ระบบคอมพิวเตอร์และสถาปัตยกรรม
(Computer System and Architecture)

บทที่ 3
การคำนวณทางคณิตศาสตร์ หน่วยประมวลผลทางคณิตศาสตร์และตรรกะ (ALU)

หน่วยประมวลผลทางคณิตศาสตร์และตรรกะ

ALU เป็นส่วนประกอบที่เป็นอิเล็กทรอนิกส์ที่ใช้พื้นฐานทางดิจิตอลลอจิก ทำหน้าที่ประมวลผลในคอมพิวเตอร์ทั้งทางด้านคณิตศาสตร์และตรรกะ โดยซีพียูจะมี ALU ประกอบอยู่ภายในซีพียู โดยซีพียูจะใช้งาน ALU ร่วมกับหน่วยควบคุม (Control Unit) รีจิสเตอร์ หน่วยความจำ และอุปกรณ์อินพุต/เอาต์พุต รูป 3.1 จะเห็นว่า ALU อยู่ภายในซีพียูโดยมีซีพียูควบคุมการทำงาน และมีรีจิสเตอร์สำหรับส่งข้อมูลเข้าสู่ ALU หลังจากนั้นผลลัพธ์ที่เกิดจากการประมวลผลของ ALU จะเก็บไว้ที่รีจิสเตอร์เช่นกัน รีจิสเตอร์นี้ทำหน้าที่เป็นหน่วยเก็บข้อมูลชั่วคราว หลังจากนั้นจะนำมาเก็บไว้ในหน่วยความจำโดยการควบคุมของซีพียู

ALU

รูป 3.1

การเปลี่ยนเป็นค่าตรงข้าม

- ใช้ sign-and-magnitude

  +21 = 00010101

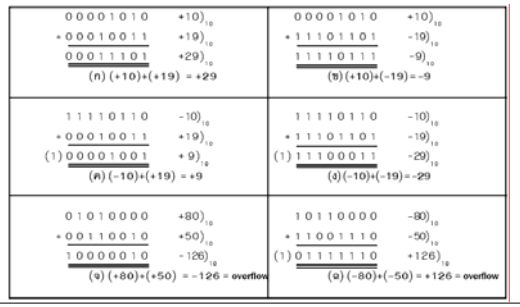
  -21 = 10010101 sign-and-magnitude

- ใช้ 2's complement

+21	=	00010101	2's complement
	=	11101010	กลับค่าแต่ละบิต
		+ 1	
-21	=	11101011	

การบวกและการลบ

- ใช้ 2's complement :
  - ใช้หลัก a-b = a+(-b)
  - ไม่คิดตัวทด และค่าสูงสุดไม่เกินของจำนวนบิต



การบวกและการลบ

- ใช้ 1's complement :
  - ถ้ามีตัวทดเกิดขึ้น จะนำไปบวกเข้ากับค่าผลลัพธ์ที่ได้

01110111 +119)10	10101111 -80)10
+ 11001101 -50)10	+ 11010111 -40)10
(1) 01000100	(1) 10000110
+ 1 end - around carry	+ 1 end - around carry
01000101 +69)10	10000111 -120)10
(ก) (+119)+(-50) = +69	(ข) (-80)+(-40) = -120

การคูณ

- จำนวนเต็มไม่มีเครื่องหมาย :
  - ตั้งตัวคูณให้ตำแหน่งขวาสุดตรงกับตัวตั้ง
  - ผลคูณย่อยที่เกิดขึ้นตำแหน่งขวาสุดให้ตรงตัวคูณ
  - นำผลคูณย่อยมารวมกัน

```
   1101     ตัวตั้ง (13)
 x 1011     ตัวคูณ (11)
 ------
   1101
  1101
 0000
1101
--------
10001111    ผลลัพธ์ (143)
```

การคูณ

- จำนวนเต็มมีเครื่องหมาย (บวก)

```
  0 0 0 1        (+1)10
x 0 0 1 1        (+3)10
---------
  0 0 0 1
  0 0 0 1
  0 0 0 0
  0 0 0 0
---------------
  0 0 0 0 0 0 1 1    (+3)10
===============
```


การคูณ

- จำนวนเต็มมีเครื่องหมาย (ลบ)

```
  1 1 1 1        (-1)10
x 0 0 1 1        (+3)10
---------
  1 1 1 1
  1 1 1 1
  0 0 0 0
  0 0 0 0
---------------
  0 0 1 0 1 1 0 1    (+45)10
===============
```

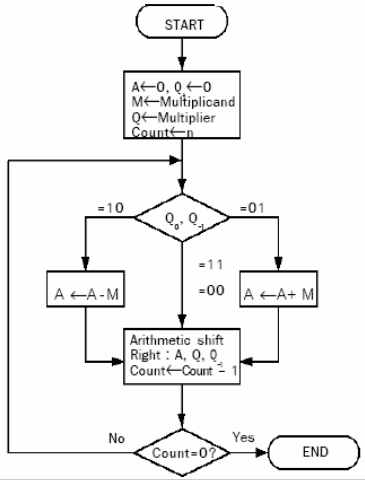
ผิด

```
  1 1 1 1 1 1 1 1    (-1)10
x         0 0 1 1    (+3)10
---------
  1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1
  0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0
---------------
  1 1 1 1 1 1 0 1    (-3)10
===============
```

ถูก (ขยายผลคูณย่อย)

การคูณ

- อัลกอริทึมของบูธ

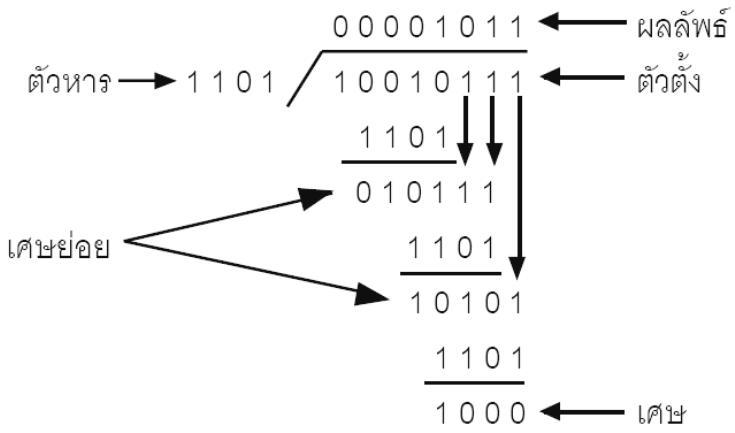


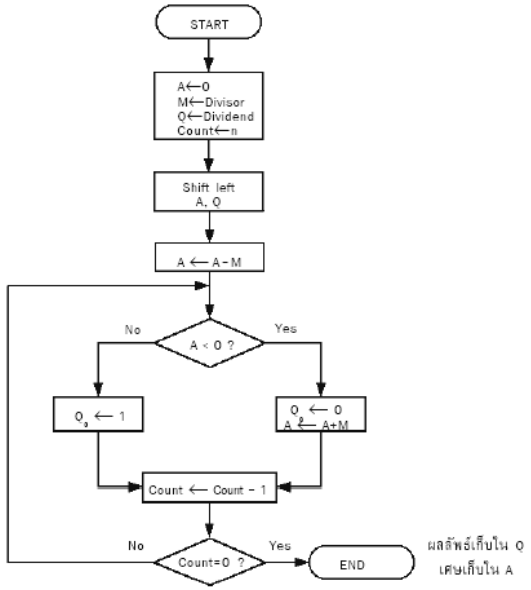
การคูณ

A (ผลลัพธ์)	Q (ตัวคูณ)	Q_{-1} (บิตพิเศษ)	M (ตัวตั้ง)		
0 0 0 0	0 0 1 1	0	0 1 1 1	ค่าเริ่มต้น	
1 0 0 1	0 0 1 1	0	0 1 1 1	A ← A – M (เนื่องจากเป็น 1-0)	วงรอบ 1
1 1 0 0	1 0 0 1	1	0 1 1 1	เลื่อนบิตทางขวา	
1 1 1 0	0 1 0 0	1	0 1 1 1	เลื่อนบิตทางขวา (เนื่องจากเป็น 1-1)	วงรอบ 2
0 1 0 1	0 1 0 0	1	0 1 1 1	A ← A + M (เนื่องจากเป็น 0-1)	วงรอบ 3
0 0 1 0	1 0 1 0	0	0 1 1 1	เลื่อนบิตทางขวา	
0 0 0 1	0 1 0 1	0	0 1 1 1	เลื่อนบิตทางขวา (เนื่องจากเป็น 0-0)	วงรอบ 4

ผลลัพธ์คือ 00010101 = $+21)_{10}$

การหาร

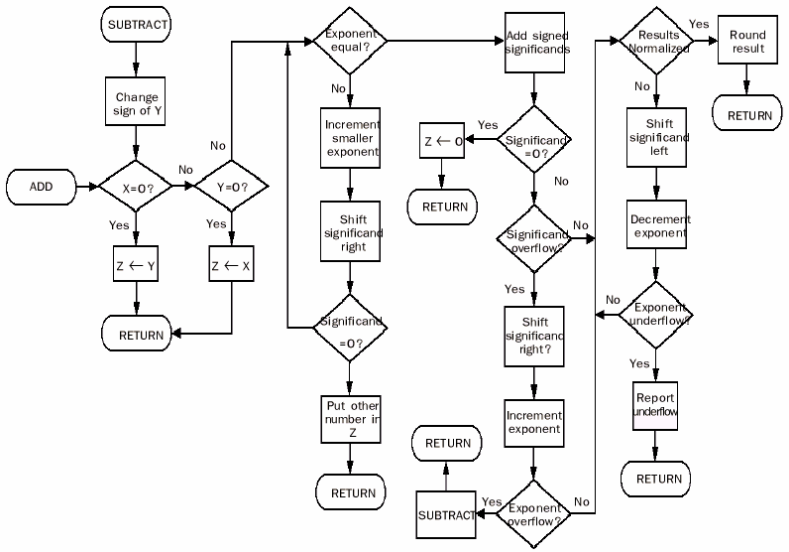




A	Q	M = 0011	A	Q	M=1101
0 0 0 0	0 1 1 1	Initial value	0 0 0 0	0 1 1 1	Initial value
0 0 0 0	1 1 1 0	shift	0 0 0 0	1 1 1 0	shift
1 1 0 1		subtract	1 1 0 1		add
0 0 0 0	1 1 1 0	restore	0 0 0 0	1 1 1 0	restore
0 0 0 1	1 1 0 0	shift	0 0 0 1	1 1 0 0	shift
1 1 1 0		subtract	1 1 1 0		add
0 0 0 1	1 1 0 0	restore	0 0 0 1	1 1 0 0	restore
0 0 1 1	1 0 0 0	shift	0 0 1 1	1 0 0 0	shift
0 0 0 0		subtract	0 0 0 0		add
0 0 0 0	1 0 0 1	set Q_0=1	0 0 0 0	1 0 0 1	set Q_0=1
0 0 0 1	0 0 1 0	shift	0 0 0 1	0 0 1 0	shift
1 1 1 0		subtract	1 1 1 0		add
0 0 0 1	0 0 1 0	restore	0 0 0 1	0 0 1 0	restore
(ก) (7) / (3)			(ข) (7) / (-3)		

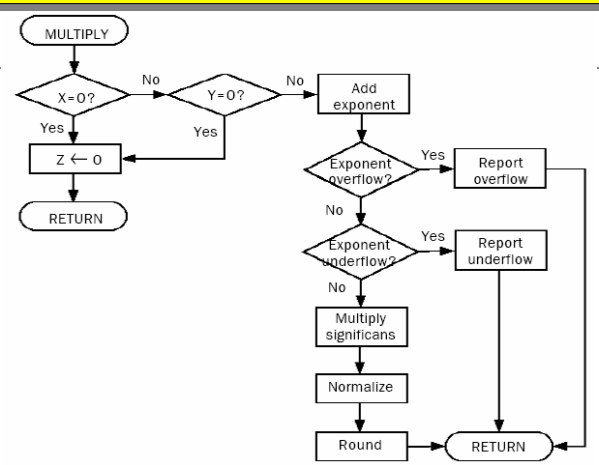
การบวกและลบเลขทศนิยม

- การตรวจสอบค่า 0
- การปรับเลขชี้กำลังให้เท่ากัน
- ทำการบวกหรือลบค่าของจำนวนนั้น (Mantissa)
- ปรับให้อยู่ในรูปแบบทั่วไป



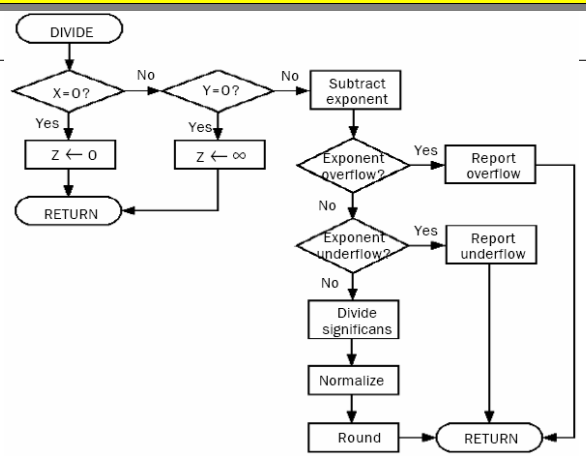
โฟลว์ชาร์ตแสดงการบวกและลบเลขทศนิยม (Z <--X+-Y)

การคูณเลขทศนิยม



แสดงการคูณเลขทศนิยม (Z <--X x Y)

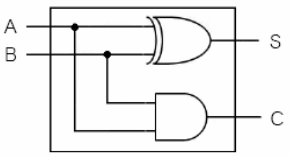
การหารเลขทศนิยม



แสดงการหารเลขทศนิยม (Z <--X / Y)

Haft Adder และ Full Adder

A	B	ผลรวม (Sum)	ตัวทด (Carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



1 = A

0 = B

1 = C_{in}

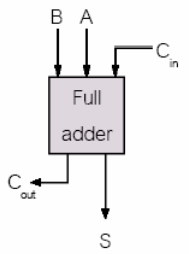
1 0

C_{out} S

ตารางค่าความจริงของการบวกเลข 2 บิต และวงจรสำหรับการบวกเลข 2 บิต

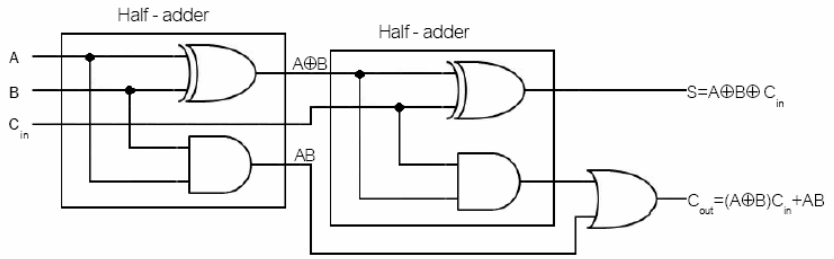
Haft Adder และ Full Adder

A	B	ตัวทดเข้า (C_{in})	ผลรวม (Sum)	ตัวทด (C_{out})
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



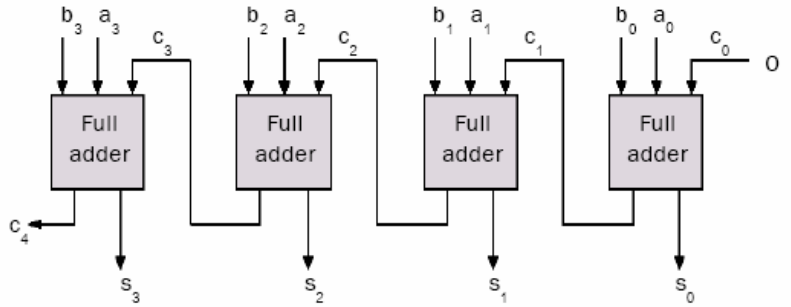
ตารางค่าความจริงการบวกเลข 3 บิต (2 บิตและมีตัวทด) และสัญลักษณ์ Full Adder

Haft Adder และ Full Adder



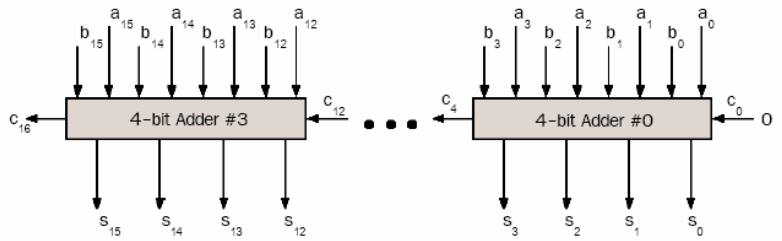
วงจร Full Adder ที่เกิดจาก Haft Adder 2 ตัว

Ripple-Carry Adder



Ripple-Carry Adder (ขนาด 4 บิต)

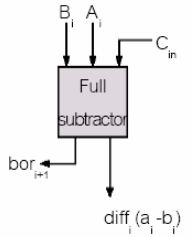
Ripple-Carry Adder



ตัวบวกขนาด 16 บิตที่เกิดจากการเรียงต่อกันของ Ripple-Carry Adder 4 ตัว

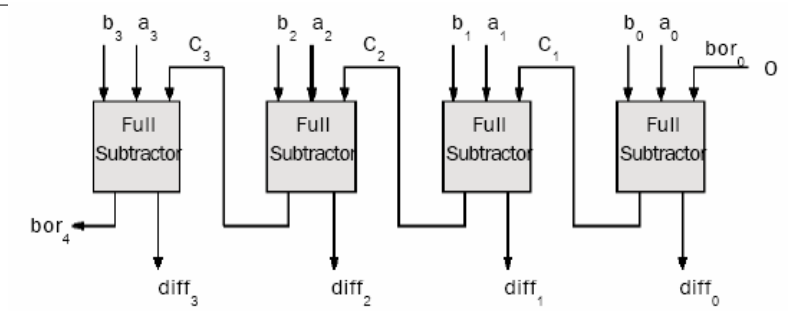
Full Subtractor

A_i	B_i	ตัวยืม (bor_i)	ผลลบ (Diff)	ตัวยืม (bor_{i+1})
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



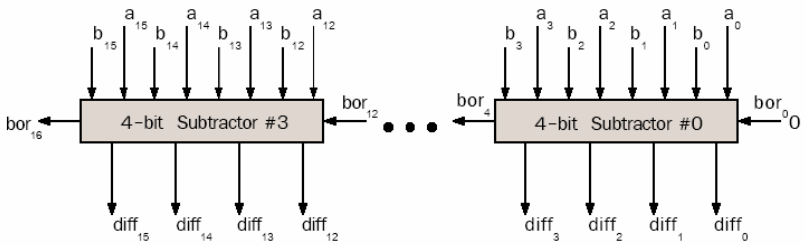
ตารางค่าความจริงและสัญลักษณ์ของ Full Subtractor

Ripple-Borrow Subtractor



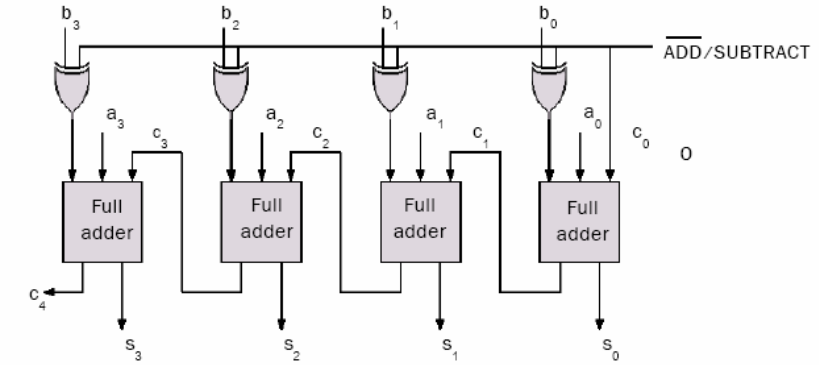
Ripple-Borrow Subtractor

Ripple-Borrow Subtractor



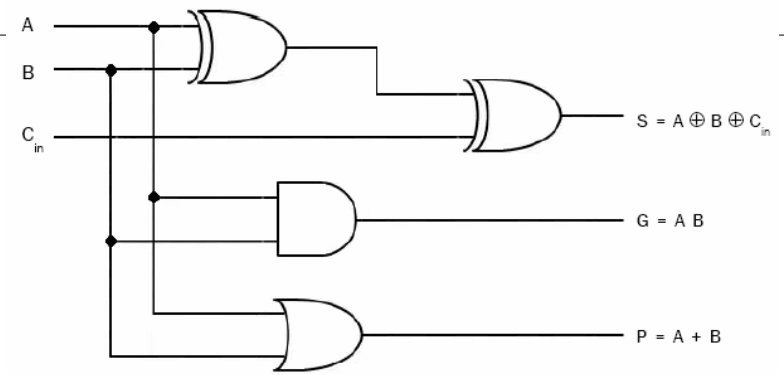
ตัวลบขนาด 16 บิตที่เกิดจากการเรียงต่อกันของ Ripple-Borrow Subtractor 4 ตัว

วงจรบวกและลบ



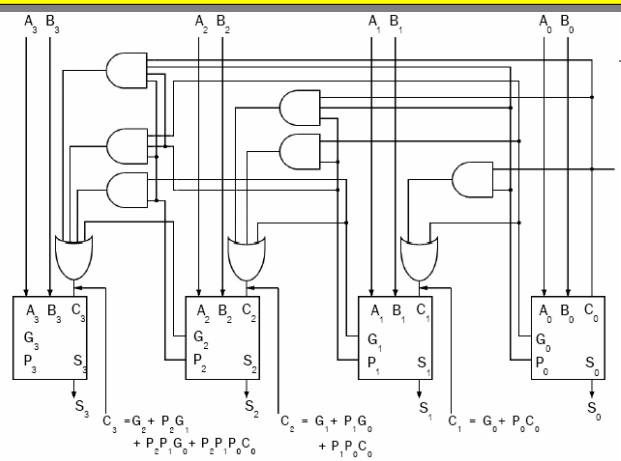
วงจรบวกและลบ

Carry-Lookahead



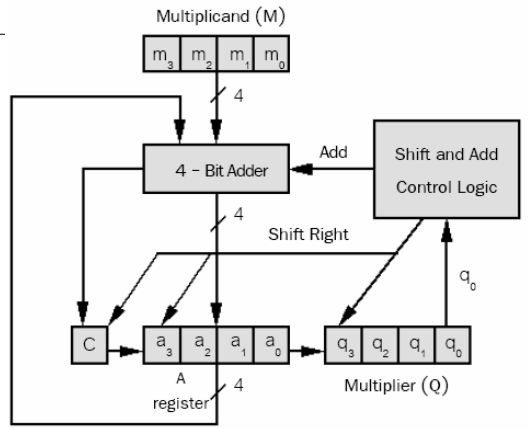
Full Adder ที่สร้างเอาต์พุตตัวทดก่อกำเนิด (G) และตัวทดแพร่ (P) สำหรับใช้ใน Carry-Lookahead

Carry-Lookahead



Full Adder ที่รวม Carry-Lookahead โดย Full Adder

Serial Multiplier



Serial Multiplier

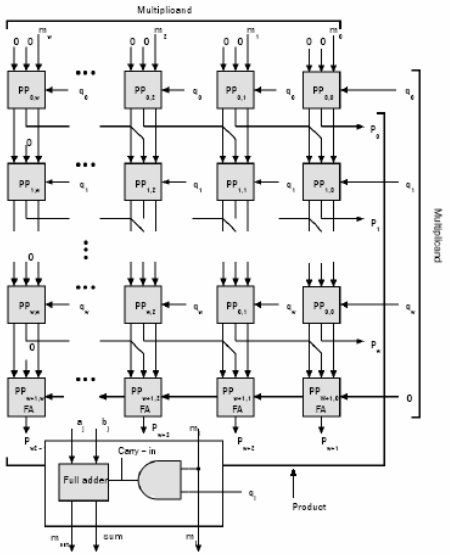
ตัวอย่างการคูณที่ใช้ Serial Multiplier

Multiplicand (M)

1 1 0 1 ← Initial Value

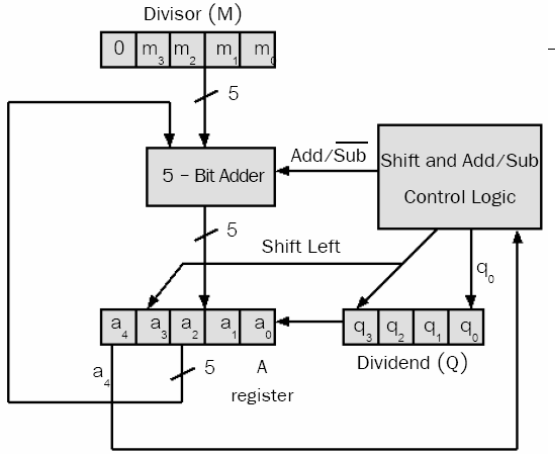
C	A	Q	
0	0 0 0 0	1 0 1 1	← Initial Value
0	1 1 0 1	1 0 1 1	Add M to A
0	0 1 1 0	1 1 0 1	Shift
1	0 0 1 1	1 1 0 1	Add M to A
0	1 0 0 1	1 1 1 0	Shift
0	0 1 0 0	1 1 1 1	Shift (no add)
1	0 0 0 1	1 1 1 1	Add M to A
0	1 0 0 0	1 1 1 1	Shift

Product



Array Multiplier

Serial Divider



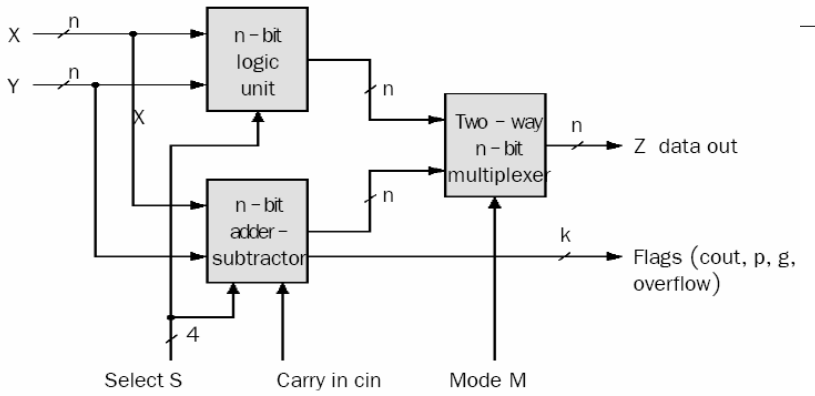
Serial Divider

A	Q	
0 0 0 0 0	0 1 1 1	Inotial value
0 0 0 0 0	1 1 1 0	Shift Left
1 1 1 0 1	1 1 1 0	Subtract M from A
0 0 0 0 0	1 1 1 0	Restore A (Add M to A)
0 0 0 0 0	1 1 1 0	Clear q_0
0 0 0 0 1	1 1 0 0	Shift Left
1 1 1 1 0	1 1 0 0	Subtract M from A
0 0 0 0 1	1 1 0 0	Restore A
0 0 0 0 1	1 1 0 0	Clear q_0
0 0 0 1 1	1 0 0 0	Shift Left
0 0 0 0 0	1 0 0 0	Subtract M from A
0 0 0 0 0	1 0 0 1	Set q_0
0 0 0 0 1	0 0 1 0	Shift Left
1 1 1 1 0	0 0 1 0	Subtract M from A
0 0 0 0 1	0 0 1 0	Restore A
0 0 0 0 1	0 0 1 0	Clear q_0

Remainder Quotient

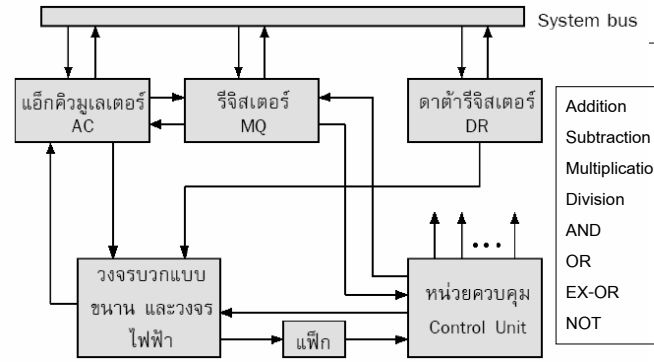
ตัวอย่างการหารโดยใช้ Serial Divider

Combination ALU



Combination ALU

Sequential ALU



Addition	AC := AC + DR
Subtraction	AC := AC – DR
Multiplication	AC.MQ := DR x MQ
Division	AC.MQ := MQ/DR
AND	AC := AC AND DR
OR	AC := AC OR DR
EX-OR	AC := AC XOR DR
NOT	AC := NOT(AC)

Sequential ALU