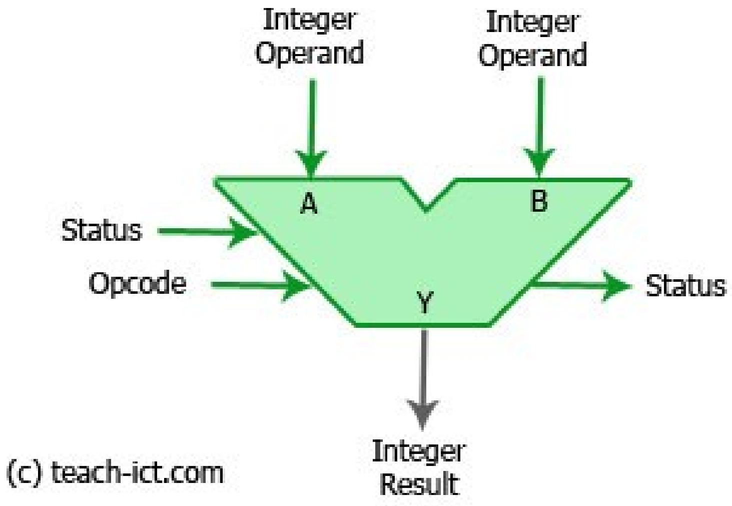
2.3 คณิตศาสตร์เลขจำนวนเต็ม

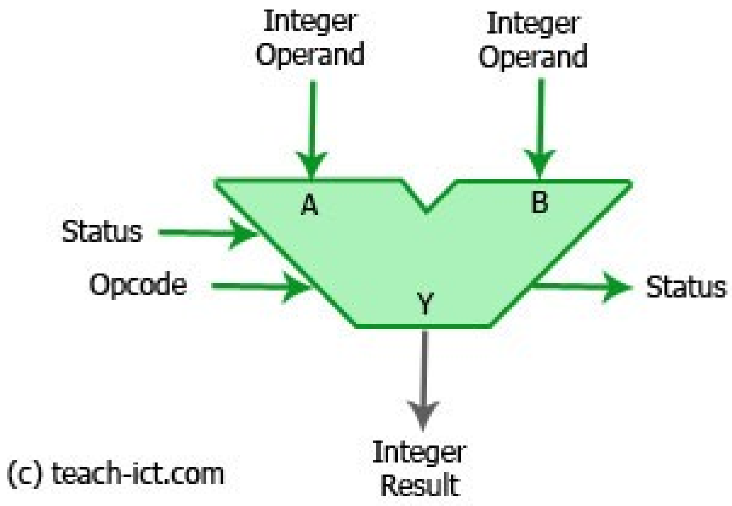
- สัญญาณ Opcode เพื่อสั่งการทำงาน เช่น บวก ลบ เป็นต้น
- ผลลัพธ์ Y เป็นจำนวนเต็มชนิดไม่มีเครื่องหมายขนาด n บิท
- สัญญาณ Status ประกอบด้วย
  - ขาเครื่องหมาย N (Negative) สำหรับเลขจำนวนเต็มชนิดมีเครื่องหมาย
  - ขา Z (Zero)=1 เพื่อบ่งบอกว่าผลลัพธ์ Y มีค่าเท่ากับศูนย์ทุกบิท
  - ขา Carry c_n สำหรับตัวทดบิทที่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

  ขาสัญญาณเหล่านี้จะบันทึกลงในรีจิสเตอร์สถานะ (Status Register) สำหรับให้วงจรและโปรแกรมเมอร์ตรวจสอบด้วยวงจรดิจิทัลและคำสั่งภาษาแอสเซมบลี ในบทที่ 4



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	



2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

$$c_{i+1}z_i = x_i + y_i + c_i \tag{2.40}$$

เมื่อ i=0, 1, 2, .. , $n-1$ โดย c_0 = 0 และสัญลักษณ์ + คือการบวกเลข ไม่ใช่การ OR กันเชิงตรรกศาสตร์

ในวิชาออกแบบวงจรดิจิทัล เราเรียกวงจรบวกเลขชนิดนี้ว่า **วงจร FUll Adder** โดยวงจรจะนำบิทข้อมูลจำนวน 3 บิทมากระทำการทางตรรกศาสตร์ได้ผลลัพธ์ z_i โดย

$$z_i = x_i \oplus y_i \oplus c_i \tag{2.41}$$

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

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

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

$$V = c_n \tag{2.43}$$

2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.1 *จงคำนวณหาค่าของ 5 + 9 ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ Unsigned ขนาด 4 บิท 5 + 9 = 14 ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิท สามารถคำนวณได้ดังนี้*

การบวกเลขขนาด 4 บิทแบบไม่มีเครื่องหมาย: 5 + 9 = 14 พร้อมตัวทด และผลลัพธ์ถูกต้องเนื่องจากไม่เกิดโอเวอร์โฟลว์ (V=c_n=0)

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

2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

ตัวอย่างที่ 2.3.2. จงคำนวณหาค่าของ 7 + 9 ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ Unsigned ขนาด 4 บิท 7 + 9 = 16 = 0 ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิท 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 เนื่องจากผลลัพธ์มีค่าอยู่นอกย่านที่เป็นไปได้ โดยสามารถตรวจสอบอย่างง่ายดายโดย c_4 = 1 (V: Overflow) เมื่อเกิดโอเวอร์โฟลว์ ผลลัพธ์ที่ได้จึงมีค่าไม่ถูกต้อง (Invalid)

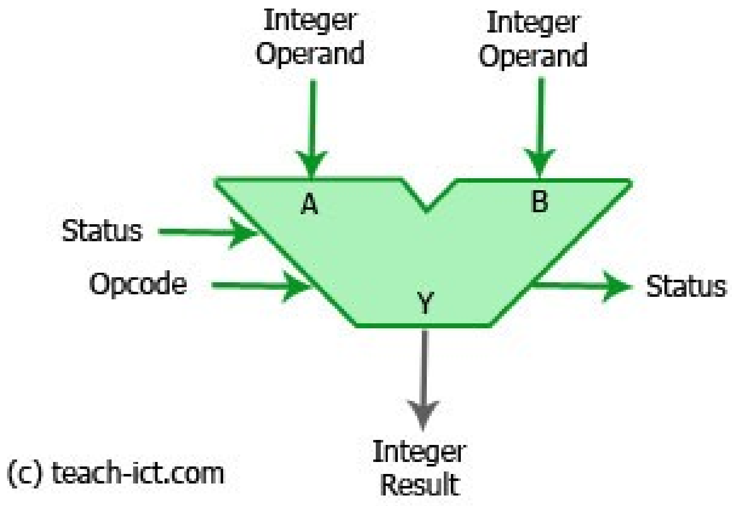
2.3.1 คณิตศาสตร์เลขจำนวนเต็ม ชนิดไม่มีเครื่องหมาย

การบวกเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย

การบวกเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย 2 จำนวน ผลลัพธ์ที่ได้จะไม่มีเครื่องหมายด้วยเช่นกัน แต่การบวกเลขขนาดใหญ่ที่เข้าใกล้ค่าสูงสุด สามารถเกิดความผิดพลาดได้ เรียกว่า **การเกิดโอเวอร์โฟลว์** (Overflow) ในสมการที่ (2.43) ซึ่งเป็นผลสืบเนื่องจากวงจรดิจิทัลที่สามารถประมวลผลได้จำกัด ตามจำนวนบิทข้อมูลสูงสุดที่ทำได้ ในตัวอย่างการแปลงเลขฐานสองเป็นฐานสิบที่ได้แสดงไปแล้ว ยกตัวอย่างเช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวัง ตามประโยคในภาษา C/C++ ประโยค i++ หรือ i=i+1 นี้ หาก i มีค่าเพิ่มขึ้นเรื่อยๆ จนถึง**ค่าสูงสุด** การบวกเพิ่มอีก 1 ไปเรื่อยๆ โดยไม่มีการตรวจจับการเกิดโอเวอร์โฟลว์ล่วงหน้า จะทำให้ค่าของ i กลายเป็น**ศูนย์**ในที่สุด ซึ่งอาจทำให้เกิดผลร้ายตามมาอย่างรุนแรง ผู้อ่านสามารถทดสอบได้ตามกิจกรรมท้ายการทดลองในภาคผนวก E

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	



2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2’s Complement

$$c_{i+1}z_i = x_i + y_i + c_i \tag{2.44}$$

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

ผลลัพธ์ของการบวกเลขจำนวน 3 บิท สามารถคำนวณได้จากวงจร Full Adder ดังนี้

$$z_i = x_i \oplus y_i \oplus c_i \tag{2.45}$$

เมื่อ $\oplus$ คือ กระบวนการ Exclusive-OR

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

เมื่อ & คือ กระบวนการ AND และ | คือ กระบวนการ OR

การเกิดโอเวอร์โฟลว์ของการบวกเลขชนิดมีเครื่องหมาย 2-Complement ได้โดย

$$\boxed{V = c_n \oplus c_{n-1}} \tag{2.47}$$

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2’s Complement

ตัวอย่างที่ 2.3.5 *จงคำนวณหาค่าของ 7 + 3 ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ 2-Complement ขนาด 4 บิท ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิท สามารถคำนวณได้ดังนี้*

		c_4	c_3	c_2	c_1	c_0	Overflow=True
		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	

ผลการคำนวณผลลัพธ์ที่ไม่ถูกต้อง (Invalid) เนื่องจากเกิดโอเวอร์โฟลว์ (Overflow) เพราะฮาร์ดแวร์คำนวณ $V=c_4 \oplus c_3$ = 0 $\oplus$ 1 = 1 ตามสมการที่ (2.47) ทำให้ซอฟท์แวร์นำคำตอบนี้ไปใช้ไม่ได้

2.3.2 คณิตศาสตร์เลขจำนวนเต็ม ชนิดมีเครื่องหมาย 2-Complement

ตัวอย่างที่ 2.3.7 *จงคำนวณหาค่าของ -3 - 6 ด้วยเลขจำนวนเต็มชนิดมีเครื่องหมาย แบบ 2-Complement ขนาด 4 บิท -3 - 6 = (-3) + (-6) ดังนั้น ในเครื่องคอมพิวเตอร์ขนาด 4 บิท ดังนี้*

		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	

ผลการคำนวณผลลัพธ์ที่ไม่ถูกต้อง (Invalid) เนื่องจากเกิดโอเวอร์โฟลว์ (Overflow) เพราะฮาร์ดแวร์คำนวณ $V = c_4 \oplus c_3$ = 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 **การเกิดโอเวอร์โฟลว์** (Overflow) ในสมการที่ (2.47) ซึ่งเป็นผลสืบเนื่องจากวงจรดิจิทัลที่สามารถประมวลผลได้จำกัดตามจำนวนบิตข้อมูลสูงสุดที่ทำได้ ยกตัวอย่างเช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวัง ตามประโยคในภาษา C/C++ ประโยค i++ หรือ i=i+1 นี้ หาก i มีค่าเพิ่มขึ้นเรื่อยๆ จนถึง**ค่าสูงสุด** การบวกเพิ่มอีก 1 ไปเรื่อยๆ โดยไม่มีการตรวจจับการเกิดโอเวอร์โฟลว์ล่วงหน้า จะทำให้ค่าของ i กลายเป็นค่า**ลบ**ในที่สุด ซึ่งอาจทำให้เกิดผลร้ายตามมาอย่างรุนแรง ผู้อ่านสามารถทดสอบได้ตามกิจกรรมท้ายการทดลองในภาคผนวก E

2.4 เลขทศนิยมฐานสองชนิดจุดตรึง (Fixed Point)

นิยามที่ 2.4.1 *กำหนดให้ F_2 เป็นเลขทศนิยมฐานสองชนิด Signed Magnitude เขียนอยู่ในรูป*

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

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

โดย F_{10} คือ ค่าฐานสิบของเลขทศนิยมฐานสองชนิดจุดตรึง F_2 สามารถคำนวณได้จาก

$$F_{10} = (-1)^s \times [x_{n-1}2^{n-1} + .. + x_1 2^1 + x_0 2^0 + y_{-1}2^{-1} + y_{-2}2^{-2} + y_{-3}2^{-3} + ... + y_{-m}2^{-m}] \tag{2.49}$$

หรือ

$$F_{10} = (-1)^s \times [x_{n-1}2^{n-1} + .. + x_1 2^1 + x_0 2^0 + \frac{y_{-1}}{2} + \frac{y_{-2}}{4} + \frac{y_{-3}}{8} + ... + \frac{y_{-m}}{2^m}] \tag{2.50}$$

2.4 เลขทศนิยมฐานสองชนิดจุดตรึง (Fixed Point)

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

$$+0.75_{10} = 000.11_2 \tag{2.51}$$

$$+3.00_{10} = 011.00_2 \tag{2.52}$$

$$-3.75_{10} = 111.11_2 \tag{2.53}$$

ผู้อ่านสามารถสามารถเขียนเลขทศนิยมฐานสองชนิดจุดคงที่ได้ตามรูปแบบนี้

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

ค่าทศนิยม (Y_2) มีความยาว m บิท เขียนเป็นสัญลักษณ์ได้ดังนี้

$$Y_2 = y_{-1}y_{-2}y_{-3}...y_{-m} \tag{2.55}$$

2.5 เลขทศนิยมฐานสองชนิดจุดลอยตัว

เลข ทศนิยม ฐาน สอง ชนิด ชนิด จุด ลอยตัว เหมาะ สำหรับ ข้อมูล ที่ มี พิสัย (Range) กว้าง และ เลข ทศนิยม ที่ต้องการความละเอียดสูง สำหรับการคำนวณทางวิทยาศาสตร์ (Scientific) ดังนี้

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

นิยามที่ 2.5.1 *เลข ทศนิยม ชนิด จุด ลอยตัว ฐาน สอง ที่ อยู่ ใน รูป นอร์มัลไลซ์ ประกอบ ด้วย 3 ส่วน คือ บิต เครื่องหมาย (Sign bit: s) ค่านัยสำคัญ (Significand: S_2) ในสมการที่ (2.61) และค่ายกกำลัง (Exponent: E_2) มีลักษณะดังนี้*

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

เลขยกกำลังเป็นเลขจำนวนเต็มฐานสองชนิด sign-magnitude ความยาว n บิต ดังนี้

$$E_2 = \pm[e_{n-1}e_{n-2}...e_0]_2 \tag{2.60}$$

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

$$S_2 = [1.y_{-1}y_{-2}y_{-3}...y_{-m}]_2 \tag{2.61}$$

ซึ่งมีความสำคัญต่อรูปแบบการเขียน เนื่องจากวงจรจะต้องทำการนอร์มัลไลซ์ผลลัพธ์ที่ได้จากการคำนวณเสมอ ค่านัยสำคัญที่นอร์มัลไลซ์ข้างต้นสามารถคำนวณหาค่าฐานสิบ ได้ดังนี้

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

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

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

- ชนิด Single precision (32-bit) ตรงกับตัวแปรชนิด float ในภาษา C/C++ และ Java
- ชนิด Double precision (64-bit) ตรงกับตัวแปรชนิด double ในภาษา C/C++ และ Java เวอร์ชั่นล่าสุดของ IEEE754 คือ ปี ค.ศ. 2019 รายละเอียดเพิ่มเติม

ตัวอย่างการประกาศและตั้งค่าตัวแปรที่ใช้มาตรฐานนี้

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

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

นิยามที่ 2.6.1 *เลขทศนิยมฐานสองชนิดจุดลอยตัวที่นอมัลไลซ์แล้ว สามารถเขียนอยู่ในรูปของเลขฐานสองต่อไปนี้*

$$F_{2,IEEE} = [s][E_{2,IEEE}][Y_2] \tag{2.63}$$

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

$$Y_2 = y_{-1}y_{-2}y_{-3}...y_{-m} \tag{2.64}$$

ค่ายกกำลัง เป็นเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย ความยาว n=8 และ 11 บิท ขึ้นกับชนิด Single Precision และ Double Precision ตามลำดับ โดยมีลักษณะคล้ายกับเลขทศนิยมฐานสองชนิดจุดลอยตัวในสมการที่ 2.60 แต่ต่างกันที่เลขยกกำลังของ IEEE754 มีค่าเป็นเลขจำนวนเต็มไม่มีเครื่องหมาย

$$E_{2,IEEE} = [e_{n-1}e_{n-2}...e_0] \tag{2.65}$$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

เลขทศนิยมฐานสองชนิดจุดลอยตัวตามมาตรฐาน IEEE754 ในสมการที่ 2.63 สามารถแปลงเป็นค่าเลขทศนิยมฐานสิบ ได้ดังนี้

$$F_{10,IEEE} = (-1)^s \times [1 + \frac{y_{-1}}{2} + \frac{y_{-2}}{4} + \frac{y_{-3}}{8} + ... + \frac{y_{-m}}{2^m}] \times 2^{(E_{2,IEEE} - E_{bias})} \quad (2.67)$$

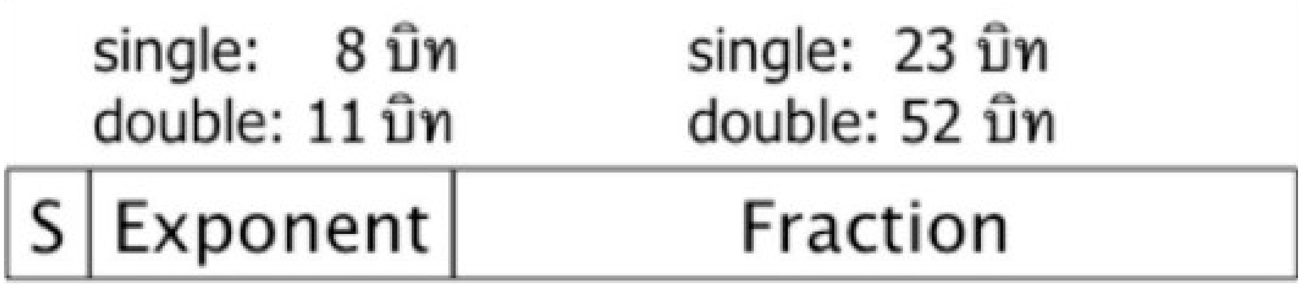
ดังนั้น เลขยกกำลังจริงจึงสามารถคำนวณได้โดย

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

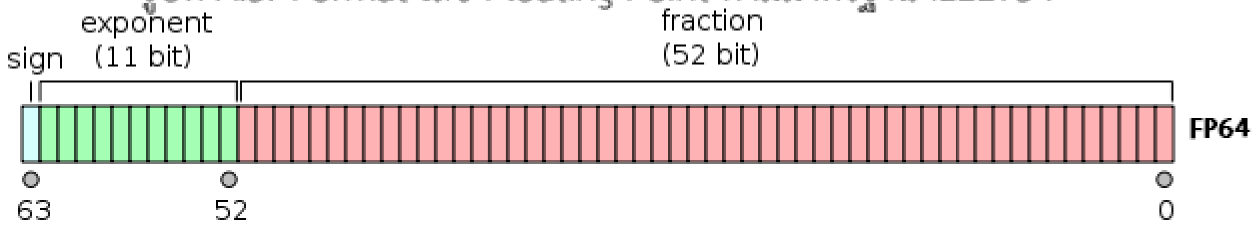
- *ชนิด Single Precision ค่า* $E_{bias} = 01111111_2 = 127_{10}$
- *ชนิด Double Precision ค่า* $E_{bias} = 01111111111_2 = 1023_{10}$

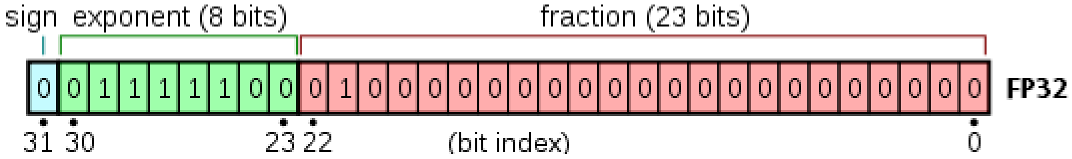
2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

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



รูปที่ A.3: Format เลข Floating-Point ตามมาตรฐาน IEEE754





2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

ตัวอย่างที่ 2.6.1 *เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754 ชนิด Single Precision ต่อไปนี้ มีค่าเท่าไรในฐานสิบตามสมการที่ (2.67)*

$4\ 2\ 2\ 8\ 0\ 0\ 0\ 0_{16}$ = [0100 0010 0010 1000 0000 0000 0000 0000]$_2$

วิธีทำ

1. แปลงจากเลขฐานสิบหกให้เป็นฐานสองและองค์ประกอบต่างๆ ได้ดังนี้

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

2. จะพบว่าบิตเครื่องหมาย $s = 0$

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

ค่าทศนิยม Y_2 = 010 1000 0000 0000 0000 0000$_2$

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

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

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

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

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

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

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

วิธีทำ

1. *แปลงเลขทศนิยมฐานสิบให้อยู่ในรูปฐานสองแบบนอร์มัลไลซ์*

$-0.75_{10} = (-1)^1 \times (0.5_{10} + 0.25_{10})$

$= (-1)^1 \times (\frac{1}{2} + \frac{1}{4})$

$= (-1)^1 \times (0.1_2 + 0.01_2)$

$= (-1)^1 \times 0.11_2 \times 2^0$

2. *ทำการนอร์มัลไลซ์ ตามสมการที่ (2.59)*

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

ดังนั้น บิตเครื่องหมาย s = 1

ค่าทศนิยม Y_2= *100 0000 0000 0000 0000 0000*$_2$

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

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน 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* $E_{2,IEEE}$ *= 0111 1110*$_2$

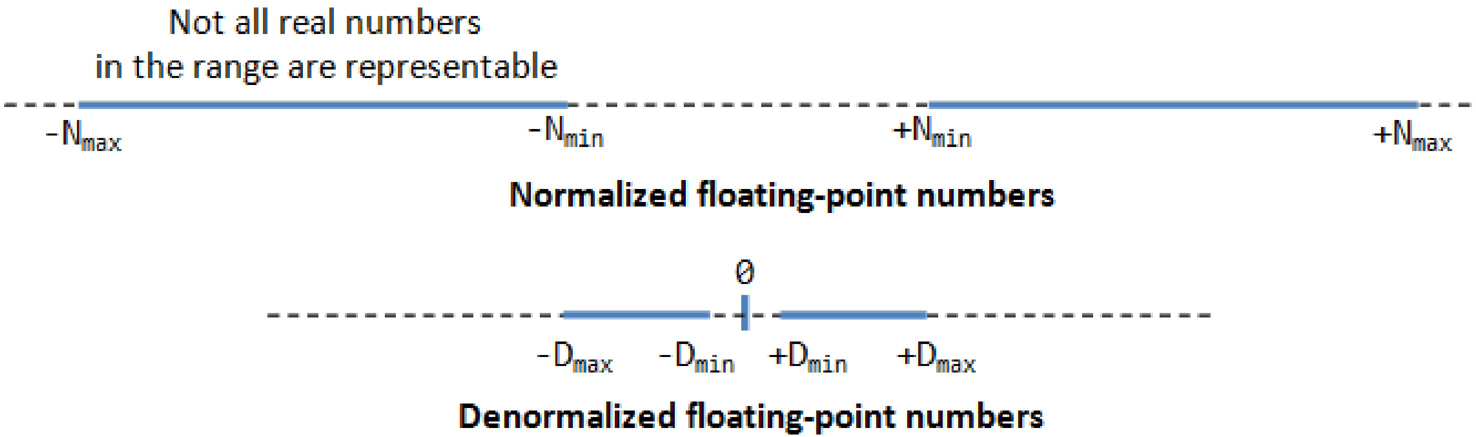
โดยชนิด Double ค่ายกกำลัง (11 บิต): $E_{2,IEEE}$*=-1 + 1023 = 1022 หรือ* $E_{2,IEEE}$ *= 011 1111 1110*$_2$

3. *ดังนั้น -0.75*$_{10}$ *เขียนในรูปของเลขทศนิยมชนิดลอยตัวตามมาตรฐาน IEEE754 ชนิด*

Single: [1][011 1111 0][100 0000 0000 0000 0000 0000]$_2$ *= BF40 0000*$_{16}$

Double: [1][011 1111 1110] [1000 0000 0000 ...0000]$_2$ *= BFE8 0000 0000 0000*$_{16}$

2.6 เลขทศนิยมฐานสองชนิดจุดลอยตัวมาตรฐาน IEEE754

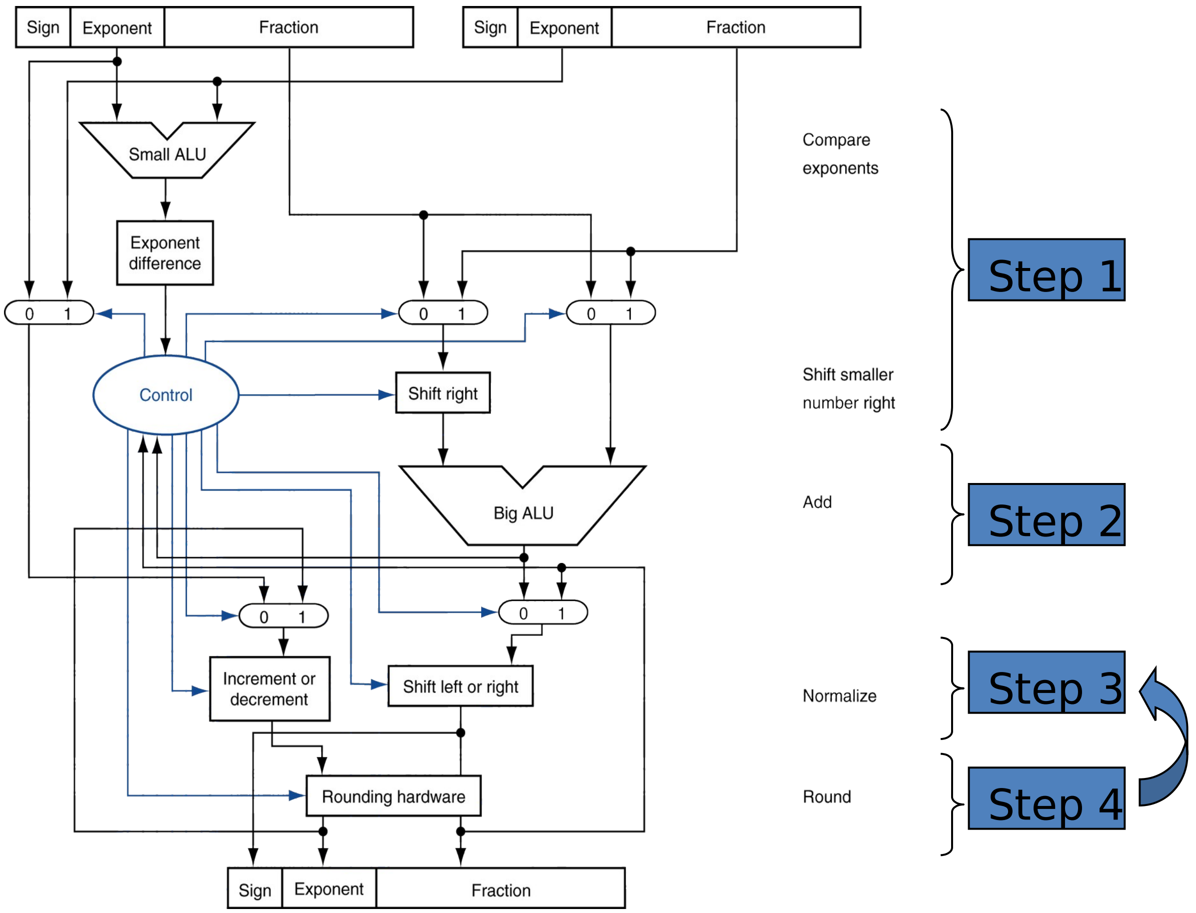


1	$\pm$	$0000\ 0001_2$- $1111\ 1110_2$	$xx.._2$	เลขฐานสิบทั่วไป (นอมัลไลซ์) สมการที่ (2.67)
2	$\pm$	$0000\ 0000_2$	$xx.._2$	เลขฐานสิบที่น้อยมาก แต่ไม่เท่ากับศูนย์ (ดีนอมัลไลซ์)
3	0	$0000\ 0000_2$	$0...0_2$	0.0_{10} (ศูนย์จุดศูนย์)
4	$\pm$	$1111\ 1111_2$	$0...0_2$	$\pm\infty$ ($\pm$อินฟินิตี)
5	0	$1111\ 1111_2$	$xx.._2$	Nan (Not a Number)

Floating-point numbers

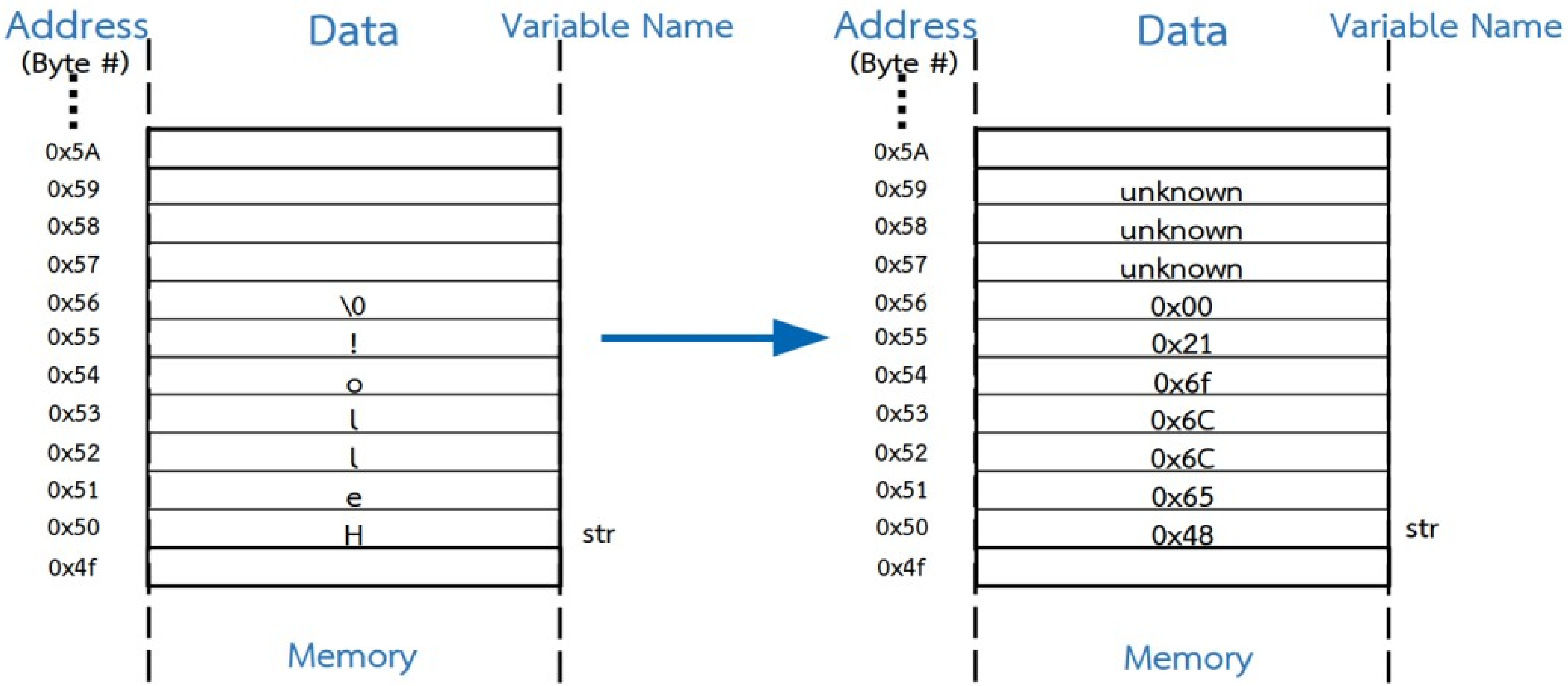
0	10000001	111 1100 0000 0000 0000 0000
0	01111100	100 0000 0000 0000 0000 0000

		Exponent		Fraction
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
		01111100		1.100 0000 0000 0000 0000 0000
Step 3		10000001		1.111 1100 0000 0000 0000 0000
	–	01111100		1.100 0000 0000 0000 0000 0000
		101 (shift amount)		
Step 4		10000001		1.111 1100 0000 0000 0000 0000
		10000001		0.000 0110 0000 0000 0000 0000 00000
Step 5		10000001		1.111 1100 0000 0000 0000 0000
		10000001	+	0.000 0110 0000 0000 0000 0000
				10.000 0010 0000 0000 0000 0000
Step 6		10000001		10.000 0010 0000 0000 0000 0000 >> 1
	+	1		
		10000010		1.000 0001 0000 0000 0000 0000
Step 7		(No rounding necessary)		
Step 8	0	10000010		000 0001 0000 0000 0000 0000



2.7 ตัวอักษร (Character)

```
char[10] str="Hello!"
```



2.7 ตัวอักษร (Character)

Codepage 874 - Thai

	-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-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-																
9-																
A-	่ 0E48	ก 0E01	ข 0E02	ฃ 0E03	ค 0E04	ฅ 0E05	ฆ 0E06	ง 0E07	จ 0E08	ฉ 0E09	ช 0E0A	ซ 0E0B	ฌ 0E0C	ญ 0E0D	ฎ 0E0E	ฏ 0E0F
B-	ฐ 0E10	ฑ 0E11	ฒ 0E12	ณ 0E13	ด 0E14	ต 0E15	ถ 0E16	ท 0E17	ธ 0E18	น 0E19	บ 0E1A	ป 0E1B	ผ 0E1C	ฝ 0E1D	พ 0E1E	ฟ 0E1F
C-	ภ 0E20	ม 0E21	ย 0E22	ร 0E23	ฤ 0E24	ล 0E25	ฦ 0E26	ว 0E27	ศ 0E28	ษ 0E29	ส 0E2A	ห 0E2B	ฬ 0E2C	อ 0E2D	ฮ 0E2E	ฯ 0E2F
D-	ะ 0E30	ั 0E31	า 0E32	ำ 0E33	ิ 0E34	ี 0E35	ึ 0E36	ื 0E37	ุ 0E38	ู 0E39	ฺ 0E3A	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	฿ 0E3F
E-	เ 0E40	แ 0E41	โ 0E42	ใ 0E43	ไ 0E44	ๅ 0E45	ๆ 0E46	็ 0E47	่ 0E48	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	ํ 0E4D	๎ 0E4E	๏ 0E4F
F-	๐ 0E50	๑ 0E51	๒ 0E52	๓ 0E53	๔ 0E54	๕ 0E55	๖ 0E56	๗ 0E57	๘ 0E58	๙ 0E59	๚ 0E5A	๛ 0E5B	¢ 00A2	¬ 00AC	¦ 00A6	00A0

- รหัส ASCII คือ มาตรฐานของรูปแบบการใช้เลขฐานสองเพื่อแทนตัวอักษรตั้งแต่อดีต ซึ่งกำหนดขึ้นมาโดยหน่วยงานชื่อว่า ANSI (American National Standard Institute)
- ตารางรหัสแอสกี้ (ASCII) กำหนดเลขฐานสองขนาด 8 บิต เพื่อแทนตัวอักษรจำนวน $2^8=256$ ตัว โดยมีรหัสเริ่มต้น คือ $00_{16} = 0000\ 0000_2$ ถึง $FF_{16} = 1111\ 1111_2$
  โดยรหัส 00_{16} แทนอักษร NULL อ่านว่า นัลล์
- ไมโครซอฟต์พัฒนารหัสภาษาไทยของตนเอง เรียกว่า Windows-874 โดยใช้มาตรฐาน TIS-620 เป็นพื้นฐานสำหรับตัวอักษรภาษาไทยสำหรับการแลกเปลี่ยนข้อมูลในระบบปฏิบัติการ Windows

2.7 ตัวอักษร (Character)

Codepage 874 - Thai

	-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-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-																
9-																
A-	่ 0E48	ก 0E01	ข 0E02	ฃ 0E03	ค 0E04	ฅ 0E05	ฆ 0E06	ง 0E07	จ 0E08	ฉ 0E09	ช 0E0A	ซ 0E0B	ฌ 0E0C	ญ 0E0D	ฎ 0E0E	ฏ 0E0F
B-	ฐ 0E10	ฑ 0E11	ฒ 0E12	ณ 0E13	ด 0E14	ต 0E15	ถ 0E16	ท 0E17	ธ 0E18	น 0E19	บ 0E1A	ป 0E1B	ผ 0E1C	ฝ 0E1D	พ 0E1E	ฟ 0E1F
C-	ภ 0E20	ม 0E21	ย 0E22	ร 0E23	ฤ 0E24	ล 0E25	ฦ 0E26	ว 0E27	ศ 0E28	ษ 0E29	ส 0E2A	ห 0E2B	ฬ 0E2C	อ 0E2D	ฮ 0E2E	ฯ 0E2F
D-	ะ 0E30	ั 0E31	า 0E32	ำ 0E33	ิ 0E34	ี 0E35	ึ 0E36	ื 0E37	ุ 0E38	ู 0E39	ฺ 0E3A	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	฿ 0E3F
E-	เ 0E40	แ 0E41	โ 0E42	ใ 0E43	ไ 0E44	ๅ 0E45	ๆ 0E46	็ 0E47	่ 0E48	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	ํ 0E4D	๎ 0E4E	๏ 0E4F
F-	๐ 0E50	๑ 0E51	๒ 0E52	๓ 0E53	๔ 0E54	๕ 0E55	๖ 0E56	๗ 0E57	๘ 0E58	๙ 0E59	๚ 0E5A	๛ 0E5B	¢ 00A2	¬ 00AC	¦ 00A6	00A0

- รหัส Unicode ถูกกำหนดให้เป็นมาตรฐานโดย ISO (International Standard Organization) เพื่อมาทดแทนรหัส ASCII เนื่องจากความต้องการใช้ภาษาทั่วโลกที่เพิ่มขึ้นเรื่อยๆ
- รหัส UCS-2 จะใช้พื้นที่ 2 ไบท์ หรือ 16 บิต ต่อ 1 ตัวษร ซึ่งทำให้สามารถใช้เลขฐานสองจำนวน 2${16}$ หรือ 65,536 แบบมาแทนตัวอักษร ตำแหน่งเริ่มต้นของตารางรหัส Unicode จะเหมือนกับตารางรหัส ASCII ตัวภาษาอังกฤษและภาษาไทย ภายในแต่ละตัวอักษร 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 ไบท์ จนถึง 4 ไบท์ โดยตัวอักษร 128 ตัวแรกคือรหัส ASCII ใช้ความยาว 1 ไบท์ ส่วนตัวอักษรในภาษาอื่นๆ จะใช้จำนวนไบท์เพิ่มขึ้น
- รหัส UTF-16 คือ การขยายรหัส UCS-2 ให้ทันสมัยมากขึ้น โดยเพิ่มการเข้ารหัสเป็นขนาด 4 ไบท์

2.8 สรุปท้ายบท

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	แอดเดรสชี้ตำแหน่งข้อมูล 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%E2%80%93computer_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