าอ่าน 0011 เป็นเลขมีเครื่องหมาย 4 บิต มันคือ $+3$ ซึ่งเป็นคู่ตรงข้ามของ $1101 = -3$ ในระบบ 2's complement พอดีครับ

⑥ กับดักของข้อนี้

ทำ 1's complement แล้วลืมบวก 1 — จะตอบ 0010 ซึ่งเป็นคำตอบของข้อ 8 ไม่ใช่ข้อ 9 ครับ วิธีกันคือ**อ่านหัวข้อโจทย์ให้ชัดก่อนว่าเขาถาม 1's หรือ 2's** เพราะข้อ 8 กับ 9 หน้าตาเหมือนกันมาก

หัวใจของข้อนี้: ถ้าบิตขวาสุดเป็น 1 ทางลัดจบใน 1 วินาที — ลอก 1 ตัวนั้น แล้วพลิกที่เหลือ

ข้อ 10

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 10

> Express -45_{10} as an 8-bit number in sign-magnitude, 1's complement, and 2's complement forms.

แปล: จงเขียน -45_{10} เป็นเลข **8 บิต** ทั้งสามรูปแบบครับ

โจทย์ซ่อนอะไรไว้: คำว่า **"as an 8-bit number"** คือหัวใจ — ต้องแปลง 45 เป็นฐานสองแล้ว**เติมศูนย์นำหน้าให้ครบ 8 บิตก่อน** ค่อยพลิก ถ้าพลิกตอนยังเป็น 6 บิตจะได้คำตอบผิดทั้งสองช่องหลังครับ

② เตรียมก่อนลงมือ

แผนคือ 3 ขั้นครับ: **① แปลง 45 เป็นฐานสอง 8 บิต → ② ทำ sign-magnitude (เปลี่ยน MSB) → ③ ทำ 1's complement (พลิก) แล้วบวก 1 ต่อเป็น 2's complement**

เลือกลำดับนี้เพราะ 2's complement ต่อยอดจาก 1's complement ได้เลย ไม่ต้องเริ่มใหม่

③ ไล่ทีละขั้นพร้อมภาพ

1

$45 = 32 + 8 + 4 + 1$
$= 101101_2$ (6 บิต)
$= 00101101_2$ (8 บิต)

แปลง 45 เป็นฐานสองแล้วเติมให้ครบ 8 บิต

$45 = 32 + 8 + 4 + 1$ ได้ 101101_2 ซึ่งมี 6 บิต ต้อง**เติมศูนย์นำหน้าอีก 2 ตัว** เป็น 00101101_2 ก่อนครับ เพราะโจทย์สั่งว่า 8-bit

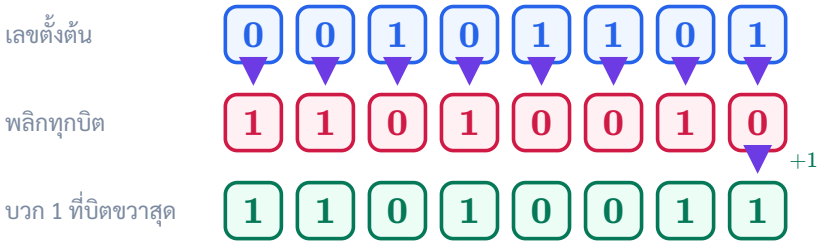
2	00101101 10101101	**sign-magnitude — เปลี่ยนเฉพาะ MSB** นิยามบอกว่า magnitude bits อยู่ในรูป `true (uncomplemented) binary` เพราะฉะนั้น 7 บิตขวาคง 0101101 ไว้เหมือนเดิม เปลี่ยนแค่ MSB เป็น 1 ครับ
3	00101101 11010010	**1's complement — พลิกทั้ง 8 บิต** พลิกจากเลข**บวก** 00101101 ไม่ใช่จาก sign-magnitude นะครับ นิยามบอกว่า `the 1's complements of the corresponding positive numbers`
4	11010010 +1 11010011	**2's complement — บวก 1 ต่อ** บิตขวาสุดของ 11010010 เป็น 0 บวก 1 ได้ 1 **ไม่มีตัวทดต่อ** จบทันทีครับ

เริ่มจากค่าบวก 45 = 00101101 ทั้งสามวิธี



เลขตัวเดียวกัน -45 แบบ 8 บิต เขียนได้สามหน้าตา ข้อสอบจะบอกว่าให้ใช้แบบไหน ถ้าไม่บอกให้ตอบทั้งสามครับ

วิธีที่ 1 — พลิกทุกบิตแล้วบวก 1



วิธีที่ 2 — ทางลัด ไล่จากขวา เจอ 1 ตัวแรกแล้วพลิกที่เหลือ



สองวิธีได้ 11010011 เท่ากันเป๊ะ

2's complement ทำได้สองทาง ผลต้องตรงกันเสมอครับ · ตรวจ: 00101101 + 11010011 = 100000000 (ล้นออกไปนอกกรอบ 8 บิต แล้วเหลือศูนย์หมด)

④ คำตอบ

sign-magnitude = 10101101_2

1's complement = 11010010_2

2's complement = 11010011_2

"−45 = 10101101 (sign-magnitude), 11010010 (1's complement), 11010011 (2's complement)."

⑤ ตรวจด้วยวิธีที่สอง — อ่านกลับเป็นฐานสิบทั้งสามช่อง

sign-magnitude 10101101 : MSB $= 1$ ลบ · magnitude $0101101_2 = 32 + 8 + 4 + 1 = 45 \rightarrow \mathbf{-45}$ ✓

1's complement 11010010 : MSB $= 1$ ลบ · พลิกกลับได้ $00101101_2 = 45 \rightarrow \mathbf{-45}$ ✓
(วิธีน้ำหนักติดลบแบบสไลด์: $-128 + 64 + 16 + 2 = -46$ แล้วบวก 1 ได้ -45 ✓)

2's complement 11010011 : MSB $= 1$ ลบ · ทำ 2's complement กลับได้ $00101101_2 = 45 \rightarrow \mathbf{-45}$ ✓
(วิธีน้ำหนักติดลบ: $-128 + 64 + 16 + 2 + 1 = -45$ ✓)

ตรวจอีกชั้น: $00101101 + 11010011 = 100000000$ ล้นออกนอกกรอบ 8 บิตแล้วเหลือศูนย์หมด แปลว่า $45 + (-45) = 0$ จริงครับ

⑥ กับดักของข้อนี้

① **พลิกจาก sign-magnitude แทนที่จะพลิกจากเลขบวก** — ถ้าเอา 10101101 ไปพลิกจะได้ 01010010 ซึ่ง MSB เป็น 0 กลายเป็นเลข**บวก 82** ผิดคนละโลกเลยครับ

② **ลืมเติมศูนย์ให้ครบ 8 บิตก่อนพลิก** — ถ้าพลิก 101101 (6 บิต) จะได้ 010010 แล้วต่อให้เติมศูนย์ทีหลังก็ได้ 00010010 ซึ่งผิดครับ **ต้องเติมศูนย์ก่อนพลิกเสมอ**

หัวใจของข้อนี้: เขียนค่าบวกให้ครบจำนวนบิตก่อน แล้วทั้งสามรูปแบบจะแตกออกจากตรงนั้น

ข้อ 11 (a)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 11 (a)

> 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

แปล: จงหาค่าฐานสิบของ 00000000_2 ทั้งสามรูปแบบครับ
โจทย์ซ่อนอะไรไว้: ข้อนี้ดูเหมือนของฟรี แต่จริง ๆ มันคือ**คำถามเรื่องศูนย์ซ้ำสองตัว**ที่สไลด์เน้นไว้ — 00000000 คือ **"บวกศูนย์"** ซึ่งใช้ได้ทั้งสามรูปแบบ ส่วน "ลบศูนย์" จะเป็นรหัสอื่นครับ

② เตรียมก่อนลงมือ

ใช้ทางลัด **“ดู MSB ก่อน”** ครับ — MSB เป็น 0 แปลว่าเป็นเลขบวก ซึ่งนิยามทั้งสามข้อบอกตรงกันว่าเลขบวกอ่านเหมือนกันหมด เพราะฉะนั้นตอบครั้งเดียวได้ทั้งสามช่อง

③ ไล่ทีละขั้นพร้อมภาพ



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 0

1's complement

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 0

2's complement

MSB = 0 แปลว่าบวก ทั้งสามแบบอ่านเหมือนกันหมด คือบวกน้ำหนักของบิตที่เป็น 1

= 0

ไล่บวกน้ำหนักทุกหลักครับ: $0 \times 128 + 0 \times 64 + 0 \times 32 + 0 \times 16 + 0 \times 8 + 0 \times 4 + 0 \times 2 + 0 \times 1 = 0$ — ไม่มีบิตไหนเป็น 1 เลย ค่าจึงเป็นศูนย์

④ คำตอบ

sign-magnitude $= 0$ · 1's complement $= 0$ · 2's complement $= 0$

“00000000 equals 0 in all three signed forms (it is the positive zero).”

⑤ ตรวจด้วยวิธีที่สอง

ตรวจด้วยตารางเทียบ 4 บิตของสไลด์: แถว Decimal $= 0$ มีค่า 0000 อยู่ในทั้งสามคอลัมน์ ✓ (ขยายเป็น 8 บิตก็คือ 00000000)

ตรวจด้วยวิธีน้ำหนักติดลบแบบสไลด์: MSB เป็น 0 จึงไม่มีน้ำหนักติดลบเข้ามาเกี่ยวเลย ผลรวมทุกหลักเป็น 0 ทั้งสามรูปแบบ ✓

ตรวจย้อนกลับ: ลองเขียน $+0$ ด้วยกฎของแต่ละรูปแบบ — sign-magnitude ได้ 00000000 · 1's complement ได้ 00000000 · 2's complement ได้ 00000000 ✓ ตรงกันหมดครับ

⑥ กับดักของข้อนี้

ตอบว่า “ไม่มีค่า” หรือ “undefined” — บางคนเห็นศูนย์ล้วนแล้วไม่กล้าตอบครับ ต้องตอบ **0** ให้ชัด

สับสนกับ −0 — จำให้แม่นว่า 00000000 คือ **บวกศูนย์** เสมอ (MSB เป็น 0) ส่วน **ลบศูนย์** เป็นรหัสคนละตัว: sign-magnitude ใช้ 10000000 · 1's complement ใช้ 11111111 · ส่วน 2's complement **ไม่มีลบศูนย์** เพราะทำ 2's complement ของ 00000000 แล้วยังได้ 00000000 เหมือนเดิมครับ

หัวใจของข้อนี้: MSB เป็น 0 เมื่อไหร่ ทั้งสามรูปแบบให้ค่าเดียวกันเสมอ

ข้อ 11 (b)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 11 (b)

> Consider the following 8-bit binary numbers. Determine the equivalent decimal values assuming sign-magnitude, 1's complement, and 2's complement forms: b. 10010100_2

แปล: จงหาค่าฐานสิบของ 10010100_2 ทั้งสามรูปแบบครับ

โจทย์ซ่อนอะไรไว้: MSB เป็น **1** — นี่คือข้อจริงของหัวข้อนี้เลยครับ บิตชุดเดียวกันจะให้**ค่าต่างกันสามค่า** และต้องติดลบทั้งสามค่า ห้ามมีช่องไหนออกมาเป็นบวก

② เตรียมก่อนลงมือ

แผนคือ: **① ยืนยันว่า MSB เป็น 1 → เป็นลบทั้งสามแบบ → ② แปลงกลับตามกฎของแต่ละรูปแบบ → ③ ใส่เครื่องหมายลบ** ครับ

พี่จะทำคู่กับ**วิธีน้ำหนักติดลบของสไลด์**ไปด้วย เพื่อให้เห็นว่าสองวิธีตรงกัน

③ ไล่ทีละขั้นพร้อมภาพ



อ่านบิตซ้ายสุดก่อนเสมอ ถ้าเป็น 0 ทั้งสามแบบให้ค่าเท่ากันหมด ถ้าเป็น 1 ค่าจะแยกกันสามทางครับ

sign-magnitude

MSB = 1 แปลว่าลบ · เอา 7 บิตที่เหลือ **0010100** มาอ่านตรง ๆ ได้ 20

= -20

1's complement

MSB = 1 แปลว่าลบ · พลิกกลับได้ **01101011** = 107

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้ว**บวก 1** ท้ายสุด: $-128 + 20 + 1 = -107$

= -107

2's complement

MSB = 1 แปลว่าลบ · ทำ 2's complement กลับได้ **01101100** = 108

อีกวิธี (แบบสไลด์) ให้ MSB มีน้ำหนักติดลบ แล้วบวกน้ำหนักทุกหลักตรง ๆ: $-128 + 20 = -108$

= -108

1

MSB $= 1 \rightarrow$ ลบ
magnitude $= 0010100_2$
$= 16 + 4 = 20$

sign-magnitude — ตัด MSB ทิ้งแล้วอ่าน 7 บิตที่เหลือ

นิยามบอกว่า magnitude bits เป็น true binary เพราะฉะนั้นอ่านตรง ๆ ได้ $0010100_2 = 16 + 4 = 20$ แล้วใส่ลบตาม MSB ได้ **−20** ครับ

2

พลิกทุกบิต:
01101011_2
$= 64+32+8+2+1 = 107$

1's complement — พลิกกลับแล้วอ่าน

พลิก 10010100 ได้ 01101011 อ่านค่าได้ 107 แล้วใส่ลบได้ **−107** ครับ

3

2's complement กลับ:
01101100_2
$= 64+32+8+4 = 108$

2's complement — ทำ 2's complement กลับแล้วอ่าน

พลิกได้ 01101011 บวก 1 ได้ 01101100 อ่านค่าได้ 108 แล้วใส่ลบได้ **−108** ครับ

④ คำตอบ

sign-magnitude = -20

1's complement = -107

2's complement = -108

“10010100 equals −20 (sign-magnitude), −107 (1's complement), and −108 (2's complement).”

⑤ ตรวจด้วยวิธีที่สอง — วิธีน้ำหนักติดลบของสไลด์

น้ำหนักของบิตที่เป็น 1 (ไม่รวม MSB) คือ $16 + 4 = 20$ และ MSB มีน้ำหนัก -128 ครับ

1's complement: $-128 + 20 + 1 = \mathbf{-107}$ ✓ (สไลด์บอกว่าต้องบวก 1 ท้ายสุด)

2's complement: $-128 + 20 = \mathbf{-108}$ ✓ (ไม่ต้องบวก 1)

sign-magnitude: ไม่ใช้น้ำหนักติดลบ แต่บวกน้ำหนักของ magnitude bits ล้วน ๆ ได้ 20 แล้วดู MSB ว่าเป็นลบ → $\mathbf{-20}$ ✓

เช็คขั้นสุดท้าย: ค่าของ 1's กับ 2's ต้องห่างกัน 1 พอดี — -107 กับ -108 ห่างกัน 1 ✓ และเขียน -20 กลับเป็น 2's complement 8 บิตได้ 11101100 ซึ่ง**ไม่ใช่** 10010100 ยืนยันว่าสามค่านี้ต่างกันจริงครับ

⑥ กับดักของข้อนี้

① **ตอบค่าเดียวใช้ทั้งสามช่อง** — ข้อนี้ MSB เป็น 1 ทั้งสามค่า**ต้องไม่เท่ากัน**ครับ ถ้าน้องได้เท่ากันหมดแปลว่าอ่านผิดรูปแบบ

② **ลืมใส่เครื่องหมายลบ** — คนคำนวณถูกแล้วเขียน 107 เฉย ๆ เสียคะแนนฟรีครับ ให้ท่องว่า **MSB เป็น 1 = คำตอบต้องมีขีดลบทุกช่อง**

③ **สับสน 1's กับ 2's** — จำง่าย ๆ ว่า 2's complement ให้ค่าที่**ลบมากกว่า**อยู่ 1 เสมอสำหรับบิตชุดเดียวกันครับ

หัวใจของข้อนี้: บิตชุดเดียวกันไม่มีค่าตายตัว จนกว่าจะบอกว่าอ่านด้วยกติกาไหน

10 เฉลย Assignment #1 ข้อ 12–13 (บวก และ ลบ)

อ่านตารางในเฉลยยังไง

ทุกข้อจะมี **ภาพไล่หลัก** (เห็นตัวทด/ตัวยืมสีแดงบนหัวทุกหลัก) · **รูปเขียนลงกระดาษสอบ** (แบบที่น้องต้องเขียนจริงตอนสอบ) · และ**ตารางไล่ทีละหลัก** ที่บอกว่าแต่ละหลักคิดยังไง ห้ามข้ามหลักเดียวครับ

ข้อ 12 (a)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 12 (a)

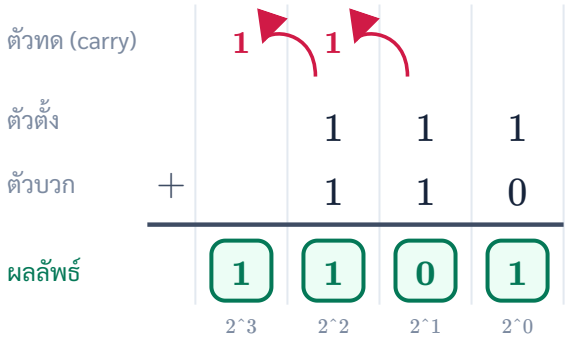
> Add the following binary numbers. a. 111+110

แปล: จงบวก $111_2 + 110_2$ ครับ
โจทย์ซ่อนอะไรไว้: ตัวตั้งเป็น 1 ทั้งสามบิต แปลว่า**ตัวทดจะวิ่งต่อกัน** และคำตอบจะ**ยาวเกิน 3 บิต** ต้องเผื่อที่ให้บิตที่ 4 ครับ

② เตรียมก่อนลงมือ

ใช้กฎบวก 4 บรรทัด ไล่จาก LSB ไป MSB ครับ — เลือกวิธีนี้เพราะโจทย์บอกให้บวกในฐานสองตรง ๆ (ห้ามแปลงไปฐานสิบแล้วแปลงกลับ เพราะข้อสอบต้องการเห็นตัวทด)

③ ไล่ทีละขั้นพร้อมภาพ



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 7 + 6 = 13

111 + 110 (แบบเขียนลงกระดาษสอบ)

```
    1 1
    1 1 1
+   1 1 0
-----------
  1 1 0 1
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	0	1	**1**	**0**
2^1	0	1	1	2	**0**	**1**
2^2	1	1	1	3	**1**	**1**

④ คำตอบ

$111_2 + 110_2 = \mathbf{1101_2}$

“111 + 110 = 1101.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $111_2 = 4+2+1 = 7$ และ $110_2 = 4+2 = 6 \rightarrow 7+6 = \mathbf{13}$

ฝั่งขวา: $1101_2 = 8+4+0+1 = \mathbf{13}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการลบกลับ: $1101 - 110 = 111$ ✓ ได้ตัวตั้งเดิมครับ

⑥ กับดักของข้อนี้

ตอบแค่ 3 บิต — ตัวทดตัวสุดท้ายที่หลัก 2^2 ต้องออกไปเป็นบิตที่ 4 ครับ ถ้าลืมจะตอบ $101_2 = 5$ ซึ่งผิดไป 8 พอดี (คือค่าของบิตที่หายไป)

หัวใจของข้อนี้: ตัวทดที่ออกจากหลักซ้ายสุด คือบิตใหม่ ห้ามทิ้ง

ข้อ 12 (b)

① โจทย์ต้นฉบับ Assignment #1 · ข้อ 12 (b)

> Add the following binary numbers. b. 1001 + 101

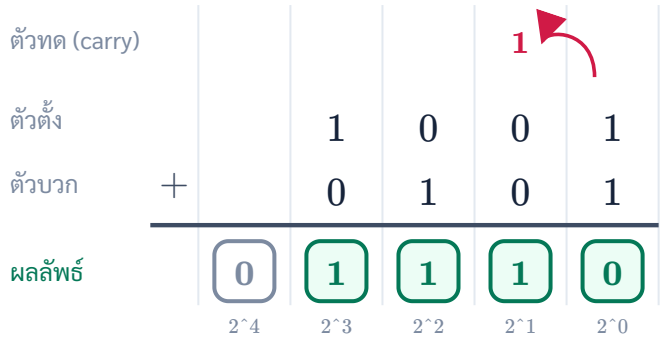
แปล: จงบวก $1001_2 + 101_2$ ครับ

โจทย์ซ่อนอะไรไว้: เลขสองตัว**ยาวไม่เท่ากัน** (4 บิต กับ 3 บิต) — ต้องเติมศูนย์นำหน้าให้เป็น 0101 ก่อนตั้งหลัก ถ้าตั้งเหลื่อมกันน้ำหนักจะผิดทันทีครับ

② เตรียมก่อนลงมือ

ขั้นแรกคือ**ทำให้ยาวเท่ากันก่อน** $101 \rightarrow 0101$ (เติมศูนย์นำหน้าไม่เปลี่ยนค่า) แล้วค่อยใช้กฎบวก 4 บรรทัดตามปกติครับ

③ ไล่ทีละขั้นพร้อมภาพ



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 9 + 5 = 14

1001 + 0101 (แบบเขียนลงกระดาษสอบ)

```
      1
    1 0 0 1
+   0 1 0 1
-----------
  0 1 1 1 0
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	1	2	**0**	**1**
2^1	1	0	0	1	**1**	**0**
2^2	0	0	1	1	**1**	**0**
2^3	0	1	0	1	**1**	**0**

④ คำตอบ

$1001_2 + 101_2 = \mathbf{1110_2}$

“1001 + 101 = 1110.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1001_2 = 8 + 1 = 9$ และ $101_2 = 4 + 1 = 5 \rightarrow 9 + 5 = \mathbf{14}$

ฝั่งขวา: $1110_2 = 8 + 4 + 2 = \mathbf{14}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการลบกลับ: $1110 - 101 = 1001$ ✓

⑥ กับดักของข้อนี้

ตั้งหลักเหลื่อม — ถ้าเอา 101 ไปวางชิดซ้ายแทนที่จะชิดขวา จะกลายเป็นบวก 1010 ได้คำตอบ $10011 = 19$ ซึ่งผิดครับ **ให้ยึดบิตขวาสุด (LSB) ให้ตรงกันเสมอ** ไม่ใช่บิตซ้ายสุด

หัวใจของข้อนี้: เลขยาวไม่เท่ากันให้เติมศูนย์นำหน้า แล้วยึด LSB ให้ตรงกัน

ข้อ 12 (c)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 12 (c)

> Add the following binary numbers. c. 1101+1011

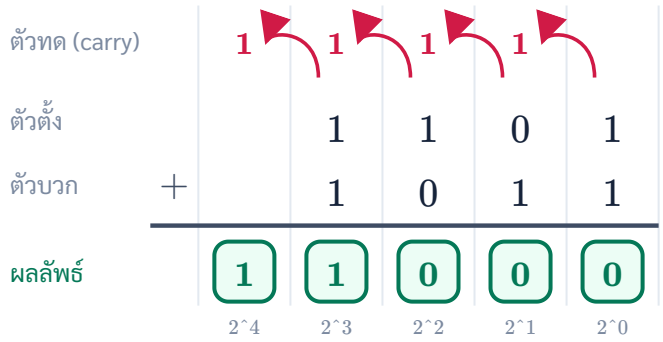
แปล: จงบวก $1101_2 + 1011_2$ ครับ

โจทย์ซ่อนอะไรไว้: ข้อนี้**ตัวทดวิ่งครบทั้ง 4 หลัก** เป็นข้อที่ยากที่สุดในสามข้อย่อย ต้องระวังกรณี $1 + 1 + 1 = 11$ (เขียน 1 ทด 1) ซึ่งจะโผล่มาที่หลักซ้ายสุดครับ

② เตรียมก่อนลงมือ

ใช้ทางลัด **“นับจำนวน 1 ในหลักนั้น รวมตัวทดด้วย — คี่เขียน 1 คู่เขียน 0 · ได้ตั้งแต่ 2 ขึ้นไปให้ทด 1”** ครับ วิธีนี้จัดการกรณี $1 + 1 + 1$ ได้โดยไม่ต้องท่องกฎเพิ่ม

③ ไล่ทีละขั้นพร้อมภาพ



บวกทีละหลักจากขวาไปซ้าย — ตัวทดสีแดงคือตัวที่หลักขวาส่งมาให้หลักนี้บวกด้วยครับ ตรวจกลับเป็นฐานสิบ: 13 + 11 = 24

1101 + 1011 (แบบเขียนลงกระดาษสอบ)

```
    1 1 1 1
      1 1 0 1
+     1 0 1 1
-------------
  1 1 0 0 0
```

หลัก	ตัวทดที่รับมา	บิตตัวตั้ง	บิตตัวบวก	รวมได้	เขียนไว้	ทดขึ้นหลักซ้าย
2^0	0	1	1	2	**0**	**1**
2^1	1	0	1	2	**0**	**1**
2^2	1	1	0	2	**0**	**1**
2^3	1	1	1	3	**1**	**1**

④ คำตอบ

$1101_2 + 1011_2 = \mathbf{11000_2}$

“1101 + 1011 = 11000.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1101_2 = 8+4+1 = 13$ และ $1011_2 = 8+2+1 = 11 \rightarrow 13+11 = \mathbf{24}$

ฝั่งขวา: $11000_2 = 16+8 = \mathbf{24}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการลบกลับ: $11000 - 1011 = 1101$ ✓

⑥ กับดักของข้อนี้

เจอ $1+1+1$ **แล้วเขียน 0 ทด 1** — ผิดครับ ต้อง**เขียน 1 ทด 1** เพราะ $1+1+1 = 3 = 11_2$ ถ้าเผลอเขียน 0 จะได้ $10000 = 16$ ผิดไป 8

วิธีกันคือใช้ทางลัดนับจำนวน 1: ได้ 3 ตัว → **เลขคี่ → เขียน 1** ครับ

หัวใจของข้อนี้: $1+1+1$ **ให้ผลไม่เหมือน** $1+1$ **— เขียน 1 ไม่ใช่ 0**

ข้อ 13 (a)

① โจทย์ต้นฉบับ Assignment #1 · ข้อ 13 (a)

> Subtract the following binary numbers. a. 1110-11

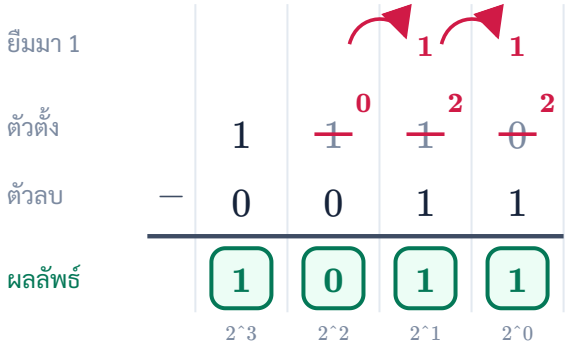
แปล: จงลบ $1110_2 - 11_2$ ครับ

โจทย์ซ่อนอะไรไว้: ยาวไม่เท่ากันอีกแล้ว ($11 \rightarrow 0011$) และ**จะเกิดการยืมสองครั้งติดกัน** เพราะบิตขวาสุดของตัวตั้งเป็น 0 แต่ตัวลบเป็น 1 ครับ

② เตรียมก่อนลงมือ

เติมศูนย์ให้ตัวลบเป็น 0011 ก่อน แล้วใช้กฎลบ 4 บรรทัด ไล่จาก LSB ครับ ทุกครั้งที่เจอ $0 - 1$ ให้จำว่ามี**สองผล**: หลักนี้ได้ +2 และ**หลักซ้ายโดนหัก −1**

③ ไล่ทีละขั้นพร้อมภาพ



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 14 − 3 = 11

1110 − 0011 (แบบเขียนลงกระดาษสอบ)

```
      1 1
  1 1 1 0
- 0 0 1 1
---------
  1 0 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	**ยืมมา +2**	—	**2**	1	2 − 1 = **1**
2^1	1	**ยืมมา +2**	**จ่ายไป −1**	**2**	1	2 − 1 = **1**
2^2	1	—	**จ่ายไป −1**	**0**	0	0 − 0 = **0**
2^3	1	—	—	**1**	0	1 − 0 = **1**

④ คำตอบ

$1110_2 - 11_2 = \mathbf{1011_2}$

“1110 − 11 = 1011.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1110_2 = 8+4+2 = 14$ และ $11_2 = 3 \rightarrow 14 - 3 = \mathbf{11}$

ฝั่งขวา: $1011_2 = 8+2+1 = \mathbf{11}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการบวกกลับ: $1011 + 0011 = 1110$ ✓ ได้ตัวตั้งเดิมครับ

⑥ กับดักของข้อนี้

ยืมแล้วลืมหักหลักที่ให้ยืม — หลัก 2^1 ให้หลัก 2^0 ยืมไป 1 ตัวมันเองจึงเหลือ 0 ไม่ใช่ 1 ครับ ถ้าลืมหักจะได้คำตอบ $1101 = 13$ ซึ่งมากเกินไป 2

ทดสอบง่าย ๆ: ผลลัพธ์ของการลบต้อง**ไม่มากกว่าตัวตั้ง**เสมอ ถ้าน้องได้คำตอบใหญ่กว่า 1110 แปลว่าผิดแน่นอนครับ

หัวใจของข้อนี้: การยืมมีสองผลเสมอ — ได้ +2 ที่หลักนี้ และ −1 ที่หลักซ้าย

ข้อ 13 (b)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 13 (b)

> Subtract the following binary numbers. b. 1100-1001

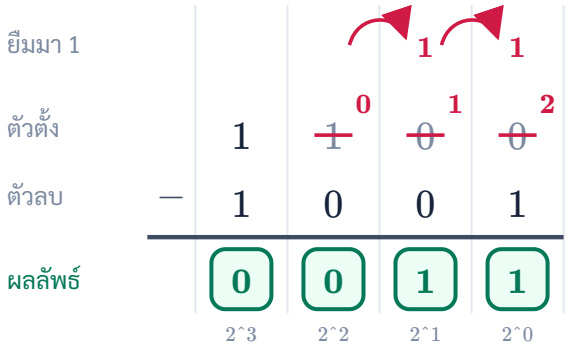
แปล: จงลบ $1100_2 - 1001_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวเลขทั้งสองยาว 4 บิตเท่ากันแล้ว แต่**บิตขวาสุดเป็น 0 ลบ 1** และ**บิตถัดไปก็เป็น 0** ด้วย ทำให้เกิด**การยืมต่อทอด** (ยืมแล้วต้องยืมต่ออีกที) ครับ

② เตรียมก่อนลงมือ

ไล่จาก LSB เหมือนเดิมครับ พี่จะเน้นให้ดูว่าเมื่อหลักที่ต้องให้ยืมเป็น 0 เอง มันจะ**ไปยืมจากหลักซ้ายของมันต่อ** — นี่คือหัวใจของข้อนี้

③ ไล่ทีละขั้นพร้อมภาพ



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 12 − 9 = 3

1100 − 1001 (แบบเขียนลงกระดาษสอบ)

```
      1 1
  1 1 0 0
- 1 0 0 1
---------
  0 0 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	**ยืมมา +2**	—	**2**	1	2 − 1 = **1**
2^1	0	**ยืมมา +2**	**จ่ายไป −1**	**1**	0	1 − 0 = **1**
2^2	1	—	**จ่ายไป −1**	**0**	0	0 − 0 = **0**
2^3	1	—	—	**1**	1	1 − 1 = **0**

④ คำตอบ

$1100_2 - 1001_2 = \mathbf{0011_2}$ **(หรือเขียนว่า** 11_2**)**

“1100 − 1001 = 0011.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1100_2 = 8 + 4 = 12$ และ $1001_2 = 8 + 1 = 9 \rightarrow 12 - 9 = \mathbf{3}$

ฝั่งขวา: $0011_2 = 2 + 1 = \mathbf{3}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการบวกกลับ: $0011 + 1001 = 1100$ ✓

⑥ กับดักของข้อนี้

ยืมข้ามหลัก — บางคนเห็นว่าหลัก 2^1 เป็น 0 เลยกระโดดไปยืมจากหลัก 2^3 ทันที **ผิดครับ** ต้องยืมทีละหลักเป็นทอด ๆ: หลัก 2^0 ยืมจาก $2^1 \cdot 2^1$ ไม่มีของก็ยืมจาก 2^2 ต่อ

ผลของการทำถูกคือหลัก 2^2 ที่เดิมเป็น 1 จะเหลือ 0 และคำตอบหลัก 2^2 ได้ 0 ครับ ถ้ายืมข้ามหลัก หลัก 2^2 จะค้างเป็น 1 แล้วหลัก 2^3 จะติดลบ ทำต่อไม่ได้

หัวใจของข้อนี้: การยืมเดินทีละหลัก ห้ามกระโดด

ข้อ 13 (c)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 13 (c)

> Subtract the following binary numbers. c. 11010-10111

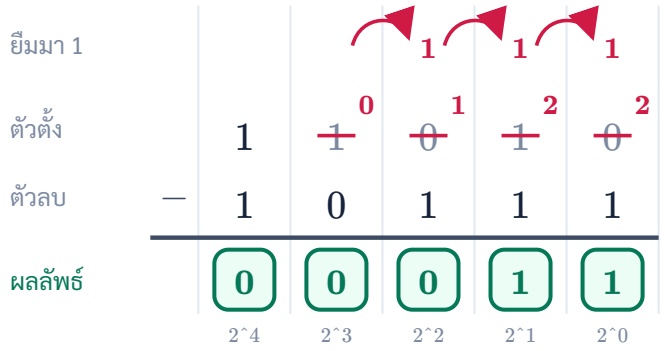
แปล: จงลบ $11010_2 - 10111_2$ ครับ

โจทย์ซ่อนอะไรไว้: ข้อนี้**ยืมติดกันถึงสามหลัก** และผลลัพธ์จะเล็กมาก (ตัวตั้งกับตัวลบต่างกันแค่ 3) เป็นข้อที่พลาดกันเยอะที่สุดของชุดนี้ครับ

② เตรียมก่อนลงมือ

ทั้งสองตัวยาว 5 บิตเท่ากันแล้ว ไม่ต้องเติมศูนย์ครับ พี่แนะนำให้**เขียนค่าที่ใช้จริงกำกับไว้บนบิตทุกตัวที่โดนแก้** (ตัวเลขสีแดงในภาพ) เพราะข้อนี้มีการยืมเยอะ ถ้าคิดในหัวล้วน ๆ จะหลงแน่นอน

③ ไล่ทีละขั้นพร้อมภาพ



ลบทีละหลักจากขวาไปซ้าย — เลข 1 สีแดงบนหัวหลักไหน แปลว่าหลักนั้นยืมจากหลักซ้ายมา 1 แล้วหลักซ้ายจะถูกหักไป 1 ทันที · ตรวจกลับเป็นฐานสิบ: 26 − 23 = 3

11010 − 10111 (แบบเขียนลงกระดาษสอบ)

```
        1 1 1
    1 1 0 1 0
  - 1 0 1 1 1
-------------
    0 0 0 1 1
```

หลัก	บิตตัวตั้ง	ยืมจากหลักซ้าย	ให้หลักขวายืม	ค่าที่ใช้จริง	บิตตัวลบ	คิดแล้วได้
2^0	0	ยืมมา +2	—	**2**	1	2 − 1 = **1**
2^1	1	ยืมมา +2	จ่ายไป −1	**2**	1	2 − 1 = **1**
2^2	0	ยืมมา +2	จ่ายไป −1	**1**	1	1 − 1 = **0**
2^3	1	—	จ่ายไป −1	**0**	0	0 − 0 = **0**
2^4	1	—	—	**1**	1	1 − 1 = **0**

④ คำตอบ

$11010_2 - 10111_2 = \mathbf{00011_2}$ **(หรือเขียนว่า** 11_2**)**

“11010 − 10111 = 00011.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $11010_2 = 16 + 8 + 2 = 26$ และ $10111_2 = 16 + 4 + 2 + 1 = 23 \rightarrow 26 - 23 = \mathbf{3}$

ฝั่งขวา: $00011_2 = 2 + 1 = \mathbf{3}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการบวกกลับ: $00011 + 10111 = 11010$ ✓ ได้ตัวตั้งเดิมพอดีครับ

⑥ กับดักของข้อนี้

① **หลัก** 2^2 **ที่ตัวตั้งเป็น 0** ต้อง**ยืมมา +2 แล้วจ่ายให้หลักขวา −1** พร้อมกัน เหลือค่าใช้จริง 1 ครับ คนที่คิดว่าค่าใช้จริงเป็น -1 จะไปต่อไม่ถูก

② **เห็นตัวเลขใกล้กันแล้วเดาคำตอบ** — 26 กับ 23 ต่างกันแค่ 3 ถ้าน้องได้คำตอบใหญ่กว่า 00100 แปลว่ายืมผิดที่ไหนสักแห่ง **ให้บวกกลับตรวจทันที**

หัวใจของข้อนี้: หลักที่ทั้งยืมมาและให้ยืมไปพร้อมกัน ค่าใช้จริงคือ บิตเดิม + 2 − 1

11 เฉลย Assignment #1 ข้อ 14–15 (คูณ และ หาร)

ข้อ 14 (a)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 14 (a)

> Perform the following binary multiplications. a. 1001 x 11

แปล: จงคูณ $1001_2 \times 11_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวคูณมี **2 บิต** แปลว่าต้องมี**แถวย่อย 2 แถวพอดี** และเพราะทั้งสองบิตเป็น 1 หมด จึงต้องลอกตัวตั้งลงมาทั้งสองแถว ไม่มีแถวศูนย์ครับ

② เตรียมก่อนลงมือ

ใช้วิธีแถวย่อยตามสไลด์ครับ: **หนึ่งแถวต่อหนึ่งบิตของตัวคูณ (เริ่มจาก LSB) → บิตเป็น 1 ลอกตัวตั้ง บิตเป็น 0 ใส่ศูนย์ → เลื่อนซ้ายทีละหลัก → บวกรวม**

เช็คขนาดคำตอบล่วงหน้า: ตัวตั้ง 4 บิต × ตัวคูณ 2 บิต → คำตอบยาวไม่เกิน 6 บิตครับ

③ ไล่ทีละขั้นพร้อมภาพ

บิตที่ 0 ของตัวคูณ = 1		1	0	0	1
บิตที่ 1 ของตัวคูณ = 1	1	0	0	1	0
บวกทุกแถวรวมกัน	**1**	**1**	**0**	**1**	**1**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 9 × 3 = 27

1001 × 11 (แบบเขียนลงกระดาษสอบ)

```
        1 0 0 1
    x       1 1
-------------------
        1 0 0 1
      1 0 0 1 0
-------------------
      1 1 0 1 1
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	1	**1001**	0 หลัก	9

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^1	1	**10010**	1 หลัก	18
บวกทุกแถวย่อยเข้าด้วยกัน				**27**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
    0 1 0 0 1
+   1 0 0 1 0
-------------
  0 1 1 0 1 1
```

④ คำตอบ

$1001_2 \times 11_2 = \mathbf{11011_2}$

“1001 × 11 = 11011.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1001_2 = 8 + 1 = 9$ และ $11_2 = 2 + 1 = 3 \rightarrow 9 \times 3 = \mathbf{27}$

ฝั่งขวา: $11011_2 = 16 + 8 + 2 + 1 = \mathbf{27}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการหารกลับ: $11011 \div 11 = 1001$ เศษ 0 ✓

⑥ กับดักของข้อนี้

ลืมเลื่อนแถวที่สอง — ถ้าเขียน $1001 + 1001$ (ไม่เลื่อน) จะได้ $10010 = 18$ ซึ่งคือ 9×2 ไม่ใช่ 9×3 ครับ

ให้จำว่า**แถวที่ k นับจาก 0 ต้องเลื่อนซ้าย k หลัก** เพราะบิตนั้นมีน้ำหนัก 2^k

หัวใจของข้อนี้: การคูณด้วย 11_2 คือ “ตัวเลขเดิม + ตัวเลขเดิมเลื่อนซ้ายหนึ่งหลัก”

ข้อ 14 (b)

① โจทย์ต้นฉบับ Assignment #1 · ข้อ 14 (b)

> Perform the following binary multiplications. b. 1101 x 1101

แปล: จงคูณ $1101_2 \times 1101_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวคูณ 4 บิต ต้องมี**แถวย่อย 4 แถว** และมี**แถวศูนย์อยู่หนึ่งแถว** (บิต 2^1 เป็น 0) ห้ามข้ามแถวนั้นไม่งั้นแถวถัดไปจะเลื่อนผิดตำแหน่งครับ

② เตรียมก่อนลงมือ

วิธีเดิมครับ แต่ข้อนี้แถวเยอะ พี่จะ**บวกสะสมทีละคู่**แทนที่จะบวกสี่แถวรวดเดียว เพราะการบวกฐานสองพร้อมกันหลายแถวทำให้ตัวทดสับสนง่ายมาก

เช็คขนาด: 4 บิต × 4 บิต → คำตอบยาวไม่เกิน 8 บิตครับ

③ ไล่ทีละขั้นพร้อมภาพ

บิตที่ 0 ของตัวคูณ = 1				1	1	0	1
บิตที่ 1 ของตัวคูณ = 0			0	0	0	0	0
บิตที่ 2 ของตัวคูณ = 1		1	1	0	1	0	0
บิตที่ 3 ของตัวคูณ = 1	1	1	0	1	0	0	0
บวกทุกแถวรวมกัน	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 13 × 13 = 169

1101 × 1101 (แบบเขียนลงกระดาษสอบ)

```
          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
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	1	**1101**	0 หลัก	13
2^1	0	**00000**	1 หลัก	0
2^2	1	**110100**	2 หลัก	52
2^3	1	**1101000**	3 หลัก	104
บวกทุกแถวย่อยเข้าด้วยกัน				**169**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
  1 1 1 1
    0 0 1 1 0 1
+   1 1 0 1 0 0
-------------------
  1 0 0 0 0 0 1
```

รวมสะสมรอบที่ 2

```
  1
    1 0 0 0 0 0 1
+   1 1 0 1 0 0 0
-------------------
  1 0 1 0 1 0 0 1
```

④ คำตอบ

$1101_2 \times 1101_2 = \mathbf{10101001_2}$

“1101 × 1101 = 10101001.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1101_2 = 8 + 4 + 1 = 13 \rightarrow 13 \times 13 = \mathbf{169}$

ฝั่งขวา: $10101001_2 = 128 + 32 + 8 + 1 = \mathbf{169}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการหารกลับ: $10101001 \div 1101 = 1101$ เศษ 0 ✓

⑥ กับดักของข้อนี้

① **ข้ามแถวศูนย์** — บิต 2^1 ของตัวคูณเป็น 0 บางคนเลยไม่เขียนแถวนั้น แล้วเลื่อนแถวถัดไปแค่ 2 หลักแทนที่จะเป็น 3 หลัก คำตอบจะผิดทันทีครับ

② **บวกสี่แถวพร้อมกันแล้วตัวทดหลง** — ในหลักที่มี 1 หลายตัวซ้อนกัน ตัวทดอาจเป็น 2 ได้ (เช่น $1 + 1 + 1 + 1 = 100_2$) ซึ่งจัดการยาก **บวกทีละคู่ปลอดภัยกว่ามาก** ตามที่พี่ทำให้ดูข้างบน

หัวใจของข้อนี้: เขียนแถวย่อยให้ครบทุกแถว แล้วบวกสะสมทีละคู่

ข้อ 14 (c)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 14 (c)

> Perform the following binary multiplications. c. 1110 x 1101

แปล: จงคูณ $1110_2 \times 1101_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวคูณเป็นตัวเดียวกับข้อ (b) เป๊ะ ต่างกันแค่ตัวตั้ง — แปลว่า**รูปแบบการเลื่อนแถวเหมือนกันทุกประการ** เปลี่ยนแค่เนื้อในของแถวครับ และตัวตั้งลงท้ายด้วย 0 แปลว่าคำตอบต้องเป็น**เลขคู่**

② เตรียมก่อนลงมือ

ใช้วิธีเดิมครับ ตัวคูณ 1101 มีบิต $2^0, 2^2, 2^3$ เป็น 1 และบิต 2^1 เป็น 0 เพราะฉะนั้นจะมีแถวที่ต้องบวกจริง 3 แถว และแถวศูนย์ 1 แถว

③ ไล่ทีละขั้นพร้อมภาพ

บิตที่ 0 ของตัวคูณ = 1				1	1	1	0
บิตที่ 1 ของตัวคูณ = 0			0	0	0	0	0
บิตที่ 2 ของตัวคูณ = 1		1	1	1	0	0	0
บิตที่ 3 ของตัวคูณ = 1	1	1	1	0	0	0	0
บวกทุกแถวรวมกัน	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**

หนึ่งแถวย่อยต่อหนึ่งบิตของตัวคูณ · บิตเป็น 1 ให้ลอกตัวตั้งลงมา บิตเป็น 0 ให้ใส่ศูนย์ แล้วเลื่อนซ้ายทีละหนึ่งหลัก · ตรวจฐานสิบ: 14 × 13 = 182

1110 × 1101 (แบบเขียนลงกระดาษสอบ)

```
          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
```

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^0	1	**1110**	0 หลัก	14

บิตของตัวคูณ (จากขวา)	ค่าบิต	แถวย่อย (partial product)	เลื่อนซ้าย	ค่าฐานสิบ
2^1	0	**00000**	1 หลัก	0
2^2	1	**111000**	2 หลัก	56
2^3	1	**1110000**	3 หลัก	112
บวกทุกแถวย่อยเข้าด้วยกัน				**182**

พี่บวกแถวย่อยทีละคู่ให้ดูเลยครับ จะได้เห็นว่าตัวทดวิ่งยังไง (แถวที่เป็นศูนย์ล้วนไม่ต้องบวก เพราะบวกแล้วก็เท่าเดิม):

รวมสะสมรอบที่ 1

```
  1 1 1
    0 0 1 1 1 0
+   1 1 1 0 0 0
  ---------------
  1 0 0 0 1 1 0
```

รวมสะสมรอบที่ 2

```
  1
    1 0 0 0 1 1 0
+   1 1 1 0 0 0 0
  ---------------
  1 0 1 1 0 1 1 0
```

④ คำตอบ

$1110_2 \times 1101_2 = \mathbf{10110110_2}$

“1110 × 1101 = 10110110.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1110_2 = 8+4+2 = 14$ และ $1101_2 = 8+4+1 = 13 \rightarrow 14 \times 13 = \mathbf{182}$

ฝั่งขวา: $10110110_2 = 128+32+16+4+2 = \mathbf{182}$ ✓ **ตรงกัน**

ตรวจเพิ่มด้วยการดูบิตท้าย: ตัวตั้งลงท้าย 0 (เลขคู่) คำตอบจึงต้องลงท้ายด้วย 0 ด้วย — 10110110 ลงท้าย 0 ✓ ถ้าน้องได้คำตอบลงท้ายด้วย 1 แปลว่าผิดแน่นอนโดยไม่ต้องคิดต่อเลยครับ

⑥ กับดักของข้อนี้

ลอกคำตอบข้อ (b) มาแก้นิดเดียว — ตัวคูณเหมือนกันจริง แต่ตัวตั้งต่างกัน ผลลัพธ์ต่างกันคนละค่า (169 กับ 182) ต้องคำนวณใหม่ทั้งหมดครับ

สับสน 1110 กับ 1101 — สองตัวนี้หน้าตาคล้ายกันมาก ให้**วงกลมตัวตั้งกับตัวคูณไว้ก่อนเริ่มทำ** จะได้ไม่สลับกันครับ

หัวใจของข้อนี้: บิตท้ายของคำตอบ = บิตท้ายของตัวตั้ง AND บิตท้ายของตัวคูณ — ใช้เช็คเร็ว ๆ ได้

ข้อ 15 (a)

① โจทย์ต้นฉบับ Assignment #1 · ข้อ 15 (a)

> Divide the following binary numbers. a. 100 ÷ 10

แปล: จงหาร $100_2 \div 10_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวหารเป็น $10_2 = 2$ ซึ่งเป็น**กำลังของสองพอดี** แปลว่าใช้ทางลัด “ตัดเลขท้าย 1 ตัว” เช็คคำตอบได้ทันทีครับ

② เตรียมก่อนลงมือ

ทำหารยาวตามสไลด์ครับ: ไล่บิตจากซ้ายมาขวา ทุกรอบถามว่า**“ลบตัวหารออกได้ไหม”** ได้ใส่ 1 ไม่ได้ใส่ 0 — และตัวตั้งมี 3 บิต จึงต้องได้ผลหารดิบ 3 บิตเสมอ

③ ไล่ทีละขั้นพร้อมภาพ

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 10 (=2) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	0	10 (=2)	ลบได้	**1**	0
3	0	0 (=0)	ยังลบไม่ได้	**0**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 4 ÷ 2 = 2 เศษ 0

100 ÷ 10 (หารยาวแบบเขียนลงกระดาษสอบ)

```
          1 0
        -----
1 0 ) 1 0 0
    - 1 0
    -----
        0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
10 x 10 = 100
100 + 0 = 100
ฐานสิบ: 2 x 2 + 0 = 4
```

④ คำตอบ

$100_2 \div 10_2 = \mathbf{10_2}$ **เศษ 0**

“100 ÷ 10 = 10 with remainder 0.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $100_2 = 4$ และ $10_2 = 2 \rightarrow 4 \div 2 = \mathbf{2}$ เศษ 0

ฝั่งขวา: $10_2 = \mathbf{2}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการคูณกลับ: $10_2 \times 10_2 = 100_2$ ✓

ตรวจด้วยทางลัด: หารด้วย $10_2 = 2^1$ ก็แค่ตัดเลขท้าย 1 ตัวของ 100 เหลือ 10 ✓

⑥ กับดักของข้อนี้

อ่านคำตอบเป็น “สิบ” — 10_2 ในข้อนี้อ่านว่า **หนึ่ง-ศูนย์** และมีค่าเท่ากับ **2** ครับ ไม่ใช่สิบ ข้อสอบวิชานี้จะไม่มีดัชนีฐานกำกับให้เสมอไป ต้องดูบริบทเอง

ลืมใส่ 0 ในรอบแรก — รอบแรกดึงได้ $1_2 = 1$ ซึ่งน้อยกว่า 2 จึงต้องใส่ 0 ครับ ถ้าไม่ใส่จะได้ผลหารสั้นไปหนึ่งบิต

หัวใจของข้อนี้: หารด้วย 2^k คือเลื่อนขวา k หลัก ตรวจได้ในหนึ่งวินาที

ข้อ 15 (b)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 15 (b)

> Divide the following binary numbers. b. 1001 ÷ 11

แปล: จงหาร $1001_2 \div 11_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวหาร $11_2 = 3$ **ไม่ใช่กำลังของสอง** จึงใช้ทางลัดตัดท้ายไม่ได้ ต้องหารยาวจริง ๆ ทีละรอบครับ

② เตรียมก่อนลงมือ

หารยาว 4 รอบ (เพราะตัวตั้งมี 4 บิต) ครับ พี่จะเขียน**ค่าฐานสิบกำกับไว้ทุกรอบ** เพื่อให้เทียบกับตัวหาร $= 3$ ได้ง่าย ๆ ว่าลบได้หรือยัง

③ ไล่ทีละขั้นพร้อมภาพ

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 11 (=3) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	0	10 (=2)	ยังลบไม่ได้	**0**	10
3	0	100 (=4)	ลบได้	**1**	1
4	1	11 (=3)	ลบได้	**1**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 9 ÷ 3 = 3 เศษ 0

1001 ÷ 11 (หารยาวแบบเขียนลงกระดาษสอบ)

```
                1 1
              -------
      1 1 ) 1 0 0 1
            - 1 1
            -----
              1 1
            - 1 1
            -----
                0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
11 x 11 = 1001
1001 + 0 = 1001
ฐานสิบ: 3 x 3 + 0 = 9
```

④ คำตอบ

$1001_2 \div 11_2 = \mathbf{11_2}$ **เศษ 0**

“1001 ÷ 11 = 11 with remainder 0.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1001_2 = 8 + 1 = 9$ และ $11_2 = 3 \rightarrow 9 \div 3 = \mathbf{3}$ เศษ 0

ฝั่งขวา: $11_2 = 2 + 1 = \mathbf{3}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการคูณกลับ: $11_2 \times 11_2 = 1001_2$ ✓ — บังเอิญข้อนี้ตัวหารกับผลหารเท่ากันพอดี เพราะ 9 เป็นกำลังสองของ 3 ครับ

⑥ กับดักของข้อนี้

เผลอใช้ทางลัดตัดท้าย — 11_2 ไม่ใช่กำลังของสอง ถ้าตัดท้าย 2 ตัวจะได้ $10_2 = 2$ ซึ่ง**ผิด** ครับ ทางลัดใช้ได้เฉพาะตอนตัวหารเป็น $10, 100, 1000, \ldots$ เท่านั้น

นับรอบไม่ครบ — ตัวตั้ง 4 บิต ต้องได้ผลหารดิบ 4 บิต (0011) แล้วค่อยตัดศูนย์นำหน้าออกเป็น 11 ครับ

หัวใจของข้อนี้: จำนวนรอบของหารยาว = จำนวนบิตของตัวตั้ง เสมอ

ข้อ 15 (c)

① โจทย์ต้นฉบับ — Assignment #1 · ข้อ 15 (c)

> Divide the following binary numbers. c. 1100 ÷ 100

แปล: จงหาร $1100_2 \div 100_2$ ครับ

โจทย์ซ่อนอะไรไว้: ตัวหาร $100_2 = 4 = 2^2$ เป็นกำลังของสองอีกครั้ง แปลว่าเช็คได้ด้วยการ**ตัดเลขท้าย 2 ตัว** ครับ และข้อนี้คือตัวอย่างเดียวกับที่สไลด์ทิ้งไว้ให้ลองพอดี

② เตรียมก่อนลงมือ

ทำหารยาวเต็มรูปแบบก่อน (เพราะข้อสอบต้องการเห็นขั้นตอน) แล้วค่อยใช้ทางลัดตัดท้ายเป็นตัวตรวจครับ ตัวตั้งมี 4 บิต → 4 รอบ

③ ไล่ทีละขั้นพร้อมภาพ

รอบที่	ดึงบิตลงมา	ค่าปัจจุบัน	ลบด้วย 100 (=4) ได้ไหม	บิตผลหาร	เหลือ
1	1	1 (=1)	ยังลบไม่ได้	**0**	1
2	1	11 (=3)	ยังลบไม่ได้	**0**	11
3	0	110 (=6)	ลบได้	**1**	10
4	0	100 (=4)	ลบได้	**1**	0

ไล่ทีละบิตจากซ้ายมาขวา ทุกรอบถามคำถามเดียวกันว่า “ตัวหารลบออกได้ไหม” ลบได้ใส่ 1 ลบไม่ได้ใส่ 0 · ตรวจฐานสิบ: 12 ÷ 4 = 3 เศษ 0

1100 ÷ 100 (หารยาวแบบเขียนลงกระดาษสอบ)

```
                1 1
              -------
      1 0 0 ) 1 1 0 0
            - 1 0 0
            -------
              1 0 0
            - 1 0 0
            -------
                  0
```

ตรวจกลับ: ผลหาร × ตัวหาร + เศษ = ตัวตั้ง

```
11 x 100 = 1100
1100 + 0 = 1100
ฐานสิบ: 3 x 4 + 0 = 12
```

④ คำตอบ

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

“1100 ÷ 100 = 11 with remainder 0.”

⑤ ตรวจด้วยวิธีที่สอง — แปลงเป็นฐานสิบทั้งสองฝั่ง

ฝั่งซ้าย: $1100_2 = 8 + 4 = 12$ และ $100_2 = 4 \rightarrow 12 \div 4 = \mathbf{3}$ เศษ 0

ฝั่งขวา: $11_2 = \mathbf{3}$ ✓ **ตรงกัน**

ตรวจซ้ำด้วยการคูณกลับ: $11_2 \times 100_2 = 1100_2$ ✓

ตรวจด้วยทางลัด: $100_2 = 2^2$ ตัดเลขท้าย 2 ตัวของ 1100 เหลือ 11 ✓ ตรงกันทั้งสามทางครับ

⑥ กับดักของข้อนี้

ตัดเลขท้ายผิดจำนวน — ตัวหารมี 3 บิต แต่**เลขศูนย์ท้ายมีแค่ 2 ตัว** ต้องตัดตามจำนวนศูนย์ท้าย ไม่ใช่ตามจำนวนบิตของตัวหารครับ ถ้าตัด 3 ตัวจะได้ $1_2 = 1$ ซึ่งผิด

สองรอบแรกลืมใส่ 0 — ตัวหารมี 3 บิต ทำให้รอบที่ 1 และ 2 ลบไม่ได้แน่นอน ต้องใส่ 0 ทั้งสองรอบ แล้วผลหารดิบจะเป็น 0011 ครับ

หัวใจของข้อนี้: หารยาวให้ครบทุกรอบก่อน แล้วค่อยใช้ทางลัดเป็นตัวตรวจ ไม่ใช่ใช้แทน

เก็บครบแล้ว

เฉลย Assignment #1 **ข้อ 8 ถึง 15 ครบทุกข้อย่อยแล้วครับ** — 8 (a, b) · 9 (a, b) · 10 · 11 (a, b) · 12 (a, b, c) · 13 (a, b, c) · 14 (a, b, c) · 15 (a, b, c) รวม **19 ข้อย่อย** ทุกข้อผ่านการตรวจด้วยการแปลงกลับเป็นฐานสิบทั้งสองฝั่งแล้ว

ส่วนข้อ 1–7 เป็นเรื่องเลขฐานและการแปลงฐาน อยู่ใน **EP01** ครับ

12 กับดักรวมของทั้ง EP + สรุปหน้าเดียว

กับดัก 10 ข้อที่ทำให้เสียคะแนนมากที่สุด

#	กับดัก	ผลที่จะเกิด	วิธีกัน
1	**ลืมตัวทดตัวสุดท้ายตอนบวก**	คำตอบสั้นไปหนึ่งบิต ค่าผิดไปเท่ากับ 2^n	เผื่อช่องว่างไว้ทางซ้ายของตัวตั้งเสมอก่อนเริ่มบวก
2	**เจอ $1+1+1$ แล้วเขียน 0**	ผลลัพธ์หลักนั้นผิด ค่าเพี้ยนไป 2^k	ใช้ทางลัดนับจำนวน 1 — คี่เขียน 1 คู่เขียน 0
3	**ยืมแล้วลืมหักหลักที่ให้ยืม**	คำตอบใหญ่เกินจริง บวกกลับแล้วไม่ตรง	เขียนค่าที่ใช้จริงกำกับบนบิตที่โดนแก้ทุกตัว
4	**ยืมข้ามหลัก**	หลักกลางไม่ถูกหัก ผลลัพธ์เพี้ยน	ยืมเป็นทอด ๆ ทีละหลักเท่านั้น
5	**ข้ามแถวศูนย์ตอนคูณ**	แถวถัดไปเลื่อนผิดตำแหน่ง คำตอบเล็กเกินจริง	เขียนให้ครบทุกแถว = จำนวนบิตของตัวคูณ
6	**ลืมใส่ 0 ในรอบที่หารไม่ได้**	ผลหารสั้นไปหนึ่งบิต ค่าผิดไปครึ่งหนึ่ง	ท่องว่า “ทุกบิตของตัวตั้ง = ผลหารหนึ่งบิต”
7	**ไม่กำหนดจำนวนบิตก่อนทำคอมพลีเมนต์**	ได้บิตยาวผิด คำตอบไม่ตรงกับที่โจทย์ถาม	โจทย์เขียนมากี่บิตตอบเท่านั้น เว้นแต่สั่งว่า “8-bit”
8	**พลิกจาก sign-magnitude แทนที่จะพลิกจากเลขบวก**	ได้ MSB เป็น 0 กลายเป็นเลขบวก ผิดทั้งข้อ	ทั้งสามรูปแบบเริ่มจาก**เลขบวก**เสมอ
9	**ตัดศูนย์นำหน้าของคอมพลีเมนต์ทิ้ง**	จำนวนบิตไม่เท่าโจทย์ ถือว่าตอบไม่ครบ	คอมพลีเมนต์ยาวเท่าเลขตั้งต้นเป๊ะเสมอ
10	**ลืมใส่เครื่องหมายลบตอนอ่านค่ากลับ**	คำนวณถูกแต่ตอบผิด เสียคะแนนฟรี	MSB เป็น 1 เมื่อไหร่ คำตอบต้องมีขีดลบทุกช่อง

ตารางเทียบวิธีทำทั้ง EP

เจอโจทย์แบบนี้	ทำแบบนี้	ตรวจแบบนี้
บวกฐานสอง	ไล่จาก LSB · นับจำนวน 1 รวมตัวทด · คี่เขียน 1 · ≥ 2 ทด 1	เอาผลลัพธ์ลบตัวบวก ต้องได้ตัวตั้งเดิม
ลบฐานสอง	ไล่จาก LSB · เจอ $0-1$ ให้ยืม (+2 ที่นี่ · -1 ที่หลักซ้าย)	เอาผลลัพธ์บวกตัวลบ ต้องได้ตัวตั้งเดิม
คูณฐานสอง	แถวย่อยหนึ่งแถวต่อหนึ่งบิตตัวคูณ · 1 ลอก 0 ใส่ศูนย์ · เลื่อนซ้ายทีละหลัก · บวกรวม	แปลงฐานสิบทั้งสองฝั่ง · เช็คบิตท้าย · เช็คความยาว $\leq m+n$

เจอโจทย์แบบนี้	ทำแบบนี้	ตรวจแบบนี้
หารยาว	ไล่บิตจากซ้าย · ทุกรอบถาม “ลบตัวหารได้ไหม” · ได้ใส่ 1 ไม่ได้ใส่ 0	ผลหาร × ตัวหาร + เศษ = ตัวตั้ง
1's complement	พลิกทุกบิต	$N + \overline{N} = 1$ ทุกหลัก
2's complement	พลิกทุกบิตแล้วบวก 1 · หรือทางลัด ไล่จากขวา เจอ 1 ตัวแรกแล้วพลิกที่เหลือ	$N + 2\text{'s comp}(N)$ ต้องล้นออกนอกกรอบ เหลือศูนย์หมด
เขียนเลขลบ n บิต	เขียนค่าบวกให้ครบ n บิตก่อน → sm เปลี่ยน MSB · 1c พลิก · 2c พลิกแล้ว +1	อ่านกลับเป็นฐานสิบต้องได้ค่าเดิม
อ่านค่ากลับ	ดู MSB ก่อน · เป็น 0 อ่านตรง ๆ · เป็น 1 แปลงกลับตามรูปแบบแล้วใส่ลบ	ค่าของ 1c กับ 2c ต้องห่างกัน 1 พอดี
พิสัย n บิต	$-(2^{n-1})$ ถึง $+(2^{n-1}-1)$ · มีทั้งหมด 2^n ค่า	นับสองฝั่งรวมกันต้องได้ 2^n

สรุปหน้าเดียว — EP02 เลขคณิตฐานสอง + เลขมีเครื่องหมาย

กฎบวก 4 บรรทัด

0+0=0 · **0+1=1** · **1+0=1** · **1+1=10** (เขียน 0 ทด 1) และกรณีที่ห้า **1+1+1=11** (เขียน 1 ทด 1)

กฎลบ 4 บรรทัด

0−0=0 · **1−1=0** · **1−0=1** · **10−1=1** คือ $0-1$ ต้อง**ยืม** — ยืมแล้วหลักนี้ได้ +2 และหลักซ้ายโดนหัก −1

คูณ

แถวย่อย 1 แถวต่อบิตตัวคูณ 1 บิต · บิต 1 **ลอกตัวตั้ง** บิต 0 **ใส่ศูนย์** · แถวที่ k เลื่อนซ้าย k หลัก · แล้วบวกรวม · คูณด้วย 2^k = เติมศูนย์ท้าย k ตัว

หารยาว

ไล่บิตจากซ้าย · ทุกรอบถาม **“ลบตัวหารออกได้ไหม”** · ได้ใส่ **1** แล้วลบจริง · ไม่ได้ใส่ **0** · เท่ากันพอดีก็นับว่าได้ · สิ่งที่เหลือตอนบิตหมดคือเศษ

คอมพลีเมนต์

1's complement = พลิกทุกบิต · **2's complement** = พลิกทุกบิตแล้ว +1 หรือทางลัด **ไล่จากขวา ลอกจนถึง 1 ตัวแรก แล้วพลิกที่เหลือ** · ทำ 2's complement สองครั้งได้เลขเดิม

สามรูปแบบของเลขมีเครื่องหมาย (-45 แบบ 8 บิต)

sign-magnitude 10101101 — เปลี่ยนแค่ MSB

1's complement 11010010 — พลิกทุกบิตของเลขบวก

2's complement 11010011 — พลิกแล้ว +1

เลขบวกเขียนเหมือนกันหมดทั้งสามแบบ

อ่านค่ากลับ (MSB คือบิตเครื่องหมาย)

MSB $= 0$ → บวก อ่านตรง ๆ ทั้งสามแบบให้ค่าเดียวกัน

MSB $= 1$ → ลบ · **sm** ตัด MSB อ่าน 7 บิตที่เหลือ · **1c** พลิกกลับ · **2c** ทำ 2's complement กลับ · แล้วใส่ลบ

วิธีสไลด์: ให้ MSB น้ำหนัก -2^{n-1} แล้วบวกทุกหลัก (1's ต้อง **+1** ท้ายสุด)

พิสัยและปัญหา -0

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

8 บิต: **−128 ถึง +127** · 16 บิต: **−32768 ถึง +32767**

ฝั่งลบมากกว่าฝั่งบวก 1 ตัวเพราะ**ศูนย์ไปนับอยู่ฝั่งบวก** · **sign-magnitude และ 1's complement มีศูนย์สองตัว** จึงแพ้ 2's complement

เช็คลิสต์ก่อนเข้าห้องสอบ

ทำได้โดยไม่ต้องเปิดชีทไหม

- ☐ บวกฐานสอง 4 บิตพร้อมเขียนตัวทดทุกหลัก
- ☐ ลบฐานสองที่มีการยืมติดกัน 3 หลัก
- ☐ คูณฐานสอง 4 บิต × 4 บิต โดยเขียนแถวย่อยครบ
- ☐ หารยาวพร้อมบอกผลหารและเศษ
- ☐ หา 1's complement ของเลข 8 บิต

ทำได้โดยไม่ต้องเปิดชีทไหม (ต่อ)

- ☐ หา 2's complement ด้วย**ทางลัด** ภายใน 5 วินาที
- ☐ เขียนเลขลบ 8 บิตได้ทั้งสามรูปแบบ
- ☐ อ่านค่าฐานสิบกลับได้ทั้งสามรูปแบบ
- ☐ บอกพิสัยของเลข n บิตได้จากสูตร
- ☐ อธิบายได้ว่าทำไม 2's complement ชนะ

นิสัยที่ทำให้ได้คะแนนเต็ม

ตรวจคำตอบทุกข้อด้วยการแปลงเป็นฐานสิบทั้งสองฝั่งครับ — ใช้เวลาไม่ถึง 20 วินาทีต่อข้อ แต่จับผิดได้เกือบทุกความผิดพลาดใน EP นี้ (ตัวทดหาย ตัวยืมลืมหัก แถวเลื่อนผิด ลืมใส่ 0)

และเวลาเขียนคำตอบลงกระดาษ ให้**เขียนตัวทด/ตัวยืมไว้บนหัวด้วยเสมอ** เพราะข้อสอบวิชานี้ให้คะแนนขั้นตอน ไม่ใช่แค่คำตอบสุดท้ายครับ

EP ต่อไป · EP03

ตอนนี้น้องบวกลบเลขจำนวนเต็มได้แล้วครับ และเก็บเลขติดลบลงบิตล้วน ๆ ได้ทั้งสามรูปแบบ

แต่เลขทศนิยมอย่าง -99.03125 คอมพิวเตอร์เก็บยังไง ในเมื่อจุดทศนิยมมันเลื่อนไปมาได้ครับ

ลองคิดตามนะครับ — ถ้าเราจองบิตไว้ตายตัวว่า “ซ้ายจุด 4 บิต ขวาจุด 4 บิต” เลขใหญ่ ๆ ก็เก็บไม่ได้ เลขละเอียด ๆ ก็เก็บไม่ได้ วิธีแก้คือให้**จุดทศนิยมลอยได้** ซึ่งเป็นที่มาของคำว่า `floating point` ครับ

EP03 จะพาไปเจอ: มาตรฐาน IEEE single precision 32 บิต (บิตเครื่องหมาย 1 + เอกซ์โพเนนต์ 8 + แมนทิสซา 23) · เลข bias 127 · การบวกลบคูณหารในรูป 2's complement เต็มรูปแบบ · และ **overflow condition** ว่าเมื่อไหร่ผลลัพธ์จะล้นกรอบจนค่าผิด ซึ่งต่อยอดจากเรื่องพิสัยที่เราเพิ่งเรียนไปครับ

แหล่งอ้างอิงของ EP นี้

แหล่ง	ส่วนที่ใช้	เอามาทำอะไรในชีท
EES 271-01 Introduction-2026 (revised).pdf	หน้า 18–25	Binary Arithmetic — กฎบวก ลบ คูณ หาร พร้อมตัวอย่างและโจทย์ที่สไลด์ทิ้งไว้ให้ลอง
EES 271-01 Introduction-2026 (revised).pdf	หน้า 35–46	Complements of Binary Numbers · Signed Numbers · The Decimal Value of Signed Numbers · Note for 1's complement · Range of signed numbers · Conversion Table
EES271 Assignment #1 (2026)	ข้อ 8–15	เฉลยละเอียดครบ 19 ข้อย่อยในตอนที่ 9–11
Floyd, *Digital Fundamentals*, 10th ed.	บทที่ 2	ตำราหลักของวิชา — สัญกรณ์และคำศัพท์ทั้งหมดในชีทนี้อิงตามเล่มนี้

หมายเหตุการตรวจสอบ

ตัวเลขทุกตัวในชีทนี้ — ผลบวก ผลลบ ตัวทด ตัวยืม แถวย่อยของการคูณ ทุกรอบของการหาร คอมพลีเมนต์ และค่าฐานสิบของเลขมีเครื่องหมาย — **ถูกคำนวณและตรวจซ้ำด้วยเครื่องมือของวิชานี้ (`tools/dig.py`) ก่อนพิมพ์ลงหน้ากระดาษทุกตัวครับ** ถ้าน้องเจอจุดที่ไม่ตรงกับที่อาจารย์สอน ให้ยึดสไลด์เป็นหลักและบอกพี่ได้เลยครับ