# Data storage I

Introduction to Computer Science

(88612159)

# จุดประสงค์การเรียนรู้

- ระบบเลขฐานสอง (Binary System)
- การแปลงจำนวนเต็มเป็นรูปแบบเลขฐานสอง
- การแปลงจำนวนจริงเป็นรูปแบบเลขฐานสอง
- การจัดเก็บจำนวนเต็ม
  - แบบมีเครื่องหมาย
  - แบบไม่มีเครื่องหมาย
  - ปัญหาการเกิด Overflow

# ระบบเลขฐานสอง
# (Binary System)

# ระบบเลขฐานสอง (Binary System)

- การจัดเก็บข้อมูลใน ระบบคอมพิวเตอร์นั้นจะแทนข้อมูลด้วยสภาวะการทำงานของอุปกรณ์ที่มี 2 สภาวะ
- เมื่อนำบิตหลายบิตมาเรียงกันจะได้สายของบิต (bit of string) ที่มีรูปแบบต่างกัน
- หากนำบิตสองบิตมาเรียงกัน โดยแต่ละบิตมีค่าที่เป็นไปได้สองค่า จะสามารถจัดสายของบิตได้ 22 หรือ 4 รูปแบบที่ไม่ซ้ำกัน
- นำรูปแบบของบิตมาใช้แทนตัวอักขระ จำนวน ภาพ และเสียง

# การแทนข้อมูลด้วยเลขฐานสองในรูปแบบต่างๆ

1. จำนวน
    1.1 จำนวนเต็ม
        จำนวนธรรมชาติ N (unsigned = ไม่มีเครื่องหมาย)
        จำนวนเต็ม Z (signed = มีเครื่องหมาย)
    1.2 จำนวนจริง R
2. ตัวอักขระ (character)
3. ภาพ
4. เสียง

# การบวกเลขฐานสอง (Binary addition)

- เริ่มบวกจากหลักที่มีนัยสำคัญต่ำสุด (ขวามือสุด) ไปหาหลักที่มีนัยสำคัญสูงสุด (ซ้ายมือสุด)
- เมื่อมีการบวกในหลักใดมีค่าเท่ากับฐาน กำหนดให้ผลบวกหลักนั้นมีค่าเป็นศูนย์และทด 1 ไปยังหลักถัดไปท้างซ้ายมือ

```
  00111010
+
  00011011
----------
==========
```

# การแทนข้อมูลจำนวนเต็มบวก (Unsigned Integer)

# การแปลงเลขจำนวนเต็ม

ฐาน 2 ↔ ฐาน 10 ↔ ฐาน 16

เลขฐานสอง (binary) แทนด้วยเลข 0 กับ 1

เลขฐานสิบ (decimal) แทนด้วยเลข 0 ถึง 9

เลขฐานสิบหก (hexadecimal) แทนด้วยเลข 0 ถึง 9 และ A ถึง F

# การหาค่าของเลขฐานสิบ

- หาค่าของเลขฐานสิบได้จากสูตร

$$\sum_{i=0}^{n} d_i \times 10^i$$

$i$ คือ เลขชี้กำลัง

$d_i$ คือตัวเลขหลักที่ $i$

<u>ตัวอย่าง</u> $193_{10}$

# การแปลงเลขฐานสองเป็นเลขฐานสิบ

- หาได้จากสูตร

$$\sum_{i=0}^{n} d_i \times 2^i$$

$i$ คือ เลขชี้กำลัง

$d_i$ คือตัวเลขหลักที่ $i$

<u>ตัวอย่าง</u> $1011_2$

ดังนั้น $1011_2$ =

# การแปลงเลขฐานสิบหกเป็นเลขฐานสิบ

- หาได้จากสูตร

$$\sum_{i=0}^{n} d_i \times 16^i$$

$i$ คือ เลขชี้กำลัง

$d_i$ คือตัวเลขหลักที่ $i$

<u>ตัวอย่าง</u> $54C_{16}$

ดังนั้น $54C_{16}$ =

# การแปลงเลขฐานใดๆ เป็นเลขฐานสิบ

- <u>สรุป</u> การแปลงเลขฐานใดๆ เป็นเลขฐานสิบ

  หาได้จากสูตร

$$\sum_{i=0}^{n} d_i \times b^i$$

$i$ คือ เลขชี้กำลัง

$d_i$ คือตัวเลขหลักที่ $i$

$b$ คือฐานของตัวเลข

# แบบฝึกหัด 1

- จงแปลงเลข $100101_2$ เป็นฐานสิบ
- จงแปลงเลข$1234_8$ เป็นฐานสิบ
- จงแปลงเลข $A9B_{16}$ เลขฐานสิบ

# การแปลงเลขฐานสิบเป็นเลขฐานสอง

<u>ตัวอย่าง</u> $117_{10}$

หารด้วย 2 ซ้ำ จนกว่าผลหารจะเป็น 0

เรียงเศษจากล่างขึ้นบน

ดังนั้น $117_{10} = 1110101_2$

# การแปลงเลขฐานสิบเป็นเลขฐานสิบหก

<u>ตัวอย่าง</u> $603_{10}$

หารด้วย 16 ซ้ำ จนกว่าผลหารจะเป็น 0

เรียงเศษจากล่างขึ้นบน

ดังนั้น $603_{10} = 25B_{16}$

# การแปลงเลขฐานสิบ เป็นเลขฐานใดๆ

- <u>สรุป</u> การแปลงเลขฐานสิบ เป็นเลขฐานใด

  วิธีทำ หารด้วยเลขฐานซ้ำจนกว่าผลหารจะเป็นศูนย์

  จากนั้นเรียงเศษที่ได้จากการหารจากล่างขึ้นบน

# แบบฝึกหัด 2

- จงแปลงเลข $165_{10}$ เป็นฐานสอง
- จงแปลงเลข $150_{10}$ เป็นฐานแปด
- จงแปลงเลข $125_{10}$ เลขฐานสิบหก

# รูปแบบบิตและ เลขฐานสิบหก

| Bit pattern | Hexadecimal representation |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

# การแทนข้อมูลจำนวนจริง (Real Number)

เช่น 3.14159

Integral part (ภาคจำนวนเต็ม)

Fractional part (ภาคเศษส่วน)

# การแปลงเลขจำนวนจริงฐานสอง เป็น เลขจำนวนจริงฐานสิบ

การแปลงจำนวนจริงในระบบทวินิยมเป็นจำนวนจริงในระบบทศนิยม

ตัวอย่าง จงแปลง $1101.1011_2$ เป็นเลขจำนวนจริงฐานสิบ

- 1101 แปลงเป็นเลขฐานสิบได้
- ตัวเลขหลังจุดทวินิยม คือ 1011 แปลงเป็นเลขฐานสิบได้โดย

$2^{-1} = 0.5$

$2^{-2} = 0.25$

$2^{-3} = 0.125$

$2^{-4} = 0.0625$

ดังนั้น $1101.1011_2$ =

# การแปลงเลขจำนวนจริงฐานสิบ เป็น เลขจำนวนจริงฐานสอง

การแปลงจำนวนจริงในระบบทศนิยมเป็นเป็นจำนวนจริงในระบบทวินิยม

ตัวอย่าง การแปลง $10.25_{10}$ เป็นจำนวนจริงฐานสอง

วิธีการแปลงเป็นเลขฐานสอง

- 10 จะได้จากการหารเอาเศษ (ตัวหาร = 2) จะได้
- 25 ได้จากการคูณเอาผลคูณ (ตัวคูณ = 2) เรียงผลคูณหน้าจุดจากบนลงล่าง จะได้

- ดังนั้น $10.25_{10}$ =

# การแปลงเลขจำนวนจริงฐานสิบ เป็น เลขจำนวนจริงฐานสอง

<u>ตัวอย่าง</u> การแปลง $25.59_{10}$ เป็นจำนวนจริงฐานสอง

- 25 จะได้จากการหารเอาเศษ (ตัวหาร = 2) จะได้
- 59 ได้จากคูณเอาผลคูณ (ตัวคูณ = 2) เรียงผลคูณหน้าจุดจากบนลงล่าง จะได้
- ต้องกำหนดจำนวนตำแหน่งทวินิยม
- หากกำหนดให้เป็น 3
- ดังนั้น $25.59_{10}$ =

# การแปลงเลขจำนวนจริงฐานสิบ เป็น เลขจำนวนจริงฐานสอง

สรุป

- คูณด้วย 2 เก็บผลคูณหน้าจุดไว้ จนกว่าผลคูณจะเป็น 0.00 หรือ ได้จำนวนทวินิยมตามที่ต้องการ
- จากนั้นเรียงผลคูณหน้าจุดจากบนลงล่าง

# แบบฝึกหัด 3

- จงแปลงจำนวนจริงฐานสิบ 9.89 เป็นจำนวนจริงฐานสอง (ต้องการเลขหลังจุดทวินิยม 4 ตำแหน่ง)

- จงแปลงจำนวนจริงฐานสอง 110.0101 เป็นจำนวนจริงฐานสิบ

# การจัดเก็บจำนวนเต็ม

# ขนาดของข้อมูล

ข้อมูลแต่ละชนิดมีขนาดจำกัด เช่น ใน ภาษา C++ และ JAVA มีขนาดตัวอักขระ และ จำนวนเต็ม ดังนี้

| ชนิด | ขนาด |
|---|---|
| char | 8 bits (1 byte ) หรือ 16 bits (2 byte ) |
| short | 16 bits (2 byte ) |
| int | 32 bits (4 byte) |
| long | 64 bits (8 byte) |

# การแทนข้อมูลชนิดอักขระ

```
char a;

a = 'Z';
```

- สิ่งที่เก็บในตัวแปร a คือ รหัสแทนอักขระ
- รหัส ASCII มีขนาด 8 บิต มี $2^8$ = 256 ตัวอักขระ แบ่งเป็น 2 ส่วน

  1) ส่วนของภาษาอังกฤษ 0 ถึง 127

  2) ส่วนของภาษาถิ่นของแต่ละประเทศ 128 ถึง 255
- รหัส Unicode มีขนาด 16 บิต มี $2^{16}$ = 65536 ตัวอักขระ

# การแทนข้อมูลชนิดจำนวนเต็ม

จำนวนเต็มแบ่งออกเป็น 2 ชนิด

- จำนวนเต็มชนิดไม่มีเครื่องหมาย (unsigned) → {0,1,2,...}
- จำนวนเต็มชนิดมีเครื่องหมาย (signed) → {...,-2,-1,0,1,2,...}

| มีเครื่องหมาย | ไม่มีเครื่องหมาย | ขนาด (bits) |
|---|---|---|
| [signed] char | unsigned char | 8, 16 |
| [signed] short | unsigned short | 16 |
| [signed] int | unsigned int | 32 |
| [signed] long | unsigned long | 64 |

# Value range (พิสัย)

- พิสัยจำนวนเต็มชนิด**ไม่มีเครื่องหมาย (unsigned)**

  คำนวณได้จาก **0 ถึง $2^b$ -1**

- พิสัยจำนวนเต็มชนิด**มีเครื่องหมาย (signed)**

  คำนวณได้จาก **$-2^{b-1}$ ถึง $2^{b-1}$ -1**

โดยที่ b คือ จำนวนบิต

# Value range (พิสัย)

| Type | Storage size (bits) | Value range |
|---|---|---|
| [signed] char | 8, 16 | -128 to 127 , -32,768 to 32,767 |
| unsigned char | 8, 16 | 0 to 255 , 0 to 65,535 |
| [signed] short | 16 | -32,768 to 32,767 |
| unsigned short | 16 | 0 to 65,535 |
| [signed] int | 32 | -2,147,483,648 to 2,147,483,647 |
| unsigned int | 32 | 0 to 4,294,967,295 |
| [signed] long | 32 or 64 | $2^{31}$ to $2^{31}$ -1 or $2^{63}$ to $2^{63}$ -1 |
| unsigned long | 32 or 64 | 0 to $2^{31}$ -1 or 0 to $2^{63}$ -1 |

# การจัดเก็บจำนวนเต็มชนิดไม่มีเครื่องหมาย

- จำนวนเต็มชนิดไม่มีเครื่องหมาย ประกอบด้วย จำนวนบวก
- แทนด้วยเลขฐานสอง ตามจำนวนบิต **“ทุกบิตแทนค่า”**
- ดังนั้น จำนวนเต็มชนิดมีเครื่องหมายขนาด n บิต แทนค่าด้วยเลขฐานสอง n บิต
- จัดเก็บในระบบเลขฐานสองให้ครบตามจำนวนบิต

<u>ตัวอย่าง</u>

```
unsigned int a; //ตัวแปร a ชนิด integer แบบไม่มีเครื่องหมาย ขนาด 32 บิต
a = 10;
```

แปลงเลขฐานสิบเป็นฐานสอง จัดเก็บให้ครบขนาด 32 บิต

จะได้ 00000000 00000000 00000000 00001010

# จำนวนเต็มชนิดมีเครื่องหมาย

- ใช้ระบบส่วนเติมเต็มสอง (2's complement)

ระบบส่วนเติมเต็มสอง

| ขนาดสามบิต | | ขนาดสี่บิต | |
|---|---|---|---|
| 011 | 3 | 0111 | 7 |
| 010 | 2 | 0110 | 6 |
| 001 | 1 | 0101 | 5 |
| 000 | 0 | 0100 | 4 |
| 111 | -1 | 0011 | 3 |
| 110 | -2 | 0010 | 2 |
| 101 | -3 | 0001 | 1 |
| 100 | -4 | 0000 | 0 |
| | | 1111 | -1 |
| | | 1110 | -2 |
| | | 1101 | -3 |
| | | 1100 | -4 |
| | | 1011 | -5 |
| | | 1010 | -6 |
| | | 1001 | -7 |
| | | 1000 | -8 |

# การจัดเก็บจำนวนเต็มชนิดมีเครื่องหมาย

- จำนวนเต็มชนิดมีเครื่องหมาย ประกอบด้วย
  - จำนวนบวก {0,1,2,...} ใช้วิธีการจัดเก็บ โดยแปลงเลขฐานสองให้ครบบิต
  - จำนวนลบ {...,-2,-1} } ใช้วิธี 2's complement (ส่วนเติมเต็มสอง) เป็นวิธีมาตราฐานในการเก็บจำนวนลบ
- ให้บิตที่มีนัยสำคัญสูงสุด (msb) แทนบิตเครื่องหมาย

  0 เป็นจำนวนบวก

  1 เป็นจำนวนลบ
- จำนวนเต็มชนิดมีเครื่องหมายขนาด n บิต แทนด้วยเลขฐานสอง

  แทนเครื่องหมาย 1 บิต

  แทนค่า n-1 บิต

# การจัดเก็บจำนวนเต็มชนิดมีเครื่องหมาย (จำนวนเต็มบวก)

แปลงเป็นเลขฐานสองให้ครบบิต

<u>ตัวอย่าง</u> จงแปลงเลขฐานสิบ $26_{10}$ เป็นเลขฐานสองขนาด 8 บิต

จะได้ $1\ 1010_2$

เติมให้ครบ 8 บิต $0001\ 1010_2$

บิตเครื่องหมาย เท่ากับ 0 เป็นจำนวนบวก

ดังนั้น $26_{10} = 0001\ 1010_2$

# ระบบ 2's complement (ส่วนเติมเต็มสอง)

## จำนวนเต็มลบใช้วิธี 2's complement

ขั้นตอนวิธี 2's complement

1. แปลง absolute ของเลขฐานสิบเป็นเลขฐานสอง
2. ทำ 1's complement โดย กลับบิตตัวเลขฐานสอง จาก 0 เป็น 1 และจาก 1 เป็น 0
3. บวกด้วย 1

# การจัดเก็บจำนวนเต็มชนิดมีเครื่องหมาย (จำนวนลบ)

จำนวนเต็มลบใช้วิธี 2's complement

<u>ตัวอย่าง</u> จงแปลงเลขฐานสิบ $-26_{10}$ เป็นเลขฐานสองขนาด 8 บิต

1. แปลง $|-26_{10}|$ เป็นเลขฐานสองให้ครบจำนวนบิต จะได้
2. ทำ 1's complement โดยการกลับบิต จะได้
3. บวกด้วย 1

# แบบฝึกหัด 4

- จงแปลงเลขฐานสิบ 25 เป็นเลขฐานสองขนาด 8 บิตในระบบส่วนเติมเต็มสอง ( 2’s complement )

- จงแปลงเลขฐานสิบ -25 เป็นเลขฐานสองขนาด 8 บิตในระบบส่วนเติมเต็มสอง ( 2’s complement )

# การแปลงเลขฐานสองในระบบ 2's complement เป็นเลขฐานสิบ

<u>ตัวอย่าง</u> จงแปลงเลขฐานสอง $1110\ 0110_2$ ในระบบ 2's complement เป็นเลขฐานสิบ

1. ทำ 1's complement โดยการกลับบิต จะได้

2. บวกด้วย 1

$$\begin{array}{r} 0001\ 1001 \\ + \\ 1 \\ \hline \end{array}$$

3. แปลงเลขฐานสองที่ได้เป็นเลขฐานสิบ และคูณด้วย -1 จะได้

$1110\ 0110_2$ (2's complement) = $-26_{10}$

# แบบฝึกหัด 5

- จงแปลงเลขฐานสอง 00011001 ในระบบส่วนเติมเต็มสอง ( 2’s complement ) เป็นเลขฐานสิบ

- จงแปลงเลขฐานสอง 11100111 ในระบบส่วนเติมเต็มสอง ( 2’s complement ) เป็นเลขฐานสิบ

# การเกิดปัญหา Overflow

## แบบชนิดไม่มีเครื่องหมาย

<u>ตัวอย่าง</u> จัดเก็บจำนวนเต็ม $256_{10}$ ขนาด 8 บิต

เมื่อแปลง 256 เป็นเลขฐานสอง จะได้ $1\ 0000\ 0000_2$

- เนื่องจากพิสัยของจำนวนเต็มขนาด 8 บิต คือ 0 ถึง 255
- ทำให้ 256 เกินพิสัย จึงเกิด Overflow ผลลัพธ์ล้นที่เก็บ (ผลลัพธ์เกินจำนวนบิต)
- ไม่ Error แต่จะตัดบิตที่เกิน (บิตที่มีนัยสำคัญสูดสุด) ทิ้ง ทำให้ผลลัพธ์ที่ได้ผิด นั่นคือ $0000\ 0000_2$

# การเกิดปัญหา Overflow

## แบบชนิดมีเครื่องหมาย

<u>ตัวอย่าง</u> จัดเก็บจำนวนเต็ม $128_{10}$ ขนาด 8 บิต

เมื่อแปลง 128 เป็นเลขฐานสอง จะได้ $1000\ 0000_2$

- แต่ $1000\ 0000_2$ (2's complement) เท่ากับ $-128_{10}$
- เนื่องจากพิสัยจำนวนเต็มขนาด 8 คือ -128 ถึง 127
- ทำให้ 128 เกินพิสัย ไม่เกิด Overflow แต่ผลลัพธ์ล้นเข้าไปในบิตเครื่องหมาย ผลลัพธ์ที่ได้จึงผิด

# การเกิดปัญหา Overflow

## แบบชนิดมีเครื่องหมาย

เมื่อมีเครื่องหมายเดียวกันนำมาบวกกัน

ตัวอย่าง จัดเก็บจำนวนเต็ม 127 + 1 ขนาด 8 บิต

127 = 0111 1111

1 = 0000 0001

0111 1111 + 0000 0001 = 1000 0000

แต่ 1000 0000 (2's complement) = -128

เนื่องจากผลลัพธ์ล้นเข้าไปบิตเครื่องหมาย ผลลัพธ์ที่ได้ผิด

# การเกิดปัญหา Overflow

## แบบชนิดมีเครื่องหมาย

เมื่อมีเครื่องหมายเดียวกันนำมาบวกกัน

ตัวอย่าง จัดเก็บจำนวนเต็ม -128 + (-1) ขนาด 8 บิต

-128 = 1000 0000 (2's complement)

-1 = 1111 1111 (2's complement)

1000 0000 + 1111 1111 = 1 0111 1111

แต่ 0111 1111 = 127

เนื่องจากผลลัพธ์ล้นเข้าไปบิตเครื่องหมาย และล้นที่เก็บ ทำให้ผลลัพธ์ที่ได้ผิด

# การเกิดปัญหา Overflow

## แบบชนิดมีเครื่องหมาย

เมื่อมีเครื่องหมายต่างกันนำมาบวกกัน

-127+127 = 0

-127 = 1000 0001 (2’s complement)

127 = 0111 1111

1000 0001 + 0111 1111 = 1 0000 0000

แต่ 0000 0000 = 0

จะเห็นว่าผลลัพธ์ล้นเข้าไปบิตเครื่องหมาย และล้นที่เก็บ แต่ผลลัพธ์ถูก

ดั้งนั้น หากเครื่องหมายต่างกัน บวกกัน ผลลัพธ์ผิดจะไม่มี

# แบบฝึกหัด 6

- จงดำเนินการบวกเลขฐานสองในระบบส่วนเติมเต็มสองขนาด 4 บิตต่อไปนี้ เกิด Overflow หรือไม่ และตรวจสอบความถูกต้องของคำตอบโดยแปลงผลลัพธ์ที่ได้เป็นเลขฐานสิบ

  1101 + → -3

  1000 → -8

# แบบฝึกหัด 7

- จงดำเนินการบวกเลขฐานสองในระบบส่วนเติมเต็มสองขนาด 4 บิตต่อไปนี้ เกิด Overflow หรือไม่ และตรวจสอบความถูกต้องของคำตอบโดยแปลงผลลัพธ์ที่ได้เป็นเลขฐานสิบ

  0101 + → 5

  0101 → 5

# แบบฝึกหัด 8

- จงดำเนินการบวกเลขฐานสองในระบบส่วนเติมเต็มสองขนาด 4 บิตต่อไปนี้ เกิด Overflow หรือไม่ และตรวจสอบความถูกต้องของคำตอบโดยแปลงผลลัพธ์ที่ได้เป็นเลขฐานสิบ

  1110 + → -2

  0011 → 3