บทที่ 7

การแทนข้อมูล

7.1 ระบบตัวเลข

7.2 การแปลงเลขฐาน

7.3 การคำนวณของเลขฐานสอง

7.4 การแทนเลขฐานที่มีเครื่องหมาย

7.5 ระบบตัวเลขกับรหัสข้อมูล

7.6 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

วัตถุประสงค์

- การแทนข้อมูลที่เป็นตัวเลข
- ระบบเลขฐาน การแปลงระหว่างเลขฐานต่างๆ
- การคำนวณของเลขฐานสอง
- การแทนเลขฐานสองที่มีเครื่องหมาย
- การแทนข้อมูลที่เป็นตัวอักขระ

ข้อมูลต่างๆ ในคอมพิวเตอร์จะถูกแทนค่าด้วยระบบเลขฐานสอง ซึ่งประกอบด้วยเลข 0 และ 1 ทั้งนี้เนื่องจากคอมพิวเตอร์ประกอบด้วยวงจรอิเล็กทรอนิกส์ต่างๆ จำนวนมากที่ออกแบบมาให้ทำงานในรูปแบบของสัญญาณไฟฟ้า (Switching) ซึ่งสามารถทำงานได้เพียง 2 สถานะ คือ เปิด (On) และ ปิด (Off) จึงเหมาะแก่การนำระบบเลขฐานสองมาใช้แทนค่าต่างๆ ภายในคอมพิวเตอร์ โดยให้ 1 แทนสถานะเปิด และ 0 แทนสถานะปิดนั่นเอง ในขณะที่เราจะคุ้นเคยกับเลขฐานสิบซึ่งใช้กันทั่วไปในชีวิตประจำวันมากกว่า นอกจากเลขฐานสองแล้ว ระบบเลขฐานอื่นๆ เช่น ฐาน 8 หรือ ฐาน 16 ก็อาจนำมาใช้ในการแทนข้อมูลได้เช่นกัน ดังนั้นเราจึงควรรู้จักระบบเลขฐาน การแปลงเลขฐานต่างๆ การคำนวณค่าโดยการใช้เลขฐานสอง รวมถึงการแทนค่าข้อมูลอักขระด้วยรหัสต่างๆ

7.1 ระบบตัวเลข

ระบบเลขฐาน r ใดจะมีเลขพื้นฐาน r ตัว คือ 0,1,2,3,...,r-1 สามารถเขียนอยู่ในรูปสัญลักษณ์แทนดังต่อไปนี้

$$N = b_n b_{n-1} b_{n-2} \ldots b_2 b_1\ b_0.\ b_{-1} b_{-2} \ldots b_m$$

โดย b เป็นเลขจำนวนเต็มซึ่งเป็นเลขพื้นฐานของระบบเลขฐาน r นั้นๆ เลขส่วนที่อยู่หน้าจุดทศนิยามเรียกว่า จำนวนเต็ม (Integer part) ส่วนเลขหลังจุดทศนิยมเรียกว่า เศษส่วน (Fraction part) เลขฐาน ระบบเลขฐานที่ใช้ในทางคอมพิวเตอร์ ประกอบด้วย

เลขฐานสอง (Binary) มีเลขพื้นฐาน 2 ตัว คือ 0,1

เลขฐานแปด (Octal) มีเลขพื้นฐาน 8 ตัว คือ 0,1,2,3,4,5,6,7

เลขฐานสิบ (Decimal) มีเลขพื้นฐาน 2 ตัว คือ 0,1,2,3,4,5,6,7,8,9

เลขฐานสิบหก (Hexadecimal) มีเลขพื้นฐาน 2 ตัว คือ 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

7.2 การแปลงเลขฐาน

การแปลงเลขฐาน r เป็นเลขฐานสิบ จะทำโดยนำผลรวมของการคูณค่าน้ำหนัก (weight) กับเลขฐาน r ในแต่ละตำแหน่ง โดยน้ำหนักแต่ละตำแหน่งแสดงในตาราง 7.1 และ 7.2

ตำแหน่ง	b_n	b_{n-1}	b_{n-2}	...	b_2	b_1	b_0
ฐาน 2	2^n	2^{n-1}	2^{n-2}		2^2	2^1	2^0
ฐาน 8	8^n	8^{n-1}	8^{n-2}		8^2	8^1	8^0
ฐาน 10	10^n	10^{n-1}	10^{n-2}		10^2	10^1	10^0
ฐาน 16	16^n	16^{n-1}	16^{n-1}		16^2	16^1	16^0

ตารางที่ 7.1 ค่าน้ำหนักประจำตำแหน่งของเลขจำนวนเต็มฐาน r

ตำแหน่ง	b_{-1}	b_{-2}	b_{-3}	...	$b_{-(m-1)}$	b_{-m}
ฐาน 2	2^{-1}	2^{-2}	2^{-3}		$2^{-(m-1)}$	2^{-m}
ฐาน 8	8^{-1}	8^{-2}	8^{-3}		$8^{-(m-1)}$	8^{-m}
ฐาน 10	10^{-1}	10^{-2}	10^{-3}		$10^{-(m-1)}$	10^{-m}
ฐาน 16	16^{-1}	16^{-2}	16^{-3}		$16^{-(m-1)}$	16^{-m}

ตารางที่ 7.2 ค่าน้ำหนักประจำตำแหน่งของเลขเศษส่วนฐาน r

การแปลงเลขฐาน r เป็นฐาน 10

ดังนั้น เมื่อต้องการแปลงจากเลขฐาน r ใดเป็นเลขฐาน 10 ก็สามารถทำได้โดยการหาผลรวมนี้ ดังตัวอย่างสมการในการแปลงเลขฐาน 2 เป็นฐาน 10 ต่อไปนี้

$$N = b_n2^n + b_{n-1}2^{n-1} + \ldots + b_22^2 + b_12^1 + b_02^0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \ldots + b_{-m}2^{-m}$$

ตัวอย่างที่ 7.1 จงแปลงเลขฐาน $(1110)_2$ เป็นเลขฐาน 10

วิธีทำ $1110 = 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0$

$= 8 + 4 + 2 + 0$

$= 14$

ตัวอย่างที่ 7.2 จงแปลงเลขฐาน $(.1101)_2$ เป็นเลขฐาน 10

วิธีทำ $.1101 = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$

$= 1/2 + 1/4 + 0 + 1/16$

$= 0.8125$

ตัวอย่างที่ 7.3 จงแปลงเลขฐาน $(1234)_8$ เป็นเลขฐาน 10

วิธีทำ $1234 = 1*8^3 + 2*8^2 + 3*8^1 + 4*8^0$

$= 512 + 128 + 24 + 4$

$= 668$

ตัวอย่างที่ 7.4 จงแปลงเลขฐาน $(.432)_8$ เป็นเลขฐาน 10

วิธีทำ $.432 = 4*8^{-1} + 3*8^{-2} + 2*8^{-3}$

$= 4/8 + 3/64 + 2/512$

$= 0.5507812$

ตัวอย่างที่ 7.5 จงแปลงเลขฐาน $(3B3D)_{16}$ เป็นเลขฐาน 10

วิธีทำ $3B3D = 3*16^3 + 11*16^2 + 3*16^1 + 13*16^0$

$= 12288 + 2816 + 48 + 13$

$= 15165$

ตัวอย่างที่ 7.6 จงแปลงเลขฐาน $(.2B)_{16}$ เป็นเลขฐาน 10

วิธีทำ

$$.2B = 2*16^{-1} + 11*16^{-2}$$
$$= 2/16 + 11/256$$
$$= 0.168$$

การแปลงเลขฐาน 10 เป็นฐาน r

เมื่อต้องการแปลงจากเลขฐาน 10 เป็นเลขฐาน r ใดๆ ในส่วนของจำนวนเต็ม สามารถทำได้โดยการหารด้วย r จนกว่าจะได้ผลหารสุดท้ายเป็น 0 ผลลัพธ์ที่ได้ คือ เศษที่ได้จากการหารด้วยค่า r โดยเรียงจากล่างขึ้นบน ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 7.6 จงแปลงเลขฐาน $(25)_{10}$ เป็นเลขฐาน 2

วิธีทำ

		เศษ
2	) 25	
2	) 12	1
2	) 6	0
2	) 3	0
2	) 1	1
	0	1

$(25)_{10} = (11001)_2$

ในส่วนของเลขทศนิยม ทำได้โดยการคูณเลขทศนิยมฐาน 10 ด้วยค่า r จนกว่าจะได้ผลคูณสุดท้ายเป็นค่า 0 หรือจนครบตำแหน่งที่ต้องการ ผลลัพธ์ที่ได้ คือ ตัวทดที่ได้การคูณโดยเรียงจากบนลงล่าง ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 7.7 จงแปลงเลขฐาน $(.375)_{10}$ เป็นเลขฐาน 2

วิธีทำ

	.375 *
	2
0	.750 *
	2
1	.500 *
	2
1	.000

$(.375)_{10} = (.011)_2$

7.3 การคำนวณของเลขฐานสอง

การบวกและลบเลขฐานสองจะคล้ายกับการบวกลบเลขฐานสิบ แต่จะง่ายกว่าเพราะมี 4 กรณีเท่านั้น

การบวกเลขฐานสอง

		carry
0 + 0	= 0	0
0 + 1	= 1	0
1 + 0	= 1	0
1 + 1	= 0	1

จะเห็นว่า 3 กรณีแรกจะเหมือนกับการบวกเลขฐานสิบ แต่กรณีสุดท้ายจะแตกต่างกัน โดย 1+1 จะมีค่าเท่ากับ 2 ในฐานสิบ เมื่อเขียนเป็นฐานสองคือ $(10)_2$ ดังนั้น 1+1 = 10 คือได้ผลบวกเป็น 0 และมีตัวทดคือ 1 เพื่อนำไปรวมกับบิตที่อยู่ทางซ้ายมือ นั่นคือ carry bit จะมีค่าเป็น 1

ตัวอย่างที่ 7.8 จงหาผลลัพธ์ของ $(11100)_2 + (11001)_2$

วิธีทำ

```
 1 1          ←  (ตัวทด)
   1 1 1 0 0 +
   1 1 0 0 1
 1 1 0 1 0 1
```

การลบเลขฐานสอง

		carry
0 - 0	= 0	0
0 - 1	= 1	1
1 - 0	= 0	0
1 - 1	= 0	0

จะเห็นว่ากรณีที่แตกต่างจากการลบเลขฐานสิบ คือ 0-1 มีค่าเท่ากับ 1 เนื่องจากบิตตัวตั้งลบไม่ได้ต้องไปยืมบิตที่อยู่ถัดไปมา $(10)_2$ หรือ $(2)_{10}$ ทำให้บิตที่ถูกยืมเหลือ 0 และบิตที่ยืมมามีค่า 2 ในฐานสิบ เมื่อเขียนในรูปเลขฐานสอง คือ 10 - 1 = 1 หรือในรูปเลขฐาน 2 คือ 2 - 1 = 1 ดังนั้นจะได้ผลลบเป็น 1 และ carry bit จะมีค่าเป็น 1 เนื่องจากมีการยืมบิตถัดไปที่อยู่ทางขวามือ

ตัวอย่างที่ 7.9 จงหาผลลัพธ์ของ $(10101)_2 - (10011)_2$

วิธีทำ

```
     1      ←  (ยืม)
 1 0 1 0 1 +
 1 0 0 1 1
 0 0 0 1 0
```

7.4 การแทนเลขฐานที่มีเครื่องหมาย

เลขจำนวนเต็มจะประกอบด้วยเลขจำนวนเต็มบวก เลขจำนวนลบ หรือเลขศูนย์ ในเลขฐานสิบมักจะใช้การเขียนเครื่องหมายลบไว้หน้าตัวเลขเพื่อบอกว่าเป็นจำนวนเต็มลบ ในขณะที่เลขจำนวนเต็มบวกอาจจะมีการเขียนเครื่องหมายบวกไว้ข้างหน้าหรืออาจไม่เขียนก็ได้ เมื่อแปลงเลขฐานสิบเหล่านี้เป็นเลขฐานสอง ก็ต้องระบุเครื่องหมายด้วยเช่นกัน เช่น +1 เป็น +001 หรือ -1 เป็น -001 แต่การใช้เครื่องบวกหรือลบนำหน้าเลขฐานสองนี้ทำให้ไม่สามารถนำไปประมวลผลได้ จึงต้องเปลี่ยนเครื่องหมายบวกและลบให้อยู่ในรูปของเลข 0 หรือ 1 ด้วยการแทนเลขฐานสองที่มีเครื่องหมายสามารถทำได้หลายวิธี แต่วิธีที่นิยมมี 3 วิธี ดังต่อไปนี้

Sign magnitude

เป็นวิธีที่ง่ายและคล้ายกับระบบเลขฐานสิบที่สุด ทำได้โดยการกำหนดบิตที่อยู่ซ้ายมือสุดเป็นบิตแสดงเครื่องหมาย (Sign bit) ถ้าบิตมีค่า 0 จะแทนเครื่องหมายบวก และบิตมีค่า 1 จะแทนเครื่องหมายลบ ส่วนบิตที่เหลือทางขวามือจะใช้แทนค่าในระบบเลขฐานสอง (magnitude) ดังแสดงในตัวอย่างที่ 7.10 ซึ่งแสดงเลขฐานสองขนาด 8 บิต ประกอบด้วย 1 บิตซ้ายสุดเป็น sign bit และบิตที่เหลือ 7 บิตแทนเลขจำนวน ดังนั้น ค่าที่เป็นไปได้ทั้งหมดสำหรับกรณีนี้คือ $2^8 = 256$ จำนวน แต่วิธีนี้ไม่เป็นที่นิยมเนื่องจากปัญหาของการแทนค่า 0 สามารถเขียนได้สองแบบ คือ +0 และ -0 แม้ในทางคณิตศาสตร์จะถือว่าเลขทั้งสองมีค่าเท่ากัน แต่ในทางคอมพิวเตอร์ซึ่งกระทำการเปรียบเทียบบิตในตำแหน่งเดียวกันทีละบิต จะพบว่าบิตซ้ายสุดแตกต่างกัน จึงสรุปว่าเลขทั้งสองไม่เท่ากัน ทำให้ต้องสร้างวงจรพิเศษเพื่อตรวจสอบกรณีค่า +0 และ -0 โดยเฉพาะ

ตัวอย่างที่ 7.10 การแทนเลขฐานสองที่มีเครื่องหมายด้วยระบบ Sign magnitude

เลขฐานสิบ	เลขฐานสองขนาด 8 บิต
-127	1 1 1 1 1 1 1 1
-126	1 1 1 1 1 1 1 0
...	...
-1	1 0 0 0 0 0 0 1
-0	1 0 0 0 0 0 0 0
+0	0 0 0 0 0 0 0 0
+1	0 0 0 0 0 0 0 1
...	...
+126	0 1 1 1 1 1 1 0
+127	0 1 1 1 1 1 1 1

Ones complement

วิธีนี้จะคล้ายกับ sign magnitude คือ กำหนดบิตซ้ายสุดให้เป็น sign bit ส่วนบิตที่เหลือใช้แทนค่าตัวเลข ในกรณีที่เป็นการแทนค่าที่เป็นจำนวนลบ จะใช้การกลับบิตเป็นค่าตรงกันข้าม (bitwise complement) คือเปลี่ยนจากบิต 0 เป็น 1 จากบิต 1 เป็น 0 ดังแสดงในตัวอย่างที่ 7.11 แต่วิธีนี้ก็ยังมีปัญหาของการแทนค่า 0 ที่เป็นได้ทั้ง +0 และ -0 ดังแสดงในตัวอย่างที่ 7.12

ตัวอย่างที่ 7.11 จงหาค่า Ones complement ขนาด 8 บิตของ $(-32)_{10}$

วิธีทำ $(+32)_{10} = (0010\ 0000)_2$

$(-32)_{10} = (1101\ 1111)_2$

ตัวอย่างที่ 7.12 การแทนเลขฐานสองที่มีเครื่องหมายด้วยระบบ ones complement

เลขฐานสิบ	เลขฐานสองขนาด 8 บิต
-127	1 0 0 0 0 0 0 0
-126	1 0 0 0 0 0 0 1
...	...
-1	1 1 1 1 1 1 1 0
-0	1 1 1 1 1 1 1 1
+0	0 0 0 0 0 0 0 0
+1	0 0 0 0 0 0 0 1
...	...
+126	0 1 1 1 1 1 1 0
+127	0 1 1 1 1 1 1 1

Twos complement

วิธีนี้จะคล้ายกับ ones complement คือ กำหนดบิตซ้ายสุดให้เป็น sign bit ส่วนบิตที่เหลือใช้แทนค่าตัวเลข ในกรณีที่เป็นการแทนค่าที่เป็นจำนวนลบ หลังจากใช้การกลับค่าบิตแล้วจะต้องบวกค่า 1 เข้ากับบิตขวาสุดด้วย ทำให้แก้ปัญหาของค่า 0 ให้เหลือเพียงค่าเดียวได้ ดังแสดงในตัวอย่างที่ 7.13 จะเห็นได้ว่าเมื่อหาค่า twos complement ของ 0 ก็จะยังคงได้ค่าเดิม นั่นคือ +0 มีค่าเท่ากับ -0 (มีการทดหรือ overflow เกิดขึ้น แต่ในกรณีนี้จะไม่สนใจ overflow) ในตัวอย่าง 7.14 แสดงการหาค่า two complement ขนาด 4 บิตของ -5 และตัวอย่าง 7.15 แสดงการแทนค่าด้วยเลขฐานสองขนาด 4 บิตโดยใช้ระบบ two complement

ตัวอย่างที่ 7.13 จงหาค่า Twos complement ขนาด 4 บิตของ $(0)_{10}$

วิธีทำ $(+0)_{10} = (0\ 0\ 0\ 0)_2$

Bitwise complement $(1\ 1\ 1\ 1)_2$

$+\ \ 1$

$(-0)_{10} = (1\ 0\ 0\ 0\ 0)_2$

ตัวอย่างที่ 7.14 จงหาค่า Twos complement ขนาด 4 บิตของ $(-5)_{10}$

วิธีทำ $(+5)_{10} = (0\ 1\ 0\ 1)_2$

Bitwise complement $(1\ 0\ 1\ 0)_2$

$+\ \ 1$

$(-5)_{10} = (1\ 0\ 1\ 1)_2$

ตัวอย่างที่ 7.15 การแทนเลขฐานสองที่มีเครื่องหมายด้วยระบบ twos complement

เลขฐานสิบ	เลขฐานสองขนาด 4 บิต
-8	1 0 0 0
-7	1 0 0 1
-6	1 0 1 0
-5	1 0 1 1
-4	1 1 1 0
-3	1 1 0 1
-2	1 1 1 0
-1	1 1 1 1
+0	0 0 0 0
+1	0 0 0 1
+2	0 0 1 0
+3	0 0 1 1
+4	0 1 0 0
+5	0 1 0 1
+6	0 1 1 0
+7	0 1 1 1

การบวกโดยใช้การแทนค่าด้วย twos complement ทำได้เช่นเดียวกับการบวกเลขฐานสองทั่วไป แต่ให้สังเกตที่ sign bit ว่าเกิด overflow หรือไม่ overflow หมายถึงผลลัพธ์ที่ได้มีขนาดใหญ่เกินกว่าขนาดของเวิร์ด หรือใหญ่เกินกว่าขนาดที่ใช้ในการแทนค่าเลขฐานสองในระบบนั้นๆ

สำหรับการลบ ทำได้โดยใช้การบวกค่า twos complement ของตัวลบเข้ากับตัวตั้ง นั่นคือ

$$A - B = A + (-B)$$

โดยวิธีการพิจารณา overflow กระทำได้ดังนี้

- กรณี sign bit ต่างกันบวกกัน จะไม่มีโอกาสเกิด overflow ถ้ามีตัวทดที่บิตซ้ายสุด (เกินขนาดบิตที่ใช้) ให้ตัดตัวทดนั้นทิ้ง ค่าที่เหลือเป็นคำตอบ
- กรณี sign bit เหมือนกัน กระทำกันแล้วเปลี่ยนเป็นค่าตรงกันข้าว แสดงว่าเกิด overflow

ตัวอย่างในรูป 7.1 แสดงขั้นตอนอย่างละเอียดของการบวกและลบในระบบเลขฐานสองขนาด 4 บิต โดยใช้ twos complement และในรูป 7.2 แสดงตัวอย่างเพิ่มเติมอย่างย่อ

0010 = 2 +1001 = –7 1011 = –5 (a) M = 2 = 0010 S = 7 = 0111 –S = 1001	0101 = 5 +1110 = –2 10011 = 3 (b) M = 5 = 0101 S = 2 = 0010 –S = 1110
1011 = –5 +1110 = –2 11001 = –7 (c) M =–5 = 1011 S = 2 = 0010 –S = 1110	0101 = 5 +0010 = 2 0111 = 7 (d) M = 5 = 0101 S =–2 = 1110 –S = 0010
0111 = 7 +0111 = 7 1110 = Overflow (e) M = 7 = 0111 S = –7 = 1001 –S = 0111	1010 = –6 +1100 = –4 10110 = Overflow (f) M = –6 = 1010 S = 4 = 0100 –S = 1100

รูปที่ 7.1 ตัวอย่างการบวกลบในระบบเลขฐานสองขนาด 4 บิตโดยใช้ twos complement

1001 = –7 +0101 = 5 1110 = –2 (a) (–7) + (+5)	1100 = –4 +0100 = 4 10000 = 0 (b) (–4) + (+4)
0011 = 3 +0100 = 4 0111 = 7 (c) (+3) + (+4)	1100 = –4 +1111 = –1 11011 = –5 (d) (–4) + (–1)
0101 = 5 +0100 = 4 1001 = Overflow (e) (+5) + (+4)	1001 = –7 +1010 = –6 10011 = Overflow (f) (–7) + (–6)

รูปที่ 7.2 ตัวอย่างการบวกลบในระบบเลขฐานสองขนาด 4 บิตโดยใช้ twos complement

7.5 ระบบตัวเลขกับรหัสข้อมูล

ข้อมูลที่ใช้ในระบบคอมพิวเตอร์ ประกอบด้วยตัวเลข ตัวอักษร และสัญลักษณ์ต่างๆ ที่ประกอบอยู่ในคำสั่ง และข้อมูล ดังนั้น จึงจำเป็นต้องมีการกำหนดรหัสข้อมูล (Data representation) ที่ใช้แทนตัวอักขระ (character) เหล่านี้ ซึ่งใช้ระบบเลขฐานเช่นเดียวกับการแทนค่าเลขจำนวน โดยอักขระแต่ละตัวจะต้องมีค่าที่ไม่ซ้ำกัน รหัสข้อมูลที่ดีควรมีจำนวนรหัสเพียงพอที่จะแทนตัวอักขระได้ครบทุกตัว สามารถใช้แทนอักษรในภาษาอื่นๆ ได้ไม่เพียงแต่ภาษาอังกฤษเท่านั้น รหัสที่ใช้มีหลายระบบ ดังตัวอย่างต่อไปนี้

BCD (Binary Coded Decimal)

เป็นรหัสที่ใช้ระบบเลขฐานสองจำนวน 6 บิต ในการแทนค่าอักขระ 1 ตัว ดังนั้นจึงสามารถแทนค่าอักขระได้ทั้งสิ้น 2^6 = 64 ตัว แบ่งรหัสออกเป็น 2 ส่วน ส่วนแรกเรียกว่า Zone bit ซึ่งใช้ 2 บิตซ้ายสุด ส่วนบิตที่เหลือ 4 บิตเรียกว่า Numeric bit ปัจจุบันไม่นิยมใช้เนื่องจากมีจำนวนรหัสน้อยเกินไป

EBCDIC (Extended Binary Coded Decimal Interchange Code)

เป็นรหัสที่พัฒนามาจากรหัส BCD โดยขยายขนาดเป็นจำนวน 8 บิต ทำให้มีคุณสมบัติเพิ่มเติมที่เรียกว่า Extended คือ สามารถเพิ่มจำนวนอักขระที่แทนค่าได้ทั้งสิ้น 256 (2^8) ตัว และมีคุณสมบัติที่เรียกว่า Interchange คือ สามารถเปลี่ยนแปลงความหมายของรหัสจากชุดอักขระหนึ่งเป็นชุดอักขระอื่นได้ รหัสแบ่งออกเป็น 2 ส่วนเช่นเดิม โดยในส่วน Zone bit จะเพิ่มขนาดเป็น 4 บิตใช้สำหรับระบุว่าเป็นชุดอักษรภาษาอังกฤษ ตัวเลข หรือสัญลักษณ์พิเศษ ส่วนบิตที่เหลือ 4 บิตเป็น Numeric bit เหมือนกับรหัส BCD เป็นตัวที่ใช้ระบุว่าอยู่ในช่วงตัวอักขระใด ดังแสดงในตารางที่ 7.1

Character	Zone bit	Numeric bit
A, B, C, D, E, F, G, H, I	1100	0001 – 1001
J, K, L, M, N, O, P, Q, R	1101	0001 – 1001
S, T, U, V, W, X, Y, Z	1110	0001 – 1001
0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1111	0001 – 1001
a, b, c, d, e, f, g, h, i	1000	0001 – 1001
j, k, l, m, n, o, p, q, r	1001	0001 – 1001
s, t, u, v, w, x, y, z	1010	0001 – 1001
blank, $, ., <, >, (, +	1011	0001 – 1001
&, !, *,), ;	0100	0001 – 1001
-, /, ‘, _, ?	0101	0001 – 1001
:, #, @, =, “	0111	0001 – 1001

ตารางที่ 7.1 ตัวอย่างรหัส EBCDIC

จากตาราง สามารถแทนค่าตัวเลข 0 ด้วยรหัส EBCDIC โดยใช้ 1111 0000 หรือตัวเลข 1 ด้วย 1111 0001 หรือ ตัวอักษร B แทนด้วย 1100 0010

ASCII (American Standard Code for International Interchange)

เป็นรหัสที่กำหนดขึ้นโดยสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI) นิยมใช้ในการส่งข้อมูลแบบอนุกรมไปตามสายโทรศัพท์ โทรเลข หรือระหว่างคอมพิวเตอร์กับอุปกรณ์อื่นๆ ในช่วงเริ่มต้นกำหนดให้ใช้ระบบเลขฐานสองจำนวน 7 บิต ในการแทนค่าอักขระ 1 ตัว ซึ่งสามารถแทนค่าอักขระได้ทั้งสิ้น 128 ตัว ต่อมาจึงได้ขยายรหัสเพิ่มเป็น 8 บิต ทำให้แทนค่าเพิ่มขึ้นเป็น 256 ตัว ดังตัวอย่างแสดงในตาราง 7.2

รหัสแบ่งออกเป็น 3 ส่วนประกอบด้วย

- Control character เป็นส่วนที่ใช้ในการควบคุมการทำงานของอุปกรณ์แสดงผล ซึ่งเป็นรหัสในช่วง 0-32
- Lower ASCII เป็นส่วนที่ใช้แทนตัวอักษรในภาษาอังกฤษ และสัญลักษณ์ต่างๆ ซึ่งเป็นรหัสในช่วง 33-127
- Higher ASCII เป็นส่วนที่ใช้แทนตัวอักษรในภาษาอื่นๆ นอกเหนือจากภาษาอังกฤษ เช่น ภาษาไทย เป็นรหัสในช่วง 128-256

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ตารางที่ 7.2 ตัวอย่างรหัส ASCII

Unicode

เป็นรหัสที่กำหนดขึ้นโดยองค์กรไม่แสวงกำไรชื่อ Unicode consortium เพื่อทำให้รหัสเป็นมาตรฐานเป็นสากลใช้ได้กับทุกภาษาทั่วโลก โดยใช้ระบบเลขฐานสองจำนวน 16 บิต ในการแทนค่าอักขระ 1 ตัว ดังนั้นจึงสามารถมีจำนวนรหัสได้ถึง 2^{16} หรือ 65,536 รหัส ซึ่งเพียงพอสำหรับตัวอักษรในภาษาอื่นๆ เช่น ภาษาจีน หรือญี่ปุ่น รวมทั้งสัญลักษณ์พิเศษต่างๆ เช่น สัญลักษณ์ทางคณิตศาสตร์ ปัจจุบันได้รับความนิยมอย่างมาก ระบบปฏิบัติการ โปรแกรมประยุกต์ รวมถึงภาษาการโปรแกรมต่างๆ ส่วนมากต่างก็สนับสนุนการใช้งานรหัส Unicode เป็นส่วนมาก

7.6 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

Chapter 9 William Stalling. Computer organization and architecture: designing for performance.

บทที่ 2 การแทนข้อมูล สุจิตรา อดุลย์เกษม. เอกสารประกอบการสอนองค์ประกอบคอมพิวเตอร์และภาษา. http://elearning.nectec.or.th/index.php?mod=Courses&op=lesson_show&uid=&cid=47&lid=363&sid=&eid=&page=1