

องค์ประกอบของเครื่องคอมพิวเตอร์  
และภาษาแอสเซมบลี:  
ARM และ RaspberryPi3

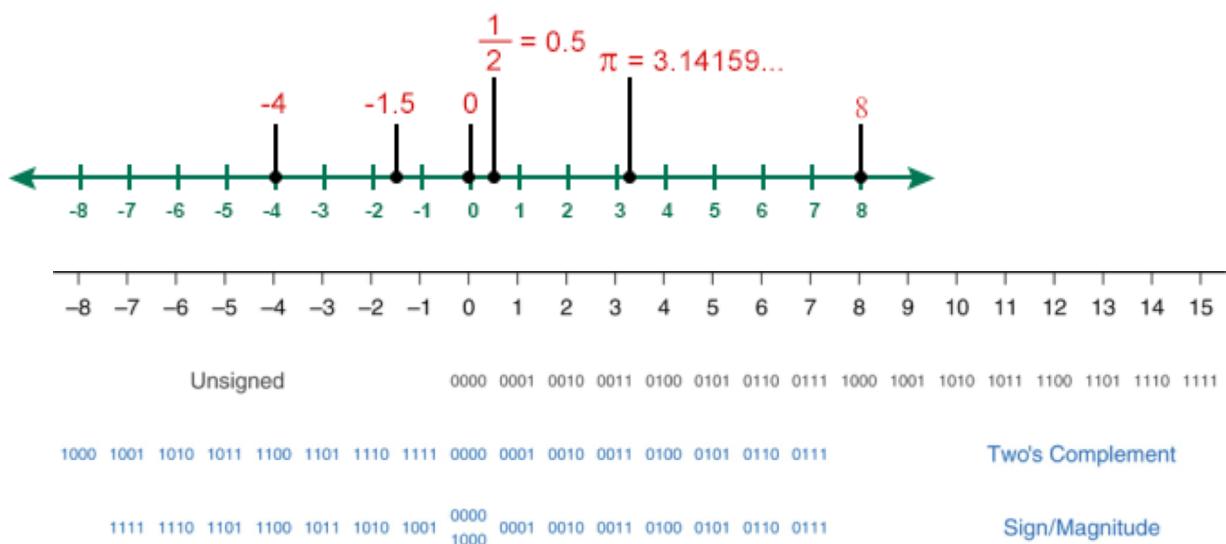
บทที่ 2 ข้อมูลและคณิตศาสตร์ในคอมพิวเตอร์

ผศ.ดร.สุรินทร์ กิตติธรภุล  
ภาควิชาศึกษาดูงานคอมพิวเตอร์ คณะวิศวกรรมศาสตร์  
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

## สารบัญ

- บทที่ 2 ข้อมูลและคณิตศาสตร์ในคอมพิวเตอร์
  - 2.1 ตัวแปรชนิดต่างในภาษา C/C++
  - 2.2 เลขจำนวนเต็มฐานสอง
  - 2.3 คณิตศาสตร์เลขจำนวนเต็มฐานสอง
  - 2.4 เลขทศนิยมฐานสองชนิดจุดคงที่ (Binary Fixed Point)
  - 2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว (Binary Floating Point)
  - 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754
  - 2.7 ตัวอักษร

## 2.1 ตัวแปรชนิดต่างในภาษา C/C++



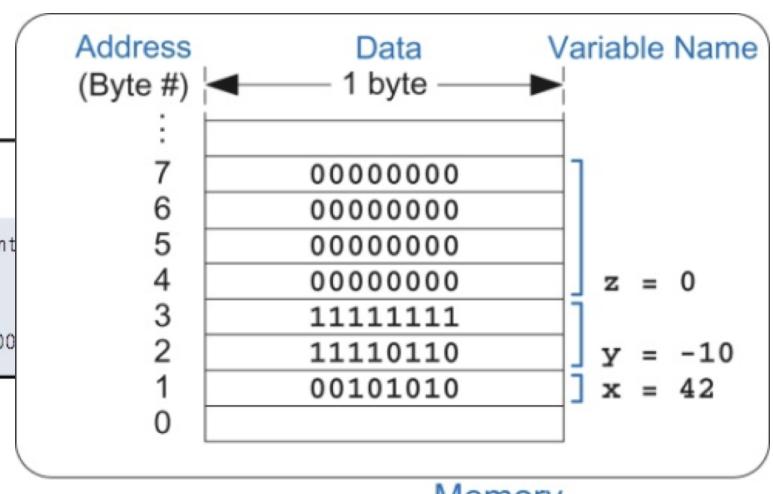
Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

3

## 2.1 ตัวแปรชนิดต่างในภาษา C/C++

### C Code Example eC.3 EXAMPLE DATA TYPES

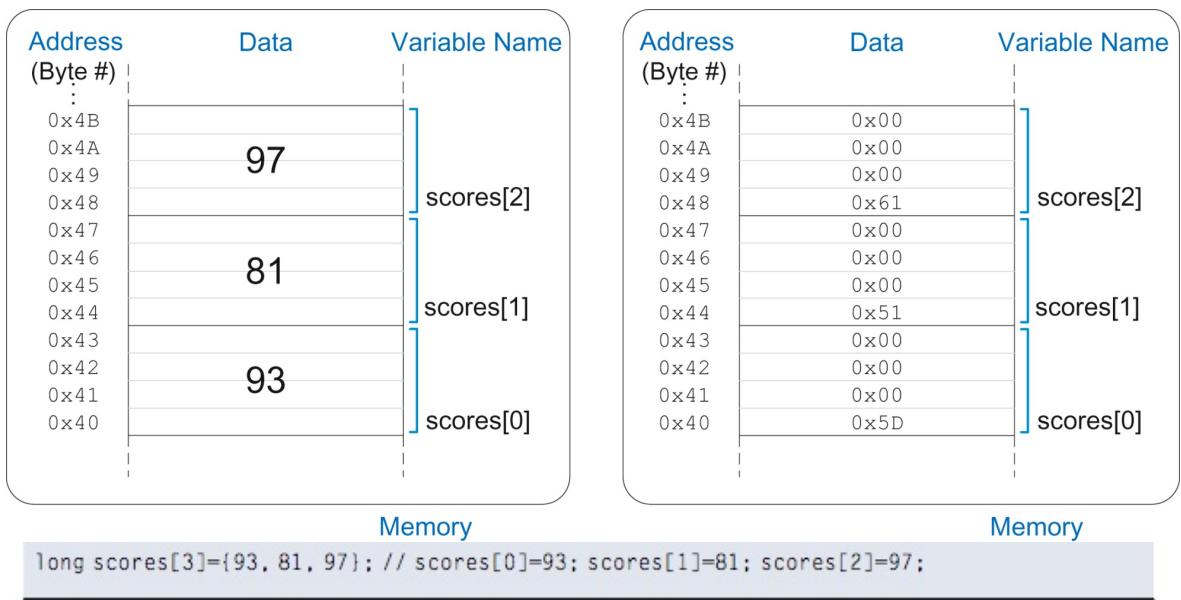
```
// Examples of several data types and their binary representation
unsigned char x = 42;           // x = 00101010
short y = -10;                 // y = 11111111 11110110
unsigned long z = 0;            // z = 00000000 00000000 00000000
```



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

4

## 2.1 ตัวแปรชนิดต่างในภาษา C/C++



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

5

## 2.1 ตัวแปรชนิดต่างในภาษา C/C++

### C Code Example eC.18

```
// Example pointer manip
int salary1, salary2;
int *ptr;

salary1 = 67500;
ptr = &salary1;
salary2 = *ptr + 1000;
```

The diagram shows the memory state for the code example. It consists of two tables labeled (a) and (b), each with columns for Address (Byte #), Data, and Variable Name.

**Table (a) - Initial State:**

Address (Byte #)	Data	Variable Name
0x7B		
0x7A		
0x79		
0x78		
0x77		
0x76		
0x75		
0x74		
0x73		
0x72		
0x71		
0x70		

**Table (b) - After Assignment:**

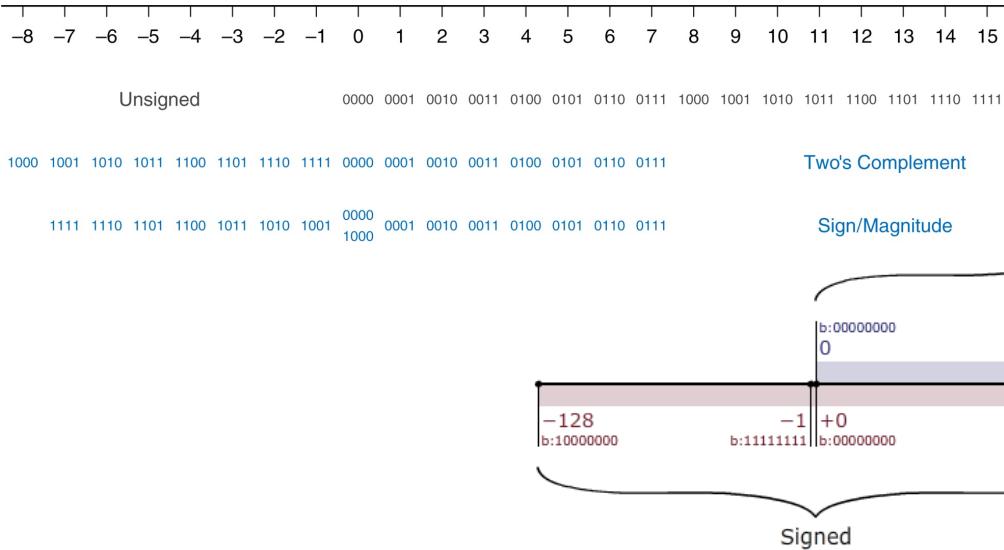
Address (Byte #)	Data	Variable Name
0x7B	0x00	ptr
0x7A	0x00	ptr
0x79	0x00	ptr
0x78	0x70	ptr
0x77	0x00	salary2
0x76	0x01	salary2
0x75	0x0B	salary2
0x74	0x94	salary2
0x73	0x00	salary1
0x72	0x01	salary1
0x71	0x07	salary1
0x70	0xAC	salary1

(a) Memory (b) Memory

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

6

## 2.2 เลขจำนวนเต็มฐานสอง



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

7

### 2.2.1 เลขจำนวนเต็มฐานสอง ชนิดไม่มีเครื่องหมาย

```
unsigned char x = 42; // x = 0b00101010
unsigned short y = 10; // y = 0b000000000000001010
unsigned int z = 0; // z = 0x00000000
```

นิยามที่ 2.2.1. กำหนดให้ เลขจำนวนเต็มฐานสองชนิดไม่มีเครื่องหมาย (Unsigned Integer)  $X_{2,u}$  เป็น เลขฐานสองเขียนอยู่ในรูป

$$X_{2,u} = x_{n-1}x_{n-2}x_{n-3}\dots x_1x_0 \quad (2.1)$$

เมื่อ  $x_i$  คือค่า “1” หรือ “0” ในตำแหน่งที่  $i$  และตำแหน่งของมือสุดคือตำแหน่งที่  $i=0$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

8

## 2.2.1 เลขจำนวนเต็มฐานสอง ชนิดไม่มีเครื่องหมาย

### 2.2.1.1 การแปลงเลขฐานสองเป็นฐานสิบ

จากนิยามที่ 2.2.1 ค่าจำนวนเต็มฐานสิบ  $X_{10,u}$  ของเลข  $X_{2,u}$  สามารถคำนวณได้จาก

$$X_{10,u} = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_12^1 + x_02^0 \quad (2.2)$$

ดังนั้น ค่าฐานสิบ  $X_{10,u}$  อยู่ในช่วง  $0 \leq X_{10,u} < 2^n - 1$

ตัวอย่างที่ 2.2.1. เมื่อ  $n = 4$  บิก เลขจำนวนเต็มฐานสองชนิดไม่มีเครื่องหมาย  $X_{2,u} = 1011_2$

ค่าฐานสิบของ  $X_{2,u}$  คือ

$$X_{10,u} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \quad (2.3)$$

$$= 8 + 0 + 2 + 1 \quad (2.4)$$

$$= 11_{10} \quad (2.5)$$

## 2.2.1 เลขจำนวนเต็มฐานสอง ชนิดไม่มีเครื่องหมาย

บิกที่	เลขฐานสิบ	ผลหาร	เศษ
-	123		
0	123/2	61	1
1	61/2	30	1
2	30/2	15	0
3	15/2	7	1
4	7/2	3	1
5	3/2	1	1
6	1/2	0	1
7	0/2	0	0

บิกที่ $i$	$2^i$	ผลลัพธ์ $- 2^i$	ผลลัพธ์ $x_i$
-		123	
7	$2^7 = 128$	123-128	123 0
6	$2^6 = 64$	123-64	59 1
5	$2^5 = 32$	59-32	27 1
4	$2^4 = 16$	27-16	11 1
3	$2^3 = 8$	11-8	3 1
2	$2^2 = 4$	3-4	3 0
1	$2^1 = 2$	3-2	1 1
0	$2^0 = 1$	1-1	0 1

## 2.2.2 เลขจำนวนเต็มฐานสอง ชนิดมีเครื่องหมาย

```
char x = -2; // x = 0b11111110
short y = -2; // y = 0b1111111111111110
int z = -2; // z = 0xFFFFFE
```

นิยามที่ 2.2.2. กำหนดให้ เลขจำนวนเต็มฐานสองชนิดมีเครื่องหมาย (Signed Integer) แบบ 2-Complement  $X_{2,s}$  เขียนอยู่ในรูป

$$X_{2,s} = x_{n-1}x_{n-2}x_{n-3}\dots x_1x_0 \quad (2.15)$$

เมื่อ ~~x~~ ทำหน้าที่เป็นบิตเครื่องหมาย (Sign bit) และ  $x_i$  คือค่า “1” หรือ “0” ในตำแหน่งที่  $i$  และตำแหน่งขวา มีสอดคือตำแหน่งที่  $i = 0$

## 2.2.2 เลขจำนวนเต็มฐานสอง ชนิดมีเครื่องหมาย

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

Unsigned

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111 Two's Complement

ตัวอย่างที่ 2.2.7. เมื่อ  $n = 4$  บิต เลขจำนวนเต็มฐานสองชนิดมีเครื่องหมายแบบ 2 Complement

$X_{2,s} = 1011_2$  มีค่าฐานสิบเท่ากับเท่าไร

ค่าฐานสิบของ  $X_{2,s}$  คือ

$$X_{10,s} = -1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \quad (2.17)$$

$$= -8 + 0 + 2 + 1 \quad (2.18)$$

$$= -5_{10} \quad (2.19)$$

เลขจำนวนเต็มฐานสองชนิดมีเครื่องหมาย 2-Complement ขนาด  $n = 4$  บิตจะมีค่าฐานสิบอยู่ในช่วง  $-8$  ถึง  $+7$

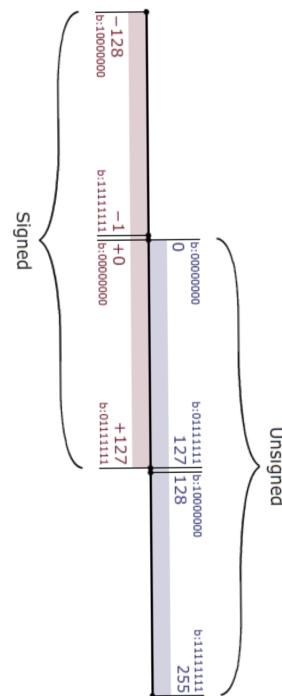
### Sign Extension

เลขฐานสอง $n=4$	$X_{10,s}$ ค่าฐานสิบ มีเครื่องหมาย	$X_{10,u}$ ค่าฐานสิบ ไม่มีเครื่องหมาย	เลขฐานสอง $n=5$	$X_{10,s}$ ค่าฐานสิบ มีเครื่องหมาย	$X_{10,u}$ ค่าฐานสิบ ไม่มีเครื่องหมาย
1000	-8	8	1 0000	-16	16
1001	-7	9	1 0111	-9	23
1010	-6	10	1 1000	-8	24
1011	-5	11	1 1001	-7	25
1100	-4	12	1 1010	-6	26
1101	-3	13	1 1011	-5	27
1110	-2	14	1 1100	-4	28
1111	-1	15	1 1101	-3	29
0000	0	0	1 1110	-2	30
0001	1	1	1 1111	-1	31
0010	2	2	0 0000	0	0
0011	3	3	0 0001	1	1
0100	4	4	0 0010	2	2
0101	5	5	0 0011	3	3
0110	6	6	0 0100	4	4
0111	7	7	0 0101	5	5
			0 0110	6	6
			0 0111	7	7
			...	...	...
			0 1111	15	15

### Sign Extension

เลขฐานสอง $n=5$	$X_{10,s}$ ค่าฐานสิบ มีเครื่องหมาย	$X_{10,u}$ ค่าฐานสิบ ไม่มีเครื่องหมาย	เลขฐานสอง $n=8$	$X_{10,s}$ ค่าฐานสิบ มีเครื่องหมาย	$X_{10,u}$ ค่าฐานสิบ ไม่มีเครื่องหมาย
1 0000	-16	16	1000 0000	-128	128
...	...	...	...	...	...
1 0111	-9	23	1111 0111	-9	
1 1000	-8	24	1111 1000	-8	248
1 1001	-7	25	1111 1001	-7	249
1 1010	-6	26	1111 1010	-6	250
1 1011	-5	27	1111 1011	-5	251
1 1100	-4	28	1111 1100	-4	252
1 1101	-3	29	1111 1101	-3	253
1 1110	-2	30	1111 1110	-2	254
1 1111	-1	31	1111 1111	-1	255
0 0000	0	0	0000 0000	0	0
0 0001	1	1	0000 0001	1	1
0 0010	2	2	0000 0010	2	2
0 0011	3	3	0000 0011	3	3
0 0100	4	4	0000 0100	4	4
0 0101	5	5	0000 0101	5	5
0 0110	6	6	0000 0110	6	6
0 0111	7	7	0000 0111	7	7
...	...	...	0000 1000	8	8
0 1111	15	15	...	...	...
			0111 1111	127	127

เลขฐานสอง $n=8$	$X_{10,s}$ ค่าฐานสิบ มีเครื่องหมาย	$X_{10,u}$ ค่าฐานสิบ ไม่มีเครื่องหมาย
1000 0000	-128	128
...	...	...
1111 0111	-9	
1111 1000	-8	248
1111 1001	-7	249
1111 1010	-6	250
1111 1011	-5	251
1111 1100	-4	252
1111 1101	-3	253
1111 1110	-2	254
1111 1111	-1	255
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
0000 0011	3	3
0000 0100	4	4
0000 0101	5	5
0000 0110	6	6
0000 0111	7	7
0000 1000	8	8
...	...	...
0111 1111	127	127



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

15

## 2.2.3 เลขจำนวนเต็มฐานสอง ชนิดมีเครื่องหมาย Sign-Magnitude

นิยามที่ 2.2.3. กำหนดให้ เลขจำนวนเต็มฐานสองชนิดมีเครื่องหมาย (Signed Integer) แบบ Sign-Magnitude  $X_{2,sm}$  เขียนอยู่ในรูป

$$X_{2,sm} = s x_{n-2} x_{n-3} \dots x_1 x_0 \quad (2.34)$$

เมื่อ  $s$  คือบิตเครื่องหมาย (Sign bit) และ  $x_i$  คือค่า “1” หรือ “0” ในตำแหน่งที่  $i$  และตำแหน่งของ  $x_0$  มีอสูดคือตำแหน่งที่  $i = 0$

การแปลงเลขจำนวนเต็มฐานสองแบบ Sign-Magnitude ให้เป็นค่าฐานสิบสามารถทำได้โดย

$$X_{10,sm} = (-1)^s \times (x_{n-2} \times 2^{n-2} + \dots + x_1 \times 2^1 + x_0 \times 2^0) \quad (2.35)$$

ดังนั้น ค่าฐานสิบ  $X_{10,s}$  อยู่ในช่วง  $-2^{n-1} + 1$  ถึง  $+2^{n-1} - 1$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

16

## 2.2.3 เลขจำนวนเต็มฐานสอง ชนิดมีเครื่องหมาย Sign-Magnitude

Table 2.8: ค่าฐานสิบแบบมีเครื่องหมาย Sign-Magnitude, แบบ 2-Complement, และแบบไม่มีเครื่องหมาย (Unsigned) ของเลขฐานสองความยาว  $n=4$  บิตทั้งหมด  $2^4=16$  แบบ

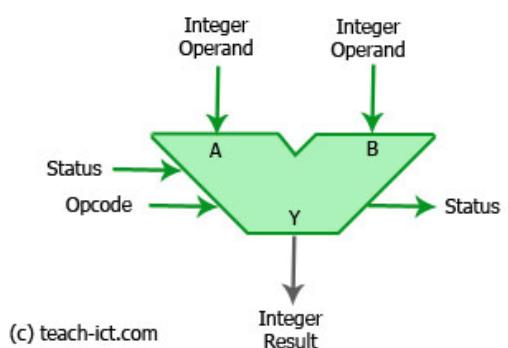
เลขฐานสอง $n=4$ บิต	$X_{10,sm}$ ค่าฐานสิบ Sign-Mag	$X_{10,s}$ ค่าฐานสิบ 2-Comp.	$X_{10,u}$ ค่าฐานสิบ Unsigned
1111	-7	-1	15
1110	-6	-2	14
1101	-5	-3	13
1100	-4	-4	12
1011	-3	-5	11
1010	-2	-6	10
1001	-1	-7	9
1000	-0	-8	8
0000	+0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

17

## 2.3 คณิตศาสตร์เลขจำนวนเต็ม

- สัญญาณ Opcode เพื่อสั่งการทำงาน เช่น บวก ลบ เป็นต้น
- ผลลัพธ์ Y เป็นจำนวนเต็มชนิดไม่มีเครื่องหมายขนาด n บิต
- สัญญาณ Status ประกอบด้วย
  - หากเครื่องหมาย N (Negative) สำหรับเลขจำนวนเต็มชนิดมีเครื่องหมาย
  - หาก Z (Zero)=1 เพื่อบ่งบอกว่าผลลัพธ์ Y มีค่าเท่ากับศูนย์ทุกบิต
  - หาก Carry  $c_n$  สำหรับตัวดิจิตที่ n
  - หาก Overflow (V) เพื่อบ่งบอกความผิดพลาด



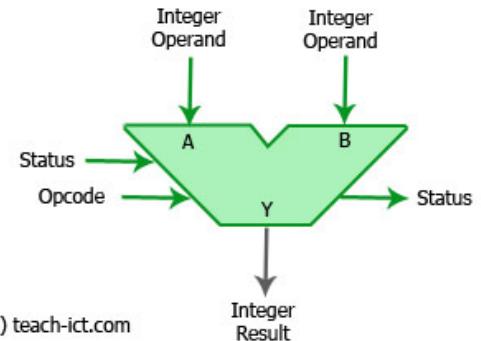
หากสัญญาณเหล่านี้จะบันทึกลงในรีจิสเตอร์สถานะ (Status Register) สำหรับให้วงจรและโปรแกรมเมอร์ตรวจสอบด้วยwangจุดดิจิทลและคำสั่งภาษาแอสเซมบลี ในบทที่ 4

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

18

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

	$c_n$	$c_{n-1}$	$c_{n-2}$	..	$c_2$	$c_1$	$c_0$	+
$X_{2,u} +$		$x_{n-1}$	$x_{n-2}$	..	$x_2$	$x_1$	$x_0$	
$Y_{2,u}$		$y_{n-1}$	$y_{n-2}$	..	$y_2$	$y_1$	$y_0$	
$Z_{2,u}$		$z_{n-1}$	$z_{n-2}$	..	$z_2$	$z_1$	$z_0$	



(c) teach-ict.com

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

19

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

$$c_{i+1}z_i = x_i + y_i + c_i \quad (2.40)$$

เมื่อ  $i=0, 1, 2, \dots, n-1$  โดย  $c_0 = 0$  และสัญลักษณ์  $+$  คือการบวกเลข ไม่ใช่การ OR กันเชิงตรรกศาสตร์ ในวิชาออกแบบจรวดิจิทัล เราเรียกว่าจะรบกวนเลขชนิดนี้ว่า วงจร Full Adder โดยวงจรจะนำบิท ข้อมูลจำนวน 3 บิตมากระทำการทางตรรกศาสตร์ได้ผลลัพธ์  $z_i$  โดย

$$z_i = x_i \oplus y_i \oplus c_i \quad (2.41)$$

เมื่อ  $\oplus$  คือ กระบวนการ Exclusive-OR และบิทตัวท้าย  $c_{i+1}$

$$c_{i+1} = (x_i \& y_i) | (x_i \& c_i) | (y_i \& c_i) \quad (2.42)$$

เมื่อ  $\&$  คือ กระบวนการ AND และ  $|$  คือ กระบวนการ OR วงจรบวกเลขชนิดไม่มีเครื่องหมายขนาด  $n$  บิต นี้สามารถตรวจสอบการเกิดโอเวอร์โฟล์วได้โดย

$$V = c_n \quad (2.43)$$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

20

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.1. จงคำนวณหาค่าของ  $5 + 9$  ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ Unsigned ขนาด 4 บิต  $5 + 9 = 14$  ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิต สามารถคำนวณได้ดังนี้

	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	Overflow
	0	0	0	1	0	$V=0$
$X=5 +$	0	1	0	1	1	+
$Y=9$	1	0	0	1		
$Z=14$	1	1	1	0		

การบวกเลขขนาด 4 บิตแบบไม่มีเครื่องหมาย:  $5 + 9 = 14$  พร้อมตัวทด และผลลัพธ์ถูกต้องเนื่องจากไม่เกิดโอเวอร์โฟล์ ( $V=c_n=0$ )

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.2. จงคำนวณหาค่าของ  $7 + 9$  ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ Unsigned ขนาด 4 บิต  $7 + 9 = 0$  ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิต ซึ่งไม่สามารถแสดงผลค่า  $16_{10}$  ได้ ดังนี้

	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	Overflow
	1	1	1	1	0	$V=c_n=1$
$X=7 +$	0	1	1	1	1	+
$Y=9$	1	0	0	1		
$Z=16$	0	0	0	0		

สาเหตุของการเกิด Overflow เนื่องจากผลลัพธ์มีค่าอยู่นอกย่านที่เป็นไปได้ โดยสามารถตรวจสอบอย่างง่ายดายโดย  $c_4 = 1$  ( $V$ : Overflow) เมื่อเกิดโอเวอร์โฟล์ ผลลัพธ์ที่ได้จึงมีค่าไม่ถูกต้อง (Invalid)

## 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.2. จงคำนวณหาค่าของ  $7 + 9$  ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ Unsigned ขนาด 4 บิต  $7 + 9 = 16 = 0$  ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิต ซึ่งไม่สามารถแสดงผลค่า  $16_{10}$  ได้ ดังนี้

	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	Overflow
	1	1	1	1	0	$V=c_n=1$
$X=7+$	0	1	1	1		+
$Y=9$	1	0	0	1		
$Z=16$	0	0	0	0		

สาเหตุของการเกิด Overflow เนื่องจากผลลัพธ์มีค่าอยู่นอกอย่างที่เป็นไปได้ โดยสามารถตรวจสอบอย่างง่ายโดย  $c_4 = 1$  ( $V$ : Overflow) เมื่อเกิดโอเวอร์เฟลว์ ผลลัพธ์ที่ได้จะมีค่าไม่ถูกต้อง (Invalid)

## 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

วงจรคูณเลขชนิดที่ 1

การคูณเลขมีความซับซ้อนสูงสำหรับใช้งานจริงทั้งด้านเวลาและความซับซ้อน การคูณเลขที่ง่ายที่สุดคือการบวกเลขแล้ววนบวกซ้ำ ตามวงจรที่จะอธิบายในรูปที่ 2.5 และ 2.6 ต่อไปนี้

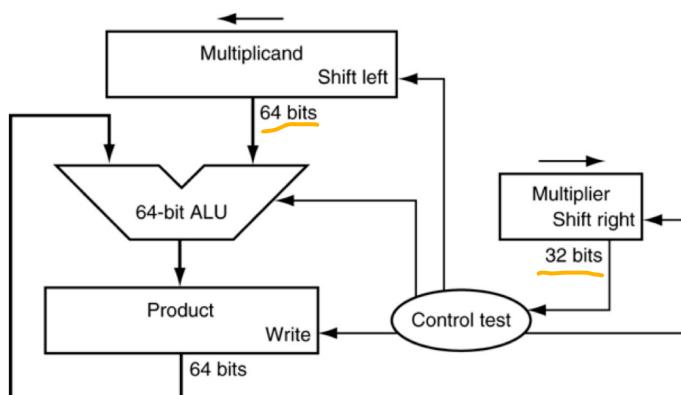


Figure 2.5: วงจรคูณเลขขนาด  $n = 32$  บิต ชนิดที่ 1 โดยใช้ ALU ขนาด  $2n = 64$  บิต และรีจิสเตอร์ตัวตั้งขนาด  $2n = 64$  บิต (ที่มา: Patterson and Hennessy (2000))

## 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.3. จงคูณ  $1010_2$  ( $10_{10}$ ) ด้วย  $1101_2$  ( $13_{10}$ ) ตามวิธีที่ 2.5

ผลคูณเลขขนาด 4 บิต ตัวตั้ง=  $1010_2$  ( $10_{10}$ ) ตัวคูณ=  $1101_2$  ( $13_{10}$ ) เท่ากับ  $130_{10}$  ด้วยวิธีในรูปที่

2.5

Table 2.9: ผลคูณเลขขนาด 4 บิต ตัวตั้ง=  $1010_2$  ( $10_{10}$ ) ตัวคูณ=  $1101_2$  ( $13_{10}$ ) เท่ากับ  $130_{10}$  ด้วยวิธีในรูปที่ 2.5 หมายเหตุ Shift R คือ การเลื่อนบิตไปทางขวา

รอบ Iteration	ตัวตั้ง Multicand	ตัวคูณ Multiplier	ผลคูณ <sub>2</sub> Product <sub>2</sub>	ผลคูณ <sub>10</sub> Product <sub>10</sub>	กระทำ Action
-	0000 1010 <sub>2</sub>	1101 <sub>2</sub>	0000 0000 <sub>2</sub>	0 <sub>10</sub>	Initialized
0	0000 1010 <sub>2</sub> 0001 0100 <sub>2</sub>	1101 <sub>2</sub> 0110 <sub>2</sub>	0000 1010 <sub>2</sub> 0000 1010 <sub>2</sub>	10 <sub>10</sub> 10 <sub>10</sub>	Add Shift R
1	0001 0100 <sub>2</sub> 0010 1000 <sub>2</sub>	0110 <sub>2</sub> 0011 <sub>2</sub>	0000 1010 <sub>2</sub> 0000 1010 <sub>2</sub>	10 <sub>10</sub> 10 <sub>10</sub>	None Shift R
2	0010 1000 <sub>2</sub> 0101 0000 <sub>2</sub>	0011 <sub>2</sub> 0001 <sub>2</sub>	0011 0010 <sub>2</sub> 0011 0010 <sub>2</sub>	50 <sub>10</sub> 50 <sub>10</sub>	Add Shift R
3	0101 0000 <sub>2</sub> 1010 0000 <sub>2</sub>	0001 <sub>2</sub> 0000 <sub>2</sub>	1000 0010 <sub>2</sub> 1000 0010 <sub>2</sub>	130 <sub>10</sub> 130 <sub>10</sub>	Add Shift R

↑ ตัวตั้ง Shift L  
↑ ตัวคูณ Shift R

## 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

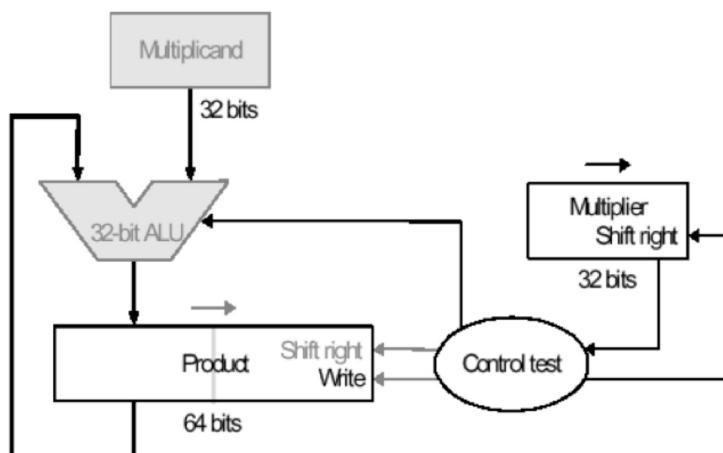


Figure 2.6: วิธีคูณเลขขนาด  $n = 32$  บิต ชนิดที่ 2 โดยใช้ ALU ขนาด  $n = 32$  บิต และรีจิสเตอร์ตัวตั้งขนาด  $n = 32$  บิต (ที่มา: Patterson and Hennessy (2000))

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.4. จงคูณ  $1010_2$  ( $10_{10}$ ) ด้วย  $1101_2$  ( $13_{10}$ ) ตามวิธีที่ 2.6

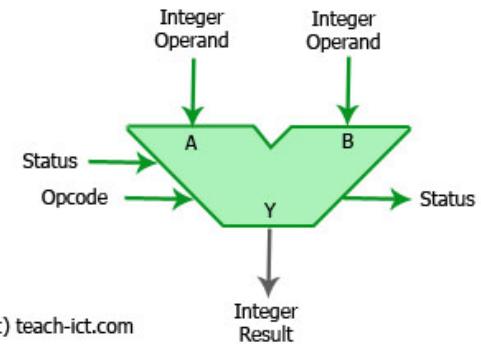
ผลคูณเลขขนาด 4 บิต ตัวตั้ง= $1010_2$  ( $10_{10}$ ) ตัวคูณ= $1101_2$  ( $13_{10}$ ) เท่ากับ  $130_{10}$  ด้วยวิธีในรูปที่ 2.6 หมายเหตุ Shift R คือ การเลื่อนบิตไปทางขวา

รอบ Iteration	ตัวตั้ง Multicand	ตัวคูณ Multiplier	ผลคูณ <sub>2</sub> Product <sub>2</sub>	ผลคูณ <sub>10</sub> Product <sub>10</sub>	กระทำ Action
-	$1010_2$	$1101_2$	$0000\ 0000_2$	$0_{10}$	Initialized
0	$1010_2$	$1101_2$	$1010\ 0000_2$	$160_{10}$	Add
	$1010_2$	$0110_2$	$0101\ 0000_2$	$80_{10}$	Shift R
1	$1010_2$	$0110_2$	$0101\ 0000_2$	$80_{10}$	None
	$1010_2$	$0011_2$	$0010\ 1000_2$	$40_{10}$	Shift R
2	$1010_2$	$0011_2$	$1100\ 1000_2$	$200_{10}$	Add
	$1010_2$	$0001_2$	$0110\ 0100_2$	$100_{10}$	Shift R
3	$1010_2$	$0001_2$	$1\ 0000\ 0100_2$	$260_{10}$	Add
	$1010_2$	$0000_2$	$1000\ 0010_2$	$130_{10}$	Shift R

### 2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

### 2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

	$c_n$	$c_{n-1}$	$c_{n-2}$	..	$c_2$	$c_1$	$c_0$	+
$X_{2,s} +$		$x_{n-1}$	$x_{n-2}$	..	$x_2$	$x_1$	$x_0$	
$Y_{2,s}$		$y_{n-1}$	$y_{n-2}$	..	$y_2$	$y_1$	$y_0$	
$Z_{2,s}$		$z_{n-1}$	$z_{n-2}$	..	$z_2$	$z_1$	$z_0$	



(c) teach-ict.com

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

29

### 2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

$$c_{i+1} z_i = x_i + y_i + c_i \quad (2.44)$$

เมื่อ  $i=0, 1, 2, \dots, n-1$  โดย  $c_0 = 0$

ผลลัพธ์ของการบวกเลขจำนวน 3 บิต สามารถคำนวณได้จากการ Full Adder ดังนี้

$$z_i = x_i \oplus y_i \oplus c_i \quad (2.45)$$

เมื่อ  $\oplus$  คือ กระบวนการ Exclusive-OR

$$c_{i+1} = (x_i \& y_i) | (x_i \& c_i) | (y_i \& c_i) \quad (2.46)$$

เมื่อ  $\&$  คือ กระบวนการ AND และ  $|$  คือ กระบวนการ OR

การเกิดโอเวอร์ฟอล์วของการบวกเลขชนิดมีเครื่องหมาย 2-Complement ได้โดย

$$V = c_n \oplus c_{n-1} \quad (2.47)$$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

30

## 2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

ตัวอย่างที่ 2.3.5. จงคำนวณหาค่าของ  $7 + 3$  ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ 2-Complement ขนาด 4 บิต ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิต สามารถคำนวณได้ดังนี้

	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	Overflow
	0	1	1	1	0	V=1
$X = 7$	0	1	1	1	1	+
$+Y = +3$	0	0	1	1		
$Z = -6$	1	0	1	0		

## 2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

ตัวอย่างที่ 2.3.7. จงคำนวณหาค่าของ  $-3 - 6 = (-3) + (-6)$  ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิต สามารถคำนวณได้แต่ผลลัพธ์กลับไม่ถูกต้องเนื่องจากเกิดโอเวอร์โฟล์ ดังนี้

	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	Overflow
	1	0	0	0	0	V=1
$X = -3 +$	1	1	0	1	1	+
$-Y = -6$	1	0	1	0		
$Z = +7$	0	1	1	1		

เนื่องจากฮาร์ดแวร์คำนวณ  $c_4 \text{ xor } c_3 = 1 \text{ xor } 0 = 1$  ทำให้ตรวจได้ว่าเกิด Overflow ทำให้ผลลัพธ์มีค่าไม่ถูกต้อง

## 2.4 เลขทศนิยมฐานสองชนิดจุดคงที่ (Fixed Point)

นิยามที่ 2.4.1. กำหนดให้  $F_2$  เป็นเลขทศนิยมฐานสองชนิด Signed Magnitude เขียนอยู่ในรูป

$$F_2 = [s][x_{n-1}x_{n-2}x_{n-3}\dots x_1x_0].[y_{-1}y_{-2}y_{-3}\dots y_{-m}] \quad (2.48)$$

นิยมใช้ชนิดขนาด-เครื่องหมาย ประกอบด้วย 3 ส่วน คือ บิตเครื่องหมาย (Sign bit:  $s$  หรือ  $\pm$ ) โดย  $s=0$  แทนเครื่องหมายบวก และ  $s=1$  แทนเครื่องหมายลบ ส่วนจำนวนเต็ม (Integer:  $X_{2,u}$ ) มีความยาว  $n$  บิต และส่วนทศนิยม (Fraction:  $F_2$ ) ยาว  $m$  บิต รวมความยาวทั้งหมด  $m+n+1$  บิต

โดย  $F_{10}$  คือ ค่าฐานสิบของเลขทศนิยมฐานสองชนิดจุดคงที่  $F_2$  สามารถคำนวณได้จาก

$$F_{10} = \pm[x_{n-1}2^{n-1} + \dots + x_12^1 + x_02^0 + y_{-1}2^{-1} + y_{-2}2^{-2} + y_{-3}2^{-3} + \dots + y_{-m}2^{-m}] \quad (2.49)$$

หรือ

$$F_{10} = \pm[x_{n-1}2^{n-1} + \dots + x_12^1 + x_02^0 + \frac{y_{-1}}{2} + \frac{y_{-2}}{4} + \frac{y_{-3}}{8} + \dots + \frac{y_{-m}}{2^m}] \quad (2.50)$$

## 2.4 เลขทศนิยมฐานสองชนิดจุดคงที่ (Fixed Point)

ตัวอย่างที่ 2.4.1. จงแปลงเลขฐานสิบท่อไปนี้เป็นเลขทศนิยมฐานสองชนิดจุดคงที่  $m=n=2$  บิต

ใน DSP  
DSP + CPU  
soc

05 1 025

$$+0.75_{10} = 000.11_2 \quad (2.51)$$

$$+3.00_{10} = 011.00_2 \quad (2.52)$$

$$\underline{-3.75_{10} = 111.11_2} \quad (2.53)$$

ผู้อ่านสามารถ Samaritans เขียนเลขทศนิยมฐานสองชนิดจุดคงที่ได้ตามรูปแบบนี้

$$F_2 = [s][X_{2,u}].[Y_2] \quad (2.54)$$

ค่าทศนิยม ( $Y_2$ ) มีความยาว  $m$  บิต เขียนเป็นสัญลักษณ์ได้ดังนี้

$$Y_2 = y_{-1}y_{-2}y_{-3}\dots y_{-m} \quad (2.55)$$

## 2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว (Floating Point)

นิยามที่ 2.5.1. เลขทศนิยมชนิดจุดลอยตัวฐานสองที่อยู่ในรูปนอมัลไลซ์ ประกอบด้วย 3 ส่วน คือ บิทเครื่องหมาย (Sign bit:  $s$ ) ค่านัยสำคัญ (Significand:  $S_2$ ) และค่ายกกำลัง (Exponent:  $E_2$ ) มีลักษณะดังนี้

$$\pm [1.y_{-1}y_{-2}y_{-3}\dots y_{-m}]_2 \times 2^{E_2} \quad (2.59)$$

เลขยกกำลังเป็นเลขจำนวนเต็มฐานสองชนิด sign-magnitude ความยาว  $n$  บิท ดังนี้

$$E_2 = \pm[e_{n-1}e_{n-2}\dots e_0]_2 \quad (2.60)$$

เมื่อ  $e_i$  แต่ละบิทมีค่า “1” หรือ “0” ในตำแหน่งที่  $i$   $s$  คือ Sign bit และ  $n$  คือ จำนวนบิทซึ่งกำหนดไว้ก่อนจะออกแบบวงจร

## 2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว (Floating Point)

จากนิยามที่ 2.59 ค่านัยสำคัญ (Significand)  $S_2$  ความยาว  $m+1$  บิท สามารถเขียนใหม่ได้ดังนี้

$$S_2 = \underline{[1.y_{-1}y_{-2}y_{-3}\dots y_{-m}]}_2 \quad (2.61)$$

โดย 1. นำ   
 Normalize

ซึ่งมีความสำคัญต่อรูปแบบการเขียน เนื่องจากจะจะต้องทำการนอมัลไลซ์ผลลัพธ์ที่ได้จากการคำนวณเสมอ ค่านัยสำคัญที่น้อมัลไลซ์ข้างต้นสามารถคำนวณหาค่าฐานสิบ ได้ดังนี้

$$S_{10} = \pm \left[ 1 + \frac{y_{-1}}{2} + \frac{y_{-2}}{4} + \frac{y_{-3}}{8} + \dots + \frac{y_{-m}}{2^m} \right] \times (2^{\oplus E_2}) \quad (2.62)$$

## 2.5 เลขศนิยมฐานสองชนิดจุดลอยตัว (Floating Point)

ตัวอย่างที่ 2.5.2. จงแปลงเลขศนิยมฐานสองชนิดจุดลอยตัวที่น้อมลัลซ์แล้ว  $k=4$  บิตให้เป็นเลขศนิยมฐานสิบ

วิธีทำ

$$(-1)^0 \times 1.1010_2 \times 2^{-3}$$

1. ปรับจุดศนิยม เพื่อให้เป็นเลขฐานสองชนิด Sign-Magnitude และเลขยกกำลังเท่ากับ 0  
 $+0.001101_2 \times 2^0$

2. แปลงค่าเลขฐานสองชนิดที่เลื่อนตำแหน่งแล้วให้เป็นฐานสิบ

$$(1 \times 2^{-3}) + (1 \times 2^{-4}) + (1 \times 2^{-6})$$

$$= 0.125 + 0.0625 + 0.015625$$

$$= 0.203125_{10}$$

## 2.5 เลขศนิยมฐานสองชนิดจุดลอยตัว (Floating Point)

ตัวอย่างที่ 2.5.4. จงบวกเลขศนิยมฐานสองชนิดจุดลอยตัวที่น้อมลัลซ์แล้ว ดังต่อไปนี้

วิธีทำ

สมมติว่ามีเลขศนิยมฐานสองขนาด 4 บิต (4-bit)

$$1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2} \text{ หรือ } (0.5_{10} + -0.4375_{10})$$

1. เลื่อนตำแหน่งจุดศนิยมและปรับเลขยกกำลังที่มีค่าน้อยกว่า ให้ตรงกับเลขยกกำลังของเลขที่มีค่ามากกว่า

$$1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$$

2. บวกค่านัยสำคัญที่เลื่อนตำแหน่งแล้ว

$$(1.000_2 - 0.111_2) \times 2^{-1} = 0.001_2 \times 2^{-1}$$

3. น้อมลัลซ์ค่าผลลัพธ์และตรวจสอบการเกิดโอเวอร์ไฟล์ว์และอันเดอร์ไฟล์ว์

$$1.000_2 \times 2^{-4}, \text{ ไม่เกิดโอเวอร์ไฟล์ว์และอันเดอร์ไฟล์ว์}$$

4. ปัดค่าให้เหลือ 4 บิตและอาจต้องน้อมลัลซ์เมื่อจำเป็น

$$1.000_2 \times 2^{-4} (\text{ไม่เปลี่ยนแปลง}) = 0.0625_{10}$$

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

- ชนิด Single precision (32-bit) ตรงกับตัวแปรชนิด float ในภาษา C/C++ และ Java
- ชนิด Double precision (64-bit) ตรงกับตัวแปรชนิด double ในภาษา C/C++ และ Java เวอร์ชันล่าสุดของ IEEE 754 คือ ปี ค.ศ. 2019 รายละเอียดเพิ่มเติม

ตัวอย่างการประมวลผลตัวตัวแปรที่ใช้มาตรฐานนี้

```
float a = -5; /* a = 0xC0A00000 */
double b = -0.75; /* b = 0xBFE8000000000000 */
```

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

- ชนิด Single precision (32-bit) ตรงกับตัวแปรชนิด float ในภาษา C/C++ และ Java
- ชนิด Double precision (64-bit) ตรงกับตัวแปรชนิด double ในภาษา C/C++ และ Java

จำนวนบิต  
ที่ใช้

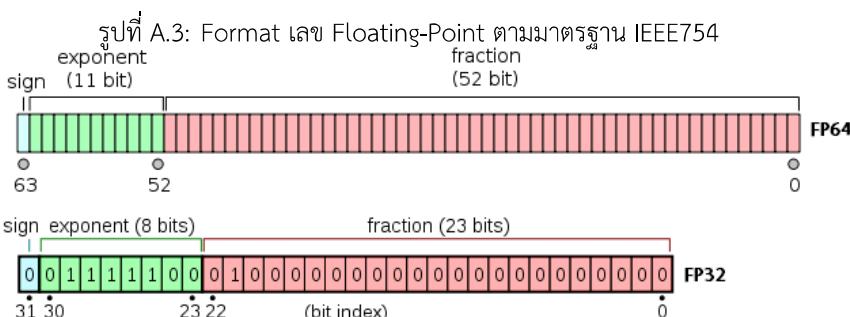
→ single: 8 บิต

→ double: 11 บิต

single: 23 บิต

double: 52 บิต

S	Exponent	Fraction
---	----------	----------



## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

เลขฐานสิบ  $-0.75$

$$-0.75 = (-1)^1 \times 0.11_2 = (-1)^1 \times 1.1_2 \times 2^{-1}$$

Sign = 1

$$\text{Fraction} = 100\ 0000\ 0000\ 0000\ 0000_2$$

Exponent =  $-1 + \text{Bias}$

$$\text{Single (8 บิต): } -1 + 127 = 126 = 0111\ 1110_2$$

$$\text{Double (11 บิต): } -1 + 1023 = 1022 = 011\ 1111\ 1110_2$$

$$\text{Single: } [1][011\ 1111][0100\ 0000\ 0000\ 0000\ 0000] = \text{BF40}\ 0000_{16}$$

$$\text{Double: } [1][011\ 1111\ 1110][1000\ 0000\ 0000\ 0000\ ..\ 0000] = \text{BFE8}\ 0000\ 0000\ 0000_{16}$$

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

ด้วยอย่างที่ 2.6.1. จงหาค่าทศนิยมฐานสิบของเลข FP-32 ที่เติมในรูปที่ 2.7 คือ เลขทศนิยมมาตรฐาน IEEE 754 ชนิด Single Precision ดังต่อไปนี้

0][011 1110 0][010 0000 0000 0000 0000 0000]

Sign      Exp       $F_{10, IEEE} = 1.01 \times 2^{(Exp - Bias)}$

(2.68)

$$= 1.01 \times 2^{-3} \quad (2.69)$$

$$= 0.00101_2 \quad (2.70)$$

$$= 2^{-3} + 2^{-5} \quad (2.71)$$

$$= \frac{1}{2^3} + \frac{1}{2^5} \quad (2.72)$$

$$= 0.125_{10} + 0.03125_{10} = 0.15625_{10} \quad (2.73)$$

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

ตัวอย่างที่ 2.6.2. จงแปลงเลข  $-0.75_{10}$  เป็นเลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE 754

ทั้งสองชนิด

วิธีทำ

$$\begin{aligned}
 -0.75_{10} &= (-1)^1 \times (0.5_{10} + 0.25_{10}) \\
 &= (-1)^1 \times (\frac{1}{2} + \frac{1}{4}) \\
 &= (-1)^1 \times (0.1_2 + 0.01_2) \\
 &= (-1)^1 \times 0.11_2 \times 2^0
 \end{aligned}$$

(1)      (2)

ทำการน้อมล้อเลิร์ ตามสมการที่ 2.59

$$-0.75_{10} = (-1)^1 \times 1.1_2 \times 2^0$$

ดังนั้น บิตเครื่องหมาย  $s = 1$

ค่าทศนิยม  $Y_2 = 100\ 0000\ 0000\ 0000\ 0000\ 0000_2$

ค่ายกกำลัง  $E_2 = -1 + 127 = 126$

โดยชนิด Single ค่ายกกำลัง (8 บิต):  $E_2 = -1 + 127 = 126$  หรือ  $E_2 = 0111\ 1110_2$

โดยชนิด Double ค่ายกกำลัง (11 บิต):  $E_2 = -1 + 1023 = 1022$  หรือ  $E_2 = 011\ 1111\ 1110_2$

ดังนั้น  $-0.75_{10}$  เชียนในรูปของเลขทศนิยมชนิดจุดลอยตัวตามมาตรฐาน IEEE 754 ชนิด

Single: [1][0111 1111 0][100 0000 0000 0000 0000] = BF40 0000<sub>16</sub>

Double: [1][0111 1111 1110][1000 0000 0000 ...0000] = BFE8 0000 0000 0000<sub>16</sub>

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

ตัวอย่างที่ 2.6.3. เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754 ชนิด Single-Precision ต่อไปนี้ มีค่าเท่าไรในฐานสิบ

$$C\ 0\ A\ 0\ 0\ 0\ 0\ 0_{16} = [1100\ 0000\ 1010\ 0000\ 0000\ 0000\ 0000\ 0000_2]$$

วิธีทำ

แปลงจากเลขฐานสิบให้เป็นฐานสองและองค์ประกอบต่างๆ ได้ดังนี้

$$s=[1][E_2=100\ 0000\ 1][Y_2=010\ 0000\ 0000\ 0000\ 0000\ 0000_2]$$

จะพบว่าบิตเครื่องหมาย  $s = 1$

ค่ายกกำลังจริง  $E_{true} = 1000\ 0001_2 - E_{bias} = 129 - 127 = 2$

ค่าทศนิยม  $Y_2 = 010\ 0000\ 0000\ 0000\ 0000_2$

$$F_{10, IEEE} = (-1)^1 \times (1 + .01_2) \times 2^2 \quad (2.74)$$

$$= (-1)^1 \times (1.01_2) \times 2^2 \quad (2.75)$$

$$= (-1)^1 \times (101.0_2) \quad (2.76)$$

$$= (-1) \times 5.0 \quad (2.77)$$

$$= -5.0_{10} \quad (2.78)$$

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

ตัวอย่างที่ 2.6.5. การบวกเลขทศนิยมฐานสองชนิดจุดลอยตัว (Single Precision) มาตรฐาน IEEE754

$$-5_{10} + -0.75_{10} \text{ หรือเท่ากับ } COA0\ 0000_{16} + BF40\ 0000_{16} = ?$$

$$\begin{array}{r} 1100\ 0000\ 1010\ 0000\ 0000\ 0000\ 0000_2 \\ + \\ 1011\ 1111\ 0100\ 0000\ 0000\ 0000\ 0000_2 \end{array}$$

วิธีทำ

ผู้อ่านจะต้องแปลงเลขฐานสองให้เป็นรูปแบบเลขฐานสองที่น้อมัลลีลซ์ ดังต่อไปนี้

$$(-1)^1 \times (1.010\ 0000\ 0000\ 0000\ 0000)_2 \times 2^2 + \text{ } \textcolor{blue}{121-127}$$

$$(-1)^1 \times (1.100\ 0000\ 0000\ 0000\ 0000)_2 \times 2^{-1} - \text{ } \textcolor{blue}{126-127}$$



1. เลื่อนตำแหน่งจุดทศนิยมและปรับเลขยกกำลังที่มีค่าบวกกว่า ให้ตรงกับเลขยกกำลังของเลขที่มีค่ามากกว่า

$$(-1)^1 \times (1.010\ 0000\ 0000\ 0000\ 0000)_2 \times 2^2 +$$

$$(-1)^1 \times (0.001\ 1000\ 0000\ 0000\ 0000)_2 \times 2^2$$

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

2. บวกค่านัยสำคัญที่เลื่อนตำแหน่งแล้ว

$$(-1)^1 \times (1.011\ 1000\ 0000\ 0000\ 0000)_2 \times 2^2$$



3. น้อมัลลีลซ์ค่าผลลัพธ์ และตรวจเช็คการเกิดโอเวอร์ฟอล์ว์และอันเดอร์ฟอล์ว์

$$(-1)^1 \times (1.011\ 1000\ 0000\ 0000\ 0000)_2 \times 2^2 \Rightarrow \text{ไม่เกิดโอเวอร์ฟอล์ว์และอันเดอร์ฟอล์ว์}$$

4. ปัดค่าให้เหลือ 23 บิตและอาจต้องน้อมัลลีลซ์เมื่อจำเป็น

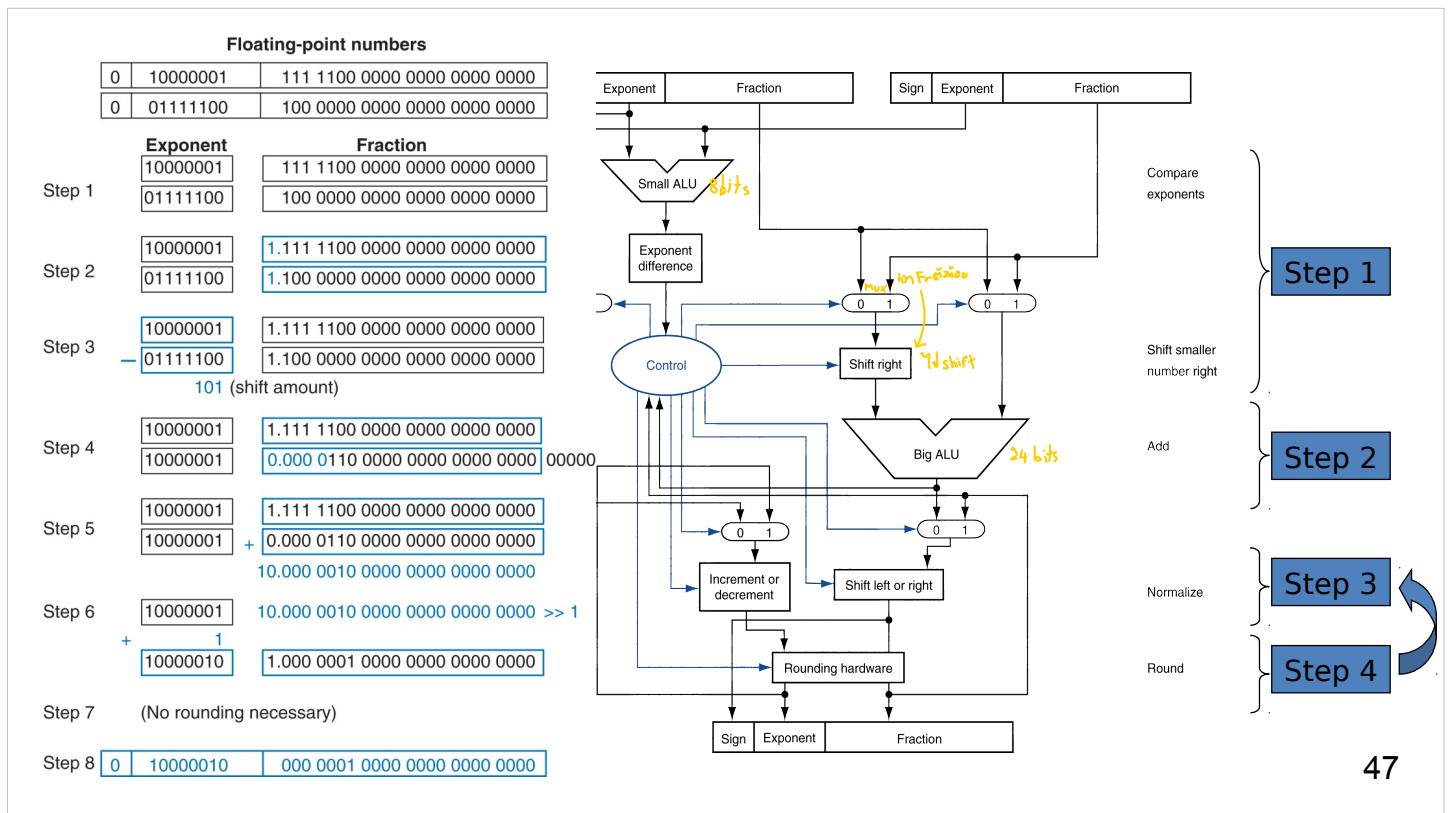
$$(-1)^1 \times (1.011\ 1000\ 0000\ 0000\ 0000)_2 \times 2^2$$

$$s = 1$$

$$\text{ค่ายกกำลัง } E_2 = 2+127 = 129 = 1000\ 0001_2$$

$$\text{ค่าทศนิยม } Y_2 = 011\ 1000\ 0000\ 0000\ 0000_2$$

$$F_{2,IEEE} = 1][100\ 0000\ 1][011\ 1000\ 0000\ 0000\ 0000]_2 \Rightarrow C0B8\ 0000_{16}$$



47

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE 754

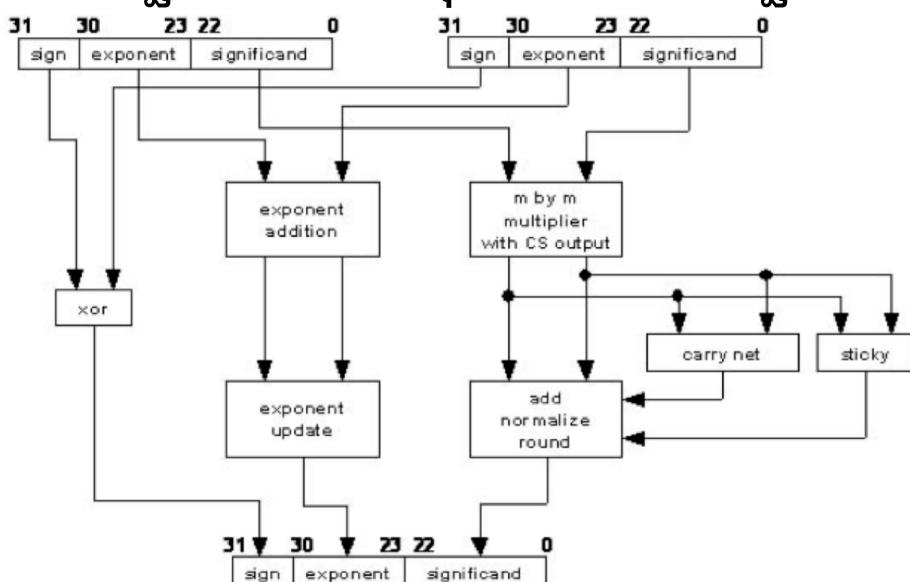


Figure 2.10: วงจรคูณเลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE 754 โดยรับค่าอินพุทจำนวน 2 ตัวด้านบน และเอาท์พุตผลลัพธ์ด้านล่าง (ที่มา: Patterson and Hennessy (2000))

48

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

ตัวอย่างที่ 2.6.7. จงคูณเลขทศนิยมฐานสองชนิดจุดลอยตัวแบบ Single Precision ดังต่อไปนี้

$$\text{EX: } -5_{10} \times -0.75_{10} = \text{COA}0\ 0000_{16} \times \text{BF}40\ 0000_{16} = ?$$

1. แปลงเลขฐานสิบหกให้เป็นเลขฐานสอง

$$1100\ 0000\ 1010\ 0000\ 0000\ 0000\ 0000_2 \times$$

$$1011\ 1111\ 0100\ 0000\ 0000\ 0000\ 0000_2$$

2. แปลงเลขฐานสองตามกฎของ IEEE 754 Single Precision รูปแบบเลขฐานสองที่น้อมลัลไล์ซ์ ดังต่อไปนี้

$$= (-1)^1 \times (1.010\ 0000\ 0000\ 0000\ 0000)_2 \times 2^2 \times$$

$$(-1)^1 \times (1.100\ 0000\ 0000\ 0000\ 0000)_2 \times 2^{-1}$$

3. บวกเลขยกกำลังเข้าด้วยกัน:  $[2^2 \times 2^{-1}] = 2^1$

$$= (-1)^1 \times (-1)^1 \times [(1.010\ 0000\ 0000\ 0000\ 0000)_2 \times$$

$$(1.100\ 0000\ 0000\ 0000\ 0000)_2] \times [2^2 \times 2^{-1}]$$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

49

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

4. คูณค่านัยสำคัญเข้าด้วยกัน

$$1.010\ 0000\ 0000\ 0000\ 0000_2 \times$$

$$_1.100\ 0000\ 0000\ 0000\ 0000_2$$

$$= 1.111\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$$

5. น้อมลัลไล์ซ์ค่าผลลัพธ์ และตรวจสอบการเกิดโอเวอร์โฟล์วและอันเดอร์โฟล์ว

$$1.111\ 0000\ 0000\ 0000\ 0000_2 \times 2^1 \text{ (ค่าเป็นปกติ)}$$

6. ปัดค่าให้เหลือ 23 บิตและอาจต้องน้อมลัลไล์ซ์อีกรอบหากจำเป็น

$$1.111\ 0000\ 0000\ 0000\ 0000_2 \times 2^1 \text{ (ไม่มีการเปลี่ยนแปลงเกิดขึ้น)}$$

7. ปรับเครื่องหมายของผลลัพธ์ให้ถูกต้องจากตัวตั้งและตัวคูณ

$$(-1)^0 \times (1.111\ 0000\ 0000\ 0000\ 0000_2 \times 2^1)$$

ดังนั้น  $-5_{10} \times -0.75_{10} = 3.75_{10}$  และในเครื่องคอมพิวเตอร์จะเห็นเป็นค่า

บิตเครื่องหมาย  $s = 0$

$$\text{COA}0\ 0000_{16} \times \text{BF}40\ 0000_{16} = 4070\ 0000_{16}$$

$$\text{ค่ายกกำลัง } E_2 = 1 + 127 = 128 = 1000\ 0000_2$$

$$\text{ค่าทศนิยม } Y_2 = 111\ 0000\ 0000\ 0000\ 0000_2$$

$$F_{2,IEEE} = 0100\ 0000\ 0111\ 0000\ 0000\ 0000\ 0000_2 \Rightarrow 4070\ 0000_{16}$$

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

50

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

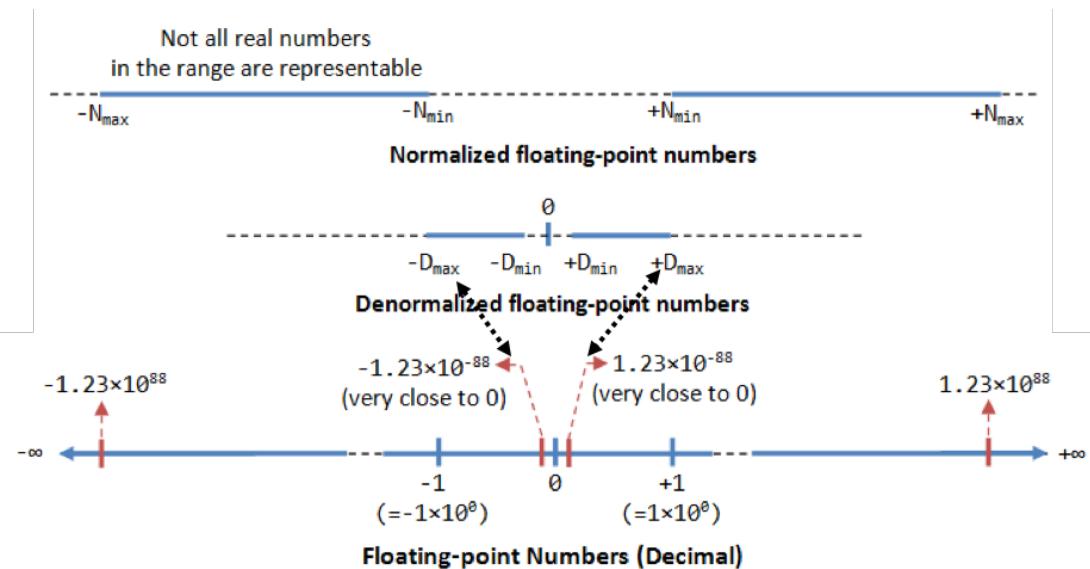
Table 2.11: ตารางสรุปความแตกต่างระหว่างเลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE 754 ชนิด Single Precision โดย  $E_2$  คือ ค่ายกกำลัง และ  $Y_2$  คือ ทศนิยมซึ่งเป็นส่วนประกอบของค่านัยสำคัญ ( $\times$  หมายถึง บิทข้อมูลมีค่าเท่ากับ '0' หรือ '1')

ลำดับที่	เครื่องหมาย $s$ (1 บิต)	ค่ายกกำลัง $E_2$ (8 บิต)	ทศนิยม $Y_2$ (23 บิต)	ความหมาย
1	±	0000 0001 <sub>2</sub> - 1111 1110 <sub>2</sub>	xx..2	เลขฐานสิบห้าไป (น้อยล่าเล็ก)
2	±	0000 0000 <sub>2</sub>	xx..2	เลขฐานสิบห้าอยมาก แต่ไม่เท่ากับศูนย์ (ดินน้อยล่าเล็ก) <span style="color: pink;">เกิด Underflow</span>
3	0	0000 0000 <sub>2</sub>	0...0 <sub>2</sub>	0.0 <sub>10</sub> (ศูนย์จุดศูนย์) <span style="color: green;">True Zero</span>
4	±	1111 1111 <sub>2</sub>	0...0 <sub>2</sub>	±∞ (±อินฟินิตี้) <span style="color: blue;">เกิดที่นี่ได้</span>
5	0	1111 1111 <sub>2</sub>	xx..2	Nan (Not a Number) <span style="color: orange;">?</span>

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

51

## 2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE 754

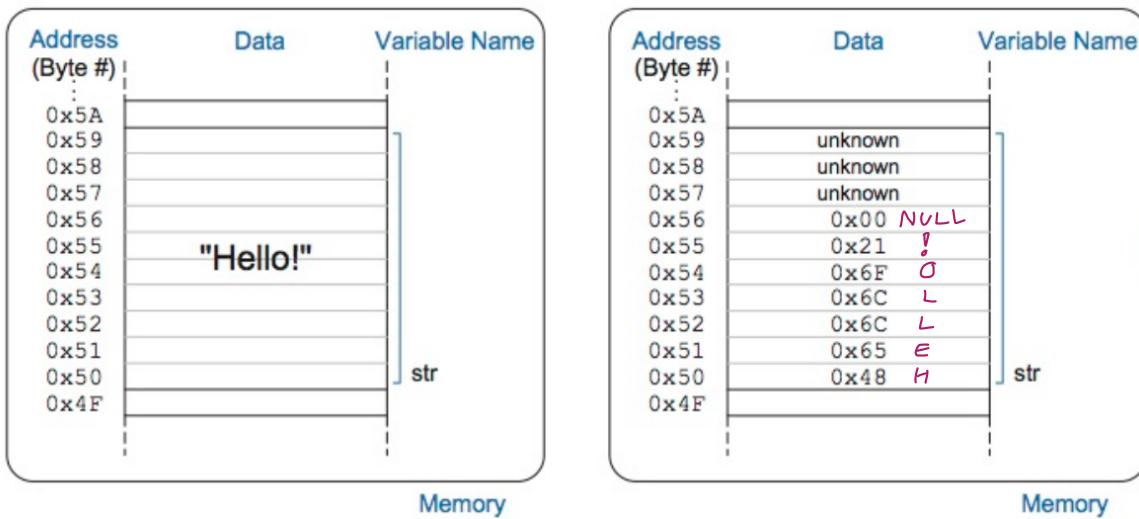


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

52

## 2.7 ตัวอักษร (Character)

char [10] str="Hello!"



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

53

Codepage 874 - Thai

-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	
0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
0020	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	^	-
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n
0070	p	q	r	s	t	u	v	w	x	y	z	{		}	~
0080															
0090															
A-	ก	ຂ	҆	ຄ	່	ແ	ແ	ແ	ແ	ແ	ແ	ແ	ແ	ແ	ແ
B-	໌	ໍ	໏	ດ	ດ	ດ	ດ	ດ	ດ	ນ	ບ	ຜ	ຜ	ພ	ພ
C-	ກ	ມ	ຢ	ຮ	ດ	ກ	ວ	ຕ	ມ	ສ	ໜ	ຝ	ອ	ໜ	ໜ
D-	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ	ດ
E-	ເ	ແ	ໂ	ງ	ໄ	ກ	ງ	ກ	ງ	ກ	ງ	ກ	ງ	ກ	ງ
F-	ອ	ກ	ໂ	ຕ	ແ	ດ	ວ	ດ	ດ	ກ	ດ	ດ	ດ	ດ	ດ

## 2.7 ตัวอักษร (Character)

รหัส Unicode ถูกกำหนดให้เป็นมาตรฐานโดย ISO (International Standard Organization) เพื่อมาตรฐานรหัส ASCII เนื่องจากความต้องการใช้ภาษาที่ไม่ใช่英文เรื่อยๆ รหัส Unicode กำหนดวิธีการเข้ารหัสที่หลากหลายตามวัตถุประสงค์การใช้งาน เช่น รหัส UTF-8 รหัส UTF-16 รหัส UCS-2 เป็นต้น

- รหัส UTF-8 นิยมใช้ในเว็บเพจต่างๆ โดยแต่ละตัวอักษรจะให้ความยาว 1 ไบต์ ดังแต่ 1 ไบต์ โอดตัวอักษร 128 ตัวแรกของรหัส ASCII ใช้ความยาว 1 ไบต์ ส่วนตัวอักษรในภาษาอื่นๆ จะให้จำนวนไบต์เพิ่มขึ้น โดยตำแหน่งเริ่มต้นของตารางรหัส Unicode จะเหมือนกับตารางรหัส ASCII ตัวภาษาอังกฤษและภาษาไทย ในรูปที่ 2.12 ภายใต้แต่ละตัวอักษร มีการรหัส Unicode กำกับอยู่ด้วยยกหัวอย่างเช่น ตัวอักษรไทยเริ่มต้นที่รหัส ASCII เท่ากับ A1 หรือ ก ในรูปของเลขฐานสองขนาด 8 บิต ตรงกับรหัส Unicode 0E01 ความยาว 16 บิต

- รหัส UCS-2 จะใช้พื้นที่ 2 ไบต์ หรือ 16 บิต ต่อ 1 ตัวอักษร ซึ่งทำให้สามารถใช้เลขฐานสองจำนวน 216 หรือ 65,536 แบบมาแทนตัวอักษร ซึ่งทำให้วิธีการนี้ได้รับความนิยม

- รหัส UTF-16 คือ การขยายรหัส UCS-2 ให้ทันสมัยมากขึ้น โดยเพิ่มการเข้ารหัสเป็นขนาด 4 ไบต์

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

54

## สรุปท้ายบท

Table 2.12: ชนิด ความยาว ข้อมูล และการประยุกต์ใช้งานเลขฐานสองชนิดต่างๆ ในคอมพิวเตอร์

ชนิด	บิต	ข้อมูล	การประยุกต์ใช้งาน
char	8	ตัวอักษร	ข้อความ อีเมล
unsigned char	8	จุดภาพ	รูปภาพขาวดำ และ Gray Scale
unsigned char	8		รูปภาพสี RGB Bitmap JPEG
unsigned int	32/64	แอดเดรส	พอยท์เตอร์ซึ่งตำแหน่งข้อมูล ระบบ 32/64 บิต
unsigned int	32/64	จำนวน	จำนวนอุปกรณ์ IoT จำนวนดาวต่างๆ
int	32/64		
ทศนิยม Fixed	16-32	เสียง	ข้อมูลเสียงดนตรี
ทศนิยม Fixed	16-32	จุดภาพ	ข้อมูลภาพความละเอียดสูง
float	32	จุดภาพ	ข้อมูลภาพความละเอียดสูง
float	32	ระยะทาง	เกม 3 มิติ
double	64	± ระยะทาง	ระยะทางไปยังดาวต่างๆ นอกโลก
double	64	± นำ้มนัก	นำ้มนักดาวต่างๆ นำ้มนักอนุภาคเล็กๆ