Introduction to Microprocessors

Introduction to Microprocessors

- เคยมีผู้นิยามความหมาย Microprocessor คือ

  “หน่วยที่ทำหน้าที่ควบคุม ในระบบของ Microcomputer, เป็นซิฟซิลิกอน ซึ่งมีลักษณะเป็นแผ่นบาง ขนาดเล็ก ประกอบไปด้วยอุปกรณ์ทางตรรกพื้นฐาน สำหรับจดจำข้อมูล, กระทำการในการคำนวณ และ ทำงานตามคำสั่งต่าง ๆ ”

การประยุกต์ใช้งาน Microprocessors

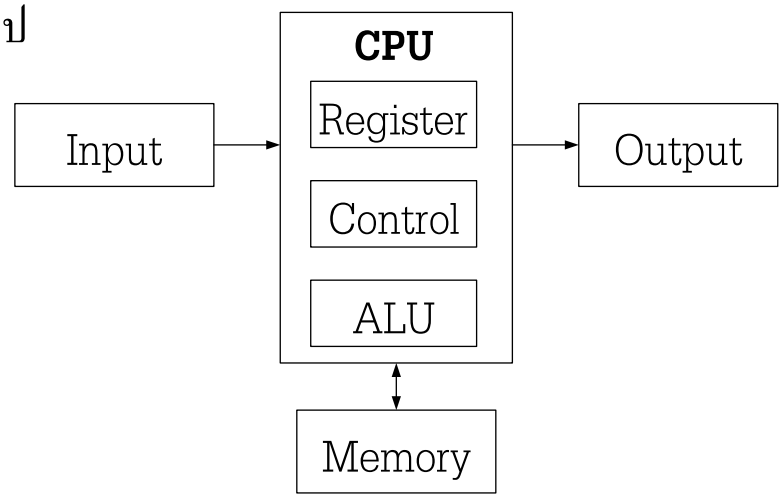
- คนทั่วไปจะคิดว่า Microprocessors จะมีอยู่เพียงในเครื่องคอมพิวเตอร์ เท่านั้น แต่แท้ที่จริงแล้ว Microprocessors ยังถูกนำมาประยุกต์ใช้งานในเครื่องใช้ต่าง ๆ มากมายรอบตัวเรา เช่น
  - เครื่องคิดเลข
  - เครื่องตอบรับโทรศัพท์
  - กล้อง VDO
  - เตาไมโครเวฟ
  - เครื่องออกกำลังกาย
  - เครื่องมือทางการแพทย์

Introduction to Microprocessors

- ซึ่งที่กล่าวมานี้ จะเป็นเพียง Microprocessor ที่ทำงานในลักษณะเฉพาะด้าน ซึ่งต่างจากเครื่องคอมพิวเตอร์ ซึ่งสามารถนำไปใช้งานได้ในด้านต่าง ๆ ซึ่งขึ้นอยู่กับโปรแกรมที่จะส่งให้เครื่องคอมพิวเตอร์ทำงาน
- จากอุปกรณ์ที่ยกตัวอย่างมานั้น ก่อนที่จะมีการนำ Microprocessors เข้ามาประยุกต์ใช้งาน อุปกรณ์เหล่านั้นก็สามารถทำงานได้ในระดับหนึ่ง แต่พอนำ Microprocessors เข้ามาประยุกต์ใช้งาน อุปกรณ์เหล่านั้นก็ มีความสามารถในการทำงานเพิ่มมากขึ้น

Microprocessors System

- โดยทั่วไปแล้ว Block Diagram ของระบบ Computer มีลักษณะดังรูป



Central Processing Unit (CPU)

- จะเป็นส่วนที่ทำหน้าที่ควบคุมลำดับขั้นตอนการทำงานตามคำสั่งต่าง ๆ ที่รับเข้ามา โดยที่คำสั่งเหล่านี้จะเก็บไว้ในหน่วยความจำของเครื่อง ในลักษณะของเลขฐาน 2

- ส่วนประกอบของ CPU จะประกอบไปด้วย
  - **Control Section : CU, DU, MMU, BIU**
  - **Arithmetic and Logic Unit (ALU) / FPU**
  - **Registers**

หน้าที่ของ Control Section

- ควบคุมการโอนย้ายข้อมูลต่าง ๆ ภายใน CPU รวมไปถึงการโอนย้ายคำสั่ง จากหน่วยความจำเข้ามายัง CPU เพื่อที่จะทำงานตามคำสั่งนั้น ๆ
- ทำการถอดรหัสคำสั่งที่อ่านเข้ามา เพื่อใช้กำหนดลำดับขั้นตอนการทำงานให้เป็นไปตามคำสั่งนั้น คำสั่งที่รับเข้ามาอาจจะเป็น การโอนย้ายข้อมูล หรือแปลงค่าข้อมูล โดยที่ในการแปลงค่าของข้อมูลนั้น จะกระทำในส่วนย่อยของ CPU ที่เรียกว่า Arithmetic and Logic Unit (ALU)
- ความเร็วในการทำงาน จะขึ้นอยู่กับสัญญาณนาฬิกา ที่ป้อนให้กับระบบ

ALU, Register

- Arithmetic Logic Unit (ALU) จะเป็นส่วนที่ทำหน้าที่ กระทำในการกระทำทางคณิตศาสตร์พื้นฐาน เช่น การบวก, การลบ, การคูณ และการหาร การกระทาง ตรรกศาสตร์ เช่น การกระทำ Complement , AND, OR และ EX-OR
- Register จะเป็นหน่วยความจำขนาดเล็ก ที่ทำหน้าที่สำหรับพักข้อมูลที่จะใช้ในการประมวลผล ซึ่ง Register บางตัวก็จะมีคุณลักษณะ พิเศษต่าง ๆ กันออกไป แล้วแต่ว่าจะออกแบบให้ทำหน้าที่ในลักษณะใด

การทำงานของคอมพิวเตอร์

- การทำงานต่าง ๆ ของคอมพิวเตอร์นั้น โดยทั่วไปแล้วจะเป็นลักษณะในการทำงานพื้นฐานคือ การนำข้อมูลเข้า, การประมวลผลข้อมูล และการแสดงผลข้อมูล ซึ่งระยะเวลาที่ใช้ในการทำคำสั่ง แต่ละคำสั่งนั้น จะมากหรือน้อยขึ้นอยู่กับความซับซ้อนของคำสั่งนั้น ๆ
- Input unit จะทำหน้าที่ในการส่งข้อมูลต่าง ๆ จากภายนอกเข้าสู่ CPU ซึ่งอาจจะมาจากแหล่งต่าง ๆ
- Output unit เป็นส่วนที่ทำหน้าที่ในการแสดงผลลัพธ์ หลังจากการทำงานต่าง ๆ ภายใน CPU

Memory

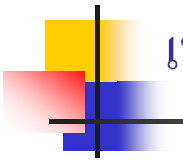
- Memory จะทำหน้าที่ในการเป็นที่สำหรับเก็บหรือพักข้อมูล
- ข้อมูลที่เข้าสู่ CPU นั้น บางครั้งอาจจะนำมาประมวลผลเลย หรือเก็บไว้ในหน่วยความจำก่อนแล้วจึงค่อยนำมาประมวลผลในภายหลัง ในกรณีของการนำเอาข้อมูลออกก็เช่นกัน อาจจะนำข้อมูลที่ได้จากการประมวลผล ออกสู่อุปกรณ์ภายนอกเลย หรืออาจจะนำไปเก็บไว้ในหน่วยความจำก่อน แล้วจึงนำข้อมูลออกในภายหลัง

Type of Memory

- ROM (Read Only Memory) เป็นหน่วยความจำ ที่สามารถเก็บรักษาข้อมูลได้เป็นเวลานานแม้ไม่มีกระแสไฟฟ้าจ่ายให้กับหน่วยความจำ หน่วยความจำชนิดนี้ จะถูกเขียนข้อมูลจากโรงงานที่ผลิต เมื่อเขียนข้อมูลลงไปแล้วจะไม่สามารถเขียนข้อมูลใหม่ทับลงไปได้
- RWM (Read Write Memory) เป็นหน่วยความจำ ที่ใช้สำหรับเป็นที่เก็บสำรองข้อมูลในการรับข้อมูลเข้า, ข้อมูลหลังจากการประมวลผล , ข้อมูลก่อนที่จะนำออกสู่อุปกรณ์ Output เช่น RAM

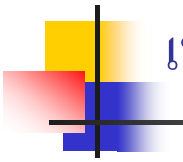
ROM

- EPROM (Erasable Programmable Read Only Memory) คือ หน่วยความจำ ROM ที่สามารถทำการเขียนข้อมูลลงไปใหม่ได้ จะใช้สำหรับในการทดสอบโปรแกรมต่าง ๆ ที่ทำการพัฒนาขึ้น ถ้าเกิดความผิดพลาดของโปรแกรม ณ.ตำแหน่งใด ก็สามารถแก้ไขข้อมูลได้ โดยใช้แสง UV (Ultra Violet) ในการลบ แล้วทำการเขียนข้อมูลลงไปใหม่
- EEPROM (Electrical Erasable Programmable Read Only Memory) คือหน่วยความจำ EPROM ที่สามารถทำการลบข้อมูลโดยใช้ ไฟฟ้าแทนที่จะใช้แสง UV (Ultra Violet)



เทคโนโลยีการผลิต Chip

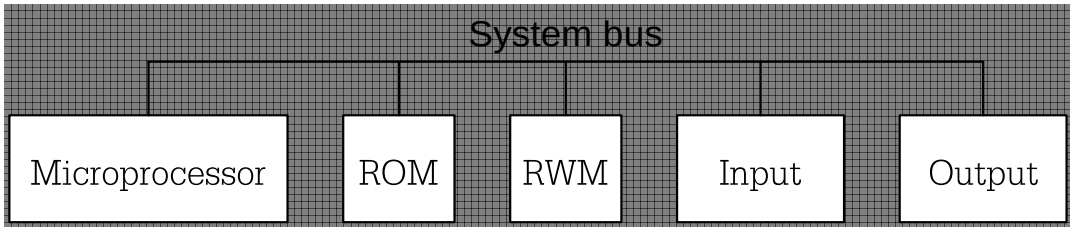
- SSI (Small Scale integration) จะเป็นการผลิตที่รวมเอาจำนวน gate พื้นฐานไม่เกิน 12 gate รวมอยู่ใน Chip เดียวกัน เช่น IC TTL พื้นฐาน
- MSI (Medium Scale Integration) จะเป็นการผลิตที่รวมเอาจำนวน gate พื้นฐานตั้งแต่ 13 - 99 gate รวมอยู่ใน Chip เดียวกัน เช่น IC จำพวก Counter, Decoder, Register, Adder และ Comparators เป็นต้น



เทคโนโลยีการผลิต Chip

- LSI (Large Scale Integration) จะเป็นการผลิตที่รวมเอาจำนวน gate ตั้งแต่ 100 gate หรือมากกว่า รวมอยู่ใน Chip เดียวกัน เช่น CPU ขนาด 8 -16 Bits
- VLSI (Very Large Scale Integration) จะเป็นการผลิตที่รวมเอา จำนวน Transistor มากกว่า 1 ล้านตัวรวมอยู่ใน Chip เดียวกัน
- จำนวน Transistor ของ 386 ~300K, 486 ~1M, PIII ~10M, Super Computer ~1G

ระบบ Bussed ของ Microprocessors



- จาก Block Diagram ของระบบ Microprocessor แต่ละ Block ก็คือระบบย่อยหรือหน่วยของ Hardware โดยเชื่อมต่อกัน ผ่านทาง System Bus
- Bus คือกลุ่มของการเชื่อมต่อซึ่งเป็นช่องทางในการโอนย้ายข้อมูลระหว่างระบบย่อยแต่ละระบบ และสัญญาณความคุมการทำงานจาก Microprocessor ที่ใช้ในการควบคุมระบบย่อยแต่ละระบบ

System bus

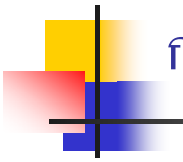
- ระบบ BUS จะถูกแบ่งออกเป็น
  - Address Bus ใช้สัญญาณที่ใช้ในการระบุตำแหน่งของหน่วยความจำ และ อุปกรณ์ หรือ I/O ที่ต้องการจะเข้าถึง
  - Data Bus เป็นช่องสัญญาณที่ใช้ ในการเป็นทางผ่านของข้อมูล
  - Control Bus เป็นช่องสัญญาณ สำหรับการควบคุมจังหวะการทำงาน ของระบบย่อยต่าง ๆ เพื่อให้ทำงานสัมพันธ์กัน
- ในระบบ Single - Chip Microprocessor (Microcontroller) จะรวมเอาหน่วยความจำ และ I/O Port ให้รวมอยู่บน Chip ตัวเดียวกัน

Programmable System

- ในระบบ Microprocessors หรือใน Chip ที่สร้างจากเทคโนโลยีที่ต่ำกว่า (SSI, MSI) การกำหนด Function การทำงานนั้น ไม่ได้ขึ้นอยู่กับ Hardware เพียงอย่างเดียว เช่น
  - ในระบบ Microprocessors สามารถทำการเขียนโปรแกรม ให้มีการทำงานในลักษณะที่ต่างกันออกไปได้
  - PLDs (Programmable Logic Devices) สามารถทำการโปรแกรมได้ว่า จะให้ทำงานตาม Function ทาง Logic Function ใด

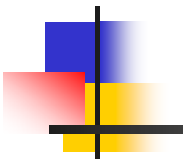
การพัฒนาโปรแกรม

- Program (Application Program) ที่จะใช้สั่งให้ Microprocessor ทำงานจะถูกเก็บไว้ในหน่วยความจำ ในลักษณะของเลข ฐาน 2 ซึ่งอาจถูกพัฒนาขึ้นได้ในลักษณะต่าง ๆ เช่น
  - Machine Language ซึ่งจะเป็นข้อมูลเลขฐาน 2
  - Assembly Language จะเขียนโดยการใช้ Mnemonics Code ในการเขียน เช่น ADD, SUB, LD หรือ JP เป็นต้น จากนั้นจึงใช้ Software ทำการแปลงให้เป็น Machine Language อีกครั้งหนึ่ง



การพัฒนาโปรแกรม

- High Level Language ใน Microprocessor บางตัวสามารถทำการเขียนโปรแกรมโดยใช้ ภาษาระดับสูง แล้วจึงทำการใช้ตัวแปลภาษา (Compilers) ทำการแปลงให้เป็น Machine Language ภาษาระดับสูง เช่น C , Pascal เป็นต้น
- การเขียนโปรแกรมโดยใช้ High Level Language จะให้ประสิทธิภาพการทำงานต่ำกว่า การเขียนด้วย Assembly Language ประสิทธิภาพนี้วัดจาก จำนวนเนื้อที่ ที่ใช้ในหน่วยความจำ และเวลาที่ใช้ในการทำงาน ประสิทธิภาพของโปรแกรมที่เขียนโดย High Level Language จะต่ำกว่าประมาณ 10 - 200%



Number system
(Review)

The Decimal Number System

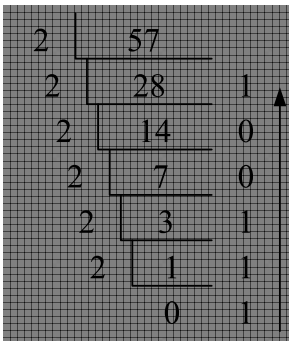
- ประกอบไปด้วย สัญลักษณ์ใช้แทนค่า จำนวน 10 ตัว คือ 0 - 9
- น้ำหนักของตำแหน่ง
  - ในกรณีหน้าจุด จะเป็น 10^{+} เริ่มจาก 10^{0} 10^{1} 10^{2} …
  - ในกรณีหลังจุด จะเป็น 10^{-} เริ่มจาก 10^{-1} 10^{-2} 10^{-3} ….
- ตัวอย่างเช่น
  - $257.32 = 2 \times 10^{2} + 5 \times 10^{1} + 7 \times 10^{0} + 3 \times 10^{-1} + 2 \times 10^{-2}$

The Binary Number System

- ประกอบไปด้วย สัญลักษณ์ใช้แทนค่า จำนวน 2 ตัว คือ 0 , 1
- น้ำหนักของตำแหน่ง
  - ในกรณีหน้าจุด จะเป็น 2^{+} เริ่มจาก 2^{0} 2^{1} 2^{2} …
  - ในกรณีหลังจุด จะเป็น 2^{-} เริ่มจาก 2^{-1} 2^{-2} 2^{-3} ….
- ตัวอย่างเช่น
  - $101.10_{2} = 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} + 1 \times 2^{-1} + 1 \times 2^{-2}$

Decimal to Binary Conversion

- ใช้วิธีการหารสั้นด้วย 2
- หารไปเรื่อย ๆ จนกว่าจนกว่าผลลัพธ์ที่ได้จะเป็น ศูนย์
- พิจารณาเฉพาะเศษที่ได้จากการหาร แต่ละครั้ง
- ตัวอย่างเช่น $57_{10} = ?_2$



$57_{10} = 111001_2$

Decimal to Binary Conversion

- ในกรณีของจุดทศนิยม จะใช้วิธีคูณจุดทศนิยมนั้นด้วย 2 แล้วพิจารณาตัวเลขที่อยู่หน้าจุด ที่ได้จากการคูณ แล้วนำตัวเลขทศนิยมไปคูณด้วยเลข 2 อีก ทำเช่นนี้เรื่อย ๆ จนกว่าตัวเลขที่อยู่หลังจุด จะเป็น ศูนย์ หมด

0.34375_{10} $= ?_2$

2 x 0.34375 $= \underline{0}.6875$

2 x 0.6875 $= \underline{1}.375$

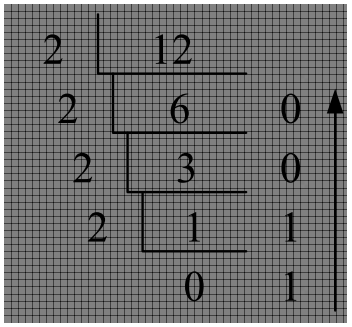
2 x 0.375 $= \underline{0}.75$

2 x 0.75 $= \underline{1}.5$

2 x 0.5 $= \underline{1}.0$ คำตอบคือ 0.01011_2

Decimal to Binary Conversion

- ในกรณีที่มีทั้งตัวเลข ที่อยู่ข้างหน้าจุดและหลังจุด ให้คิดแยกกันแล้วนำมารวมกันภายหลัง
- $12.125_{10} = ?_2$



$2 \times 0.125 = 0.25$

$2 \times 0.25 = 0.5$

$2 \times 0.5 = 1.0$

$12_{10} = 1100_2$ $\quad 0.125_{10} = 0.001_2$

- คำตอบ $12.125_{10} = 1100.001_2$

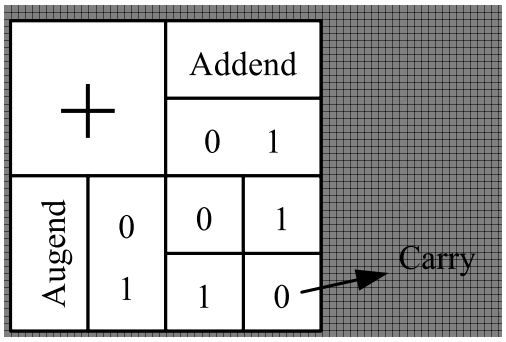
Hexadecimal vs. Binary Number

- เลขฐาน 16 และ ฐาน 2 มีความสัมพันธ์กัน สามารถทำการการแปลงฐานซึ่งกันและกันได้
- เลขฐาน 2 จำนวน 4 หลัก จะเท่ากับเลขฐาน 16 จำนวน 1 หลัก โดยเริ่มนับแบ่งตั้งแต่จุดทศนิยมไปทางซ้ายและขวา
- ถ้าไม่ครบ 4 หลักให้ทำการเพิ่มเลข 0 เข้าไปจนครบ 4 หลัก
- เช่น $1001111010.010101_2 = ?_{16}$

0010	0111	1010 .	0101	0100_2
2	7	A .	5	4_{16}

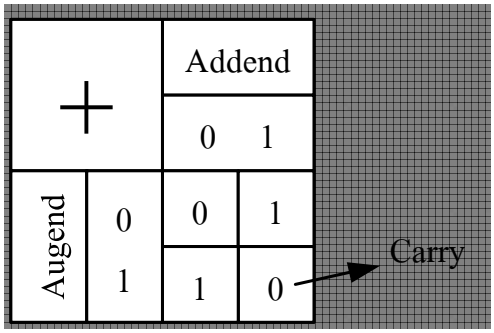
Binary Addition

Term	Binary Addition
Carry	*1 1 1 1 1 1 1*
Addend	0 1 1 0 0 0 1 1
Augend	0 1 0 1 1 1 1 1
Sum	1 1 0 0 0 0 1 0



Binary Addition

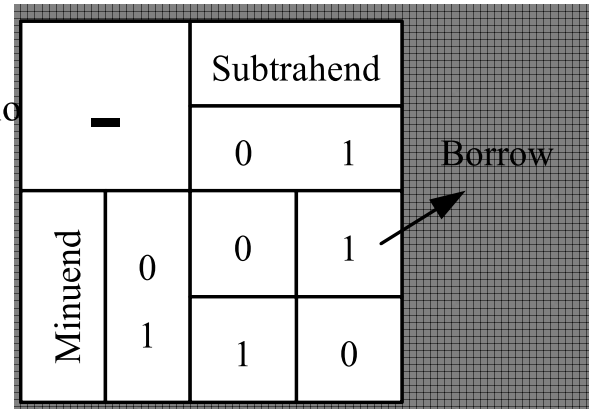
Term	Binary Addition
Carry	0 0 0 1 1 1 1
Addend	1 0 0 0 1 0 0 1
Augent	0 0 0 0 1 1 1 1
Sum	1 0 0 1 1 0 0 0



Term	Binary Addition
Carry	1 0 1 1 1 1 0 1 1 1 1 1 1 0 0
Addend	0 1 0 0 0 0 1 0 0 1 1 0 1 1 0 0
Augent	0 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0
Sum	1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0

Binary Subtraction

Term	Binary Subtractio
Borrow	0 1 1 0 0 0 0
Minuend	0 1 1 0 1 1 0 1
Subtrahend	0 0 1 1 0 0 0 1
Difference	0 0 1 1 1 1 0 0



Binary Subtraction

Term	Binary Subtraction
Borrow	0 0 0 1 1 1 0
Minuend	1 1 1 1 0 0 0 1
Subtrahend	0 0 0 0 0 0 1 1
Difference	1 1 1 0 1 1 1 0

Term	Binary Subtraction
Borrow	1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0
Minuend	1 0 0 1 0 0 1 1 1 0 0 1 1 0 0 1
Subtrahend	0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1
Difference	0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 1

การเก็บตัวเลขแบบคิดเครื่องหมาย

- แบบ Sign and Magnitude method
  - จะใช้ Bit ที่แรกสุด ในการเก็บเครื่องหมาย ถ้าเป็น 0 คือ ค่าบวก และถ้าเป็น 1 จะเป็น ค่าลบ เช่น

Binary	Decimal
0001 1100	+28
1001 1100	-28

การเก็บตัวเลขแบบคิดเครื่องหมาย

- แบบ One's Complement
  - จะให้ Bit แรกเป็น Sign Bit เช่นกัน แต่ถ้าหากข้อมูลเป็นลบ จะทำการกลับ Bit ของข้อมูลจาก 0 เป็น 1 และ จาก 1 เป็น 0

Binary	Decimal
0001 1100	+28
1110 0011	-28

การเก็บตัวเลขแบบคิดเครื่องหมาย

- Two's Complement
  - จะกระทำเช่นเดียวกับ การทำ 1's แต่ในกรณีที่ข้อมูลมีค่าเป็นลบ จะทำการบวกเพิ่มขึ้นไปอีก 1

Binary	Decimal
0001 1100	+28
1110 0100	-28

การลบเลขด้วยวิธี 1's complement

- นำเอาตัวลบไปทำ 1's
- นำเอาตัวตั้งและตัวลบ (1's) มารวมกัน แต่จำนวนของหลังของตัวตั้งและตัวลบต้องเท่ากัน
- ผลที่ได้จากการรวมกัน
  - ถ้ามีตัวทด ให้นำไปบวกเพิ่มจากผลที่ได้ คำตอบจะมีมาเป็นค่าบวก
  - ถ้าไม่มีตัวทด ให้นำผลที่ได้ ไปทำ 1's อีกครั้งหนึ่ง คำตอบจะออกมาเป็นค่าลบ

การลบเลขด้วยวิธี 1's complement

1 0100 0000 - 1110 1101

1's ของ 0 1110 1101 = 1 0001 0010

1 0100 0000

\+

1 0001 0010

(1) 0 0101 0010 * มีตัวทด

\+

1

0 0101 0011

คำตอบคือ 1010011

การลบเลขด้วยวิธี 1's complement

1 0010 1000 - 10 1001 0010

1's ของ 10 1001 0010 = 01 0110 1101

01 0010 1000

\+

01 0110 1101

(0) 10 1001 0101 * ไม่มีตัวทด

01 0110 1010 * ทำ 1's

คำตอบคือ - 1 0110 1010

การลบเลขด้วยวิธี 2's

- นำเอาตัวลบไปทำ 2's
- นำเอาตัวตั้งและตัวลบ (2's) มารวมกัน แต่จำนวนของหลังของตัวตั้งและตัวลบต้องเท่ากัน
- ผลที่ได้จากการรวมกัน
  - ถ้ามีตัวทด ให้ตัดตัวทดทิ้ง คำตอบจะออกมาเป็นค่าบวก
  - ถ้าไม่มีตัวทด ให้นำผลที่ได้ ไปทำ 2's อีกครั้งหนึ่ง คำตอบจะออกมาเป็นค่าลบ

การลบเลขด้วยวิธี 2's

1 0100 0000 - 1110 1101

2's ของ 0 1110 1101 = 1 0001 0011

1 0100 0000

\+

1 0001 0011

~~1~~ 0 0101 0011 * มีตัวทด

0 0101 0011

คำตอบคือ 1010011

การลบเลขด้วยวิธี 2's

1 0010 1000 - 10 1001 0010

1's ของ 10 1001 0010 = 01 0110 1110

	01 0010 1000	
	01 0110 1110	+
(0)	10 1001 0110	* ไม่มีตัวทด
	01 0110 1010	* ทำ 2's

คำตอบคือ - 1 0110 1010

Binary Multiplication

X		Multiplication	
		0	1
Multiplier	0	0	0
	1	0	1

Binary Multiplication

- กำหนดให้ตัวเก็บผลรวมมีค่าเท่ากับ ศูนย์
- นำตัวคูณทำการ Shift ขวา 1 ครั้ง ถ้า Carry ที่ออกมาเป็น 1 ให้นำตัวตั้งไปบวกเพิ่ม ในตัวเก็บผลรวม ถ้า Carry ที่ออกมาคือ 0 ไม่ต้องนำไปบวกเพิ่ม
- ทำการ Shift ตัวตั้งไปทางซ้าย 1 ครั้ง
- ทำซ้ำในข้อ 2 และ 3 จนกว่า ข้อมูลในตัวคูณจะหมด

Binary Multiplication

0 0 0 1 0 0 0 1

x

0 0 0 0 1 1 0 0

0 0 0 0 0 0 0 0	**Shift ครั้งที่ 1 ได้ 0**
0 0 0 0 0 0 0 0	**Shift ครั้งที่ 2 ได้ 0**
0 0 0 1 0 0 0 1	**Shift ครั้งที่ 3 ได้ 1**
0 0 0 1 0 0 0 1	**Shift ครั้งที่ 4 ได้ 1**
0 0 0 0 0 0 0 0	**Shift ครั้งที่ 5 ได้ 0**
0 0 0 0 0 0 0 0	**Shift ครั้งที่ 6 ได้ 0**
0 0 0 0 0 0 0 0	**Shift ครั้งที่ 7 ได้ 0**
0 0 0 0 0 0 0 0	**Shift ครั้งที่ 8 ได้ 0**

0 0 0 0 0 0 0 1 1 0 0 1 1 0 0

การหาร (Division)

- ในกรณี เลขฐาน 10 ทำได้ดังนี้

```
        17
    ------
12 | 204
     12
     ---
     84
     84
     --
      0
```

Quotient

Divisor | Dividend

Binary Division

ขั้นตอนในการหารเลขฐาน 2

- (1) นำ Divisor มาทำ 2's โดยเพิ่ม Sign bit เข้าไป

12_{10} $= 1100_2$ จะได้

Sign bit ⟶ (0) 1100

1's 1 0011

2's 1 0100

- (2) นำค่าของ 2's ของ Divisor ที่ได้ ไปรวมกับ dividend (คือการลบนั่นเอง) โดยเขียนให้แต่ละหลังตรงกัน โดยเริ่มจากทางด้านซ้าย

Binary Division

- (3) ถ้าผลของ Sign bit ออกมาเป็น 0
  - แสดงว่า Quotient ในตำแหน่งนั้นคือ 1
  - แล้วทำการ Shift ผลที่ได้จากข้อ 2 ไปทางซ้าย 1 bit
  - แล้วกลับไปทำข้อ 2 ใหม่
- (4) ถ้าผลของ Sign bit ออกมาเป็น 1
  - แสดงว่า Quotient ในตำแหน่งนั้นคือ 0
  - จะต้องทำการ บวกเพิ่มเข้าไปอีกเท่ากับจำนวนของ Divisor แล้วทำการ Shift ผลที่ได้จากข้อ 2 ไปทางซ้าย 1 bit
  - แล้วกลับไปทำข้อ 2 ใหม่

Binary Division

- ทำเช่นนี้ไปเรื่อย ๆ จนกว่าผลที่ออกมาจาก การรวมจะมีค่าเท่ากับศูนย์
- ถ้าผลรวมออกมาเป็น ศูนย์ ในขณะที่จำนวนหลักของผลที่ได้มากกว่าจำนวนหลักของ Divisor ให้เพิ่มศูนย์ที่ Quotient เท่ากับจำนวนหลักที่เกินมา

Binary Division

- $204 \div 12$

$204_{10} = 11001100_2$

$12_{10} = 1100_2$

$= 0\ 1100$

1's $= 1\ 0011$

2's $= 1\ 0100$

0	1 1 0 0 1 1 0 0	Dividend
1	0 1 0 0 0 0 0 0	Subtract 12
0	0 0 0 0 1 1 0 0	

Quotient = 1x

Binary Division

0	0 0 0 1 1 0 0	Shift result
1	0 1 0 0 0 0 0	Subtract 12
1	0 1 0 1 1 0 0	

Quotient = 10x

1	0 1 0 1 1 0 0	
0	1 1 0 0 0 0 0	Add 12
0	0 0 0 1 1 0 0	

Binary Division

0	0 0 1 1 0 0	Shift result
1	0 1 0 0 0 0	Subtract 12
1	0 1 1 1 0 0	

Quotient = 100x

1	0 1 1 1 0 0	
0	1 1 0 0 0 0	Add 12
0	0 0 1 1 0 0	

Binary Division

0	0 1 1 0 0	Shift result
1	0 1 0 0 0	Subtract 12
1	1 0 1 0 0	

Quotient = 1000x

1	1 0 1 0 0	
0	1 1 0 0 0	Add 12
0	0 1 1 0 0	

Binary Division

0	1 1 0 0	Shift result
1	0 1 0 0	Subtract 12
0	0 0 0 0	

Quotient = 10001_2

= 17_{10}

Binary Division

$101000_2 \div 100_2$

100_2 = 0 100

1's = 1 011

2's = 1 100

0	1 0 1 0 0 0	Dividend
1	1 0 0 0 0 0	Subtract 100_2
0	0 0 1 0 0 0	Quotient = 1x

Binary Division

0	0 1 0 0 0	Shift result
1	1 0 0 0 0	Subtract 100_2
1	1 1 0 0 0	
		Quotient = 10x

1	1 1 0 0 0	
0	1 0 0 0 0	Add 100_2
0	0 1 0 0 0	

Binary Division

0	1 0 0 0	Shift result
1	1 0 0 0	Subtract 100_2
0	0 0 0 0	
		Quotient = 101x

- จำนวนหลัก มีมากกว่าตัวหารอยู่ 1 หลัก จะต้องเพิ่มศูนย์เข้าไปที่ Quotient อีก 1 ตัว

  คำตอบคือ 1010_2

Floating Point Number

- ในเลขฐาน 10

  $N = M \times 10^E$

  M = Mantissa E = Exponent

เช่น 65,535 สามารถเขียนได้เป็น

65535×10^0 6553.5×10^1

655.35×10^2 65.535×10^3

6.5535×10^4 0.65535×10^5

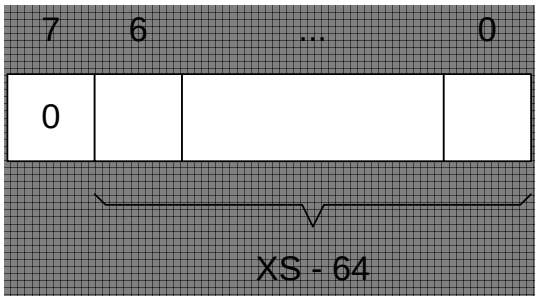
0.065535×10^6 0.0065535×10^7

Floating Point Format

- ในฐาน 2
  - Mantissa บางครั้งเรียกว่า Coefficient or Fraction
  - Exponent บางครั้งเรียกว่า Characteristic or Power
- Exponent
  - จะเก็บอยู่ในรูปแบบที่ Equivalent ของเลข 2'S ยกตัวอย่างเช่น การเก็บแบบ XS-64

Floating Point Format

- Exponent



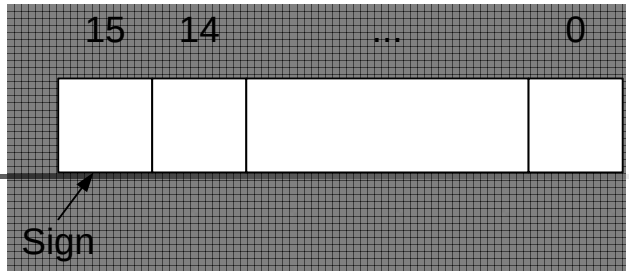
- Rang

0000 0000 to 0111 1111

-64 to +63

7 - bit	XS - 64
+ 63	111 1111
….	
+32	110 0000
….	
+1	100 0001
0	100 0000
-1	011 1111
….	
-32	010 0000
….	
-63	000 0001
-64	000 0000

- Mantissa



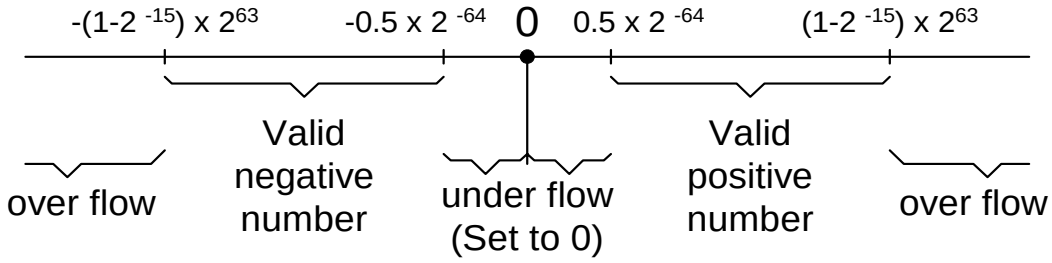
- Rang : 0100 0000 0000 0000 to 0111 1111 1111 1111

0.5 to $1-2^{-15}$

And 1100 0000 0000 0000 to 1000 0000 0000 0001

-0.5 to $-(1-2^{-15})$

Combined format $\pm$ (0.5 to $1.0-2^{-15}$)



- จากตัวอย่างสามารถ อ้างจำนวนได้

$(0.5 \times 2^{-64}$ to $(1-2^{-15}) \times 2^{63})$

$(2.7105 \times 10^{-19}$ to $0.9223 \times 10^{19})$

Binary Code Decimal Number

- หรือเรียกอีกอย่างหนึ่งว่า BCD number

Decimal Number	BCD Number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Addition of Unsigned BCD Number

- การบวกเลข BCD ทำการบวกเหมือนกับเลขฐาน 16 ธรรมดา
- ถ้าการบวกในหลักใด มีค่ามากกว่า 9 ให้ทำการบวกเพิ่มไปอีก 6

7 + 6 = ?

7 = 0000 0111

6 = <u>0000 0110</u>

0000 1101 เกิน 9

<u>0000 0110</u>

0001 0011 = 13

Addition of Unsigned BCD Number

67 + 34 = ?

7 = 0111

4 = <u>0100</u> +

1011 > 9

<u>0110</u> +

1 **0001** = 1

Carry

3 = 0011

6 = <u>0110</u> +

1001

<u>0001</u> + (Carry จาก 7+4)

1010 > 9

<u>0110</u> +

1 0000

คำตอบคือ 0001 0000 0001

1 0 1