หลักการสำหรับการดำเนินการเชิงเลขคณิต (บวก ลบ คูณ หาร) สามารถนำมาใช้ในระบบเลขฐานสองได้ เพียงแต่ว่าตัวเลขที่ใช้ในระบบฐานสองมีเพียงสองตัว คือ 0 และ 1 เท่านั้น

สำหรับการแปลงเลขฐานสองเป็นเลขฐานสิบสามารถทำได้ตรงไปตรงมา ดังเช่นตัวอย่างที่ 2.1.1 ในทางกลับกันการแปลงเลขฐานสิบเป็นเลขฐานสอง สามารถทำได้ง่ายเช่นกัน ในที่นี้จะแสดงวิธีที่ง่าย ๆ สำหรับคำนวณด้วยตนเองหรือเครื่องคิดเลข โดยแบ่งการแปลงเป็นสองส่วนคือ

ส่วนที่ 1 วิธีการแปลงเลขจำนวนเต็มฐานสิบเป็นฐานสอง

ให้ x เป็นเลขจำนวนเต็มบวกฐานสิบ ต้องการเขียน x ในรูป

$$x = c_n 2^n + c_{n-1} 2^{n-1} + \cdots + c_1 2^1 + c_0 2^0 \tag{2.1.1}$$

เมื่อ $c_0, c_1, \ldots, c_n$ เป็น 0 หรือ 1 ดังนั้น

$$(x)_{10} = (c_n c_{n-1} \ldots c_1 c_0)_2 \tag{2.1.2}$$

จากสมการ (2.1.1) เริ่มหา ส.ป.ส. c_0 ก่อน โดยการเอา 2 หาร x เศษที่ได้คือ c_0 และเรียกผลหารที่ได้เป็น q_1 ซึ่งอยู่ในรูป

$$q_1 = c_n 2^{n-1} + c_{n-1} 2^{n-2} + \cdots + c_1 2^0 \tag{2.1.3}$$

ต่อไปหา ส.ป.ส. c_1 โดยเอา 2 หาร q_1 ในสมการ (2.1.3) เรียกผลหารที่ได้เป็น q_2 และเศษที่ได้ก็คือ c_1 ดำเนินการต่อไปในทำนองเดียวกัน ผลคือ ได้ส.ป.ส. $c_2, c_3, c_4, \ldots, c_n$ มาตามลำดับ

ส่วนที่ 2 วิธีการแปลงเลขเศษส่วนฐานสิบเป็นฐานสอง

ให้ y เป็นเลขเศษส่วนฐานสิบ ซึ่งน้อยกว่า 1 และมีค่าเป็นบวก ต้องการเขียน y ในรูป

$$y = d_1 2^{-1} + d_2 2^{-2} + d_3 2^{-3} + \cdots \qquad (2.1.4)$$

เมื่อ $d_1, d_2, d_3, \ldots$ เป็น 0 หรือ 1 ดังนั้น

$$(y)_{10} = (.d_1\, d_2\, d_3\, \ldots)_2 \qquad (2.1.5)$$

ในทางตรงกันข้ามกับส่วนที่ 1 จากสมการ (2.1.4) จะเห็นได้ว่า หา ส.ป.ส. d_1 ได้โดยการเอา 2 คูณ y ทำให้ได้ผลคูณที่เป็นจำนวนเต็ม d_1 และเรียกเศษส่วนที่ได้เป็น f_1 นั่นคือ

$$2y = d_1 + d_2 2^{-1} + d_3 2^{-2} + \cdots = d_1 + f_1 \qquad (2.1.6)$$

ต่อไปหา ส.ป.ส. d_2 โดยการเอา 2 คูณ f_1 ได้ผลคูณที่เป็นจำนวนเต็มคือ d_2 และเรียกเศษส่วนที่ได้เป็น f_2 ดำเนินการต่อไปในลักษณะนี้ ผลคือ ได้ ส.ป.ส. $d_3, d_4, \ldots$ มาตามลำดับ

ตัวอย่างที่ 2.1.2 จงแปลงเลขฐานสิบ 21.09375 เป็นเลขฐานสอง

วิธีทำ

แบ่ง 21.09375 เป็นสองส่วน โดยให้ $x = 21$ และ $y = .09375$ แล้วแปลงเป็นเลขฐานสองตามวิธีการที่อธิบายไว้ ได้ผลของการคำนวณเป็นลำดับดังนี้

$$
\begin{array}{lllll}
x = 21 & q_1 = 10, & c_0 = 1 \\
 & q_2 = 5, & c_1 = 0 \\
 & q_3 = 2, & c_2 = 1 \\
 & q_4 = 1, & c_3 = 0 \\
 & q_5 = 0, & c_4 = 1
\end{array}
$$

$$\Rightarrow \quad (21)_{10} = (10101)_2$$

$$y = 0.09375$$

$$
\begin{array}{lll}
2y = .1875 & f_1 = .1875, & d_1 = 0 \\
2f_1 = .375 & f_2 = .375, & d_2 = 0 \\
2f_2 = .75 & f_3 = .75, & d_3 = 0 \\
2f_3 = 1.5 & f_4 = .5, & d_4 = 1 \\
2f_4 = 1.0 & f_5 = 0, & d_5 = 1
\end{array}
$$

$$\Rightarrow \quad (.09375)_{10} = (.00011)_2$$

ดังนั้น

$$(21.09375)_{10} = (10101.00011)_2$$

□

ตัวอย่างที่ 2.1.3 จงแปลงเลขฐานสิบ 11.1 เป็นเลขฐานสอง

วิธีทำ

ในทำนองเดียวกับตัวอย่างที่ 2.1.2 ได้ว่า

$$(11)_{10} = (1011)_2$$

และให้ $y = .1$ โดยใช้หลักการคูณด้วย 2 ได้

$(8502)_{16} = (8\times16^3) + (5\times16^2) + 0 + (2\times16^0) = (34050)_{10}$

$(AF1)_{16} = (10\times16^2) + (15\times16^1) + (1\times16^0) = (2801)_{10}$

$(AF1.17)_{16} = (10\times16^2) + (15\times16^1) + (1\times16^0) + (1\times16^{-1}) + (7\times16^{-2}) =$

$(2801)_{10}$

$$
\begin{array}{lll}
2y = .2 & f_1 = .2, & d_1 = 0 \\
2f_1 = .4 & f_2 = .4, & d_2 = 0 \\
2f_2 = .8 & f_3 = .8, & d_3 = 0 \\
2f_3 = 1.6 & f_4 = .6, & d_4 = 1 \\
2f_4 = 1.2 & f_5 = .2, & d_5 = 1 \\
2f_5 = .4 & f_6 = .4, & d_6 = 0 \\
\quad\vdots & &
\end{array}
$$

ฉะนั้นเมื่อแปลง .1 ฐานสิบเป็นเลขฐานสอง ผลคือ เศษส่วนฐานสองแบบซ้ำไม่รู้จบ นั่นคือ

$$(.1)_{10} = (.0001100110011\ldots)_2$$

สาเหตุที่เป็นแบบซ้ำไม่รู้จบก็เนื่องมาจาก $.1 = \dfrac{1}{10} = \dfrac{1}{2}\cdot\dfrac{1}{5}$ ซึ่ง 5 ไม่เป็นตัวประกอบของฐาน 2 □

2.2 ระบบเลขฐานสิบหก (Hexadecimal Number System)

ในทำนองเดียวกับการแทนเลขจำนวนใดๆในระบบเลขฐานสิบและระบบเลขฐานสองในระบบเลขฐานสิบหก ฐานก็คือ 16 และตัวเลขทั้ง 16 ตัวในระบบ คือ

$$0 \ 1 \ 2 \ \ldots \ 8 \ 9 \ A \ B \ C \ D \ E \ F$$

โดยมีความหมายเมื่อเปรียบเทียบกับเลขฐานสิบดังนี้

$$
\begin{array}{lll}
(0)_{16} = (0)_{10}, & (1)_{16} = (1)_{10}, & (2)_{16} = (2)_{10}, \\
\quad\ldots, & (8)_{16} = (8)_{10}, & (9)_{16} = (9)_{10}, \\
(A)_{16} = (10)_{10}, & (B)_{16} = (11)_{10}, & (C)_{16} = (12)_{10}, \\
(D)_{16} = (13)_{10}, & (E)_{16} = (14)_{10}, & (F)_{16} = (15)_{10}
\end{array}
$$

ตัวอย่างที่ 2.2.1 จงแปลงเลขฐานสิบหก $2F.17$ เป็นเลขฐานสิบและฐานสอง

วิธีทำ

เขียน $2F.17$ ในรูปผลบวกของผลคูณของสิบหกยกกำลังจำนวนเต็มได้ดังนี้

$$\begin{aligned} 2F.17 &= (2\times 16^1 + 15\times 16^0 + 1\times 16^{-1} + 7\times 16^{-2})_{10} \\ &= (47.08984375)_{10} \end{aligned}$$

การแปลง $(2F.17)_{16}$ เป็นเลขฐานสองทำได้โดยง่าย เพราะว่า

$$\begin{aligned} (2)_{16} &= (0010)_2 \qquad (F)_{16} = (1111)_2 \\ (1)_{16} &= (0001)_2 \qquad (7)_{16} = (0111)_2 \end{aligned}$$

ดังนั้น

$$(2F.17)_{16} = (101111.00010111)_2$$ □

ตัวอย่างที่ 2.2.2 จงแปลงเลขฐานสอง 10101.110101 เป็นเลขฐานสิบหก

วิธีทำ

แบ่งกลุ่มจากจุดทวินิยม (binary point) เป็นกลุ่มละ 4 บิต นั่นคือ

$$(0001\ 0101.1101\ 0100)_2 = (15.D4)_{16}$$ □

คอมพิวเตอร์เชิงตัวเลขโดยทั่วไปจะใช้ระบบการแทนเลขในคอมพิวเตอร์เป็นระบบฐานสอง สำหรับคอมพิวเตอร์ IBM ใช้ระบบการแทนเลขเป็นระบบเลขฐานสิบหก

2.3 การแทนแบบอิงดรรชนี (Floating-Point Representation)

พิจารณาเลขฐานสิบ 3.14159 เห็นได้ชัดว่า สามารถเขียนเลขจำนวนนี้ได้หลายแบบ โดยการเลื่อนจุดทศนิยมและปรับเลขชี้กำลังของฐาน 10ให้ถูกต้อง เช่น

$$31.4159 \times 10^{-1} \quad \text{หรือ} \quad .0314159 \times 10^{2}$$

อีกนัยหนึ่ง สามารถเขียนจำนวนจริงใด ๆ x ในรูป

$$x = (-1)^{s} \times a \times 10^{e} \qquad (2.3.1)$$

เมื่อ s เป็น 0 หรือ 1 แทนเครื่องหมาย เรียก a ว่า แมนทิสซา (mantissa) ซึ่งมีค่า $\boxed{0 \le a < 1}$ นั่นคือ a แทนเศษส่วน และเรียก e ซึ่งเป็นจำนวนเต็มว่า เลขชี้กำลัง (exponent) ซึ่งบอกตำแหน่งของจุดทศนิยมเมื่อเทียบกับ a

การแทนแบบอิงดรรชนี ก็คือ การแทนเลขจำนวนจริงใด ๆ โดยอาศัยองค์ประกอบสามส่วนคือ เครื่องหมาย แมนทิสซาและเลขชี้กำลัง โดยทั่วไปการแทนแบบอิงดรรชนีสำหรับเลขฐาน b คือ

$$x = (-1)^{s} \times a \times b^{e} \qquad (2.3.2)$$

เมื่อตัวเลขในแมนทิสซา a เป็นตัวเลขในระบบนั้น ๆ นอกจากนี้ เรียก ลำดับของตัวเลขในแมนทิสซาไม่รวมตัวเลขนำหน้าที่เป็นศูนย์ว่า เลขนัยสำคัญ (significant digits)

การแทนเลขแบบอิงดรรชนี สำหรับจำนวนจริงจำนวนเดียวกันทำได้หลายแบบ จึงมีการแทนแบบอิงดรรชนีบรรทัดฐาน (normalized floating-point representation) โดยกำหนดว่า ตัวเลขที่หนึ่งของแมนทิสซาต้องไม่เป็นตัวเลขศูนย์ ดังนั้น ค่าของ a ในสมการ (2.3.1) และ (2.3.2) ต้องสอดคล้องตาม

$$0.1 \le a < 1$$

ตัวอย่างที่ 2.3.1 จงเขียน $(12.462)_{10}$, $(-5/8)_{10}$, $(101)_2$ และ $(-.0011)_2$ เป็นแบบอิงดรรชนีบรรทัดฐาน

วิธีทำ

$$
\begin{aligned}
(12.462)_{10} &= +.12462 \times 10^2 \\
(-5/8)_{10} &= -.625 \times 10^0 \\
(101)_2 &= +.101 \times 2^3 \\
(-.0011)_2 &= -.11 \times 2^{-2}
\end{aligned}
$$

□

2.4 การแทนเลขในคอมพิวเตอร์

เลขที่ใช้ในการคำนวณด้วยดิจิทัลคอมพิวเตอร์ โดยทั่วไปจัดอยู่ใน 3 ภาวะ คือ

1. ภาวะจำนวนเต็ม (integer mode)
2. ภาวะจำนวนจริง (real mode)
3. ภาวะจำนวนเชิงซ้อน (complex mode)

แต่ละภาวะขึ้นอยู่กับการกำหนดชนิดของข้อมูลตัวเลขซึ่งเป็นข้อมูลเข้า (input data) ผลลัพธ์ (output) และตัวแปรในโปรแกรม ในที่นี้จะกล่าวถึงการแทนจำนวนเต็มและจำนวนจริงในคอมพิวเตอร์เท่านั้น สำหรับการแทนจำนวนเชิงซ้อนในคอมพิวเตอร์ ก็คือ การแทนส่วนจริง (real part) และส่วนจินตภาพ (imaginary part) ซึ่งต่างก็เป็นจำนวนจริงนั่นเอง

โดยที่ขนาดของข้อมูลซึ่งคอมพิวเตอร์ประมวลได้ในการดำเนินการแต่ละครั้ง ขึ้นอยู่กับความยาวคำ (word length) ในหน่วยความจำว่า ประกอบด้วยจำนวยบิตเป็นจำนวนเท่าใด ความยาวคำจะแตกต่างกันออกไปตามชนิดของคอมพิวเตอร์ ดังนั้นการแทนข้อมูลตัวเลขจึงต้องเป็นแบบจำกัด (finite form) ทำให้มีขอบเขตและการจำกัดจำนวนตัวเลขที่สามารถแทนได้ในคอมพิวเตอร์ ด้วยเหตุนี้ จึงต้องตระหนักว่า คอมพิวเตอร์ไม่สามารถแทนจำนวนจริงทุก ๆ จำนวนได้อย่างแม่นยำ และผลลัพธ์ในการดำเนินการพื้นฐานแต่ละครั้งจะถูกปัดเศษ (rounding) หรือตัดเศษ (chopping) ตามข้อจำกัดในการแทนเลขของคอมพิวเตอร์นั้น ๆ

2.4.1 การแทนจำนวนเต็ม

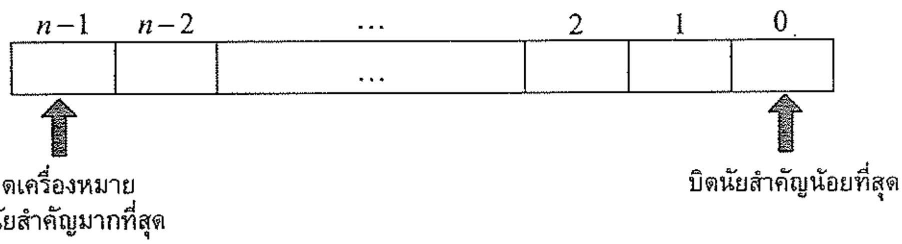
สมมติว่าความยาวคำ คือ n บิต ดังนั้น ลำดับของตัวเลข 0 และ 1 ในแต่ละบิตของคำมีได้ทั้งหมด 2^n แบบ นั่นคือ คำยาว n บิต สามารถแทนจำนวนเต็มได้ทั้งหมด 2^n จำนวน เช่น

แทนจำนวนเต็มซึ่งไม่เป็นลบจาก 0 ถึง $2^n - 1$ ด้วยเลขฐานสองโดยตรง

หรือ

แทนจำนวนเต็มลบจาก -2^{n-1} ถึง -1

รูปที่ 2.4.1 แสดงคำยาว n บิต กำกับด้วยหมายเลข 0 ถึง $n-1$ จากขวาสุดไปซ้ายสุด เริ่มจาก 0 ถึง $n-1$ บิตที่ $n-1$ หรือบิตซ้ายสุด คือ บิตนัยสำคัญมากที่สุด และบิต 0 หรือบิตขวาสุด คือ บิตนัยสำคัญน้อยที่สุด และโดยทั่วไป บิตซ้ายสุดเป็นบิตระบุเครื่องหมาย (sign bit) เช่น ถ้าเป็น 0 หมายถึง เครื่องหมายบวก และถ้าเป็น 1 หมายถึงเครื่องหมายลบ



รูปที่ 2.4.1

ตัวอย่างที่ 2.4.1 สมมติว่าคอมพิวเตอร์แทนจำนวนเต็มด้วย 8 บิต ดังนั้น สามารถแทนจำนวนเต็มได้ทั้งหมด $2^8 = 256$ จำนวน เช่น

แทนจำนวนเต็มจาก 0 ถึง $2^8 - 1 = 255$ ดังแสดงในตารางที่ 2.4.1 (ก)

หรือ

แทนจำนวนเต็มจาก $-2^{8-1} = -128$ ถึง $2^{8-1} - 1 = 127$
ดังแสดงในตารางที่ 2.4.1 (ข)

ซึ่งเป็นแบบที่จะกล่าวถึงต่อไป สังเกตได้จากตารางที่ 2.4.1 (ข) ว่า บิต 7 เป็น 1 สำหรับจำนวนเต็มลบ และเป็น 0 สำหรับจำนวนเต็มบวก

เลขฐานสิบ 0 ถึง 255	บิต 7 $\rightarrow$ บิต 0
0	0000 0000
1	0000 0001
2	0000 0010
⋮	⋮
253	1111 1101
254	1111 1110
255	1111 1111

ตารางที่ 2.4.1 (ก)

เลขฐานสิบ -128 ถึง 127	บิต 7 $\rightarrow$ บิต 0
-128	1000 0000
-127	1000 0001
⋮	⋮
-2	1111 1110
-1	1111 1111
0	0000 0000
1	0000 0001
2	0000 0010
⋮	⋮
126	0111 1110
127	0111 1111

ตารางที่ 2.4.1 (ข)

□

จากตัวอย่างที่ 24.1 จะสังเกตได้ว่า มีวิธีการแทนจำนวนเต็บลบอย่างมีระบบ โดยเฉพาะบิตที่ระบุเครื่องหมาย แต่ทั้งนี้มิได้หมายความว่า คอมพิวเตอร์เปลี่ยนจำนวนเต็มบวกให้เป็นจำนวนเต็มลบ โดยการเปลี่ยนค่าของบิตระบุเครื่องหมายนี้ ทั้งนี้ต้องขึ้นอยู่กับวิธีการแทนจำนวนเต็มลบที่ใช้ในคอมพิวเตอร์ ซึ่งมีอยู่ 3 วิธี ดังนี้

วิธีที่ 1 การแทนแบบคอมพลีเมนต์ของหนึ่ง (one's complement representation) คือ การเปลี่ยนจำนวนเต็ม x ให้เป็น $-x$ โดยการเปลี่ยนค่าในบิตทั้งหมดที่แทน x เช่น สำหรับคำ 8 บิต

0000 1101 แทน 13

จะได้ว่า การแทนแบบคอมพลีเมนต์ของหนึ่งสำหรับ -13 คือ

1111 0010 แทน -13

วิธีที่ 2 การแทนแบบคอมพลีเมนต์ของสอง (two's complement representation) คือ การเปลี่ยนจำนวนเต็ม x ให้เป็น $-x$ โดยใช้การแทนแบบคอมพลีเมนต์ของหนึ่งก่อน แล้วตามด้วยการบวกด้วยหนึ่ง เช่น สำหรับคำ 8 บิต

0000 1101 แทน 13

จะได้ว่า การแทนแบบคอมพลีเมนต์ของสองสำหรับ -13 คือ

1111 0011 แทน -13

วิธีการแทนแบบคอมพลีเมนต์ของสอง ยังสามารถพิจารณาได้ในอีกสองลักษณะคือ ลักษณะที่หนึ่ง คือ สำหรับคำ n บิต ที่ใช้แทนจำนวนเต็ม กำหนดให้น้ำหนักของบิตนัยสำคัญมากที่สุดเป็น -2^{n-1} ดังนั้นสำหรับคำ 8 บิต เช่น

7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

เลขฐานสองในคำนี้ คือ

$$-2^7 + 2^6 + 2^5 + 2^4 + 0\times 2^3 + 0\times 2^2 + 2^1 + 2^0 = -13$$

ลักษณะที่สองคือ ต้องการใช้คำ n บิต เพื่อแทนจำนวนเต็มในช่วง -2^{n-1} ถึง 2^n-1 โดยการบวกด้วยค่าคงที่จำนวนเต็ม 2^{n-1} ทำให้สามารถแทนจำนวนเต็มในช่วงนี้ด้วยช่วงใหม่ คือ 0 ถึง 2^n-1+2^{n-1} ซึ่งประกอบด้วยจำนวนเต็มบวกหมดยกเว้น ศูนย์ รูปแบบนี้มีชื่อเรียกว่า แบบแคแรกเทอริสติก (characteristic form) ซึ่งนิยมใช้แทนเลขชี้กำลัง และเรียกค่าคงที่จำนวนเต็มนี้ว่า ไบแอส (bias) ตารางที่ 2.4.2 แสดงการเปรียบเทียบการแทนจำนวนเต็ม -128 ถึง 127 ด้วยแบบคอมพลีเมนต์ของสองและแบบแคแรกเทอริสติกด้วยคำ 8 บิต จะเห็นได้ว่า บิตระบุเครื่องหมายถูกเปลี่ยนหมด ทำให้ได้ว่า 0 ถูกแทนด้วย 1000 0000 สำหรับแบบแคแรกเทอริสติก

เลขฐานสิบ	เลขฐานสิบ บวก 128	แบบคอมพลีเมนต์ของสอง	แบบแคแรกเทอริสติก
-128	0	1000 0000	0000 0000
-127	1	1000 0001	0000 0001
-126	2	1000 0010	0000 0010
⋮	⋮	⋮	⋮
-1	127	1111 1111	0111 1111
0	128	0000 0000	1000 0000
+1	129	0000 0001	1000 0001
⋮	⋮	⋮	⋮
+126	254	0111 1110	1111 1110
+127	255	0111 1111	1111 1111

ตารางที่ 2.4.2

วิธีที่ 3 การแทนแบบขนาดมีเครื่องหมาย (signed magnitude representation) คือ การเปลี่ยนจำนวนเต็ม x ให้เป็น $-x$ โดยการเปลี่ยนค่าของบิตที่ระบุเครื่องหมาย เช่น ใช้คำ 8 บิตแทนจำนวนเต็ม -127 ถึง 127 โดยใช้บิต 7 เป็นบิตระบุเครื่องหมาย ซึ่งมีค่าเป็น 0 สำหรับเครื่องหมายบวก และมีค่าเป็น 1 สำหรับเครื่องหมายลบ และที่เหลืออีก 7 บิต ใช้แทนจำนวนเต็ม 0 ถึง 127 สังเกตจากตารางที่ 2.4.3 จะเห็นว่า มีการแทน 0 ได้สองแบบ คือ -0 และ $+0$

เลขฐานสิบ	แบบขนาดมีเครื่องหมาย บิต 7 → บิต 0
-127	1111 1111
-126	1111 1110
⋮	⋮
-1	1000 0001
-0	1000 0000
+0	0000 0000
+1	0000 0001
⋮	⋮
+126	0111 1110
+127	0111 1111

ตารางที่ 2.4.3

ในปัจจุบัน แบบที่นิยมใช้แทนจำนวนเต็มลบ คือ แบบคอมพลีเมนต์ของสอง เพราะว่า มีความสะดวกในการบวกจำนวนเต็มในคอมพิวเตอร์ โดยคอมพิวเตอร์ไม่ต้องตรวจสอบว่า ค่าของบิตระบุเครื่องหมายเป็นเท่าใด หรือต้องตัดสินว่า ในกรณีของการลบ ตัวตั้งหรือตัวไปลบออก ตัวใดมีค่ามากกว่ากัน พิจารณาจากตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.4.2 จงใช้คำ 8 บิต แทนจำนวนเต็ม 9 และ –23 แบบคอมพลีเมนต์ของสอง แล้วแสดงการหาค่าของ $9 - 23$ หรือ $9+(-23)$ ในรูปแบบฐานสอง

วิธีทำ

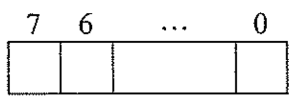
9	แทนด้วย	0000 1001
23	แทนด้วย	0001 0111
–23	แทนด้วย	1110 1001

ดังนั้น

$$\begin{array}{rr} 0000\ \ 1001 & 9 \\ \underline{+\,1110\ \ 1001} & \underline{+(-23)} \\ \underline{\underline{=1111\ \ 0010}} & \underline{\underline{=-14}} \end{array}$$

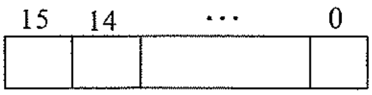
เพราะว่า 1111 0010 แทนจำนวนเต็ม –14 ในแบบคอมพลีเมนต์ของสอง □

ตัวอย่างที่ 2.4.3 DEC 4000 ระบบ AXP แทนจำนวนเต็มด้วยคำ 8 บิต 16 บิต 32 บิต และ 64 บิต ตามขนาดที่ระบุด้วยภาษาฟอร์แทรนเป็น INTEGER*1 INTEGER*2 INTEGER*4 และ INTEGER*8 ตามลำดับ (ถ้าระบุเพียง INTEGER จำนวนบิตที่ใช้แทนจำนวนเต็ม คือ 32 บิต) การแทนจำนวนเต็มลบนั้น DEC ใช้แบบคอมพลีเมนต์ของสอง โดยใช้บิตซ้ายสุดหรือบิตนัยสำคัญมากที่สุดเป็นบิตระบุเครื่องหมาย ซึ่งมีค่าเป็น 0 สำหรับเครื่องหมายบวก และมีค่าเป็น 1 สำหรับเครื่องหมายลบ ในรูปที่ 2.4.2 (ก) - (ง) บิต 7 บิต 15 บิต 31 และ บิต 63 เป็นบิตระบุเครื่องหมาย และบิตที่เหลือเป็นเลขฐานสอง



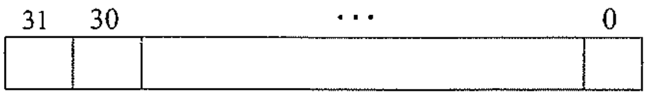
INTEGER*1 ใช้ 1 ไบท์หรือ 8 บิต แทนจำนวนเต็มจาก -2^7 ถึง 2^7-1 หรือ –128 ถึง 127

รูปที่ 2.4.2 (ก)



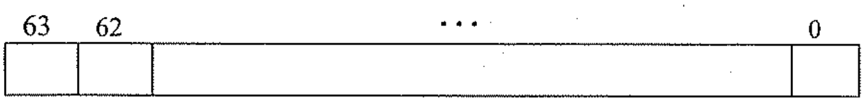
INTEGER*2 ใช้ 2 ไบท์หรือ 16 บิต แทนจำนวนเต็มจาก -2^{15} ถึง $2^{15}-1$ หรือ -32768 ถึง 32767

รูปที่ 2.4.2 (ข)



INTEGER*4 หรือ INTEGER ใช้ 4 ไบท์หรือ 32 บิต แทนจำนวนเต็มจาก -2^{31} ถึง $2^{31}-1$ หรือ $-2,147,483,648$ ถึง 2,147,483,647

รูปที่ 2.4.2 (ค)



INTEGER*8 ใช้ 8 ไบท์ หรือ 64 บิต แทนจำนวนเต็มจาก -2^{63} ถึง $2^{63}-1$ หรือ $-9, 223, 372, 036, 854, 775, 808$ ถึง
9, 223, 372, 036, 854, 775, 807

รูปที่ 2.4.2 (ง)

แบบฝึกหัดบทที่ 2

1. จงแปลงเลขฐานสิบต่อไปนี้เป็นเลขฐานสอง

(1) 0.782

(2) 23.58

(3) 47.1

(4) 5.321

2. จงแปลงเลขฐานสองต่อไปนี้เป็นเลขฐานสิบ

(1) 11011.101101

(2) 1010.10

(3) .010101...

(4) 111.111

3. จงแปลงเลขฐานสิบแปดต่อไปนี้เป็นเลขฐานสิบ

(1) $AAA.111$

(2) $8D2.01B$

(3) $CF3.1234$

(4) $12.12A$

3. จงแปลงเลขฐานสิบแปด $\underbrace{AAA\ldots A}_{k\text{ ตัว}}.\underbrace{111\ldots 1}_{m\text{ ตัว}}$ เป็นเลขฐานสิบ

4. จงแปลงเลขฐานสอง $\underbrace{111\ldots 1}_{k\text{ ตัว}}.\underbrace{010101\ldots 01}_{2m\text{ ตัว}}$ เป็นเลขฐานสิบ

5. จงแปลงเลขฐานสองต่อไปนี้เป็นเลขฐานสิบแปดและฐานแปด

(1) 1101101.10110101

(2) 101101.0110011

(3) 101.101101

(4) 11111111.101101101

6. จงแปลงเป็นเลขฐานสิบแปดต่อไปนี้เป็นเลขฐานสอง

(1) $AA.081$

(2) $2B.2C$

(3) $D301.28E$

(4) $B567$

(5) $FF.F$

7. จงแสดงว่าจำนวนจริง x ซึ่งเมื่อแปลงเป็นเลขฐานสองแล้วได้เลขฐานสองจำนวนจำกัด หรือแบบรู้จบ (finite representation) ก็ต่อเมื่อ จำนวนจริง x อยู่ในรูป $\pm \frac{m}{2^n}$ เมื่อ m และ n เป็นจำนวนเต็มบวก

8. จงแสดงว่าเมื่อแปลงเลขฐานสองใดๆ ซึ่งเป็นเลขฐานสองจำนวนจำกัด เป็นเลขฐานสิบแล้วย่อมได้ เลขฐานสิบจำนวนจำกัดหรือแบบรู้จบ ในทางกลับกันไม่จริง หาตัวอย่างประกอบ

9. จงเขียนเลขฐานสิบต่อไปนี้เป็นเลขอิงดรรชนี

(1) 0.78214

(2) 23.5803

(3) 47.0114

(4) 5.32110

10. จงเขียนเลขฐานสิบต่อไปนี้เป็นเลขอิงดรรชนีบรรทัดฐาน

(1) 0.78214

(2) 23.5803

(3) 47.0114

(4) 41355.32110

บทที่ 3

พหุนามเทย์เลอร์ (Taylor Polynomials)

3.1 การคำนวณฟังก์ชันพหุนาม (Polynomial Function Evaluation)

ฟังก์ชันพหุนามระดับขั้น n (degree n) ในรูปทั่วไป คือ

$$p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n \tag{3.1.1}$$

เมื่อสัมประสิทธิ์ $a_0, a_1, \ldots, a_{n-1}, a_n$ เป็นค่าคงตัว โดยที่ $a_n \neq 0$ และ n เป็นจำนวนเต็มบวก เรียก a_n ว่า สัมประสิทธิ์นำ ถ้า $a_n = 1$ เรียก $p_n(x)$ ว่า พหุนามโมนิก (monic polynomial) ยังสามารถเขียน $p_n(x)$ ในรูปฟังก์ชันพหุนามรอบจุด x_0 ได้

$$p_n(x) = b_0 + b_1(x-x_0) + b_2(x-x_0)^2 + \cdots + b_{n-1}(x-x_0)^{n-1} + b_n(x-x_0)^n \tag{3.1.2}$$

เมื่อสัมประสิทธิ์ $b_0, b_1, \ldots, b_n$ เป็นค่าคงตัว และเรียก x_0 ว่า ศูนย์กลาง ดังนั้น $p_n(x)$ ในสมการ (3.1.2) มีศูนย์กลางที่ 0 เช่น

$$p_3(x) = 2 - 3x + 4x^3$$

เมื่อเขียน $p_3(x)$ รอบจุด 1 จะได้

$$p_3(x) = 3 + 9(x-1) + 12(x-1)^2 + 4(x-1)^3$$

เมื่อกำหนด $p_n(x)$ มาให้ สามารถเขียน $p_n(x)$ ในรูปศูนย์กลางที่ x_0 ได้ โดยคำนวณสัมประสิทธิ์ $b_0, b_1, \ldots, b_n$ ในสมการ (3.1.2) จาก

$$b_k = \frac{p_n^{(k)}(x_0)}{k!}, \quad k = 0, 1, 2, \ldots, n \tag{3.1.3}$$

การคำนวณฟังก์ชันพหุนามสามารถทำได้อย่างมีประสิทธิภาพ โดยเขียนในรูปกำลังซ้อนใน (nested power form) พิจารณาจากกรณีต่อไปนี้

$$p_3(x) = 1 - 5x + 4x^2 - 2x^3$$

ถ้าคำนวณ $p_3(x)$ ในรูปนี้โดยตรงแล้ว จำนวนครั้งในการคูณเป็น $1+2+3=6$ ครั้ง แต่ถ้าเขียน $p_3(x)$ ในรูปกำลังซ้อนใน

$$p_3(x) = 1 + x(-5 + x(4 - 2x))$$

จำนวนครั้งในการคูณเป็น 3 ครั้ง สำหรับ

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1} + a_n x^n \tag{3.1.4}$$

ถ้าคำนวณ $p_n(x)$ ในรูปกำลัง (power form) นี้โดยตรงแล้ว จำนวนครั้งในการคูณของแต่ละพจน์ เช่น พจน์ $a_k x^k$ ต้องใช้การคูณ k ครั้ง ดังนั้น รวมจำนวนครั้งในการคูณของทุก ๆ พจน์ของ $p_n(x)$ ในสมการ (3.1.4) ได้

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{3.1.5}$$

และเมื่อรวมการบวกอีก n ครั้ง จำนวนครั้งในการดำเนินการ หรือเรียกโดยย่อว่า flops (floating-point operations) เป็น

$$\frac{n(n+1)}{2}+n = \frac{n(n+3)}{2} \tag{3.1.6}$$

ถ้าเขียน $p_n(x)$ ในรูปกำลังซ้อนใน

$$p_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x))\ldots)) \tag{3.1.7}$$

จำนวนครั้งในการคูณลดลงเหลือ n ครั้ง และรวมการบวก n ครั้ง ทำให้ได้จำนวน flops ในการคำนวณ $p_n(x)$ ในรูปกำลังซ้อนในสมการ (3.1.7) เป็น $2n$ ดังนั้น สัดส่วนของ flops ในการคำนวณ $p_n(x)$ ในรูปกำลังเทียบกับรูปกำลังซ้อนในเป็น

$$\frac{n(n+3)/2}{2n} = \frac{n+3}{4} \tag{3.1.8}$$

เห็นได้ชัดว่าถ้า n มีค่าใหญ่ เช่น $n=1000$ จำนวน flops ของการคำนวณ $p_n(x)$ ในรูปกำลังเป็น 501500 และรูปกำลังซ้อนในเป็น 2000 หรือ สัดส่วนของ flops ในการคำนวณ $p_n(x)$ ในรูปกำลังเทียบกับรูปกำลังซ้อนในเป็น 250 โดยประมาณ

วิธีการเขียนขั้นตอนการคำนวณ $p_n(x)$ ในรูปกำลังซ้อนใน พิจารณาได้จากการคำนวณกรณี $p_5(x)$ ดังนี้

$$p_5(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

เขียน $p_5(x)$ ในรูปกำลังซ้อนใน

$$p_5(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + a_5 x))))$$

การคำนวณ $p_5(x)$ ในรูปกำลังซ้อนใน จึงเริ่มจากวงเล็บข้างในสุด เช่น ต้องการคำนวณ $p_5(6)$ มีขั้นตอนดังนี้ คือ