บทที่ 2

การกระทำทางคณิตศาสตร์กับเลขฐานสอง (Binary Arithmetic)

2.1 บทนำ

ข้อมูลที่ถูกเก็บอยู่ในคอมพิวเตอร์จะอยู่ในรูปของเลขฐานสอง (Binary Digits) ซึ่งมีหน่วยเรียกเป็นบิท (BIT). ซึ่งในคอมพิวเตอร์นั้นมีหน่วยประมวลผลทางคณิตศาสตร์ที่เรียกว่า ALU (Arithmetic Logic Unit) ซึ่งการประมวลผลทางคณิตศาสตร์ใน ALU ถูกลดให้เหลือเพียงการบวกหรือการลบเท่านั้น เพื่อให้ง่ายในการออกแบบ ถึงแม้ว่าจะเจอกับการประมวลผลที่เป็นการคูณหรือหารก็สามารถให้คำตอบที่ถูกต้องได้โดยจะใช้เทคนิคในการคำนวณที่มีพื้นฐานบนการบวกและการลบเท่านั้น

2.2 การบวกเลขฐานสอง (Adding Binary Numbers)

การบวกเลขฐานสองมีหลักการเหมือนกับการบวกเลขฐานสิบ คือนำตัวตั้งมาบวกกับตัวบวกทีละหลักจากหลักที่มีค่าต่ำสุดไปยังหลักที่มีค่าสูงสุด (LSD ไปยัง MSD สำหรับเลขฐานสิบ) (LSB ไปยัง MSB สำหรับเลขฐานสอง) ในกรณีเลขฐานสิบเมื่อผลลัพธ์จากการบวกที่ได้มีค่ามากกว่า 9 (ในกรณีเลขฐานสองเมื่อค่าที่ได้มากกว่า 1) ก็ให้ทำการลบผลลัพธ์ที่ได้ด้วย 10 (ในกรณีเลขฐานสองต้องลบด้วย 2) แล้วนำเศษที่เหลือไปเป็นผลลัพธ์ของหลักนั้นและทดไปหลักถัดไปด้วยค่า 1 ทำอย่างนี้ไปเรื่อยๆจนครับทุกหลัก ดังในตัวอย่างต่อไปนี้

จงหาผลบวกของ 256 + 178

(ตัวทด)	1	1			
(ตัวตั้ง)	2	5	6		
(ตัวบวก)	1	7	8	+	
(ผลบวก)	4	13	14		=> ผลลัพธ์มากกว่า 10 ต้องลบออก
(ค่าที่ใช้ลบ)		10	10	-	=> เลขฐานสิบต้องลบออกด้วย 10
(ผลลัพธ์)	4	3	4		

256 + 178 = 434

ลองมาดูตัวอย่างในกรณีของการบวกเลขฐานสอง

จงหาผลบวกของ $1011_2 + 1101_2$

(ตัวทด)	1	1	1	1		
(ตัวตั้ง)		1	0	1	1	
(ตัวบวก)		1	1	0	1	+
(ผลบวก)		3	2	2	2	=> ผลลัพธ์มากกว่า 1 ต้องลบออก
(ค่าที่ใช้ลบ)		2	2	2	2	- => เลขฐานสองต้องลบออกด้วย 2
(ผลลัพธ์)	1	1	0	0	0	

$1011_2 + 1101_2 = 11000_2$

ลองมาดูอีกตัวอย่างเพื่อให้เข้าใจยิ่งขึ้น

จงหาผลบวกของ $1011_2 + 1010_2$

(ตัวทด)	1		1			
(ตัวตั้ง)		1	0	1	1	
(ตัวบวก)		1	0	1	0	+
(ผลบวก)		2	1	2	1	=> ผลลัพธ์มากกว่า 1 ต้องลบออก
(ค่าที่ใช้ลบ)		2		2		- => เลขฐานสองต้องลบออกด้วย 2
(ผลลัพธ์)	1	0	1	0	1	

$1011_2 + 1010_2 = 10101_2$

2.3 การลบเลขฐานสอง (Subtracting Binary Numbers)

การลบเลขฐานสองนั้นมีอยู่ 2 วิธี คือวิธีการลบแบบธรรมดาเหมือนกับที่ใช้ลบเลขฐานสิบและอีกวิธีหนึ่งคือการลบแบบใช้วิธีของการทำคอมพลีเมนท์เข้ามาช่วยในการลบ

2.3.1 การลบเลขฐานสองแบบธรรมดา (Subtracting Binary Numbers Using The Conventional Method)

การลบเลขฐานสองมีหลักการเหมือนกับการลบเลขฐานสิบ คือนำตัวตั้งมาลบออกจากตัวลบทีละหลักจากหลักที่มีค่าต่ำสุดไปยังหลักที่มีค่าสูงสุด (LSD ไปยัง MSD สำหรับเลขฐานสิบ) (LSB ไปยัง MSB สำหรับเลขฐานสอง) ในกรณีที่หลักใดๆที่ตัวตั้งมีค่าน้อยกว่าตัวลบก็ให้ยืมมาจากหลักที่มีค่าสูงกว่าซึ่งหลักที่ถูกยืมจะต้องลบออกด้วย 1 และหลักที่ยืมมาจะบวกเพิ่มด้วย 10 (บวกเพิ่มด้วย 2 ในกรณีเลขฐานสอง) แล้วจึงนำไปลบกับตัวลบอีกทีหนึ่ง ทำอย่างนี้ไปจนครบทุกหลัก ดังในตัวอย่างต่อไปนี้

จงหาผลลัพธ์ของ 423 – 198

		1	1		
(ตัวตั้ง)	4	~~2~~	3		=> ตัวตั้งน้อยกว่าตัวลบต้องยืม
(ตัวลบ)	1	9	8	-	

	3	1			
(ตัวตั้ง)	~~4~~	1	$\boxed{13}$		=> นำ 3 ไปบวกกับตัวที่ยืมมา
(ตัวลบ)	1	9	8	-	(ตัวที่ยืมมามีค่าเท่ากับ 10)
			5		

(ตัวตั้ง)	3	$\boxed{11}$	13		=> นำ 1 ไปบวกกับตัวที่ยืมมา
(ตัวลบ)	1	9	8	-	(ตัวที่ยืมมามีค่าเท่ากับ 10)
(ผลลัพธ์)	2	2	5		

423 – 198 = 225

ลองมาดูตัวอย่างในกรณีของการลบเลขฐานสอง

จงหาผลลัพธ์ของ 110_2 - 011_2

		0	1		
(ตัวตั้ง)	1	~~1~~	0		=> ตัวตั้งน้อยกว่าตัวลบต้องยืม
(ตัวลบ)	0	1	1	-	

	0	1			
(ตัวตั้ง)	~~1~~	0	$\boxed{2}$		=> นำ 0 ไปบวกกับตัวที่ยืมมา
(ตัวลบ)	0	1	1	-	(ตัวที่ยืมมามีค่าเท่ากับ 2)
			1		

(ตัวตั้ง)	0	$\boxed{2}$	2		=> นำ 0 ไปบวกกับตัวที่ยืมมา
(ตัวลบ)	0	1	1	-	(ตัวที่ยืมมามีค่าเท่ากับ 2)
	0	1	1		

$110_2 – 011_2 = 011_2$

จงหาผลลัพธ์ของ 100_2 - 011_2

	MSB		LSB	
		1	1	
(ตัวตั้ง)	~~1~~	0	0	=> ตัวตั้งน้อยกว่าตัวลบต้องยืม
(ตัวลบ)	0	1	1	-

(เนื่องจากถูก LSBยืมไป
เดิมยืมมาจาก MSB มาบวกกับ 0
เท่ากับ 2 แต่ถูก LSB ยืมไปอีก
ทอดหนึ่งจึงเหลือมีค่าเท่ากับ 1)

(ตัวตั้ง)	0	$\boxed{1}$ →	$\boxed{2}$	=> นำ 0 ไปบวกกับตัวที่ยืมมา
(ตัวลบ)	0	1	1	- (ตัวที่ยืมมามีค่าเท่ากับ 2)
	0	0	1	

$100_2 - 011_2 = 001_2$

2.3.2 การลบเลขฐานสองโดยใช้การคอมพลีเมนท์ (Subtracting Binary Numbers Using The Complement Method)

การลบเลขฐานสองโดยใช้การคอมพลีเมนท์นั้นมีอยู่ 2 แบบคือ One’s Complement และ Two’s Complement

2.3.2.1 One’s Complement

การลบเลขฐานสองโดยใช้ One’s Complement มีหลักการดังนี้ ให้ทำการคอมพลีเมนท์แบบ One’s Complement กับตัวที่นำมาลบ คือให้ทำการกลับบิทที่เป็น ‘1’ ให้เป็น ‘0’ และ กลับบิทที่เป็น ‘0’ ให้เป็น ‘1’ ทำอย่างนี้ให้ครบทุกบิท จากนั้นนำตัวเลขที่ได้จากการทำคอมพลีเมนท์ มาบวกกับตัวตั้ง ในกรณีที่การบวกในบิท MSB มีตัวทดเกิดขึ้นให้นำตัวทดที่ได้ไปบวกกับบิท LSB ของผลลัพธ์อีกทีหนึ่งก็จะได้ผลลัพธ์สุดท้ายออกมา ดังแสดงในตัวอย่างต่อไปนี้

จงหาผลลัพธ์ของ 110101_2 - 101011_2 โดยใช้วิธีการ One's Complement

นำตัวลบ มาทำ One's Complement

		1	0	1	0	1	1
(One's Complement)		0	1	0	1	0	0
(ตัวทด)	1	1		1			
(ตัวตั้ง)		1	1	0	1	0	1
(ตัวบวก)		0	1	0	1	0	0 +
(ผลบวก)		0	0	1	0	0	1
							1 +
(ผลลัพธ์)		0	0	1	0	1	0

110101_2 - 101011_2 = 001010_2

2.3.2.2 Two's Complement

การลบเลขฐานสองโดยใช้ Two's Complement มีหลักการดังนี้ ให้ทำการคอมพลีเมนท์แบบ One's Complement กับตัวที่นำมาลบ คือให้ทำการกลับบิทที่เป็น '1' ให้เป็น '0' และ กลับบิทที่เป็น '0' ให้เป็น '1' ทำอย่างนี้ให้ครบทุกบิท จากนั้นนำตัวเลขที่ได้จากการทำคอมพลีเมนท์ มาบวกกับ 1 แล้วจึงนำมาบวกกับตัวตั้งอีกทีหนึ่ง ในกรณีที่การบวกในบิท MSB มีตัวทดเกิดขึ้นก็ไม่ต้องนำมาคิด ตัวอย่างต่อไปนี้

จงหาผลลัพธ์ของ 110101_2 - 101011_2 โดยใช้วิธีการ Two's Complement

นำตัวลบ มาทำ Two's Complement

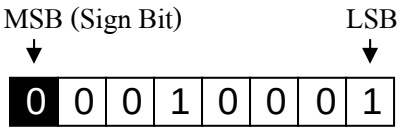
		1	0	1	0	1	1
(One's Complement)		0	1	0	1	0	0
							1 +
(Two's Complement)		0	1	0	1	0	1
		ไม่ต้องนำมาคิด					
(ตัวทด)	1	1		1		1	
(ตัวตั้ง)		1	1	0	1	0	1
(ตัวบวก)		0	1	0	1	0	1 +
(ผลลัพธ์)		0	0	1	0	1	0

110101_2 - 101011_2 = 001010_2

2.4 Signed Numbers

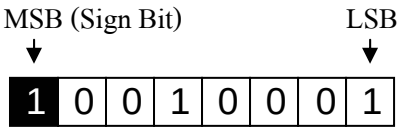
ในระบบตัวเลขนั้นมีทั้งตัวเลขที่เป็นค่าบวก (Positive) และตัวเลขที่เป็นค่าลบ (Negative) ซึ่งจะใช้เครื่องหมายบวก (+) นำหน้าตัวเลขที่เป็นค่าบวก และใช้เครื่องหมายลบ (-) นำหน้าตัวเลขที่เป็นค่าลบ ซึ่งเราเรียกเลขพวกนี้ว่า Signed Numbers

ตัวเลขที่ถูกเก็บอยู่ในคอมพิวเตอร์นั้น ตำแหน่งของ MSB ในรีจิสเตอร์นั้นจะถูกสงวนไว้สำหรับบิทที่เรียกว่า Sign Bit , ซึ่งจะใช้ในการแสดงเครื่องหมายของตัวเลขที่ถูกเก็บอยู่ในรีจิสเตอร์ว่าเป็นบวกหรือลบนั่นเอง ถ้าตัวเลขในตำแหน่ง MSB เป็น 0 หมายความว่าตัวเลขที่ถูกเก็บอยู่ในรีจิสเตอร์นั้นมีค่าเป็นบวก ในทางกลับกันถ้าตัวเลขในตำแหน่ง MSB เป็น 1 หมายความว่าตัวเลขที่ถูกเก็บอยู่ในรีจิสเตอร์มีค่าเป็นลบ ดังแสดงในตัวอย่างต่อไปนี้



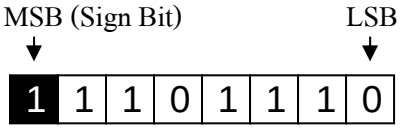
รูปที่ 2-1

จากรูปที่ 2-1 แสดงตัวอย่างของรีจิสเตอร์ขนาด 8 bit ที่เก็บตัวเลขซึ่งมีค่าเป็น 17 เนื่องจาก Sing Bit มีค่าเป็น 0



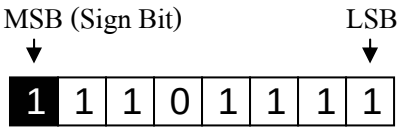
รูปที่ 2-2

จากรูปที่ 2-2 แสดงตัวอย่างของรีจิสเตอร์ขนาด 8 bit ที่เก็บตัวเลขซึ่งมีค่าเป็น -17 เนื่องจาก Sing Bit มีค่าเป็น 1



รูปที่ 2-3

จากรูปที่ 2-3 แสดงตัวอย่างของรีจิสเตอร์ขนาด 8 bit ที่เก็บตัวเลขซึ่งมีค่าเป็น -17 ซึ่งอยู่ในรูปของ One’s Complement



รูปที่ 2-4

จากรูปที่ 2-4 แสดงตัวอย่างของรีจิสเตอร์ขนาด 8 bit ที่เก็บตัวเลขซึ่งมีค่าเป็น -17 ซึ่งอยู่ในรูปของ Two’s Complement

2.5 การบวกและการลบเลขที่มีเครื่องหมาย (Adding and Subtracting Singed Numbers)

กฎในการบวกและลบเลขที่มีเครื่องหมายนั้น มีหลักการเหมือนกับการบวกและลบเลขฐานสิบที่มีเครื่องหมายโดยทั่วไป ดังนี้คือ

1. การบวกเลขที่มีเครื่องหมายเหมือนกัน ผลลัพธ์ที่ได้ก็จะมีเครื่องหมายเหมือนเดิม เช่น การบวกเลขที่มีเครื่องหมายบวกเหมือนกันผลลัพธ์ที่ได้ก็จะเป็นเลขบวก และการบวกเลขที่มีเครื่องหมายลบเหมือนกันก็จะได้ผลลัพธ์เป็นเลขลบเช่นกัน
2. การบวกเลขที่มีเครื่องหมายต่างกัน ผลลัพธ์ที่ได้ก็จะมีเครื่องหมายเหมือนกับเลขที่มีค่ามากกว่า (พิจารณาที่ค่าของตัวเลขไม่ต้องพิจารณาเครื่องหมาย) เช่นตัวตั้งเป็นเลขลบและตัวบวกเป็นเลขบวกซึ่งมีค่ามากกว่าตัวตั้งผลลัพธ์ที่ได้ก็จะมีค่าเป็นเลขบวก
3. การลบเลขที่มีเครื่องหมาย ให้ทำเปลี่ยนเครื่องหมายของตัวลบเป็นตรงกันข้ามกับของเดิม เช่นถ้าตัวลบเป็นเลขบวกก็ให้เปลี่ยนเป็นเลขลบ และถ้าตัวตั้งเป็นเลขลบก็ให้เปลี่ยนเป็นเลขบวก แล้วทำการบวกเลขตามกฎในข้อที่ 1 และ 2

<u>หมายเหตุ</u> การบวกเลขที่มีเครื่องหมายในรูปของเลขฐานสองจำเป็นต้องทำ Two's Complement กับเลขที่มีเครื่องหมายเป็นลบ ทั้งที่เป็นตัวตั้งและตัวบวก ถ้าผลลัพธ์ที่ได้เป็นเลขลบ จะถูกแสดงอยู่ในรูปของ Two's Complement

จงหาผลลัพธ์ของ +7 บวกด้วย +10 ในรูปของเลขฐานสอง

+7 แปลงเป็นเลขฐานสอง => 00000111_2

+10 แปลงเป็นเลขฐานสอง => 00001010_2

0	0	0	0	0	1	1	1	
0	0	0	0	1	0	1	0	+
0	0	0	1	0	0	0	1	

+7 บวกด้วย +10 = +17 (00010001_2)

จงหาผลลัพธ์ของ -5 บวกด้วย -14 ในรูปของเลขฐานสอง

-5 ทำเป็น Two's Complement => 11111011_2

-14 ทำเป็น Two's Complement => 11110010_2

1	1	1	1	1	0	1	1	
1	1	1	1	0	0	1	0	+
1	1	1	0	1	1	0	1	

-5 บวกด้วย -14 = -19 (11101101_2) (ผลลัพธ์อยู่ในรูปของ Two's Complement)

จงหาผลลัพธ์ของ +8 บวกด้วย -11 ในรูปของเลขฐานสอง

+8 แปลงเป็นเลขฐานสอง => 00001000_2

-11 ทำเป็น Two's Complement => 11110101_2

0	0	0	0	1	0	0	0	
1	1	1	1	0	1	0	1	+
1	1	1	1	1	1	0	1	

+8 บวกด้วย -11 = -3 (11111101_2) (ผลลัพธ์อยู่ในรูปของ Two's Complement)

จงหาผลลัพธ์ของ +12 บวกด้วย -6 ในรูปของเลขฐานสอง

+12 แปลงเป็นเลขฐานสอง => 00001100_2

-6 ทำเป็น Two's Complement => 11111010_2

0	0	0	0	1	1	0	0	
1	1	1	1	1	0	1	0	+
0	0	0	0	0	1	1	0	

+12 บวกด้วย -6 = +6 (00000110_2)

จงหาผลลัพธ์ของ +15 ลบด้วย +5 ในรูปของเลขฐานสอง

+15 แปลงเป็นเลขฐานสอง => 00001111_2

+5 เปลี่ยนเครื่องหมายของตัวลบ => -5 (ตามกฎข้อที่ 3)

-5 ทำเป็น Two's Complement => 11111011_2

0	0	0	0	1	1	1	1	
1	1	1	1	1	0	1	1	+
0	0	0	0	1	0	1	0	

+15 ลบด้วย +5 = +10 (00001010_2)

จงหาผลลัพธ์ของ -14 ลบด้วย -3 ในรูปของเลขฐานสอง

-14 ทำเป็น Two's Complement => 11110010_2

-3 เปลี่ยนเครื่องหมายของตัวลบ => +3 (ตามกฎข้อที่ 3)

+3 แปลงเป็นเลขฐานสอง => 00000011_2

1	1	1	1	0	0	1	0	
0	0	0	0	0	0	1	1	+
1	1	1	1	0	1	0	1	

-14 ลบด้วย -3 = -11 (11110101_2) (ผลลัพธ์อยู่ในรูปของ Two's Complement)

2.6 Overflow

ความหมายของ Overflow คือ ล้น เหมือนกับที่เรารินนำใส่แก้วแล้วปริมาณของน้ำที่เราใส่มีมากกว่าความจุของแก้วก็จะเกิดการล้นขึ้น ในระบบตัวเลขก็เหมือนกันถ้าผลลัพธ์ที่ได้จากการกระทำทางคณิตศาสตร์นั้นมีค่ามากเกินกว่าที่จะเก็บเอาไว้ได้ก็จะเกิดการล้นเช่นกัน ยกตัวอย่างเช่น ในรีจิสเตอร์ขนาด 8 bit ตัวเลขที่มีเครื่องหมายเป็นบวกที่สามารถเก็บได้สูงสุดคือ 01111111_2(+127)
และตัวเลขที่มีเครื่องหมายเป็นลบที่สามารถเก็บได้มีค่าสูงสุดคือ 10000000_2(-128) ดังนั้นถ้าผลลัพธ์ที่ได้จากการกระทำทางคณิตศาสตร์นั้นมีค่ามากกว่าค่าดังกล่าวแล้วจะทำให้เกิด Overflow ขึ้นนั่นเอง

การเกิด Overflow เกิดจากข้อผิดพลาดซึ่งสามารถตรวจพบได้ดังต่อไปนี้

1. มีการทดเลข (Carry) ไปยังตำแหน่งของบิทเครื่องหมาย (Sign Bit) โดยที่ไม่มี Carry-Out ออกจากบิทดังกล่าว
2. ไม่มีการทดเลข (Carry) ไปยังตำแหน่งของบิทเครื่องหมาย (Sign Bit) โดยที่มี Carry-Out ออกจากบิทดังกล่าว

2.7 Sixteen's and Eight's Complement Method's

ในการบวกหรือลบเลขฐานสองที่มีเครื่องหมายซึ่งมีจำนวนบิทมากๆ นั้นทำให้ไม่มีความสะดวกสักเท่าไหร่ ดังนั้นจึงได้มีการนำ Sixteen's Complement มาใช้เพื่อแก้ปัญหาดังกล่าว การทำ Sixteen's Complement ได้จากการทำ Fifteen's Complement บวกด้วยหนึ่ง มีหลักการเหมือนกับการทำ Two's Complement จาก One's Complement โดยที่การทำ Fifteen's Complement นั้นได้จากการนำเลขฐานสิบหกมาแยกตัวเลขแต่ละหลักออกจากกันแล้วแทนแต่ละหลักที่แยกออกมาด้วยเลขฐานสองขนาด 4 บิทแล้วจึงทำ One's Complement กับเลขฐานสองนั้นจากนั้นนำผลลัพธ์ที่ได้แปลงกลับไปเป็นเลขฐานสิบหกอีกทีหนึ่งโดยทำทีละ 4 บิท จะได้ Fifteen's Complement แล้วจึงนำไปบวกกับหนึ่ง ก็จะได้ Sixteen's Complement

การทำ Eight's Complement ก็มีหลักการคล้ายกับการทำ Sixteen's Complement เพียงแต่ Eight's Complement จะได้มาจากการนำ Seven's Complement มาบวกด้วยหนึ่ง โดยที่เราจะนำเลขฐานแปดและเมื่อเราแยกเลขฐานแปดแต่ละหลักออกจากกันก็จะแทนที่เลขฐานแปดเหล่านั้นด้วยเลขฐานสองจำนวน 3 บิทแล้วจึงทำ One's Complement กับเลขฐานสองที่ได้ เมื่อได้ผลลัพธ์จากการทำ One's Complement แล้วจึงทำการแปลงเลขฐานสองที่ได้กลับไปเป็นเลขฐานแปดอีกทีหนึ่งโดยทำทีละ 3 บิท ซึ่งผลลัพธ์ที่ได้คือ Seven's Complement จากนั้นจึงนำไปทำเป็น Eight's Complement โดยการบวกเพิ่มเข้าไปอีกหนึ่ง

<u>หมายเหตุ</u> การทำ Sixteen's Complement และ Eight's Complement ใช้กับเลขที่มีเครื่องหมายเป็นลบซึ่งทำให้สะดวกในการคำนวณ และถ้าผลลัพธ์จากการคำนวณอยู่ในรูปของ Sixteen's Complement และ Eight's Complement หมายถึงคำตอบเป็นเลขลบ

จงหาผลลัพธ์ของ –D76 บวกกับ +52A โดยใช้วิธีของ Sixteen's Complement

นำ –D76 มาทำ Sixteen's Complement

(-)	D	7	6	
	↓	↓	↓	
	1101	0111	0110	
	↓	↓	↓	
	0010	1000	1001	
	↓	↓	↓	
	2	8	9	=> Fifteen's Complement
			+ 1	
	2	8	A	=> Sixteen's Complement
	5	2	+ A	
	7	B	4	

–D76 บวกกับ +52A = $7B4_{16}$ (ผลลัพธ์อยู่ในรูปของ Sixteen's Complement)

จงหาผลลัพธ์ของ +456 ลบกับ +357 โดยใช้วิธีของ Eight's Complement

เปลี่ยนเครื่องหมายของตัวลบจาก +357 ไปเป็น -357 แล้วจึงทำ Eight's Complement

(-)	3	5	7	
	↓	↓	↓	
	011	101	111	
	↓	↓	↓	
	100	010	000	
	↓	↓	↓	
	4	2	0	=> Seven's Complement
			+ 1	
	4	2	1	=> Eight's Complement
	4	5	6	
	4	2	+ 1	
	0	7	7	

+456 ลบกับ +357 = 077_8

2.8 การบวกและการลบเลขบีซีดี (BCD Addition and Subtraction)

ถึงแม้ว่าคอมพิวเตอร์ส่วนใหญ่จะใช้เลขฐานสองในการกระทำทางคณิตศาสตร์ แต่ก็ยังมีคอมพิวเตอร์บางชนิดที่ใช้เลขบีซีดีในการประมวลผลตัวอย่างเช่นเครื่องคิดเลขเป็นต้น แต่การใช้เลขบีซีดีนั้นมีข้อเสียคือ เลขบีซีดีจะใช้เลขฐานสองจำนวน 4 บิทในการแทนตัวเลขหนึ่งหลักซึ่งค่าที่เป็นไปได้คือ 0 ถึง 9 ซึ่งต่างกับเลขฐานสองที่มีขนาด 4 บิทเพราะค่าที่เป็นไปได้คือ 0 ถึง 15

การบวกเลขบีซีดีนั้นจะใช้หลักการคล้ายกับการบวกเลขฐานสอง แต่ก็มีข้อแตกต่างดังที่ได้กล่าวไว้ในข้างต้น ดังนั้นเมื่อทำการบวกเลขบีซีดี (ในแต่ละหลัก) แล้วค่าที่ได้เกินกว่า 9 ผลลัพธ์ที่ได้ก็จะผิด ดังนั้นจึงต้องมีการนำผลลัพธ์ที่ผิดนั้นมาบวกด้วย 6 ซึ่งเป็นเลขบีซีดีก็จะทำให้ได้ผลลัพธ์ที่ถูกต้อง

จงหาผลลัพธ์ของ $0101_{BCD} + 0111_{BCD}$ ซึ่งเป็นเลขบีซีดี

5	<=	0	1	0	1		(BCD)
7	<=	0	1	1	1	+	(BCD)
ผลลัพธ์ที่ได้ผิด	=>	1	1	0	0		(BCD)
ต้องบวกด้วย 6	=>	0	1	1	0	+	(BCD)
0 0 0 1		0	0	1	0		(BCD)
1		2					

$0101_{BCD} + 0111_{BCD} = 0001\ 0010_{BCD}$

จงหาผลลัพธ์ของ 138 + 259 ในรูปของเลขบีซีดี

หลักร้อยหลักสิบ	หลักหน่วย		
	1		(ตัวทด)
0001	0011	1000	
0010	0101	1001	+
		0001	(ผลลัพธ์ที่ได้มากกว่า 9 ถือว่าผิด)
		0110	+ (ต้องนำไปบวกด้วย 6)
		0111	(ผลลัพธ์ที่ถูกต้องของหลักหน่วย)
	1001		(ผลลัพธ์ที่ถูกต้องของหลักสิบ)
0011			(ผลลัพธ์ที่ถูกต้องของหลักร้อย)

$138 + 259 = 397\ (0011\ 1001\ 0111_{BCD})$

ในการลบเลขบีซีดีนั้นจำเป็นต้องมีการทำ Nine's Complement ให้กับตัวลบ แล้วจึงนำไปบวกกับตัวตั้ง ซึ่งคล้ายกับการลบเลขฐานสองที่ต้องทำ Two's Complement นั่นเอง แต่การลบเลขบีซีดีด้วยวิธีการทำ Nine's Complement นี้ต้องพิจารณาตัวทดที่อยู่เกิดจากหลัก MSB ด้วย ซึ่งถ้าตัวทดที่ได้จาก MSB คือ 0 แสดงว่าไม่มีการทด หมายความว่าผลลัพธ์ที่เกิดขึ้นเป็นเลขลบให้นำผลลัพธ์ที่ได้ไปทำ Nine's Complement อีกทีหนึ่งก็จะได้คำตอบที่ถูกต้อง แต่ถ้าตัวทดจาก MSB คือ 1 แสดงว่าผลลัพธ์เป็นเลขบวกให้นำตัวทดดังกล่าวไปบวกกับ LSB อีกทีหนึ่ง

DECIMAL	BCD	Nine's Complement of BCD
0	0000	1001
1	0001	1000
2	0010	0111
3	0011	0110
4	0100	0101
5	0101	0100
6	0110	0011
7	0111	0010
8	1000	0001
9	1001	0000

ตารางที่ 2-1 แสดงการเปรียบเทียบระหว่างเลขฐานสิบ เลขบีซีดี และ Nine's Complement ของเลขบีซีดี

จงหาผลลัพธ์ของ 888 – 564 ในรูปของเลขบีซีดี

564	=>	0101	0110	0100	(BCD)
		0100	0011	0101	(Nine's Complement of BCD)
หลักร้อย		หลักสิบ		หลักหน่วย	
1		1			(ตัวทด)
1000		1000		1000	
0100		0011		0101 +	
				1101	(ผลลัพธ์ที่ได้มากกว่า 9 ถือว่าผิด)
				0110 +	(ต้องนำไปบวกด้วย 6)
				0011	
(ตัวทด)		1100			(ผลลัพธ์ที่ได้มากกว่า 9 ถือว่าผิด)
		0110 +			(ต้องนำไปบวกด้วย 6)
1		0010			(ผลลัพธ์ที่ถูกต้องของหลักสิบ)
1101					(ผลลัพธ์ที่ได้มากกว่า 9 ถือว่าผิด)
0110 +					(ต้องนำไปบวกด้วย 6)
0011					(ผลลัพธ์ที่ถูกต้องของหลักร้อย)
				⊕	(นำตัวทดไปบวกกับ LSB)
				0100	(ผลลัพธ์ที่ถูกต้องของหลักหน่วย)

888 – 564 = 324 (0011 0010 0100_{BCD})

จงหาผลลัพธ์ของ 342 – 436 ในรูปของเลขบีซีดี

436	=>	0100	0011	0110	(BCD)
		0101	0110	0011	(Nine's Complement of BCD)
หลักร้อยหลักสิบ			หลักหน่วย		
1					(ตัวทด)
0011		0100		0010	
0101		0110		0011	+
				0101	
		1010			(ผลลัพธ์ที่ได้มากกว่า 9 ถือว่าผิด)
		0110	+		(ต้องนำไปบวกด้วย 6)
		0000			
1001					(ไม่มีตัวทดเกิดขึ้นในตำแหน่ง MSB แสดงว่าผลลัพธ์ติดลบ ต้องทำ Nine's Complement)
(-) 0000		1001		0100	(ผลลัพธ์ที่ถูกต้อง)

342 – 436 = -94 (-1001 0100_{BCD})