Written by Thapanapong Rukkanchanunt

Data Representation

ตอนที่ 2 การแทนข้อมูลในคอมพิวเตอร์

เค้าโครงเนื้อหาในตอนที่ 2

1. หน่วยของข้อมูล และระบบเลขเลขฐาน 2
2. การเก็บเลขจำนวนเต็ม
3. การเก็บเลขทศนิยม
4. การบีบอัดข้อมูล
5. การเก็บข้อมูลรูปภาพ

Bit และ Byte

- หน่วยพื้นฐานของข้อมูลในระบบคอมพิวเตอร์เรียกว่า bit ซึ่งมาจากการย่อคำว่า **Bi**nary Digi**t** โดยแทนด้วยตัวเลขหนึ่งหลัก 0 หรือ 1
- bit เปรียบเสมือนสถานะทางไฟฟ้า ซึ่งมี 2 สถานะคือ เปิด (แทนด้วย 1) และ ปิด (แทนด้วย 2)
- ในปี 2507 คอมพิวเตอร์ IBM เริ่มต้นการเรียกกลุ่มของ bit จำนวน 8 bits ว่า Byte

Words

- Words คือกลุ่มของ Byte ตั้งแต่ 1 Bytes ขึ้นไป ที่มักจะถูกเรียกใช้งานพร้อมกัน
- ขนาดของ Word คือขนาดของข้อมูลที่คอมพิวเตอร์นั้น ๆ ประมวลผลได้อย่างมีประสิทธิภาพ
- ในปัจจุบันขนาดของ Word มีสองแบบที่ใช้กันอย่างแพร่หลาย คือ 32 bits และ 64 bits

ระบบเลขฐานสิบ

- เลขจำนวนที่เราใช้อยู่ในปัจจุบันคือเลขฐานสิบ
- ในเลขฐานสิบ จะมีตัวเลขทั้งหมด 10 ตัวคือ 0 1 2 3 4 5 6 7 8 และ 9
- ในระบบเลขฐาน ค่าของตัวเลขจะเพิ่มขึ้นในแต่ละหลักด้วยการยกกำลังฐานนั้น
- เช่น เลข 129 มีค่าเป็น $1\times10^2 + 2\times10^1 + 9\times10^0$
- ในบางครั้ง เราจะเขียนฐานกำกับไว้ เพื่อให้ทราบว่าเลขนี้มาจากฐานใด เช่น 129_{10}

ระบบเลขฐานสอง

- ระบบเลขฐานสองเป็นระบบเลขที่นำมาใช้กับการกระทำของข้อมูลในระบบคอมพิวเตอร์
- ในระบบเลขฐานสอง จะมีตัวเลขอยู่สองตัวคือ 0 และ 1 (เช่นเดียวกับ bit)
- ค่าของเลขฐานสองเมื่อเทียบกับฐานสิบ สามารถทำได้เช่นกับเลขฐานสิบ
- $101001_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 41_{10}$

การแปลงเลขฐาน (จำนวนเต็ม)

- การแปลงเลขฐานสิบให้เป็นเลขฐานสอง สามารถทำได้โดยการหารเก็บเศษด้วย 2
- เช่น ถ้าต้องการเปลี่ยน 202_{10} ให้เป็นเลขฐานสอง สามารถทำได้ดังนี้

202 ÷ 2 = 101 เศษ 0
101 ÷ 2 = 50 เศษ 1
50 ÷ 2 = 25 เศษ 0
25 ÷ 2 = 12 เศษ 1
12 ÷ 2 = 6 เศษ 0
6 ÷ 2 = 3 เศษ 0
3 ÷ 2 = 1 เศษ 1
1 ÷ 2 = 0 เศษ 1

$202_{10} = 11001010_2$

การแปลงเลขฐาน (ทศนิยม)

- ถ้าต้องการเปลี่ยนเลขฐานในส่วนที่เป็นทศนิยม สามารถทำได้โดยการคูณฐานนั้นไปเรื่อย ๆ จนส่วนที่เป็นทศนิยมมีค่า 0 หรือได้จำนวนตำแหน่งทศนิยมที่ต้องการ โดยดูที่ส่วนที่เป็นจำนวนเต็มของแต่ละการคูณ
- เช่น ถ้าต้องการเปลี่ยน 0.325_{10} ให้เป็นเลขฐานสอง

$0.375 \times 2 = 0.75$

$0.75 \times 2 = 1.5$

$0.5 \times 2 = 1.0$

(ถ้าส่วนเป็นจำนวนจริงมีค่ามากกว่า 0 ให้ละเว้นในการคูณขั้นถัดไป)

$$0.375_{10} = \overrightarrow{0.011}_{2}$$

การแปลงเลขฐาน (ต่อ)

- ตัวเลขทศนิยมที่เป็นทศนิยมรู้จบในเลขฐานหนึ่งอาจจะเป็นทศนิยมไม่รู้จบในอีกเลขฐานหนึ่งได้ ซึ่งก็ถือเป็นข้อจำกัดของคอมพิวเตอร์
- คอมพิวเตอร์ไม่สามารถเก็บเลข 0.1 แบบตรง ๆ ได้ เนื่องจาก

0.1×2	$= 0.2$	0.6×2	$= 1.2$
0.2×2	$= 0.4$	0.2×2	$= 0.4$
0.4×2	$= 0.8$	0.4×2	$= 0.8$
0.8×2	$= 1.6$	...	

สังเกตเห็นว่ารูปแบบเริ่มซ้ำ ดังนั้น $0.1_{10} = 0.\overline{00011}_{2}$

การเก็บเลขจำนวนเต็มในคอมพิวเตอร์

- คอมพิวเตอร์เก็บเลขจำนวนเต็มในรูปแบบของเลขฐานสอง
- จำนวนหลักของเลขฐานสองที่เก็บคือขนาดของ Word ของคอมพิวเตอร์นั้น ๆ
- ถ้า Word มีขนาด n bits ตำแหน่งซ้ายสุดจะเป็นกำหนดเครื่องหมายของตัวเลข โดยที่ 0 คือจำนวนเต็มบวก และ 1 คือจำนวนเต็มลบ n-1 bits ที่เหลือคือค่าของตัวเลขนั้น
- เช่น ถ้าต้องการเก็บเลข 25_{10} ก็ต้องแปลงเป็นเลขฐานสองก่อนนั่นคือ 11001_2 ถ้า Word มีขนาด 8 bits ค่าที่เก็บในคอมพิวเตอร์คือ

0001 1001

ในกรณีที่เลขฐานสองมีจำนวนตำแหน่งน้อยกว่าจำนวน bit ที่เก็บได้ให้เพิ่มเลข 0 ไว้ด้านหน้า

การบวกเลขฐานสอง

- การบวกเลขฐานสองเหมือนกันการบวกเลขฐานสิบ แต่เนื่องจากตัวเลขในเลขฐานสองมีแค่สองตัว 1 + 1 จึงเท่ากับ 0 แล้วจะต้องมีการทดไปยังตำแหน่งถัดไป
- การนับเครื่องหมายถือเป็นเรื่องสำคัญ
  - ถ้าเลขทั้งสองจำนวนมีเครื่องหมายเหมือนกัน ผลลัพธ์ที่ได้ก็จะมีเครื่องหมายเหมือนกัน
  - ถ้าเลขทั้งสองจำนวนมีเครื่องหมายต่างกัน ผลลัพธ์จะมีเครื่องหมายของเลขที่ใหญ่กว่า

ตัวอย่างการบวกเลขจำนวนเต็ม

- ต้องการบวก 0010 1010 + 0000 1111 ในคอมพิวเตอร์ 8 bits

ทด	1 1 1	
	0 1 0 1 0 1 0	42
+	0 0 0 1 1 1 1	+15
	0 1 1 1 0 0 1	57

ทั้งสองจำนวนมีเครื่องหมายเหมือนกัน ดังนั้นคำตอบคือ 0011 1011

เหตุการณ์ Overflow

- เหตุการณ์ Overflow คือเหตุการณ์ที่ผลลัพธ์ของการบวกเลขมีจำนวนหลักมากกว่าค่าที่เก็บได้ (ทดเกินจำนวนหลัก)
- เช่นถ้าต้องการบวก 75 + 100 ในคอมพิวเตอร์ขนาด 8 Bits

ทด	1	1 1 1	
		1 0 0 1 0 1 1	75
+		1 1 0 0 1 0 0	+100
		0 1 0 1 1 1 1	47

คำตอบที่ได้คือ 0010 1111 หรือ 47 ซึ่งเป็นคำตอบที่ไม่ถูกต้อง

การลบเลขจำนวนเต็ม

- การลบเลขฐานสองมีลักษณะคล้ายเลขฐานสิบคือมีการยืมค่าจะหลักที่ใหญ่กว่า ในที่ค่าที่ยืมได้จะเป็น 2 แทนที่จะเป็น 10 เนื่องจากเป็นเลขฐานสอง
- การคำนวณเครื่องหมายทำให้การลบซับซ้อน เนื่องจากต้องคำนวณว่าเลขตัวไหนมีขนาดใหญ่กว่า
- เนื่องจากการลบโดยวิธีปกติทั่วไปยุ่งยากสำหรับคอมพิวเตอร์ จึงมีการคิดวิธีที่เปลี่ยนการลบให้กลายเป็นการบวกด้วยระบบที่เรียกว่า Complement
- ระบบ Complement มีทั้งสิ้นสองแบบคือ One’s Complement และ Two’s Complement

ระบบ Complement

- สมมติว่าเราต้องการลบเลข 67 – 52 ในระบบเลขฐานสิบ เราจะใช้วิธีการบวก 167 ด้วย -52 ดังนั้นเราจำเป็นต้องแปลง -52 ให้อยู่ในรูปของ Complement
- Complement ของตัวเลขเลขหนึ่ง คือตัวเลขที่เมื่อนำไปบวกเลขนั้นได้ค่าที่สูงสุดที่มีจำนวนหลักเท่ากัน เช่น 47 คือ Complement ของ 52 เพราะ 47 + 52 = 99
- ถ้าหากเราลองเอา 67 มาบวกกับ 47 จะได้ 114 ซึ่งถ้าเรานำหลักที่เกินมาบวกเพิ่มกับเลขที่เหลืออยู่จะได้เป็น 14 + 1 = 15 ซึ่งคือผลลัพธ์ของ 67 – 52
- ดังนั้นการลบเลขคือการบวกด้วย Complement ของเลขนั้น
- สำหรับเลขฐานสองแล้วการหา Complement สามารถทำได้ง่ายมาก

ระบบ One’s Complement

- เราจะแปลงเลขติดลบให้อยู่ในรูปของ Complement
- Complement ของเลขฐานสองใช้วิธีเดียวกับ Complement ของเลขฐานสิบ เช่น Complement ของ 0100 0111 คือ 1011 1000 เนื่องจากนำมาบวกกันได้ 1111 1111
- เราจะสังเกตเห็นว่าการหา Complement ของเลขฐานสองสามารถทำได้ง่ายมากโดยการเปลี่ยน 0 เป็น 1 และ 1 เป็น 0

การลบเลขด้วยระบบ One's Complement

- ถ้าต้องการลบ 0100 0011 (67) ด้วย 0011 0100 (52)
- แปลง 0011 0100 ให้เป็น Complement นั่นคือ 1100 1011

ทด 1 1 1 1

0 1 0 0 0 0 1 1 67

\+ 1 1 0 0 1 0 1 1 +(-52)

0 0 0 0 1 1 1 0

นำหลักที่เกินกลับเข้ามาบวก จะได้ 0000 1110 + 1 = 0000 1111 = 15

การลบเลขด้วยระบบ One’s Complement

- ถ้าต้องการลบ 0011 0100 (52) ด้วย 0100 0011 (67)
- แปลง 0100 0011 ให้เป็น Complement นั่นคือ 1011 1100

ทด	1 1 1 1	
	0 0 1 1 0 1 0 0	52
+	1 0 1 1 1 1 0 0	+(-67)
	1 1 1 1 0 0 0 0	(-15)

1111 0000 คือ Complement ของ 0000 1111 หรือ 15 ดังนั้นคำตอบคือ -15

การตรวจ Overflow ในระบบ Complement

- เราสามารถตรวจสอบเหตุการณ์ Overflow ในระบบ Complement ด้วยการตรวจสอบว่าเลขทดที่ตำแหน่งสุดท้ายตรงกับเลขทดที่เกินมาหรือไม่

ทด	0 1	
	0 1 0 0 1 0 1 1	75
+	0 1 1 0 0 1 0 0	+100
	1 0 1 0 1 1 1 1	(-80)

สังเกตว่าเลขทดที่เกินมาคือ 0 ส่วนเลขทดที่ตำแหน่งสุดท้ายคือ 1 ไม่ตรงกันดังนั้นการบวกนี้ทำให้เกิดเหตุการณ์ Overflow

การเก็บเลขทศนิยมในคอมพิวเตอร์

- เลขทศนิยมจะถูกแบ่งออกเป็นสามส่วนในระบบคอมพิวเตอร์คือ
  - เครื่องหมาย
  - เลขยกกำลัง
  - จำนวนหลังจุดทศนิยม
- เพื่อง่ายต่อการทำความเข้าใจเราจะใช้ระบบ 14 bits ในการเก็บเลขทศนิยมโดยแบ่งเป็น
  - เครื่องหมาย 1 bit
  - เลขยกกำลัง 5 bit
  - จำนวนหลังจุดทศนิยม 8 bit

ตัวอย่างการเก็บเลขทศนิยม

- ถ้าเราต้องการเก็บเลข 3.125 เราต้องแปลงเลขนี้ให้กลายเป็นเลขฐานสองก่อน

$$3 = 11.001$$

- จากนั้นปรับให้อยู่ในรูปของ $0.A \times 2^B$

$$11.001 = 0.11001 \times 2^2$$

- เมื่อนำแต่ละส่วนมารวมกันจะได้ค่าที่เก็บคือ

$$0\ 00010\ 11001000$$

สังเกตว่าถ้าจำนวนเลขหลังจุดทศนิยมไม่ครบให้เติม 0 ต่อท้าย

กรณีเลขยกกำลังติดลบ

- ถ้าตัวเลขมีขนาดเล็กมากทำให้เมื่อแปลงมาอยู่ในรูปของ $0.A \times 2^B$ แล้ว B เป็นจำนวนติดลบ
- หนึ่งวิธีที่เราสามารถทำได้คือการให้ bit ซ้ายสุดระบุเครื่องหมาย แต่ในทางปฏิบัติแล้วเราจะใช้วิธีการปรับขอบเขตของตัวเลข โดยการบวกค่ากึ่งกลางของขอบเขตนั้นให้กับเลขยกกำลัง เนื่องจากเราใช้ 5 bits ในการเก็บเลขยกกำลัง ขอบเขตของตัวเลขคือ 0 – 31 ซึ่งค่ากลางคือ 16
- ดังนั้นเลข 0.11001×2^2 จะถูกเก็บเป็น 0 10010 11001000 (18 = 10010)
- ส่วนเลข 0.11001×2^{-2} จะถูกเก็บเป็น 0 01100 11001000 (14 = 01100)

ขนาดของข้อมูลในหน่วยความจำ

- ข้อมูลที่เป็นอักขระจะใช้พื้นที่ในการเก็บ 8 bits หรือ 1 Byte ซึ่งขนาดของข้อมูลในหนึ่งไฟล์คือจำนวนอักขระนั่นเอง เช่น ข้อความ I LOVE ICECREAM มีทั้งสิ้น 15 อักขระจึงใช้พื้นที่ในการเก็บข้อมูล 15 Bytes (ช่องว่างถือเป็นอักขระ)
- สำหรับอักขระในภาษาอื่น อาจจะใช้พื้นที่ในการเก็บมากกว่า 1 Byte เนื่องจากอาจจะมีตัวอักษรเกิน 256 (รวมตัวเลขและสัญลักษณ์ต่าง ๆ)

การบีบอัดข้อมูล

- เราสามารถลดขนาดของข้อมูลลงได้โดยการปรับเปลี่ยนการแทนค่าของตัวอักษร
- เราจะสังเกตเห็นว่าตัวอักขระบางตัวปรากฏบ่อยกว่าอักขระตัวอื่น ถ้าเราใช้จำนวน bit ที่น้อยลงแทนค่าตัวอักขระที่เจอบ่อยแล้วเพิ่มจำนวน bit ให้มากขึ้นกับตัวอักษรที่แทบจะไม่ได้เจอ เราสามารถจะลดขนาดของข้อมูลได้
- วิธีการของ Huffman เป็นวิธีที่สร้างการแทนค่าของตัวอักษรที่ทำให้ขนาดของข้อมูลน้อยลงอย่างมากและมีประสิทธิภาพสูง

ตัวอย่างการบีบอัดข้อมูล

- สมมุติว่าเรามีตัวอักขระอยู่ 4 ตัวคือ A B C และ D โดยแต่ละตัวปรากฏทั้งหมด 5, 20, 30 และ 45 ครั้งตามลำดับ
- โดยปกติแล้วเราจะให้ทุกอักขระแทนด้วยจำนวน bit ที่เท่ากันเช่น A = 00, B = 01, C = 10, D = 11 จะทำให้ไฟล์นี้มีขนาด 5 × 2 + 20 × 2 + 30 × 2 + 45 × 2 = 200 bits
- แต่ถ้าเราแทนค่าใหม่เป็น A = 111, B = 110, C = 10, D = 0 ไฟล์เดิมจะมีขนาดเป็น 5 × 3 + 20 × 3 + 30 × 2 + 45 × 1 = 180 bits ซึ่งลดลงจากเดิม

ข้อมูลรูปภาพ

- ข้อมูลรูปภาพจะถูกเก็บอยู่ในรูปแบบของตาราง โดยหนึ่งจุดในรูปภาพเราจะเรียกว่า Pixel
- สำหรับภาพขาวดำ แต่ละ Pixel สามารถแทนด้วยแรกตัวเดียวได้โดยจะเป็นความสว่างของแสง (0 แทนสีดำ และ ค่ามากที่สุดแทนสีขาว)
- สำหรับภาพสี แต่ละ Pixel จะประกอบด้วยเลขสามตัวคือความเข้มแสงของสีแดง สีเขียว และสีฟ้า

ความละเอียดของรูปภาพ

- ความละเอียดของรูปภาพแบ่งออกเป็นสองส่วนคือขนาดของรูปภาพและความลึกของสี
- ขนาดของรูปภาพวัดโดยจำนวน Pixel ในแนวกว้าง (Width) และแนวสูง (Height)
- ความลึกของสีหรือ Color Depth วัดจากจำนวน bit ที่เก็บแต่ละ Pixel เช่น ถ้า Color Depth เท่ากับ 8 bit แต่ละ Pixel จะสามารถเก็บค่าของสีได้กว้าง 0 – 255 (มีทั้งหมด 256 เฉดสีที่แตกต่างกัน)
- ปัจจุบัน Color Depth ที่นิยมใช้กันคือ True Color (24-bit) ซึ่งจะมีทั้งหมด 16 ล้านกว่าเฉดสี่ ในขณะที่สายตาคนปกติสามารถแยกแยะได้ประมาณ 10 ล้านเฉดสี

คำถามท้ายบทเรียน

- จงแปลง 204.202 เป็นเลขฐานสอง (ทศนิยม 5 ตำแหน่ง)
- จงหา Complement ของ 2559_{10}
- จงหา Complement ของ $-0011\ 0111_{2}$
- จงหาค่าของ $0110\ 0011_{2} - 0110\ 1111_{2}$
- 12.345 จะถูกเก็บอย่างไรโดยใช้ระบบ 14 bits
- ถ้าในไฟล์มีตัวอักขระปรากฏอยู่ 3 ตัวคือ A, B และ C โดยแต่ละตัวปรากฏทั้งสิ้น 5, 30, 65 ครั้งตามลำดับ ถ้าไฟล์นี้ใช้ 2 bits ในการแทนอักขระหนึ่งตัว จงหาว่าถ้าเลือกการแทนค่าเป็น A = 10, B = 0, C = 11 แล้วขนาดไฟล์จากเพิ่มหรือลดลงกี่เปอร์เซ็นต์

เอกสารอ้างอิง

- The Essentials of Computer Organization and Architecture *by Linda Null and Julia Lobur*