บทที่2
การแทนข้อมูลในระบบคอมพิวเตอร์

(Data Representation in Computer Systems)

หน่วยของข้อมูลในคอมพิวเตอร์

บิต (Bit)

- เก็บสถานะของการเปิดหรือปิด (“on” or “off”) ในวงจรดิจิทัล
- บางครั้งก็เก็บสถานะแรงดันไฟฟ้าสูงหรือต่ำ (“high” or “low”) แทนสถานะเปิดหรือปิด

ไบต์ (Byte)

- กลุ่มของบิตที่เรียงต่อกัน 8 บิต
- เป็นหน่วยข้อมูลที่เล็กที่สุดของคอมพิวเตอร์ที่สามารถเก็บได้ในหน่วยความจำ

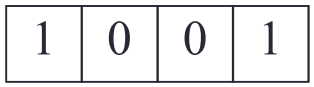
หน่วยของข้อมูลในคอมพิวเตอร์

เวิร์ด (Word)

- กลุ่มของไบต์ที่เรียงต่อกัน
- อาจแสดงเป็นเลขจำนวนของบิตหรือไบต์
- ขนาดของเวิร์ดพื้นฐานคือ 16 32 และ 64 บิต
- ในระบบที่ใช้ระบบแอดเดรสแบบเวิร์ด จะเป็นข้อมูลที่เล็กที่สุดของคอมพิวเตอร์ที่สามารถเก็บได้ในหน่วยความจำ

นิบเบิล (Nibble)

- กลุ่มของบิตที่เรียงต่อกัน 4 บิต



- 1 ไบต์ ประกอบด้วย 2 nibble คือ “high-order nibble,” และ “low-order nibble”

ระบบจำนวน (Number System)

- ข้อมูล 1 ไบต์ จะเก็บค่าข้อมูลในแต่ละบิต แต่ละบิตจะแทนด้วยค่ายกกำลังของ 2 เรียกว่า ระบบไบนารี่ (Binary System) หรือระบบเลขฐานสอง
- ระบบตัวเลขของมนุษย์คือระบบเลขฐานสิบ (Decimal System)
- เลขจำนวนเต็มสามารถแทนด้วยเลขฐานได้ เช่น
- เลขฐานสิบ 947 จะมีค่าในแต่ละตำแหน่งเท่ากับค่ายกกำลังของ 10 คือ

  $9x10^2 + 4x10^1 + 7x10^0$

- เลขฐานสิบ 5836.47 จะมีค่าในแต่ละตำแหน่งเท่ากับค่ายกกำลังของ 10 คือ

  $5x10^3 + 8x10^2 + 3x10^1 + 6x10^0 + 4x10^{-1} + 7x10^{-2}$

- เลขฐานสอง 11001 จะมีค่าในแต่ละตำแหน่งเท่ากับค่ายกกำลังของ 2 คือ

  $1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$

ระบบจำนวน (Number System)

- เลขฐานจะมีค่าในแต่ละตำแหน่งเท่ากับค่ายกกำลังของฐานนั้น และมีสัญลักษณ์ของตัวเลขตามจำนวนของฐาน โดยเริ่มจาก 0 เช่น
- เลขฐาน 2 มีสัญลักษณ์ 2 ตัว คือ 0 และ 1
- เลขฐาน 3 มีสัญลักษณ์ 3 ตัว คือ 0 1 และ 2
- เลขฐาน 16 มีสัญลักษณ์ 16 ตัว คือ 0 – 9 และ A (10) – F (15) เป็นต้น
- การเขียนตัวเลขฐานใด ๆ ให้เขียนฐานของตัวเลขนั้นเป็นตัวห้อยต่อท้าย เช่น
- $11001_2 = 25_{10}$

ฝึกหาค่าเลขฐานต่าง ๆ

$111_2 = ?_{10}$

$101_2 = ?_{10}$

$10000001_2 = ?_{10}$

$121_3 = ?_{10}$

$202_3 = ?_{10}$

$10_8 = ?_{10}$

$12_8 = ?_{10}$

$41_{16} = ?_{10}$

$10_{16} = ?_{10}$

$28_{16} = ?_{10}$

ฝึกหาค่าเลขฐานต่าง ๆ

$111_2 = 1x2^2 + 1x2^1 + 1x2^0 = 7_{10}$

$101_2 = 1x2^2 + 0x2^1 + 1x2^0 = 5_{10}$

$10000001_2 = 1x2^7 + 0x2^6 + 0x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0 = 129_{10}$

$121_3 = 1x3^2 + 2x3^1 + 1x3^0 = 16_{10}$

$202_3 = 2x3^2 + 0x3^1 + 2x3^0 = 20_{10}$

$10_8 = 1x8^1 + 0x8^0 = 8_{10}$

$12_8 = 1x8^1 + 2x8^0 = 10_{10}$

$41_{16} = 4x16^1 + 1x16^0 = 65_{10}$

$10_{16} = 1x16^1 + 0x16^0 = 16_{10}$

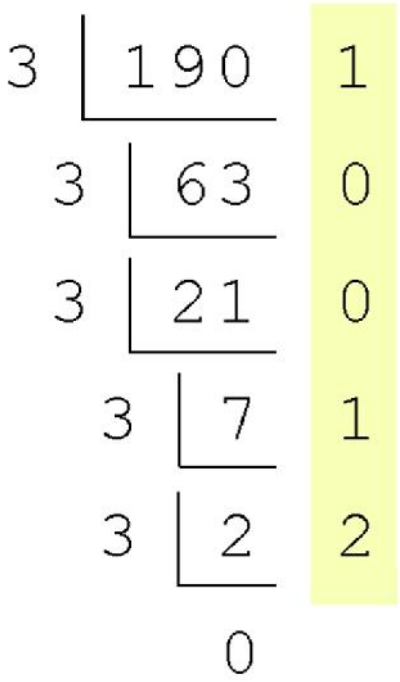
$28_{16} = 2x16^1 + 8x16^0 = 40_{10}$

การแปลงเลขฐานสิบเป็นเลขฐานสอง

- เลขฐานสองเป็นระบบตัวเลขที่ใช้แทนข้อมูลในคอมพิวเตอร์
- การรู้ระบบเลขฐานสองจะทำให้เข้าใจการทำงานของคอมพิวเตอร์ และสามารถออกแบบสถาปัตยกรรมชุดคำสั่งของคอมพิวเตอร์ได้
- การแปลงค่าตัวเลขจากฐานสิบเป็นฐานใด ๆ ทำได้โดยการนำตัวเลขนั้นมาหารด้วยค่าฐานที่ต้องการแปลงไปเรื่อย ๆ จนกว่าผลลัพธ์จากการหารจะมีค่าน้อยกว่าค่าฐาน โดยแต่ละครั้งที่หารให้เขียนค่าเศษที่เหลือจากการหารไว้ ค่าผลลัพธ์ที่แปลงได้จะเป็นการนำค่าเศษที่เขียนไว้มาเขียนเรียงตามลำดับจากค่าสุดท้ายไปยังค่าแรก

ตัวอย่างการแปลงค่า 190_{10} เป็นเลขฐาน 3

- $190_{10} = 21001_3$



แปลงค่าเลขฐาน 10 เป็นฐานอื่น

$100 = ?_{2}$

$63 = ?_{2}$

$255 = ?_{2}$

$100 = ?_{8}$

$63 = ?_{8}$

$255 = ?_{8}$

$100 = ?_{16}$

$57 = ?_{16}$

$258 = ?_{16}$

แปลงค่าเลขฐาน 10 เป็นฐานอื่น

$100 = 01100100_2$

$63 = 00111111_2$

$255 = 11111111_2$

$100 = 144_8$

$63 = 77_8$

$255 = 377_8$

$100 = 64_{16}$

$57 = 39_{16}$

$258 = 102_{16}$

การแปลงเลขทศนิยมให้เป็นเลขฐานสอง

- ค่าเศษส่วนสามารถประมาณค่าได้ในทุกระบบเลขฐาน
- ค่าเศษส่วนไม่จำเป็นต้องใช้ในทุกระบบเลขฐาน
- ค่า ½ สามารถแทนค่าในระบบเลขฐานสอง และฐานสิบได้ แต่ไม่สามารถแทนค่าในระบบเลขฐานสามได้
- เลขทศนิยมในระบบจำนวนจะไม่มีเลข 0 อยู่ทางด้านขวาสุดของตัวเลข
- ค่าตัวเลขที่อยู่ด้านขวามือของจุดทศนิยมจะมีค่าเท่ากับค่าฐานยกกำลังเลขจำนวนเต็มลบโดยเริ่มจาก -1, -2, -3, … ไปเรื่อย ๆ เช่น

$$0.47_{10} = 4 \times 10^{-1} + 7 \times 10^{-2}$$

$$0.11_2 = 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= \frac{1}{2} + \frac{1}{4}$$

$$= 0.5 + 0.25 = 0.75$$

การแปลงเลขทศนิยมฐานสิบเป็นฐานสอง

- จะใช้วิธีการคูณด้วยสองเฉพาะตัวเลขที่อยู่หลังจุดทศนิยมไปเรื่อย ๆ จนกว่าผลลัพธ์ของตัวเลขหลังจุดทศนิยมจะมีค่าเป็น 0
- ผลลัพธ์จากการแปลงก็คือการเรียงตัวเลขที่อยู่หน้าจุดทศนิยมที่ได้จากการคูณในแต่ละครั้ง เริ่มจากค่าแรกไปจนถึงค่าสุดท้ายตามลำดับ เช่น

$$0.8125_{10} = 0.1101_2$$

```
  .8125
  × 2
1.6250

  .6250
  × 2
1.2500

  .2500
  × 2
0.5000

  .5000
  × 2
1.0000
```

การแปลงเลขเป็นฐานสอง

- ระบบเลขฐานสองมีความสำคัญกับคอมพิวเตอร์มากที่สุด
- ตัวเลขที่มีค่ามาก เมื่อแทนด้วยเลขฐานสองจะเป็นตัวเลขที่ยาวมากซึ่งยากต่อการอ่านและเขียน เช่น
  - $11010100011011_2 = 13595_{10}$
- เพื่อให้เขียนได้สั้นกะทัดรัด และง่ายต่อการอ่าน จึงมักจะเขียนแทนค่าโดยใช้ระบบตัวเลขฐานสิบหก (Hexadecimal)
- การแปลงค่าระหว่างเลขฐานสอง และฐานสิบหก สามารถทำได้ง่าย เนื่องจาก $16 = 2^4$ ดังนั้นจะใช้เลขฐานสอง 4 หลักต่อเลขฐานสิบหก 1 หลัก
- กลุ่มของเลขฐานสอง 4 หลัก เรียกว่า Hextet

การแปลงเลขเป็นฐานสอง

- การแปลงกลุ่ม Hextets ของตัวเลข 11010100011011_2 (= 13595_{10}) เป็นฐานสิบหก

0011	0101	0001	1011
3	5	1	B

- การแปลงระหว่างเลขฐานสองเป็นฐานแปดจะใช้กลุ่มของเลขฐานสอง 3 หลัก ($8 = 2^3$)

011	010	100	011	011
3	2	4	3	3

- การใช้เลขฐานแปดจะมีประโยชน์มากในระบบคอมพิวเตอร์ที่ใช้เวิร์ดแบบหกบิต

การแทนค่าเลขจำนวนเต็ม (Integer) ในคอมพิวเตอร์

- ระบบคอมพิวเตอร์จะใช้บิตสูงสุดเป็นบิตเครื่องหมายเพื่อบ่งบอกค่าตัวเลขว่าเป็นเลขจำนวนเต็มบวก หรือลบ
- บิตสูงสุด (high-order bit) คือ บิตที่อยู่ทางซ้ายสุดของไบต์นั้น หรือเรียกว่า บิตที่มีนัยสำคัญสูงสุด (Most Significant Bit : MSB)
- ส่วนบิตที่เหลือจะเป็นค่าของตัวเลขนั้น
- การเก็บค่าเลขจำนวนเต็มในคอมพิวเตอร์มี 3 วิธี คือ
    1. การกำหนดบิตเครื่องหมาย (Signed magnitude)
    2. 1’s complement (One’s complement)
    3. 2’s complement (Two’s complement)

การกำหนดบิตเครื่องหมาย (Signed magnitude)

- ในระบบเวิร์ดแบบ 8 บิต บิตเครื่องหมายจะบ่งบอกว่าเป็นเลขจำนวนเต็มบวก หรือ ลบ ของค่าตัวเลขอีก 7 บิตที่เหลือ เช่น

  3 จะเก็บเป็น 00000011

  -3 จะเก็บเป็น 10000011

- เวลาคอมพิวเตอร์คำนวณค่าตัวเลขจะไม่สนใจเครื่องหมายคล้ายกับการคำนวณเลขของคน จะนำเครื่องหมายมาใส่หลังจากคำนวณเสร็จเรียบร้อยแล้ว

การบวกเลขฐานสอง

ของการแทนค่าเลขจำนวนเต็มโดยการกำหนดบิตเครื่องหมาย

- การบวกเลขฐานสองมี 4 ลักษณะ คือ
- $0 + 0 = 0$ $0 + 1 = 1$
- $1 + 0 = 1$ $1 + 1 = 10$
- ตัวอย่างการบวกเลข 75 และ 46
- แปลงเลข 75 และ 46 เป็นเลขฐานสอง โดยแยกบิตเครื่องหมายออก
- บวกและทดค่าตัวเลขคล้ายกับบวกเลขฐานสิบ
- ถ้าผลลัพธ์จากการคำนวณมีค่าไม่เกินจำนวนบิตที่ใช้เก็บข้อมูลจะไม่เกิดปัญหาใด ๆ และจะได้ผลลัพธ์ที่ถูกต้อง

```
0   1001011
0 + 0101110
```

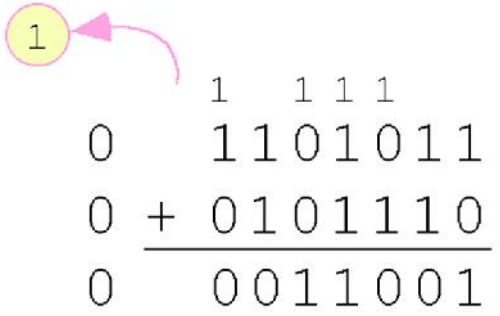
```
0   1001011
0 + 0101110
          1
```

```
      1
0   1001011
0 + 0101110
         01
```

```
    1 1 1
0   1001011
0 + 0101110
0   1111001
```

การบวกเลขฐานสอง
ของการแทนค่าเลขจำนวนเต็มโดยการกำหนดบิตเครื่องหมาย

- ตัวอย่างการบวกเลข 107 และ 46
- ถ้าผลลัพธ์จากการคำนวณมีค่าเกินจำนวนบิตที่ใช้เก็บข้อมูลจะเกิดปัญหา Overflow ซึ่งค่าในบิตที่เกินจะถูกลบออก และจะได้ผลลัพธ์ที่ไม่ถูกต้อง
- ผลลัพธ์ของ 107 + 46 = 25 (ผลลัพธ์ผิดเนื่องจากผลลัพธ์ที่ได้เป็นตัวเลขที่มีค่าเกิน 7 บิต)



การบวกเลขฐานสอง
ของการแทนค่าเลขจำนวนเต็มโดยการกำหนดบิตเครื่องหมาย

- ตัวอย่างการบวกเลข -46 และ –25
- วิธีคำนวณจะทำคล้ายกับการบวกเลขด้วยมือเช่นกัน
  คือ แยกเครื่องหมายออกแล้วนำตัวเลขมาบวกกัน
  จากนั้นจึงนำเครื่องหมายมาใส่ทีหลัง

```
      1 1
1   0101110
1 + 0011001
-----------
1   1000111
```

การลบเลขฐานสอง
ของการแทนค่าเลขจำนวนเต็มโดยการกำหนดบิตเครื่องหมาย

- การลบเลขฐานสองก็สามารถทำคล้ายกับการบวก
- มีการยืมค่าตัวเลขในหลักถัดไปเหมือนกับการลบเลขฐานสิบ
  - **ตัวอย่างการหาผลลบของ 46 และ -25**
- **บิตเครื่องหมายที่จะใส่หน้าผลลัพธ์ของการลบจะเป็นบิตเครื่องหมายของค่าตัวเลขที่มากกว่า**

```
          0 2     0 2
0    0 1 0 1 1 1 0
1  + 0 0 1 1 0 0 1
  ---------------
0    0 0 1 0 1 0 1
```

ปัญหาของการแทนค่าเลขจำนวนเต็ม โดยการกำหนดบิตเครื่องหมาย

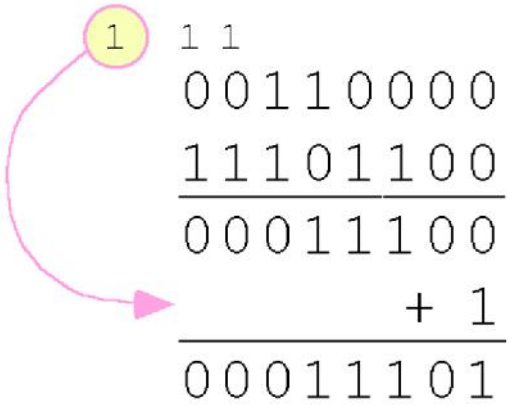
- การแทนค่าเลขจำนวนเต็มโดยการกำหนดบิตเครื่องหมายนั้นทำให้ง่ายต่อการเข้าใจ แต่ต้องใช้ฮาร์ดแวร์ที่มีความซับซ้อน
- การจัดเก็บค่าตัวเลข 0 จะเก็บได้สองค่า คือ +0 และ -0 (0 0000000 และ 1 0000000) ซึ่งในทางคอมพิวเตอร์จะต้องสร้างวงจรพิเศษในการตรวจสอบเพิ่มเติม

การแทนเลขจำนวนเต็มแบบ 1's complement (One's complement)

- ในระบบเลขฐานโดยทั่วไปเลขจำนวนเต็มลบจะเป็นเลขที่ตรงกันข้ามกับเลขจำนวนเต็มบวก เช่น +3 และ -3
- ในระบบเลขฐานสองที่แทนเลขจำนวนเต็มแบบ 1's complement จะมีการกำหนดบิตเครื่องหมายที่บิตซ้ายสุดเช่นเดียวกับการแทนโดยการกำหนดบิตเครื่องหมาย ส่วนบิตที่เหลือจะเป็นค่าตัวเลขนั้น ซึ่งเลขจำนวนเต็มลบก็จะมีค่าแต่ละบิตตรงกันข้ามกับเลขจำนวนเต็มบวกนั้น ๆ เช่น การเก็บค่าตัวเลขขนาด 8 บิต ของ 3 และ -3
- 3 จะแทนค่าเป็น 0 0000011
- -3 จะแทนค่าเป็น 1 1111100

การหาผลรวมของตัวเลขในระบบ 1's complement

- ไม่จำเป็นต้องมีวงจรสำหรับการลบเลข
- การหาผลรวมของตัวเลขจะใช้วิธีการบวก
- ในกรณีที่เกิด overflow จะนำบิตนั้นมาบวกเพิ่มกับผลลัพธ์เบื้องต้น ดังตัวอย่าง การหาผลรวมของ 48 (00110000) และ -19 (11101100)
- ค่า 19 คือ 0 0010011
- ค่า -19 คือ 1 1101100



ข้อดี-ข้อเสียของการแทนเลขจำนวนเต็มแบบ 1’s complement

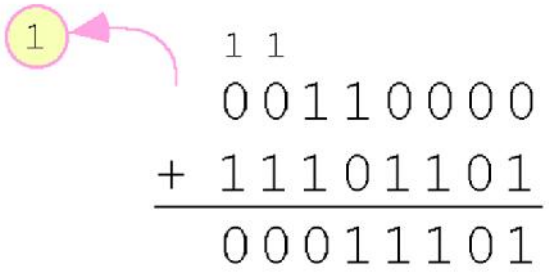
- ระบบสร้างได้ง่ายกว่าแบบกำหนดบิตเครื่องหมาย
- การแทนค่าเลข 0 ยังคงมีได้ 2 ค่า

การแทนเลขจำนวนเต็มแบบ 2's complement

- จะคล้ายกับการเก็บแบบ 1's complement แต่การแทนตัวเลขที่เป็นจำนวนเต็มลบจะมีการบวก 1 เพิ่มเข้าไป
- ตัวอย่างการเก็บค่าตัวเลขขนาด 8 บิต:
- 3 คือ 00000011
- -3 ในระบบ 1's complement คือ 11111100
- -3 ในระบบ 2's complement จะบวก 1 เพิ่มเข้าไป คือ 11111101

การหาผลรวมค่าตัวเลขในระบบ 2's complement

- การหาผลรวมจะคล้ายกับระบบ 1's complement แต่ถ้าเกิด overflow ขึ้น จะทิ้งบิตที่ overflow นั้น
- ตัวอย่างการหาผลรวมของ 48 และ -19



- 19 ในระบบ 1's complement คือ 00010011
- -19 ในระบบ 1's complement คือ 11101100
- -19 ในระบบ 2's complement คือ 11101101

ข้อดี-ข้อเสียของตัวเลขในระบบ 2's complement

- การเก็บค่าตัวเลข 0 จะมีเพียง 1 ค่าเท่านั้น
- ไม่สามารถขจัดปัญหาการเกิด overflow ได้ แต่สามารถตรวจสอบได้ โดยดูจากบิตเครื่องหมายของตัวเลขที่นำมารวมกัน ถ้าผลลัพธ์มีค่าของบิตเครื่องหมายต่างไปจากเดิม แสดงว่าเกิด overflow ขึ้น
- ตัวอย่างการหาผลรวม 107 และ 46 ได้ผลลัพธ์เป็น -103 ซึ่งพบว่าผลลัพธ์ที่ได้มีบิตเครื่องหมายต่างไปจากตัวเลขที่นำมารวมกัน

```
  1 1   1 1 1
  01101011
+ 00101110
----------
  10011001
```

การแทนค่าเลขทศนิยม (Floating-point) ในคอมพิวเตอร์

- การเขียนเลขทศนิยมปกติ

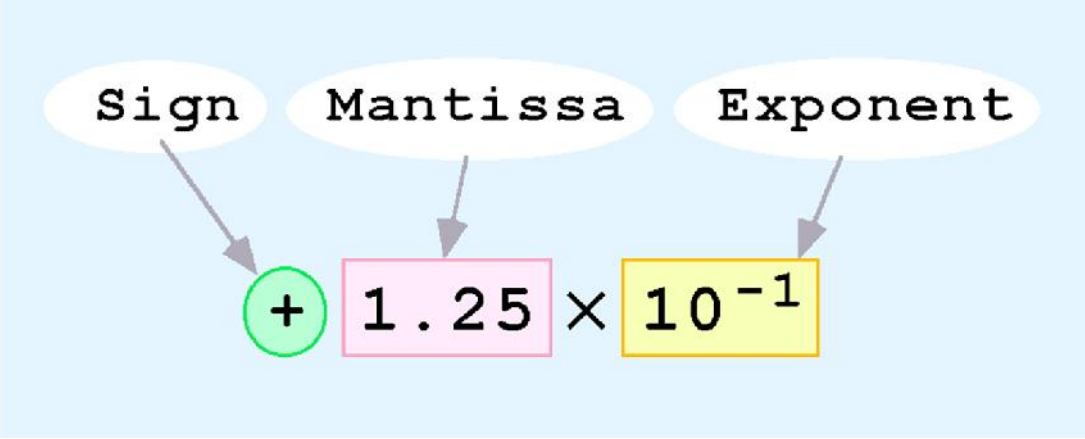
  $0.5 \times 0.25 = 0.125$

- การเขียนเลขทศนิยมในรูปแบบทางวิทยาศาสตร์ (Exponential)

  $0.125 = 1.25 \times 10^{-1}$

  $5{,}000{,}000 = 5.0 \times 10^{6}$

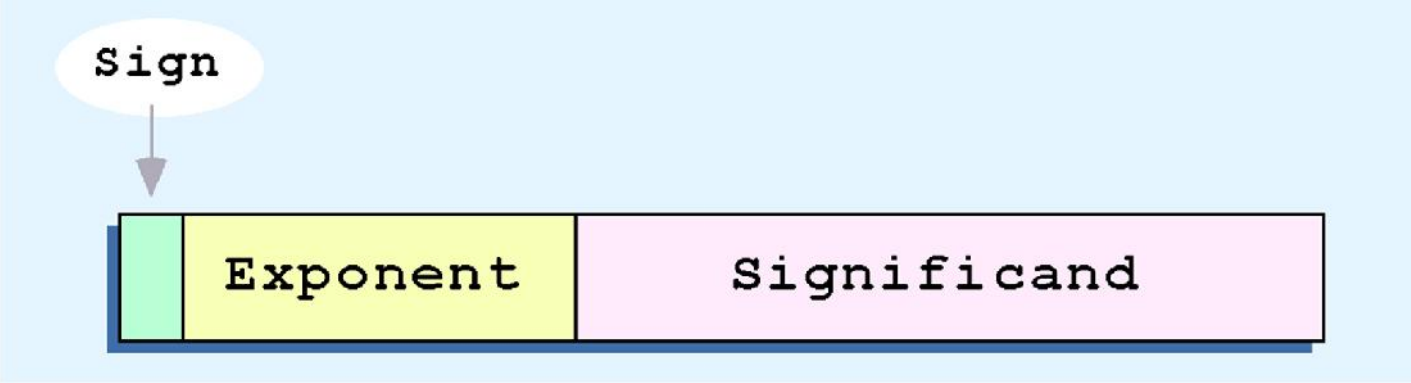
- การเขียนเลขทศนิยมในรูปแบบทางวิทยาศาสตร์ประกอบด้วย 3 ส่วน คือ



การแทนค่าเลขทศนิยม (Floating-point) ในคอมพิวเตอร์

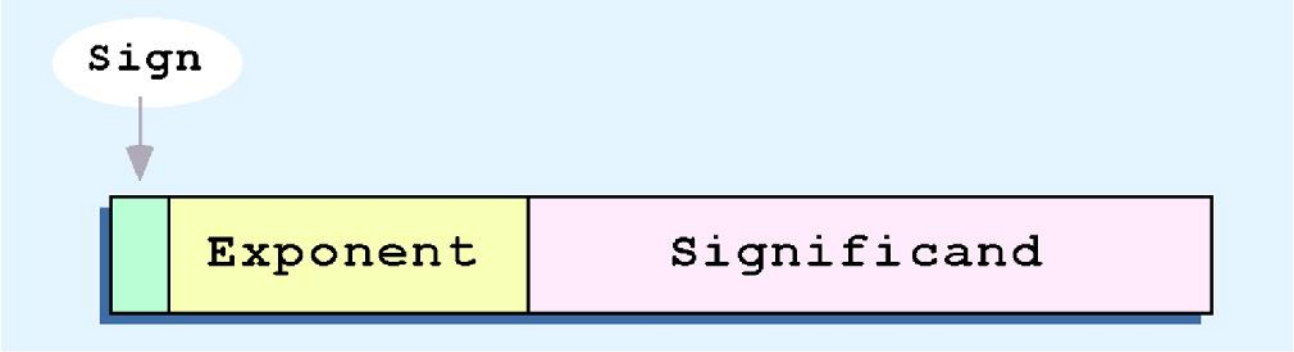
- การแทนตัวเลขทศนิยมในคอมพิวเตอร์จะประกอบด้วย 3 ส่วน คือ

1. บิตเครื่องหมาย (Sign)
2. ค่าเลขยกกำลังของ 2 (Exponent)
3. ค่านัยสำคัญ (Significand) หรือ แมนทิสซา (Mantissa) คือ ค่าความเที่ยง หรือ ค่าเลขทศนิยม



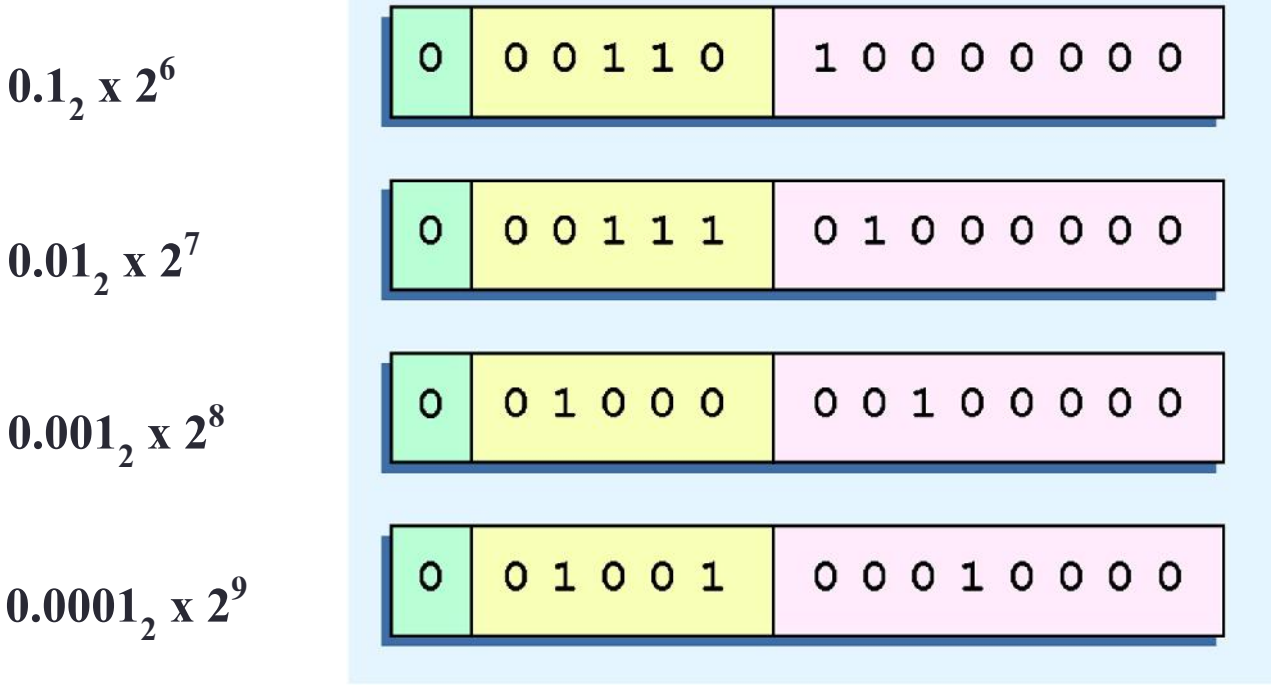
การแทนค่าเลขทศนิยม (Floating-point) ในคอมพิวเตอร์

- มาตรฐาน IEEE-754 กำหนดค่าทศนิยมความเที่ยง 1 เท่า (single precision floating point) จะกำหนด exponent มีขนาด 8 บิต และ significand มีขนาด 23 บิต
- มาตรฐาน IEEE-754 กำหนดค่าทศนิยมความเที่ยง 2 เท่า (double precision floating point) จะกำหนด exponent มีขนาด 11 บิต และ significand มีขนาด 52 บิต



การแทนค่าเลขทศนิยม (Floating-point) ในคอมพิวเตอร์

- ตัวอย่างการแทนตัวเลข 32_{10} ด้วยค่าทศนิยมขนาด 14 บิต (Exponent 5 บิต และ Significand 8 บิต)
- $32_{10} = 2^5$ เขียนในรูปแบบ Exponential : $32_{10} = 1.0 \times 2^5 = 0.1 \times 2^6$
- ส่วนของ exponent จะมีค่า 110 (6_{10}) และส่วนของ significand จะมีค่า 1



รหัสอักขระ (Character Codes)

- การกำหนดรหัสข้อมูลที่ใช้แทนตัวอักขระ (character) จะใช้ตัวเลขเช่นเดียวกับการแทนค่าตัวเลข
- รหัสอักขระแต่ละตัวจะต้องมีค่าที่ไม่ซ้ำกัน และมีจำนวนรหัสเพียงพอที่จะแทนตัวอักขระได้ครบทุกตัว สามารถใช้แทนอักษรในภาษาอื่น ๆ ได้ ไม่เพียงแต่ภาษาอังกฤษเท่านั้น รหัสอักขระในคอมพิวเตอร์มีหลายระบบ ได้แก่

    1. BCD (Binary Coded Decimal)
    2. EBCDIC (Extended Binary Coded Decimal Interchange Code)
    3. ASCII (American Standard Code for International Interchange)
    4. Unicode

BCD (Binary Coded Decimal)

- กำหนดขึ้นเพื่อใช้ในเครื่องเมนเฟรมของบริษัท IBM ในปี 1950 - 1960
- ใช้ระบบเลขฐานสองจำนวน 6 บิต ในการแทนค่าอักขระ 1 ตัว จึงสามารถแทนค่าอักขระได้ทั้งสิ้น $2^6 = 64$ ตัว
- รหัสถูกแบ่งออกเป็น 2 ส่วน ส่วนแรกเรียกว่า Zone bit ซึ่งใช้ 2 บิตซ้ายสุด ส่วน 4 บิตที่เหลือเรียกว่า Numeric bit
- ปัจจุบันไม่นิยมใช้เนื่องจากมีจำนวนรหัสน้อยเกินไป

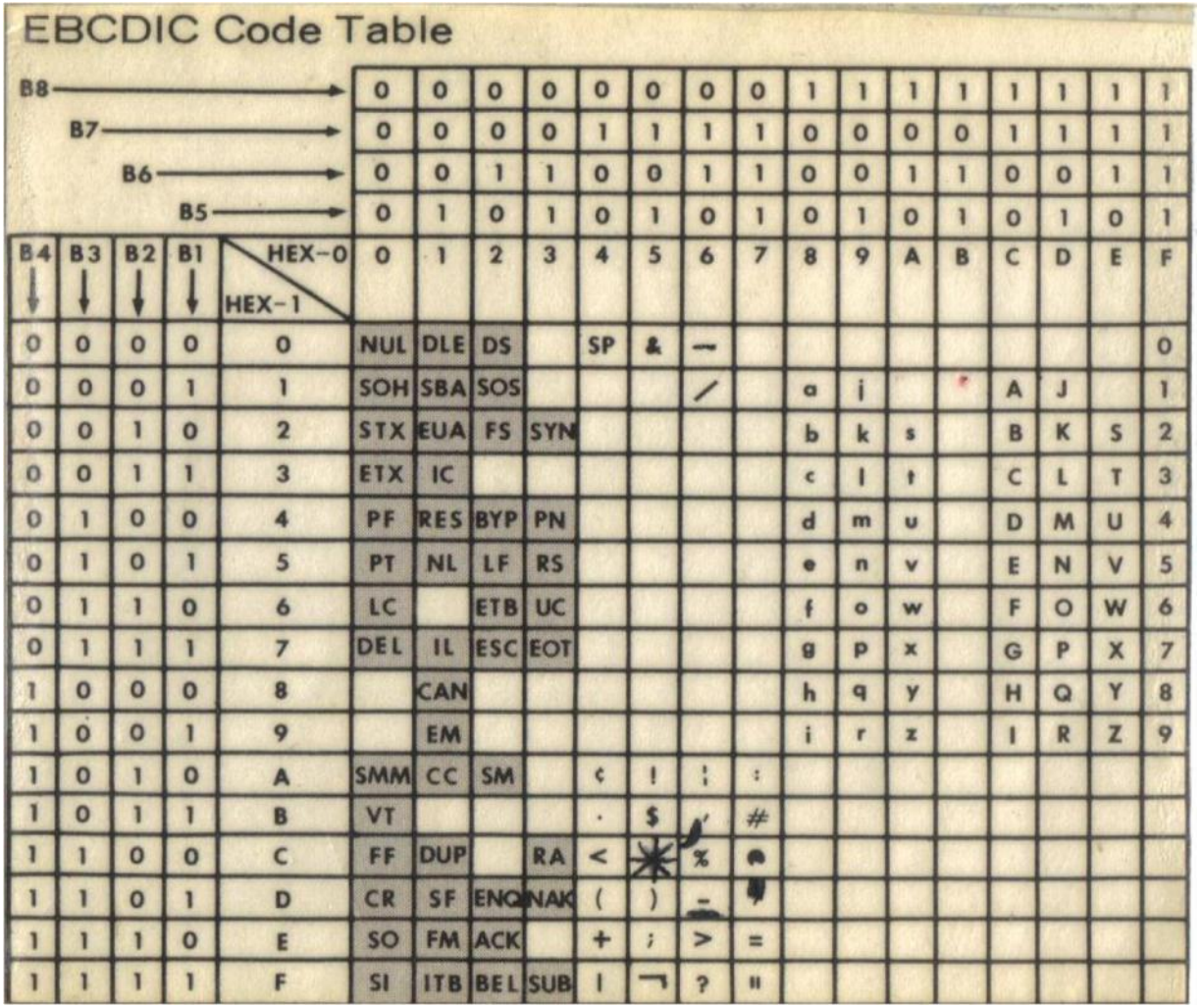
EBCDIC

(Extended Binary Coded Decimal Interchange Code)

- พัฒนามาจากรหัส BCD ในปี 1964 โดยขยายขนาดเป็น 8 บิต ทำให้มีคุณสมบัติเพิ่มเติมที่เรียกว่า Extended คือ สามารถแทนอักขระได้ 256 ตัว (28) ตัว และมีคุณสมบัติที่เรียกว่า Interchange คือ สามารถเปลี่ยนแปลงความหมายของรหัสจากชุดอักขระหนึ่งเป็นชุดอักขระอื่นได้
- รหัสแบ่งออกเป็น 2 ส่วนเช่นเดิม โดยในส่วน Zone bit จะเพิ่มขนาดเป็น 4 บิตใช้สำหรับระบุว่าเป็นชุดอักษรภาษาอังกฤษตัวพิมพ์เล็ก ตัวพิมพ์ใหญ่ ตัวเลข หรือสัญลักษณ์พิเศษ ส่วนบิตที่เหลือ 4 บิตเป็น Numeric bit เหมือนกับรหัส BCD เป็นตัวที่ใช้ระบุว่าอยู่ในช่วงตัวอักขระใด
- ทั้ง EBCDIC และ BCD ยังคงใช้ในเครื่อง IBM mainframes ในปัจจุบัน

EBCDIC

(Extended Binary Coded Decimal Interchange Code)



EBCDIC Code Table

B8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
B5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

B4	B3	B2	B1	HEX-0 / HEX-1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	NUL	DLE	DS		SP	&	—									0
0	0	0	1	1	SOH	SBA	SOS				/		a	j			A	J		1
0	0	1	0	2	STX	EUA	FS	SYN					b	k	s		B	K	S	2
0	0	1	1	3	ETX	IC							c	l	t		C	L	T	3
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0	1	0	1	5	PT	NL	LF	RS					e	n	v		E	N	V	5
0	1	1	0	6	LC		ETB	UC					f	o	w		F	O	W	6
0	1	1	1	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1	0	0	0	8		CAN							h	q	y		H	Q	Y	8
1	0	0	1	9		EM							i	r	z		I	R	Z	9
1	0	1	0	A	SMM	CC	SM		¢	!	¦	:								
1	0	1	1	B	VT				.	$	,	#								
1	1	0	0	C	FF	DUP		RA	<	*	%	@								
1	1	0	1	D	CR	SF	ENQ	NAK	(	)	_	'								
1	1	1	0	E	SO	FM	ACK		+	;	>	=								
1	1	1	1	F	SI	ITB	BEL	SUB	|	¬	?	"								

ASCII

(American Standard Code for International Interchange)

- กำหนดขึ้นโดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI)
- นิยมใช้ในการส่งข้อมูลแบบอนุกรมไปตามสายโทรศัพท์ โทรเลข หรือระหว่างคอมพิวเตอร์กับอุปกรณ์อื่น
- ช่วงเริ่มต้นจะใช้ระบบเลขฐานสอง 7 บิต ในการแทนค่าอักขระ 1 ตัว ซึ่งสามารถแทนได้ 128 ตัว ต่อมาจึงขยายเพิ่มเป็น 8 บิต ทำให้แทนค่าเพิ่มขึ้นเป็น 256 ตัว
- รหัสแบ่งออกเป็น 3 ส่วน
    1. Control character ใช้ในการควบคุมการทำงานของอุปกรณ์แสดงผล ซึ่งเป็นรหัสในช่วง 0-31
    2. Lower ASCII ใช้แทนตัวอักษรในภาษาอังกฤษ และสัญลักษณ์ต่าง ๆ ซึ่งเป็นรหัสในช่วง 32-127
    3. Higher ASCII ใช้แทนตัวอักษรในภาษาอื่น เช่น ภาษาไทย เป็นรหัสในช่วง 128-255

ASCII

(American Standard Code for International Interchange)

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Unicode

- กำหนดขึ้นโดยองค์กรไม่แสวงกำไรชื่อ Unicode consortium เพื่อให้เป็นมาตรฐานสากลใช้ได้กับทุกภาษาทั่วโลก
- ใช้ระบบเลขฐานสอง 16 บิต ในการแทนค่าอักขระ 1 ตัว
- สามารถมีจำนวนรหัสได้ถึง 2^{16} หรือ 65,536 รหัส ซึ่งเพียงพอสำหรับตัวอักษรทั่วโลก เช่น ภาษาจีน หรือญี่ปุ่น รวมทั้งสัญลักษณ์พิเศษต่าง ๆ เช่น สัญลักษณ์ทางคณิตศาสตร์
- ปัจจุบันได้รับความนิยมอย่างมากในระบบปฏิบัติการ โปรแกรมประยุกต์รวมถึงภาษาการโปรแกรมต่าง ๆ ส่วนมากสนับสนุนการใช้งานรหัส Unicode