10. 컴퓨터 산술 연산

산술 연산

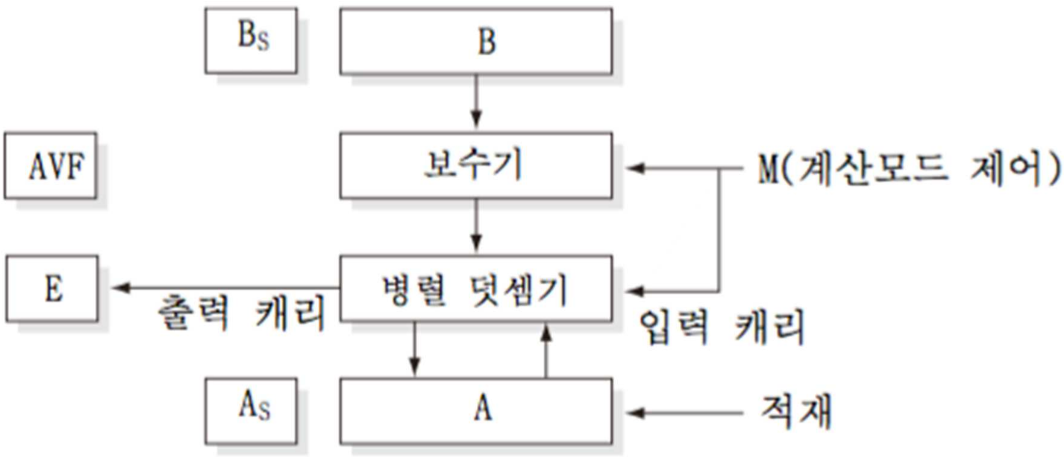
- 수를 대상으로 한 연산
- 덧셈, 뺄셈, 곱셈, 나눗셈
- 데이터
  - 이진 또는 십진
  - 고정 소수점 또는 부동 소수점
  - 정수 : 부호-절대값, 보수 표현(예: 1의 보수, 2의 보수)

덧셈과 뺄셈

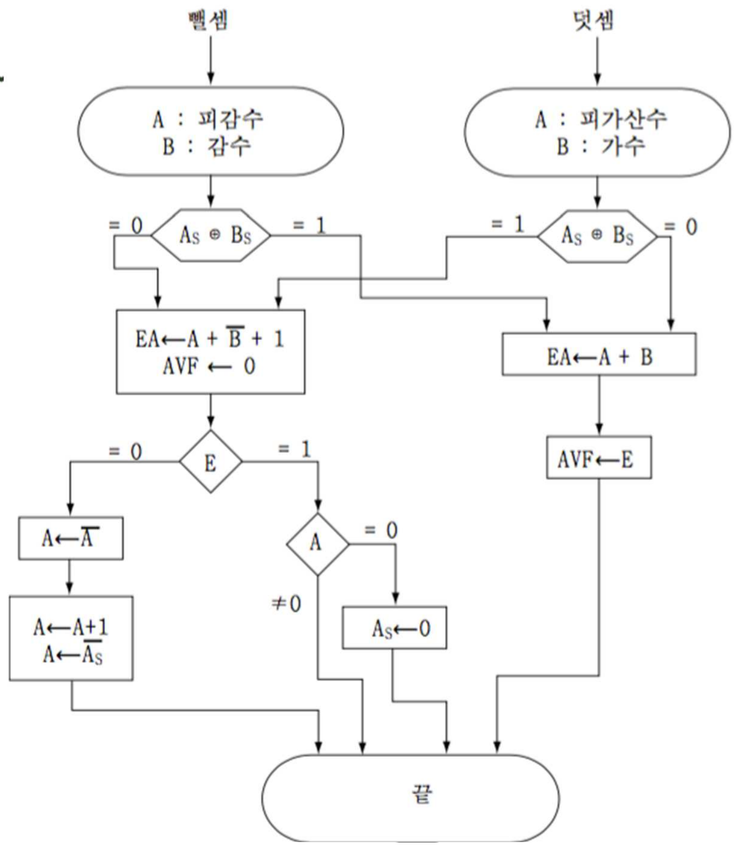
부호-절대값 덧셈/뺄셈

연산	크기 덧셈	크기 뺄셈		
		$A > B$	$A < B$	$A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

부호-절대값 덧셈/뺄셈 하드웨어 구성

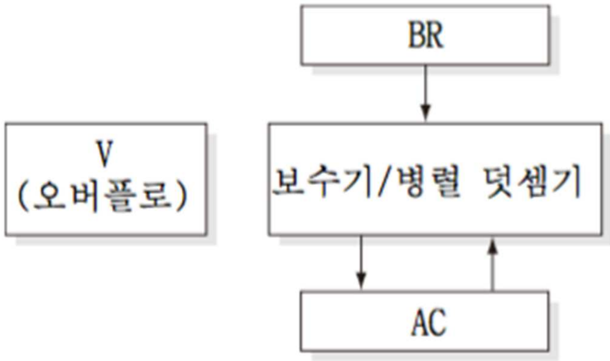


부호-절대값 연산 알고리즘

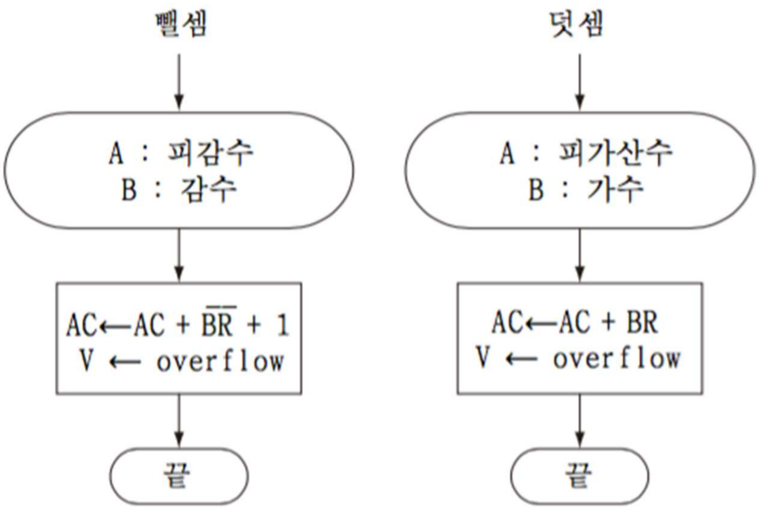


2의 보수 데이터 덧셈/뺄셈

- 부호 역할을 하는 비트도 다른 비트와 동일하게 연산에 사용
- ‘그냥‘ 덧셈함.
- 하드웨어



2의 보수 데이터 덧셈/뺄셈 알고리즘



곱셈 알고리즘

부호-절대값 형식의 곱셈

- 시프트와 덧셈 연산의 반복

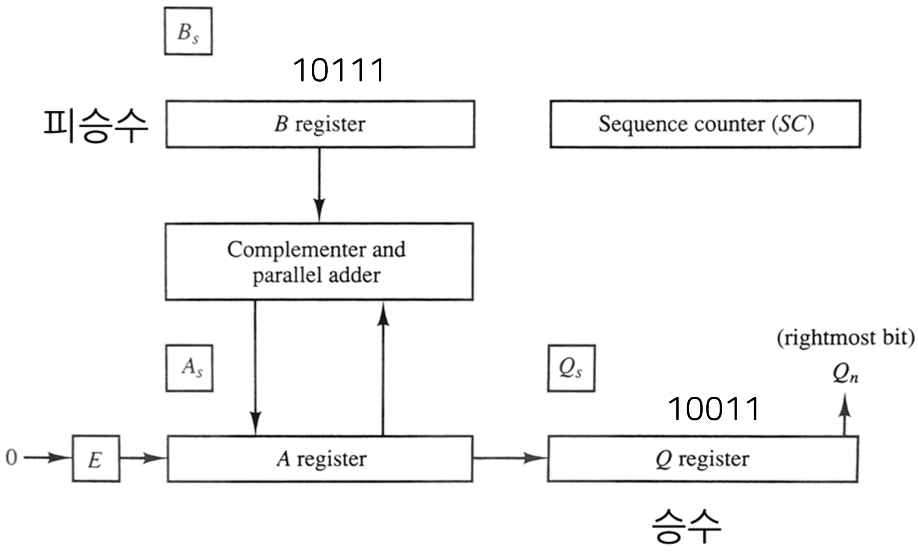
23	× 10111		Multiplicand
19	10011		Multiplier
	10111		
	10111		
	00000	+	
	00000		
	10111		
437	110110101		Product

1. 피승수를 부분곱에 덧셈
2. 부분곱을 오른쪽 시프트
3. 승수의 비트가 0이면 시프트만

결과의 부호:

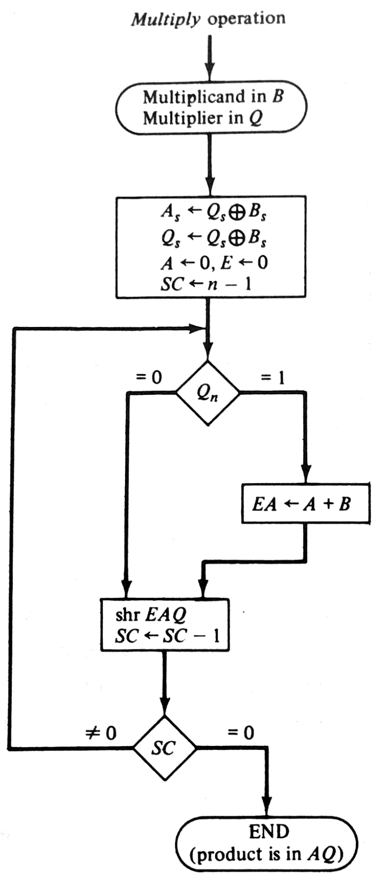
1. 두 수의 부호가 같으면 : +
2. 두 수의 부호가 다르면 : -

부호-절대값 형식의 곱셈 하드웨어



10011	10111_10011 0_01011_11001
11001	1_00010_11001 0_10001_01100
01100	0_10001_01100 0_01000_10110
10110	0_01000_10110 0_00100_01011
01011	0_11011_01011 0_01101_10101

부호-절대값 형식의 곱셈 알고리즘



Booth 곱셈 알고리즘

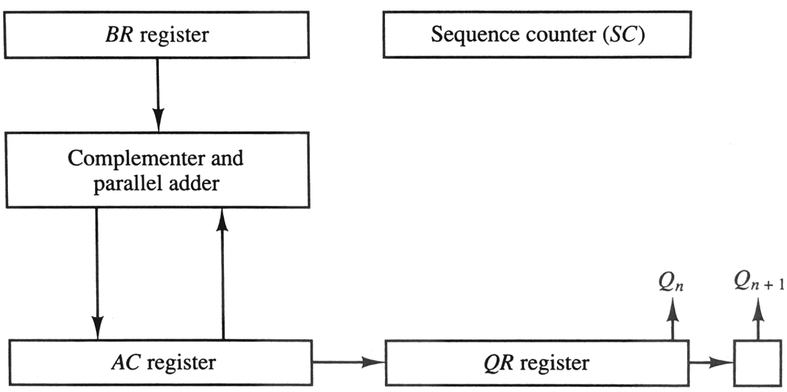
- 2의 보수 표기법의 곱셈

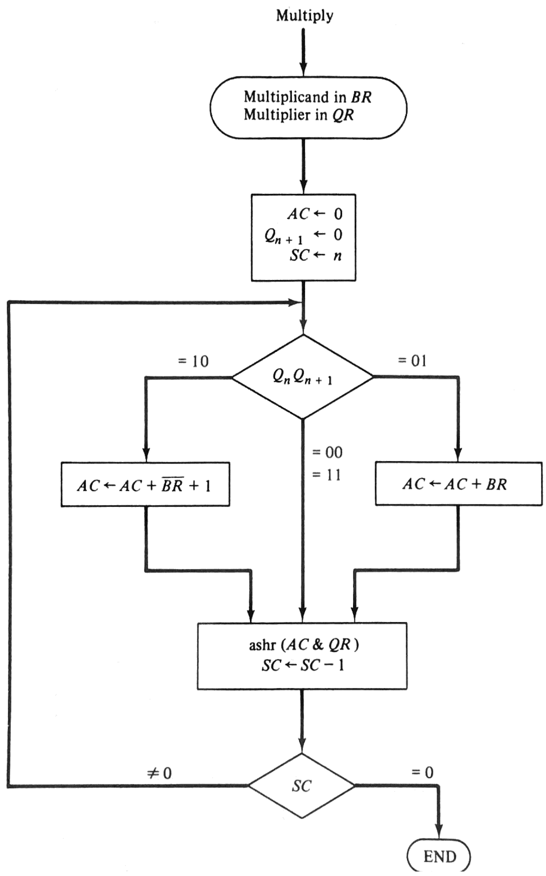
$$N = \cdots 0\ \overset{2^i}{1}\ 1\ 1\ \overset{2^j}{1}\ 0 \cdots = 2^{(i+1)} - 2^j$$

$$M \times N = \cdots + M \times 2^{(i+1)} - M \times 2^j + \cdots$$

  - 승수를 오른쪽에서 왼쪽으로 비트 단위로 스캔
  - 처음 1을 만나면 부분곱에서 피승수를 뺌
  - 1이 계속되다 0이 나오면 부분곱에서 피승수를 더함
  - 이전 비트와 같은 비트가 나오면 변경없음.

Booth 곱셈 알고리즘 하드웨어





Booth 곱셈 알고리즘의 예

BR: 10111(-9)

AC QR
00000_10011_0 -
01001_10011_0 sh

00100_11001_1 sh

00010_01100_1 +
11001_01100_1 sh

11100_10110_0 sh

11110_01011_0 –
00111_01011_0 sh

$\boxed{00011_10101}$_1

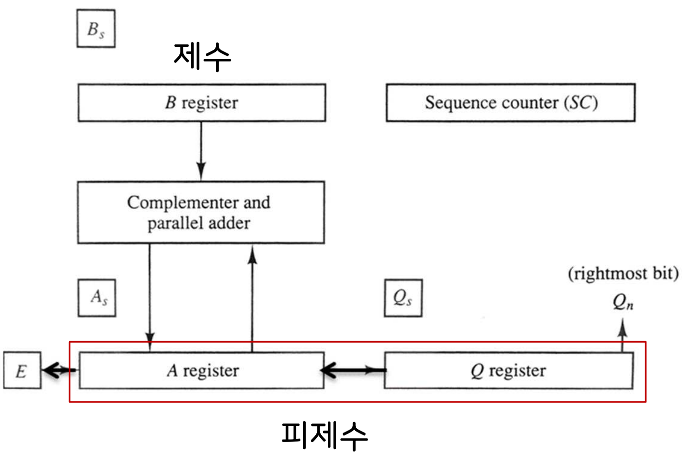
나눗셈 알고리즘

부호-절대값 정수의 나눗셈

- 비교, 시프트, 뺄셈 연산으로 구성
  - '비교 결과 큰 수이면 몫은 1'← 이진수 이므로

Divisor:	11010	Quotient = Q
B = 10001	)0111000000	Dividend = A
	01110	5 bits of $A < B$, quotient has 5 bits
	011100	6 bits of $A \geqslant B$
	-10001	Shift right B and subtract; enter 1 in Q
	-010110	7 bits of remainder $\geqslant B$
	--10001	Shift right B and subtract; enter 1 in Q
	--001010	Remainder $< B$; enter 0 in Q; shift right B
	---010100	Remainder $\geqslant B$
	----10001	Shift right B and subtract; enter 1 in Q
	----000110	Remainder $< B$; enter 0 in Q
	-----00110	Final remainder

부호-절대값 정수의 나눗셈



Divisor B = 10001, $\quad \overline{B} + 1$ = 01111

	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\overline{B}$ + 1		01111		
E = 1	1	01011		
Set Q_n = 1	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\overline{B}$ + 1		01111		
E = 1	1	00101		
Set Q_n = 1	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\overline{B}$ + 1		01111		
E = 0; leave Q_n = 0	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2
shl EAQ	0	10100	01100	
Add $\overline{B}$ + 1		01111		
E = 1	1	00011		
Set Q_n = 1	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\overline{B}$ + 1		01111		
E = 0; leave Q_n = 0	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A:		00110		
Quotient in Q:			11010	

A, B 크기 비교

- (A-B) 한 후 캐리(E) 값 확인
- A≥B 이면 E = 1

$$\begin{aligned}&\boldsymbol{A-B}\\&=\boldsymbol{A+(-B)}\\&=\boldsymbol{A+\ 2^n-B}\\&=\boldsymbol{2^n+(A-B)}\end{aligned}$$

- A < B이면 E = 0

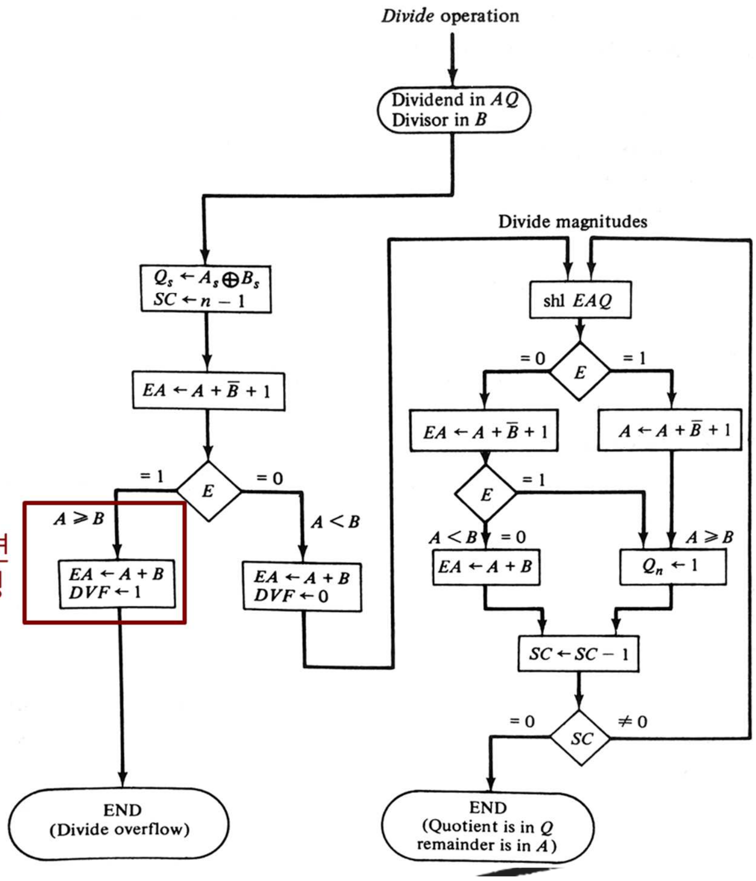
$$\begin{aligned}&\boldsymbol{A-B}\\&=\boldsymbol{A+(-B)}\\&=\boldsymbol{A+\ 2^n-B}\\&=\boldsymbol{2^n+(A-B)}\\&=\boldsymbol{2^n-(B-A)}\end{aligned}$$

나눗셈 오버플로

- '피제수(*2n* 비트)의 상위 *n* 비트의 값이 제수(*n*비트)보다 큰 경우'

➔몫은 (*n+1*)비트가 필요

나눗셈 알고리즘



처음부터 A≥B이면
몫에 저장에 오버플로 발생

부동 소수점 산술 연산

부동 소수점 덧셈/뺄셈

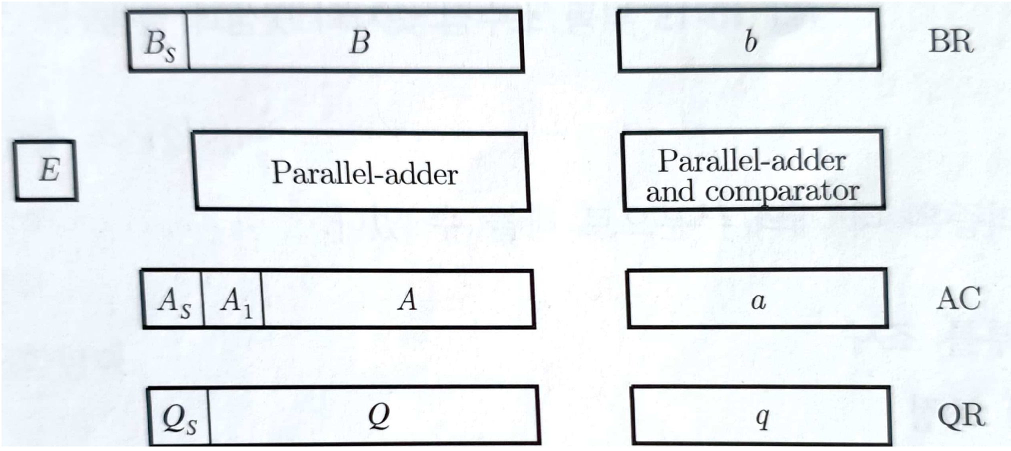
- 단계
  - 0인지 조사
  - 지수가 같도록 가수 조정
  - 가수 덧셈/뺄셈
  - 결과 정규화

※ 산술 파이프라인 참고

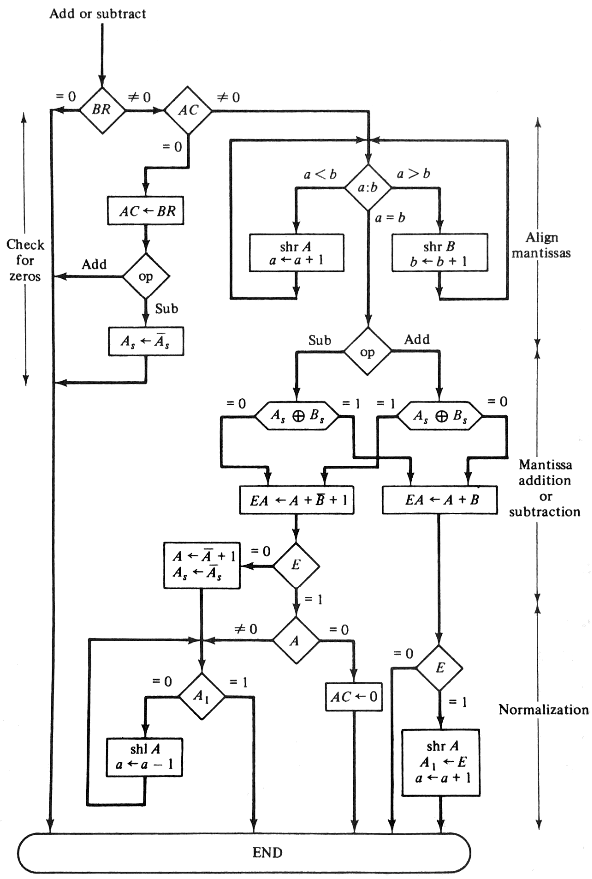
부동 소수점 곱셈/나눗셈

- 곱셈
  - 0인지 조사
  - 지수를 더함
  - 가수를 곱함
  - 결과를 정규화
- 나눗셈
  - 0인지 조사
  - 부호 결정
  - 피제수의 위치 조정
  - 지수 뺄셈
  - 가수 나눗셈

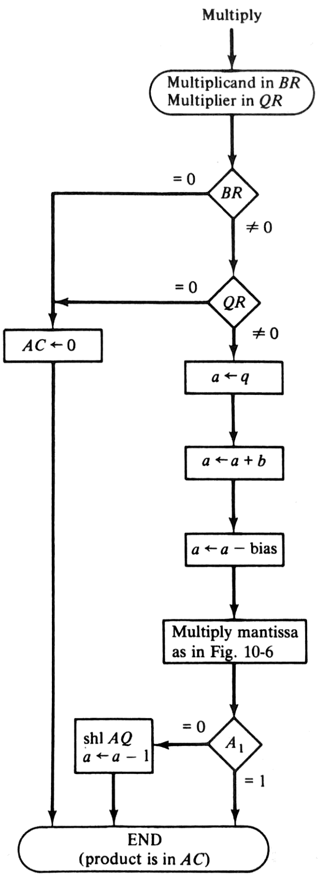
부동 소수점 연산을 위한 레지스터



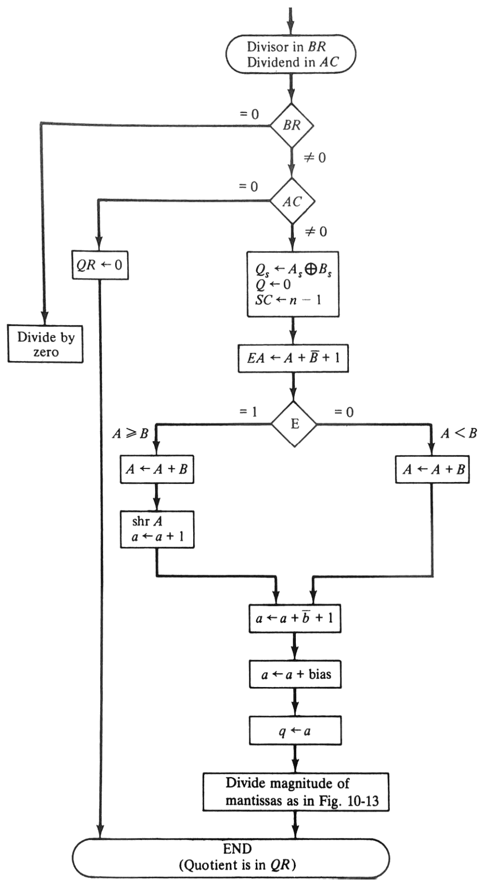
부동 소수점 덧셈/뺄셈



부동소수점 곱셈



부동 소수점 나눗셈



Q&A