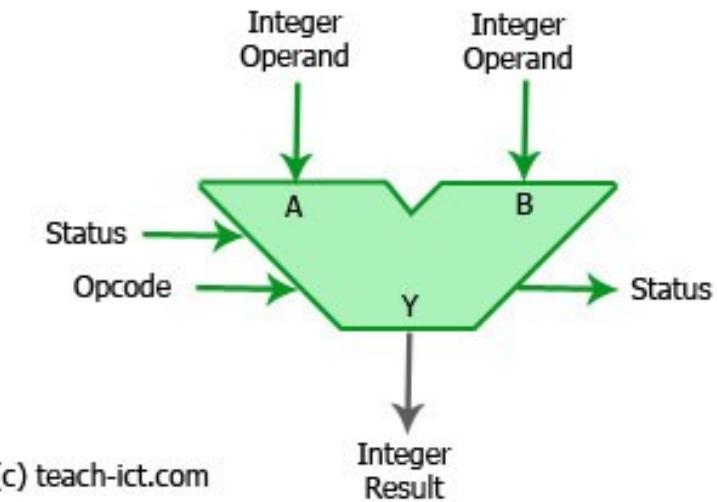


2.3 คณิตศาสตร์เลขจำนวนเต็ม

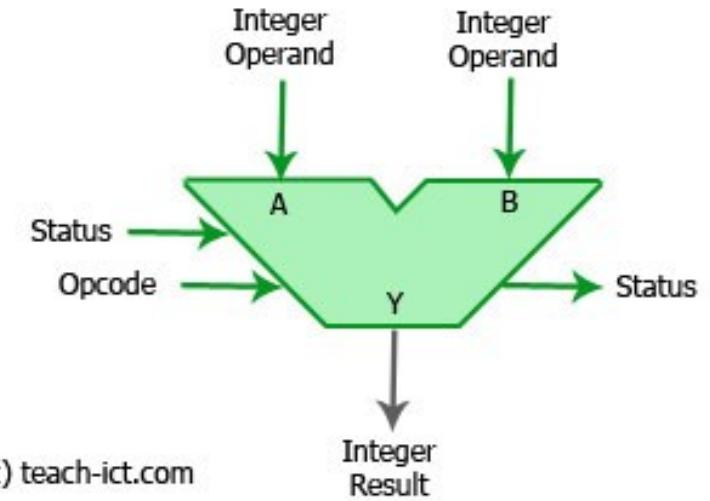
- สัญญาณ Opcode เพื่อสั่งการทำงาน เช่น บวก ลบ เป็นต้น
- ผลลัพธ์ Y เป็นจำนวนเต็มชนิดไม่มีเครื่องหมายขนาด n บิต
- สัญญาณ Status ประกอบด้วย
 - ขาเครื่องหมาย N (Negative) สำหรับเลขจำนวนเต็มชนิดมีเครื่องหมาย
 - ขา Z (Zero)=1 เพื่อบ่งบอกว่าผลลัพธ์ Y มีค่าเท่ากับศูนย์ทุกบิต
 - ขา Carry c_n สำหรับตัวทดบิตที่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



ขาสัญญาณเหล่านี้จะบันทึกลงในรีจิสเตอร์สถานะ (Status Register) สำหรับใช้งานและโปรแกรมเมอร์ตรวจสอบด้วยวงจรดิจิทัลและคำสั่งภาษาแอสเซมบลี ในบทที่ 4

2.3.1 គណិតសាស្ត្រលេខាំងវាទែម បង្កើតមិនមៀត្រីរងមាយ

	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

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 ในวิชาออกแบบวงจรดิจิทัล เราเรียกวงจรบวกเลขชนิดนี้ว่า วงจร Full Adder โดยวงจรจะนำบิท ข้อมูลจำนวน 3 บิทมากระทำการทางตรรกศาสตร์ได้ผลลัพธ์ z_i โดย

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

เมื่อ \oplus คือ กระบวนการ Exclusive-OR และบิทตัวที่ 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 บิท นี้สามารถตรวจจับการเกิดโอเวอร์ฟอลว์ได้โดย

$$V = c_n$$

unsigned

$$(2.43)$$

2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

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

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

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

เกิด Overflow

ผลลัพธ์ เฟลช

ต้องซึ่ง exception

*ลองตรวจสอบว่าเกิด overflow หรือปะกัน ตามที่เขียนไว้ในโปรแกรม หาเหตุ因
สาเหตุที่ไม่เกิด

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		

Hardware คือ

สาเหตุของการเกิด Overflow เนื่องจากผลลัพธ์มีค่าอยู่นอกย่านที่เป็นไปได้ โดยสามารถตรวจสอบอย่างง่ายโดย $c_4 = 1$ (V : Overflow) เมื่อเกิดโอเวอร์เฟล์ ผลลัพธ์ที่ได้จะมีค่า ไม่ถูกต้อง (Invalid)

2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

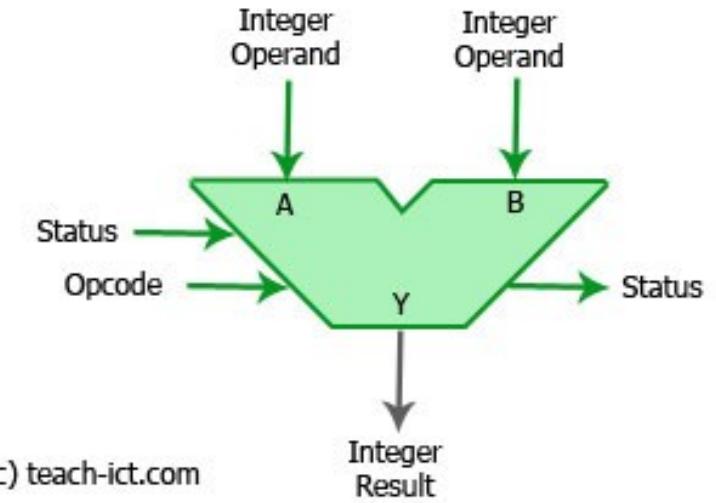
การบวกเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย

การบวกเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย 2 จำนวน ผลลัพธ์ที่ได้จะไม่มีเครื่องหมายด้วยเช่นกัน แต่การบวกเลขขนาดใหญ่ที่เข้าใกล้ค่าสูงสุด สามารถเกิดความผิดพลาดได้ เรียกว่า การเกิดโอเวอร์โฟล์ว (Overflow) ในสมการที่ (2.43) ซึ่งเป็นผลสืบเนื่องจากการจัดจิทัลที่สามารถประมวลผลได้จำกัด ตามจำนวนบิตข้อมูลสูงสุดที่ทำได้ ในตัวอย่างการแปลงเลขฐานสองเป็นฐานสิบที่ได้แสดงไปแล้ว ยกตัวอย่างเช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวัง ตามประโยคในภาษา C/C++ ประโยค $i++$ หรือ $i = i + 1$ นี้ หาก i มีค่าเพิ่มขึ้นเรื่อยๆ จนถึงค่าสูงสุด การบวกเพิ่มอีก 1 ไปเรื่อยๆ โดยไม่มีการตรวจจับการเกิดโอเวอร์โฟล์วล่วงหน้า จะทำให้ค่าของ i กลายเป็นศูนย์ในที่สุด ซึ่งอาจทำให้เกิดผลร้ายตามมาอย่างรุนแรง ผู้อ่านสามารถทดสอบได้ตามกิจกรรมท้ายการทดลองในภาคผนวก E

int

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2's 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

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2's 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

$$z_i = x_i \oplus y_i \oplus c_i$$

ผลลัพธ์จะได้ 3 บิต
(2.45)

หลักการคือ การตรวจสอบ พอดาวน์เพิร์ก์เพิร์ก
ที่สาม bit ซึ่งถ้าเป็น 0 ให้บวก 1 ไปเรื่อยๆ

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

ถ้ามีตัวที่บวกมา ผ่านไปทุกครั้ง

เมื่อ \oplus คือ กระบวนการ Exclusive-OR

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

การเกิดโอเวอร์โฟล์วของการบวกเลขชนิดมีเครื่องหมาย 2-Complement ได้โดย

$$V = c_n \oplus c_{n-1}$$

check overflow (2.47)

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2's Complement

$\text{int}((-8) \oplus 7)$

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

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

↑ 3 บิต สำหรับ ตัวเลขบวกที่อยู่ใน bit ก่อน

Invalid

ผลการคำนวณผลลัพธ์ที่ไม่ถูกต้อง (Invalid) เนื่องจากเกิดโอเวอร์ฟล็อว (Overflow) เพราะหารด้วยร
คำนวณ $V = c_4 \oplus c_3 = 0 \oplus 1 = 1$ ตามสมการที่ (2.47) ทำให้ออฟท์แวร์นำคำตอบนี้ไปใช้ไม่ได้

Signed
Signed
Signed
Signed

+/-

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

ตัวอย่างที่ 2.3.7 จงคำนวณหาค่าของ $-3 - 6$ ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ 2-Complement ขนาด 4 บิต $-3 - 6 = (-3) + (-6)$ ดังนี้ ในเครื่องคอมพิวเตอร์ขนาด 4 บิต ดังนี้

5 bits

อาจรู้

$-9 < 4 \text{ bit}$
Signed

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

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

ผลการคำนวณผลลัพธ์ที่ไม่ถูกต้อง (Invalid) เนื่องจากเกิดโอเวอร์โฟล์ (Overflow) เพราะหารด้วย
คำนวณ $V=c_4 \oplus c_3 = 1 \oplus 0 = 1$ ตามสมการที่ (2.47) ทำให้ซอฟท์แวร์นำคำตอบนี้ไปใช้ไม่ได้

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

การบวก/ลบเลขจำนวนเต็มชนิดมีเครื่องหมาย

การบวก/ลบเลขจำนวนเต็มชนิดมีเครื่องหมาย 2 จำนวน ผลลัพธ์ที่ได้จะมีเครื่องหมายด้วยเช่นกัน แต่การบวก/ลบเลขขนาดใหญ่ที่เข้าใกล้ค่าสูงสุดหรือค่าต่ำสุด สามารถเกิดความผิดพลาดได้ เรียกว่า การเกิดโอเวอร์โฟล์ว (Overflow) ในสมการที่ (2.47) ซึ่งเป็นผลสืบเนื่องจากการจัดจิจิทัลที่สามารถประมวลผลได้จำกัดตามจำนวนบิตข้อมูลสูงสุดที่ทำได้ ยกตัวอย่างเช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวัง ตามประโยชน์ในภาษา C/C++ ประโยชน์ $i++$ หรือ $i = i + 1$ นี้ หาก i มีค่าเพิ่มขึ้นเรื่อยๆ จนถึงค่าสูงสุด การบวกเพิ่มอีก 1 ไปเรื่อยๆ โดยไม่มีการตรวจจับการเกิดโอเวอร์โฟล์วล่วงหน้า จะทำให้ค่าของ i กลับเป็นค่าลบในที่สุด ซึ่งอาจทำให้เกิดผลลัพธ์ตามมาอย่างรุนแรง ผู้อ่านสามารถทดสอบได้ตามกิจกรรมท้ายการทดลองในภาคผนวก E



fixed
point

2.4 เลขทศนิยมฐานสองชนิดจุดตรึง (Fixed Point)¹

นิยามที่ 2.4.1 กำหนดให้ F_2 เป็นเลขทศนิยมฐานสองชนิด Signed Magnitude เขียนอยู่ในรูป

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

นิยามใช้ชนิดขนาด-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 และ ส่วนทศนิยม (Fraction: F_2) ยาว m บิต รวมความยาวทั้งหมด $m+n+1$ บิต

โดย F_{10} คือ ค่าฐานสิบของเลขทศนิยมฐานสองชนิดจุดตรึง F_2 สามารถคำนวณได้จาก

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

หรือ

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

2.4 เลขศนิยมฐานสองชนิดจุดตรีง (Fixed Point)

ตัวอย่างที่ 2.4.1. จงแปลงเลขฐานสิบต่อไปนี้เป็นเลขศนิยมฐานสองชนิดจุดคงที่ $m = n=2$ บิท

Diagram illustrating the binary addition of 0.75_{10} and 3.00_{10} to get 3.75_{10} . The numbers are aligned by their decimal points. The sum is 111.11_2 , which is -3.75_{10} . A yellow cloud highlights the fractional part of the sum.

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

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

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

ผู้อ่านสามารถสามารถเขียนเลขศนิยมฐานสองชนิดจุดคงที่ได้ตามรูปแบบนี้

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

ค่าทศนิยม (Y_2) มีความยาว m บิท เขียนเป็นสัญลักษณ์ได้ดังนี้

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

2.5 เลขทศนิยมฐานสอง ชนิดจุดลอยตัว (Floating Point)

เลขทศนิยมฐานสอง ~~ชนิด~~ ชนิด จุด ลอยตัว หมายความว่า สำหรับข้อมูลที่มี ~~พิสัย~~ (Range) กว้าง และเลขทศนิยมที่ต้องการความละเอียดสูง สำหรับการคำนวณทางวิทยาศาสตร์ (Scientific) ดังนี้

- -2.34×10^{56} ซึ่งเขียนอยู่ในลักษณะที่นอร์มัลไลซ์ (Normalize) แล้ว
- $+0.002 \times 10^{-4}$ ซึ่งจะต้องนอร์มัลไลซ์ต่อไปเป็น $+2.000 \times 10^{-7}$
- $+987.02 \times 10^9$ ซึ่งจะต้องนอร์มัลไลซ์ต่อไปเป็น $+9.8702 \times 10^{11}$

ไม่ว่าจะบวก ลบ อลบัฟฟ์

ไม่ใช่ 1 คงที่ ไป x เว็บไซต์ ก้าว

นิยามที่ 2.5.1 เลขทศนิยม ชนิด จุด ลอยตัว ฐานสอง ที่อยู่ในรูปนอร์มัลไลซ์ ประกอบด้วย 3 ส่วน คือ **บิตเครื่องหมาย** (Sign bit: s) **ค่านัยสำคัญ** (Significand: S_2) ในสมการที่ (2.61) และค่ายกกำลัง (E_2) มีลักษณะดังนี้

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

เลขยกกำลังเป็นเลขจำนวนเต็มฐานสองชนิด sign-magnitude ความยาว n บิต ดังนี้

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

2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว

เมื่อ e_i แต่ละบิตมีค่า “1” หรือ “0” ในตำแหน่งที่ i s คือ Sign bit และ n คือ จำนวนบิตซึ่งกำหนดไว้ก่อนจะออกแบบวงจร

จากนิยามที่ 2.59 ค่านัยสำคัญ (Significand) S_2 ความยาว $m+1$ บิต สามารถเขียนใหม่ได้ ดังนี้

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

ซึ่งมีความสำคัญต่อรูปแบบการเขียน เนื่องจากจะจะต้องทำการนормัลไลซ์ผลลัพธ์ที่ได้จากการคำนวณเสมอ ค่านัยสำคัญที่นิยามไว้ในรูปแบบนี้สามารถคำนวณหาค่าฐานสิบ ได้ดังนี้

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

2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว

ตัวอย่างที่ 2.5.1 จงแปลงเลขทศนิยมฐานสองชนิดจุดลอยตัวที่นอร์มัลไลซ์แล้ว $(-1)^1 \times 1.0101_2 \times 2^3$ ให้เป็นเลขทศนิยมฐานสิบตามสมการที่ (2.62)

วิธีทำ

1. ปรับจุดทศนิยม เพื่อให้เป็นเลขฐานสองชนิด Sign-Magnitude และเลขยกกำลังเท่ากับ 0
 $-1010.1_2 \times 2^0$
2. แปลงค่าเลขฐานสองชนิดที่เลื่อนตำแหน่งแล้วให้เป็นฐานสิบ
 $-(1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}) = -(8 + 0 + 2 + 0 + 0.5) = -10.5$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

สูตร โฉลก เลขไม่เป็นป

มาตรฐานของเลขทศนิยมฐานสองชนิดจุดลอยตัวได้ถูกกำหนดโดย IEEE (Institute of Electrical and Electronics Engineers) เรียกว่า มาตรฐาน IEEE754 ในปี ค.ศ.1985 เพื่อให้โปรแกรมคอมพิวเตอร์สามารถคำนวณค่าเลขทศนิยมฐานสองบนเครื่องที่ใช้ชิปปี้เดาๆ ได้แล้วให้ผลลัพธ์ตรงกัน ปัจจุบันนี้มาตรฐานได้รับการยอมรับอย่างแพร่หลายและปรับปรุงอย่างต่อเนื่อง สามารถรองรับเลขทศนิยมจุดลอยตัว 2 รูปแบบหลัก คือ

IEEE754

- 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 IEEE754 คือ ปี ค.ศ. 2019 รายละเอียดเพิ่มเติม

ตัวอย่างการประกาศและตั้งค่าตัวแปรที่ใช้มาตรฐานนี้

float a = -5; /* a = 0xC0A00000 */ ร่างรีบ กดลงที่นี่ ช่วยด้วย
double b = -0.75; /* b = 0xBFE8000000000000 */ 64 บิต

รีบ

2.6 เลขคณิตมฐานส่องชนิดจุดลอยตัวมาตรฐาน IEEE754

นิยามที่ 2.6.1 เลขคณิตมฐานส่องชนิดจุดลอยตัวที่น้อมลัลซ์แล้ว สามารถเขียนอยู่ในรูปของเลขฐานสองต่อไปนี้

$$F_{2,IEEE} = [s][E_{2,IEEE}][Y_2] \quad (2.63)$$

↑ Sign ↑ Ex ↓ Fraction

โดยค่าทศนิยม (Fraction): Y_2 ตามสมการที่ (2.64) มีความยาว $m=23$ และ 51 บิต ตามชนิด Single Precision และ Double Precision ตามลำดับ สามารถเขียนเป็นลัญลักษณ์คล้ายกับเลขคณิตมฐานส่องชนิดจุดตรึง ดังนี้

$$Y_2 = y_{-1}y_{-2}y_{-3}\dots y_{-m} \quad (2.64)$$

ค่ายกกำลัง เป็นเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย ความยาว $n=8$ และ 11 บิต ขึ้นกับชนิด Single Precision และ Double Precision ตามลำดับ โดยมีลักษณะคล้ายกับเลขคณิตมฐานส่องชนิดจุดลอยตัวในสมการที่ 2.60 แต่ต่างกันที่เลขยกกำลังของ IEEE754 มีค่าเป็นเลขจำนวนเต็มไม่มีเครื่องหมาย

$$E_{2,IEEE} = [e_{n-1}e_{n-2}\dots e_0] \quad (2.65)$$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

เลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE754 ในสมการที่ 2.63 สามารถแปลงเป็นค่าเลขทศนิยมฐานสิบ ได้ดังนี้

Normalize และ Mag

แปลงจากฐาน 2 เป็นฐาน 10 ในส่วนของเลข

$$F_{10,IEEE} = (-1)^s \times [1 + \frac{y_1}{2} + \frac{y_2}{4} + \frac{y_3}{8} + \dots + \frac{y_m}{2^m}] \times 2^{(E_{2,IEEE} - E_{bias})} \quad (2.67)$$

ดังนั้น เลขยกกำลังจริงจึงสามารถคำนวณได้โดย

$$E_{2,IEEE} = E_2 + E_{bias} \quad (2.66)$$

- ชนิด Single Precision ค่า $E_{bias} = 0111111_2 = 127_{10}$

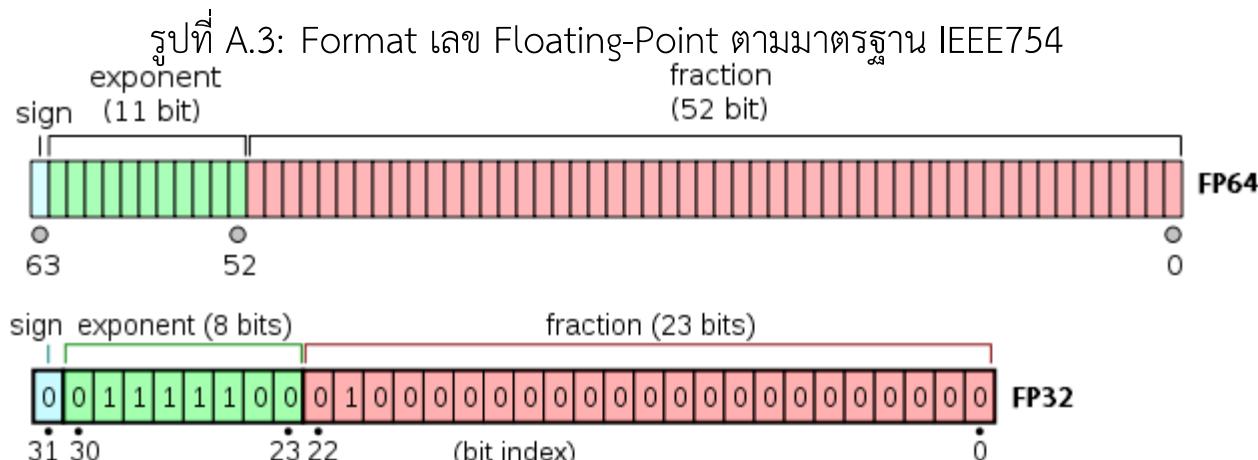
- ชนิด Double Precision ค่า $E_{bias} = 0111111111_2 = 1023_{10}$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

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

single: 8 บิต
double: 11 บิต

single: 23 บิต
double: 52 บิต



$$S_0 \{ F_{2, IEEE} = [S] [E_{2, IEEE}] [Y] \}$$

$$\begin{aligned}
 -122 &\longrightarrow X_{2,u} = [\overline{X_{10,r_2}}] + 1_2 \\
 &\simeq [\overline{0111} \quad \overline{1010}] + 1 \\
 &\simeq 1000 \quad 0101 + 1 \\
 &\simeq 1000 \quad 0110
 \end{aligned}$$

$$-0.625 = 0.5 + 0.125$$

$$2(-1)^1 + \frac{1}{2} + \frac{1}{8} = (-1)^1 + 0.1 + 0.001$$

$$\simeq [-1]^1 + 0.101_2 \times 2^0$$

Normalize

$$-0.625 = (-1)^1 + 1.01_2 \times \bar{1}^1 \quad \boxed{\text{0110100101010101}}$$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

ตัวอย่างที่ 2.6.1 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754 ชนิด Single Precision ต่อไปนี้ มีค่าเท่าไรในฐานลิบตามสมการที่ (2.67)

$$422.8000_0_{16} = [0|100\ 0010\ 0010\ 1000\ 0000\ 0000\ 0000\ 0000]_2$$

วิธีทำ

1. แปลงจากเลขฐานลิบหากให้เป็นฐานสองและองค์ประกอบต่างๆ ได้ดังนี้

$$s=[0][E_{2,IEEE}=100\ 0010\ 0][Y_2=010\ 1000\ 0000\ 0000\ 0000\ 0000]_2$$

2. จะพบว่าบิตเครื่องหมาย $s = 0$

$$\text{ค่ายกกำลังจริง } E_{true} = 1000\ 0100_2 - E_{bias} = 132 - 127 = 5$$

$$\text{ค่าทศนิยม } Y_2 = 010\ 1000\ 0000\ 0000\ 0000\ 0000_2$$

๒๖

เต็มทั้งหมด

คงเหลือ

$$F_{10,IEEE} = (-1)^0 \times (1 + .0101_2) \times 2^5 \quad (2.68)$$

$$= (-1)^0 \times (1.0101_2) \times 2^5 \quad (2.69)$$

$$= (-1)^0 \times (101010.0_2) \quad (2.70)$$

$$= (+1) \times (32 + 8 + 2) \quad (2.71)$$

$$= 42.0_{10} \quad (2.72)$$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

ตัวอย่างที่ 2.6.4 จงแปลงเลข -0.75_{10} เป็นเลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE754 ทั้งสองชนิด

วิธีทำ

sign

1. แปลงเลขทศนิยมฐานสิบให้อยู่ในรูปฐานสองแบบอร์มัลไลซ์

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

2. ทำการนอร์มัลไลซ์ ตามสมการที่ (2.59)

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

ดังนั้น บิตเครื่องหมาย $s = 1$

shift Right

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

$$\text{ค่ายกกำลัง } E_{2,IEEE} = -1 + E_{bias} = 126_{10}$$

2.6 เลขคณิตมฐานสองชnidจุดลอยตัวมาตรฐาน IEEE754

ดังนั้น บิตเครื่องหมาย $s = 1$

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

ค่ายกกำลัง $E_{2,IEEE} = -1 + E_{bias}$

โดยชนิด Single ค่ายกกำลัง (8 บิต): $E_{2,IEEE} = -1 + 127 = 126$ หรือ $E_{2,IEEE} = \boxed{0111\ 1110_2}$

โดยชนิด Double ค่ายกกำลัง (11 บิต): $E_{2,IEEE} = -1 + 1023 = 1022$ หรือ $E_{2,IEEE} = 011\ 1111\ 1110_2$

ทริก $\rightarrow s = 0111\ 111_2 = 127_{10}$

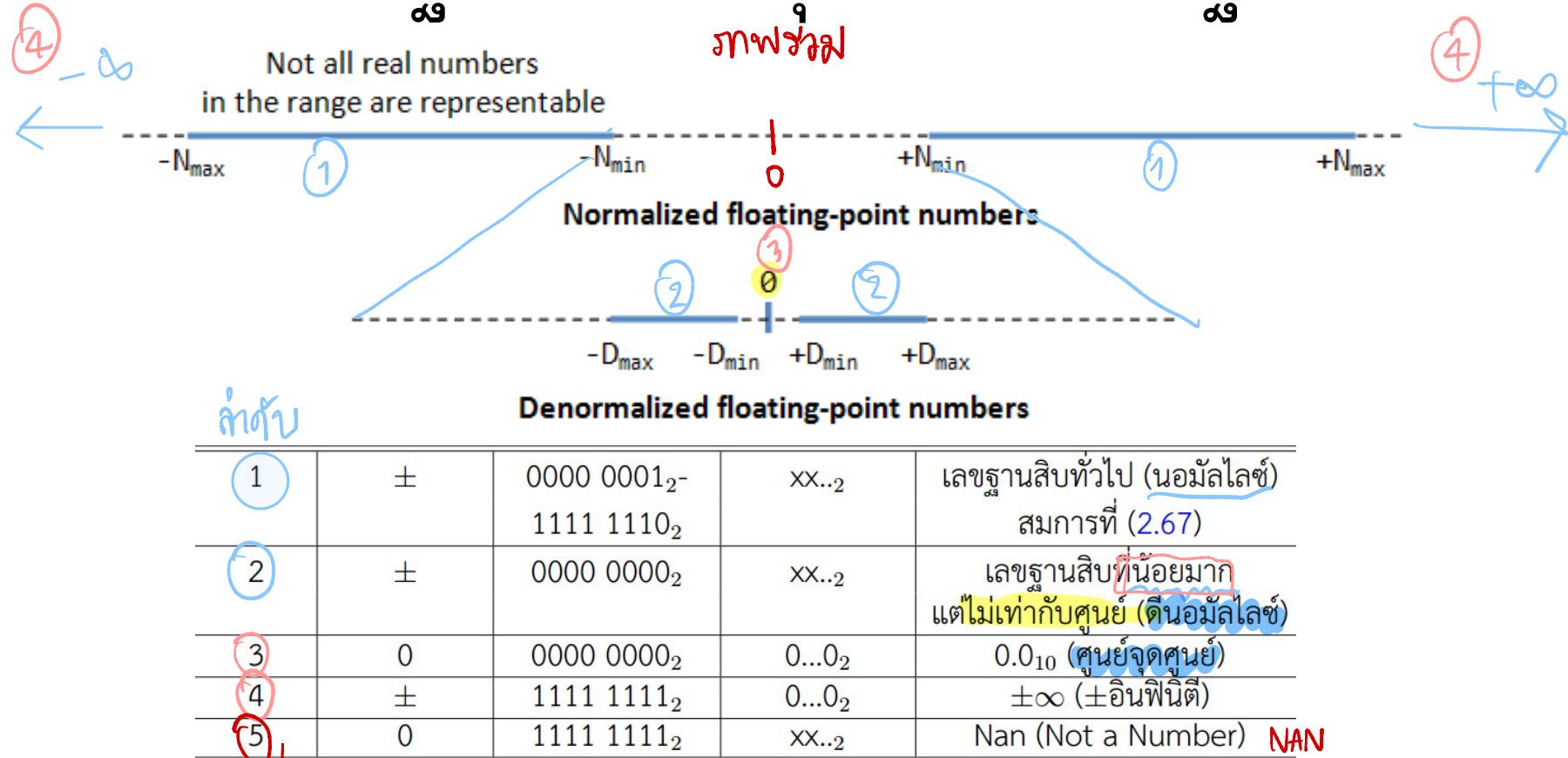
3. ดังนั้น -0.75_{10} เขียนในรูปของเลขคณิตมฐานสองชnidจุดลอยตัวตามมาตรฐาน IEEE754 ชนิด

Single: [1][011 1111 0] 100 0000 0000 0000 0000 0000₂ = BF40 0000₁₆

Single 32 บิต

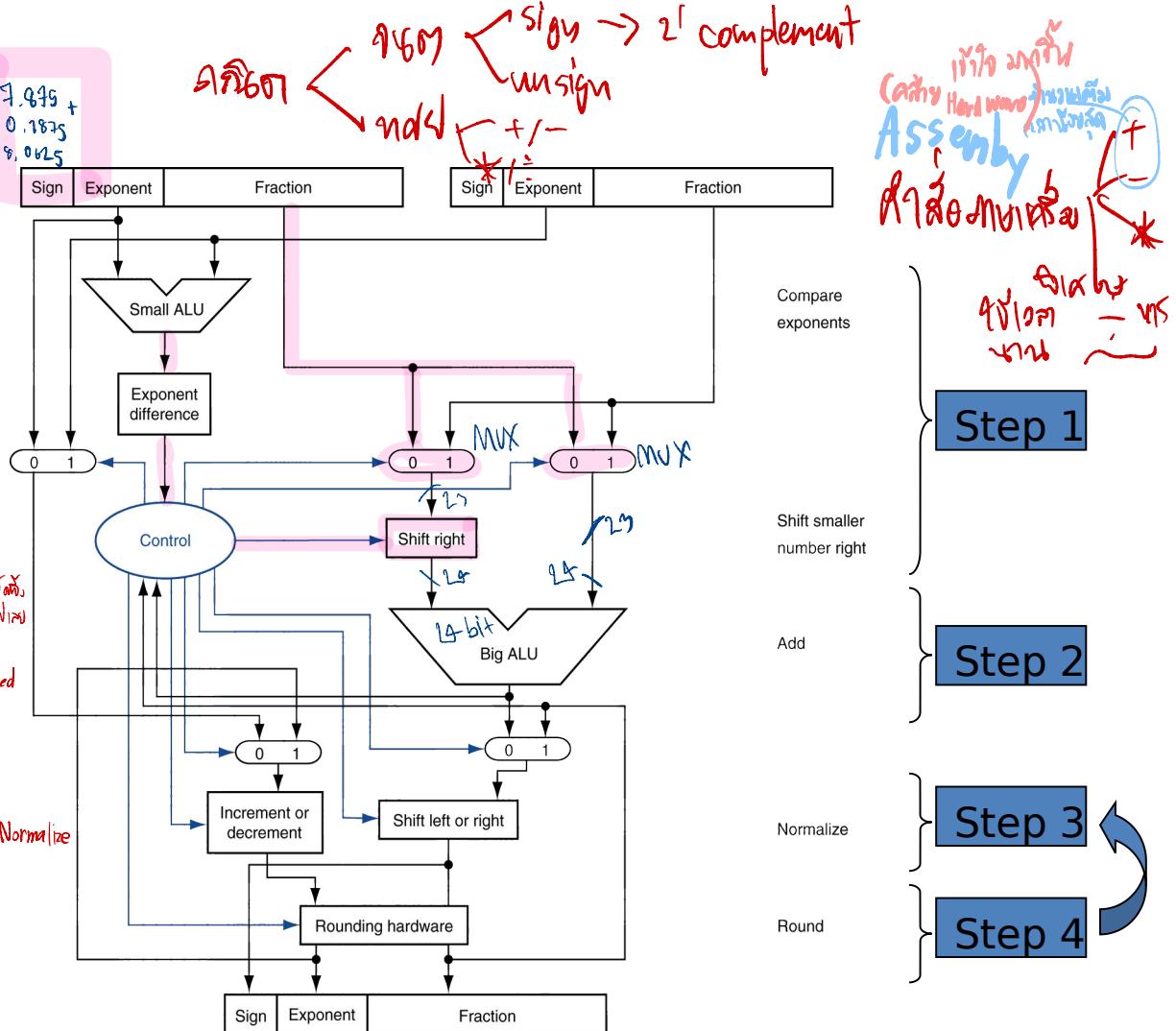
Double: [1][011 1111 1110] [1000 0000 0000 ...0000]₂ = BFE8 0000 0000 0000₁₆ Double + 4 บิต

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

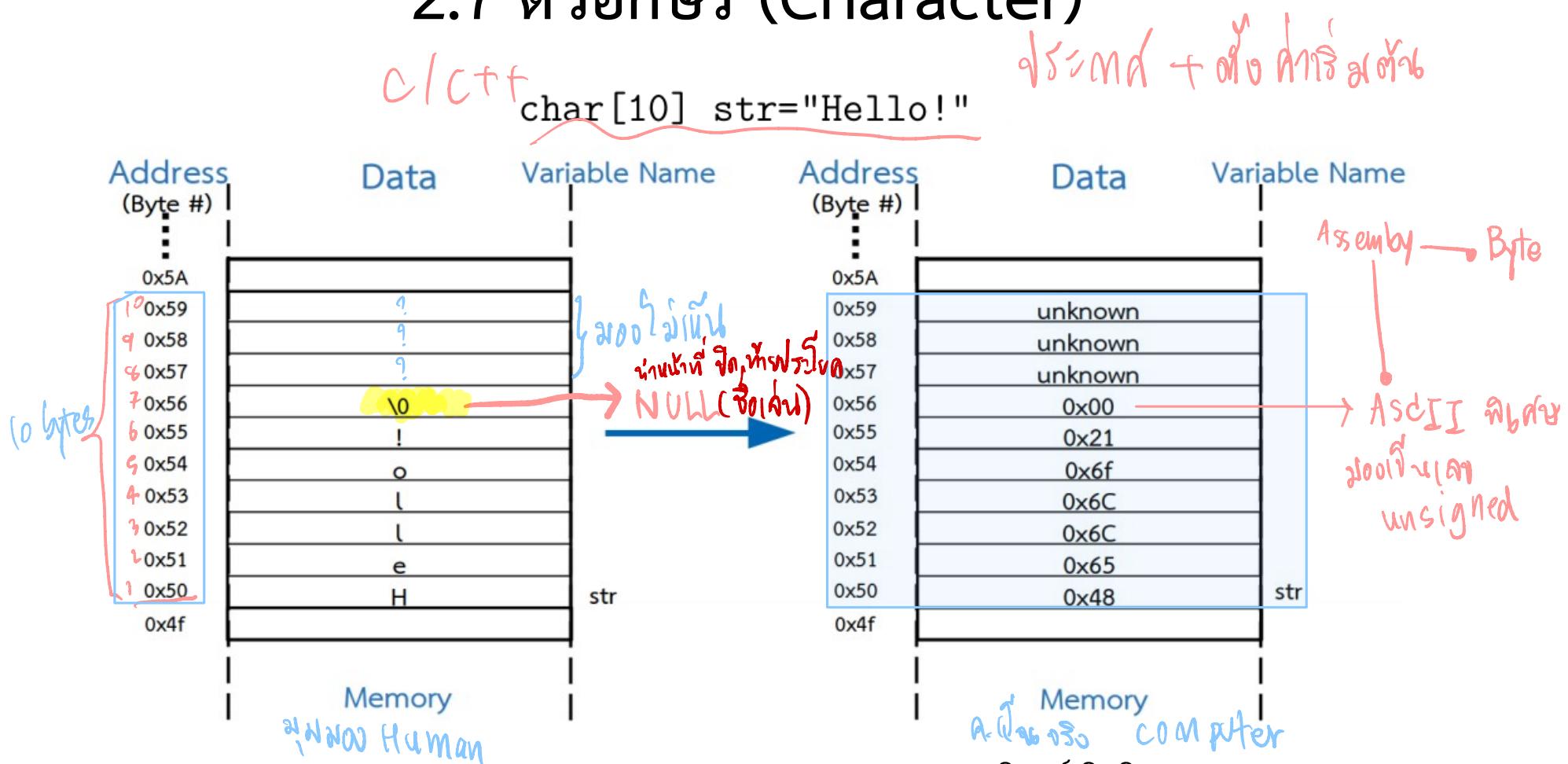


IEEE Floating-point numbers Single

	$0 \quad 10000001_{127}$	$111\ 1100\ 0000\ 0000\ 0000\ 0000$	$2^3 +$
	$0 \quad 01111100_{124}$	$100\ 0000\ 0000\ 0000\ 0000\ 0000$	2^3
Exponent			
Step 1	10000001	$111\ 1100\ 0000\ 0000\ 0000\ 0000$	
	01111100	$100\ 0000\ 0000\ 0000\ 0000\ 0000$	
Step 2	10000001	$1.111\ 1100\ 0000\ 0000\ 0000\ 0000$	24 บิต
	01111100	$1.100\ 0000\ 0000\ 0000\ 0000\ 0000$	24 บิต
Step 3	10000001	$1.111\ 1100\ 0000\ 0000\ 0000\ 0000$	Intenger ALU unsigned
	01111100	$1.100\ 0000\ 0000\ 0000\ 0000\ 0000$	
	$5_{10} = 101_2$	(shift amount) เก็บตัวเลขอ้างอิง	
Step 4	10000001	$1.111\ 1100\ 0000\ 0000\ 0000\ 0000$	
	10000001	$0.000\ 0110\ 0000\ 0000\ 0000\ 0000$	00000 บิต
Step 5	10000001	$1.111\ 1100\ 0000\ 0000\ 0000\ 0000$	↑ 1 bit shift left 5 บิต
	10000001	$+ 0.000\ 0110\ 0000\ 0000\ 0000\ 0000$	unsigned
Step 6	10000001	$10.000\ 0010\ 0000\ 0000\ 0000\ 0000$	overflow บิตที่ 9 ตัวที่ 10
	10000010	$>> 1$	บิตที่ 9 = Normalize
Step 7	$(No rounding necessary)$	$1.000\ 0001\ 0000\ 0000\ 0000\ 0000$	บิตที่ 9
Step 8	$0 \quad 10000010$	$1000\ 0001\ 0000\ 0000\ 0000\ 0000$	23 บิต



2.7 ตัวอักษร (Character)



-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	
0-	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	
1-	៩១ ₁₆ 0010	០១១ 0011	០២១ 0012	០៣១ 0013	០៤១ 0014	០៥១ 0015	០៦១ 0016	០៧១ 0017	០៨១ 0018	០៩១ 0019	០១៨ 001A	០១៩ 001B	០១១៣ 001C	០១១៤ 001D	០១១៥ 001E	០១១៥ 001F
2-	៩២ ₁₆ 0020	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3-	០០៣០	១	២	៣	៤	៥	៦	៧	៨	៩	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[\	^	_	
6-	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	ៗ	
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8-																
9-																
A-	៩	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ
B-	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ
C-	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ
D-	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ
E-	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ
F-	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ	ឃ

2.7 ត៊វអ៊កម្រ (Character)

ប្រចាំប្រចុង unicode

សំណើរដ្ឋមន្ត្រី ពេលវេលា

- រាជស ASCII គឺ មាត្រូនាមទំនាក់ទំនង ដែលបានការពារឡើងដោយ ANSI (American National Standard Institute) ដើម្បីផ្តល់ជូនសារអកម្រព័ត៌មាន។
- តារាងរាជសអេសកី (ASCII) កំណត់លេខចុងសង្គមទាំង 8 បិត ដើម្បីផ្តល់អកម្រតាមរយៈលក្ខណៈ ដែលមានចំនួន $2^8 = 256$ ព័ត៌មាន ដើម្បីរាជសទីផ្សារ។
- ឯកសាររាជសអេសកី ត្រូវបានការពារឡើងដោយកូដលើកូដដើម្បីរាជសទីផ្សារ។
- ឯកសាររាជសអេសកី ត្រូវបានការពារឡើងដោយកូដលើកូដដើម្បីរាជសទីផ្សារ។

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

2.7 ตัวอักษร (Character)

- รหัส Unicode ถูกกำหนดให้เป็นมาตรฐานโดย ISO (International Standard Organization) เพื่อมาทดแทนรหัส ASCII เนื่องจากความต้องการใช้ภาษาทั่วโลกที่เพิ่มขึ้นเรื่อยๆ
- รหัส UCS-2 จะใช้พื้นที่ 2 ไบต์ หรือ 16 บิต ต่อ 1 ตัวอักษร ซึ่งทำให้สามารถใช้เลขฐานสองจำนวน 2^{16} หรือ 65,536 แบบมาแทนตัวอักษร ตำแหน่งเริ่มต้นของตารางรหัส Unicode จะเหมือนกับตารางรหัส ASCII ตัวภาษาอังกฤษและภาษาไทย ภายใต้แต่ละตัวอักษร มีค่ารหัส Unicode กำกับอยู่ด้วย ยกตัวอย่างเช่น ตัวอักษรไทยเริ่มต้นที่ รหัส ASCII เท่ากับ A1 คือ ก ในรูปของเลขฐานสองขนาด 8 บิต ตรงกับรหัส 0E01 ความยาว 16 บิต
- UTF-8 นิยมใช้ในเว็บเพจต่างๆ โดยแต่ละตัวอักษรจะใช้ความยาว ตั้งแต่ 1 ไบต์ จนถึง 4 ไบต์ โดยตัวอักษร 128 ตัวแรกคือรหัส ASCII ใช้ความยาว 1 ไบต์ ส่วนตัวอักษรในภาษาอื่นๆ จะใช้จำนวนไบต์เพิ่มขึ้น
- รหัส UTF-16 คือ การขยายรหัส UCS-2 ให้ทันสมัยมากขึ้น โดยเพิ่มการเข้ารหัสเป็นขนาด 4 ไบต์

2.8 สtruปทัยบท

ตารางที่ 2.13: ชนิด ความยาว ข้อมูล และการประยุกต์ใช้งานเลขฐานสองชนิดต่างๆ ในคอมพิวเตอร์

ชนิด	บิต	ข้อมูล	การประยุกต์ใช้งาน
char	8	ตัวอักษร	ข้อความ อีเมล ชื่อ นามสกุล
unsigned char	8	จุดภาพ	รูปภาพขาวดำ และ Gray Scale
unsigned char	8	จุดภาพ	รูปภาพสี RGB Bitmap JPEG
unsigned int	32/64	พอยน์เตอร์	แอดเดรสซ์ตำแหน่งข้อมูล ระบบ 32/64 บิต
unsigned int	32/64	จำนวน	จำนวนอุปกรณ์ IoT จำนวนดาวต่างๆ
int	32/64	จำนวน	เลขจำนวนเต็ม
ทศนิยมจุดตรึง	16-32	เสียง	ข้อมูลเสียงดนตรี
ทศนิยมจุดตรึง	16-32	จุดภาพ	ข้อมูลภาพความละเอียดสูง
float	32	จุดภาพ	ข้อมูลภาพความละเอียดสูง
float	32	ระยะทาง	เกม 3 มิติ
double	64	± ระยะทาง	ระยะทางไปยังดาวต่างๆ นอกโลก
double	64	± น้ำหนัก	น้ำหนักดาวต่างๆ น้ำหนักอนุภาคเล็กๆ

References

- https://www.researchgate.net/figure/Block-Diagram-of-Micro-SD-card_fig6_306236972
- <https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/in-memory-layout/>
- <https://freedompenguin.com/articles/how-to/learning-the-linux-file-system>
- <https://www.techpowerup.com/174709/arm-launches-cortex-a50-series-the-worlds-most-energy-efficient-64-bit-processors>
- https://www.researchgate.net/figure/NVIDIA-Tegra-2-mobile-processor-11_fig1_221634532
- Harris, D. and S. Harris (2013). Digital Design and Computer Architecture (1st ed.). USA: Morgan Kauffman Publishing.
- <https://learn.adafruit.com/resizing-raspberry-pi-boot-partition/edit-partitions>

References

- https://en.wikipedia.org/wiki/Human-computer_interaction
- <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/programmer-s-guide-for-armv8-a>
- https://xdevs.com/article/rpi3_oc/
- https://www.gsmarena.com/a_look_inside_the_new_proprietary_apple_a6_chipset-news-4859.php
- https://www.slideshare.net/kleinerperkins/2012-kpcb-internet-trends-yearend-update/25-Global_Smartphone_Tablet_Shipments_Exceeded
- <https://www.aliexpress.com/item/32329091078.html>
- <https://www.raspberrypi.org/forums/viewtopic.php?t=63750>
- <https://www.youtube.com/watch?v=2ciyXehUK-U>