COMPUTER ARITHMETIC

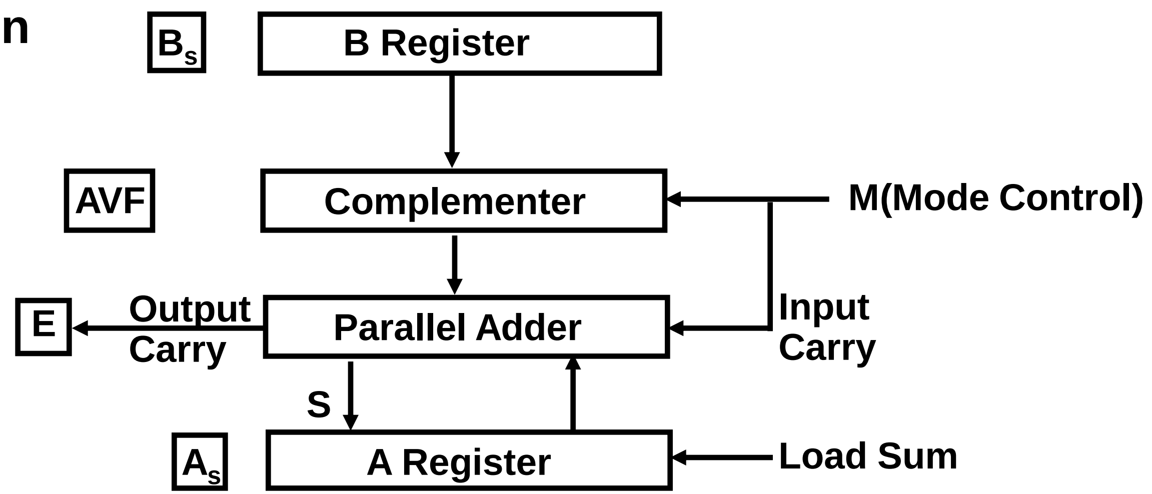
- **Arithmetic with Signed-2's Complement Numbers**
- **Multiplication and Division**
- **Floating-Point Arithmetic Operations**
- **Decimal Arithmetic Unit**
- **Decimal Arithmetic Operations**

SIGNED MAGNITUDEADDITION AND SUBTRACTION

Addition: A + B ; A: Augend; B: Addend
Subtraction: A - B: A: Minuend; B: Subtrahend

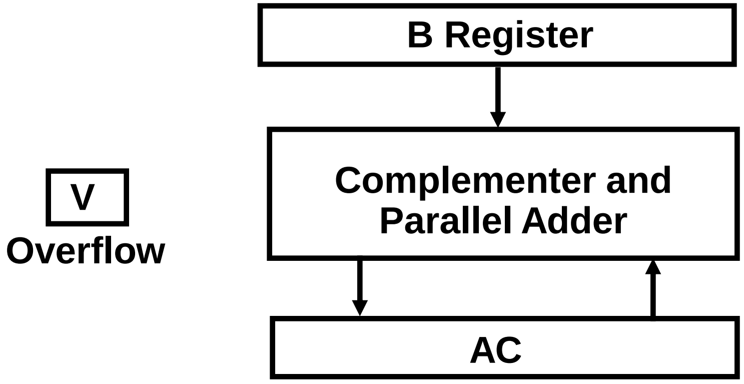
Operation	Add Magnitude	Subtract Magnitude		
		When A>B	When A<B	When A=B
(+A) + (+B)	+(A + B)			
(+A) + (- B)		+(A - B)	- (B - A)	+(A - B)
(- A) + (+B)		- (A - B)	+(B - A)	+(A - B)
(- A) + (- B)	- (A + B)			
(+A) - (+B)		+(A - B)	- (B - A)	+(A - B)
(+A) - (- B)	+(A + B)			
(- A) - (+B)	- (A + B)			
(- A) - (- B)		- (A - B)	+(B - A)	+(A - B)

Hardware Implementation

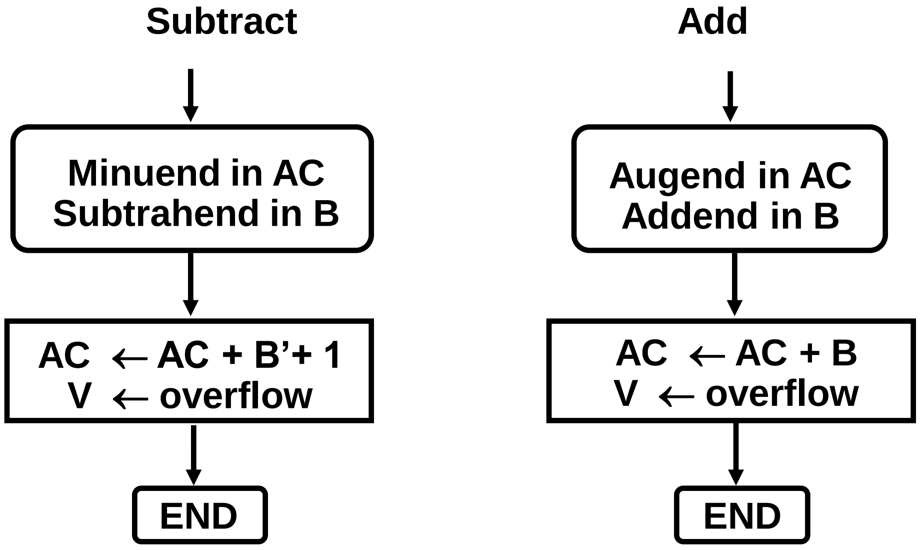


SIGNED 2'S COMPLEMENT ADDITION AND SUBTRACTION

Hardware



Algorithm



MULTIPLICATION

Multiplication: B * A; B: Multiplicand; A: Multiplier; P: Partial Product

Multiplication of Unsigned Positive Numbers

$$A = A_{n-1}A_{n-2} \dots A_0$$
$$B = B_{n-1}B_{n-2} \dots B_0$$

$$P = B * A$$
$$= B * \left(\sum_{i=0}^{n-1} 2^i * A_i \right)$$
$$= A_{n-1} * (\underbrace{B2^{n-1}}_{\text{B shifted left n-1 bits}}) + A_{n-2} * (\underbrace{B2^{n-2}}_{\text{B shifted left n-2 bits}}) + \dots + A_0 * (\underbrace{B2^{0}}_{\text{B shifted left 0 bits = A}})$$

Or

$$P = A_{n-1}*(\underbrace{\overline{B2^{n-1}} * 2^0}_{B2^{n-1}}) + A_{n-2}*(\underbrace{\overline{B2^{n-1}} * 2^{-1}}_{B2^{n-1}\text{ shifted right 1 bit}}) + \dots + A_0*(\underbrace{\overline{B2^{n-1}} * 2^{-(n-1)}}_{B2^{n-1}\text{ shifted right (n-1) bits}})$$

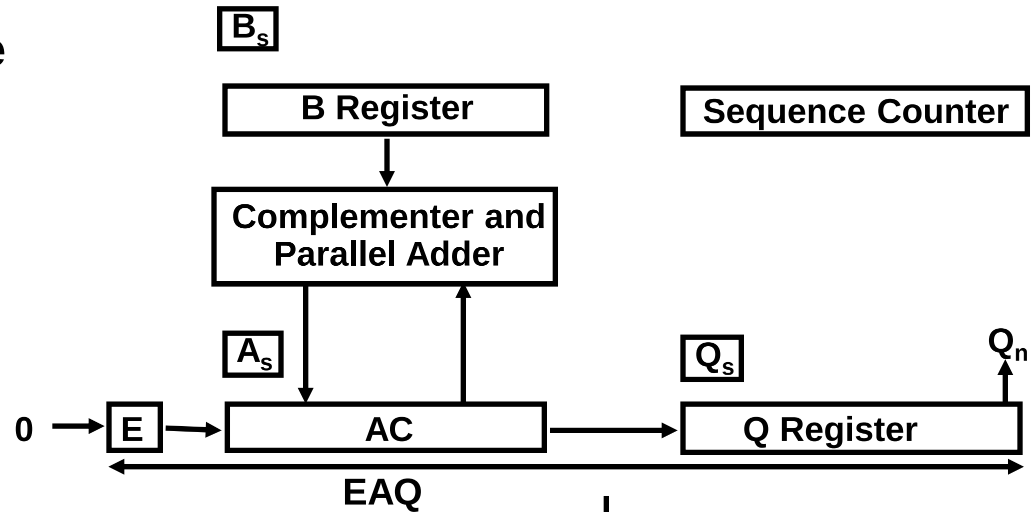
B shifted (n-1) bits to the left

EXAMPLE

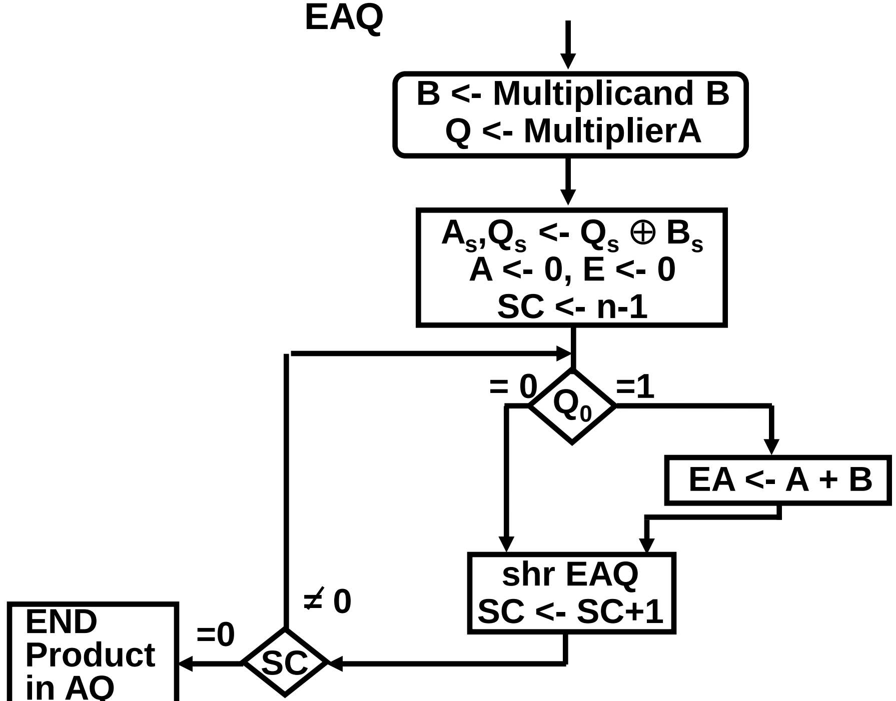
Multiplicand B=10111	E	A	Q	SC
Multiplier in Q	0	00000	1001$\boxed{1}$	101
Q_0 = 1; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	1100$\boxed{1}$	100
Q_0 = 1; add B		10111		
Second Partial Product	1	00010		
Shift right EAQ	0	10001	0110$\boxed{0}$	011
Q_0 = 0; shift right EAQ	0	01000	1011$\boxed{0}$	010
Q_0 = 0; shift right EAQ	0	00100	0101$\boxed{1}$	001
Q_0 = 1; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final Product in AQ = 0110110101				

SIGNED MAGNITUDE MULTIPLICATION

Hardware



Algorithm



BOOTH MULTIPLICATION ALGORITHM FOR SIGNED 2'S COMPLEMENT

Multiplier

Strings of 0's: No addition; Simply shifts

Strings of 1's: String of 1's from m_p to m_q: $2^{p+1} - 2^q$

Example

001110 (14) -> p = 3, q = 1

$001110 = 2^{3+1} - 2^1$

$M * 14 = M2^4 - M2^1$

Algorithm

[1] Subtract multiplicand for the first least significant 1 in a string of 1's in the multiplier

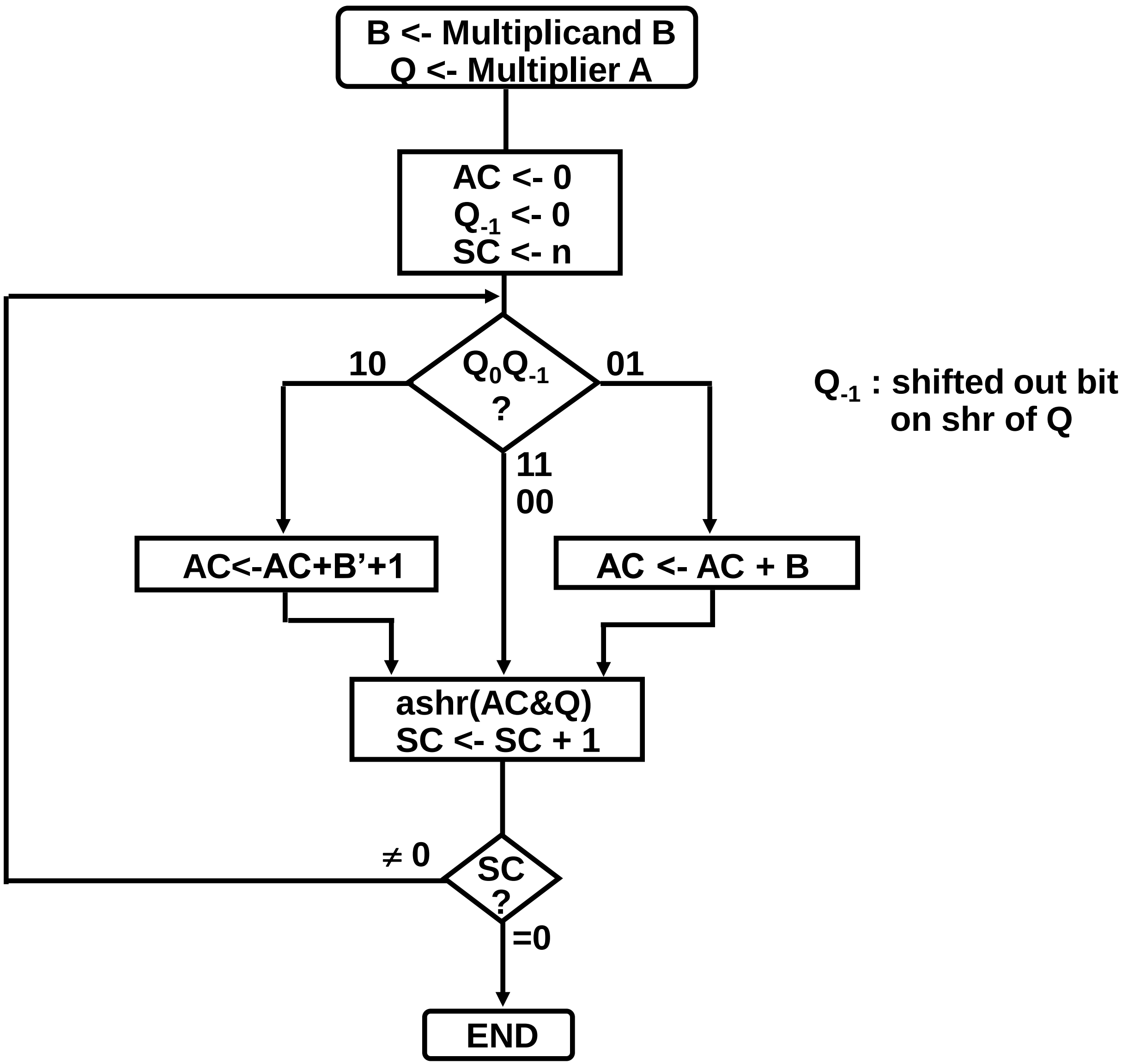
[2] Add multiplicand for the first 0 after the string of 1's in the multiplier

[3] Partial Product does not change when the multiplier bit is identical to the previous bit

$110010 = -2^4 + 2^2 - 2^1 = -16 + 4 - 2 = -14$

subtract 2^4 Add 2^2 subtract 2^1

BOOTH ALGORITHM FOR SIGNED 2'S COMPLEMENT



EXAMPLE OF BOOTH MULTIPLIER

B = 10111

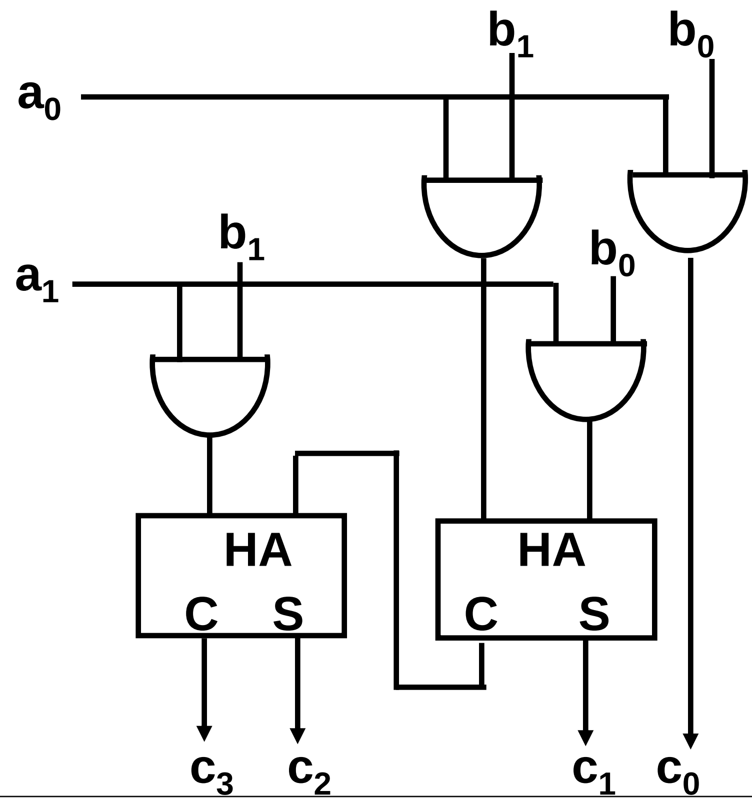
Q_0Q_{-1}	B'+1=01001	AC	Q	Q_{-1}	SC
	Initial	00000	10011	0	101
10	Subtract B	01001			
		01001			
	ashr	00100	11001	1	100
11	ashr	00010	01100	1	011
01	Add B	10111			
		11001			
	ashr	11100	10110	0	010
00	ashr	11110	01011	0	001
10	Subtract B	01001			
		00111			
	ashr	00011	10101	1	000

ARRAY MULTIPLIER

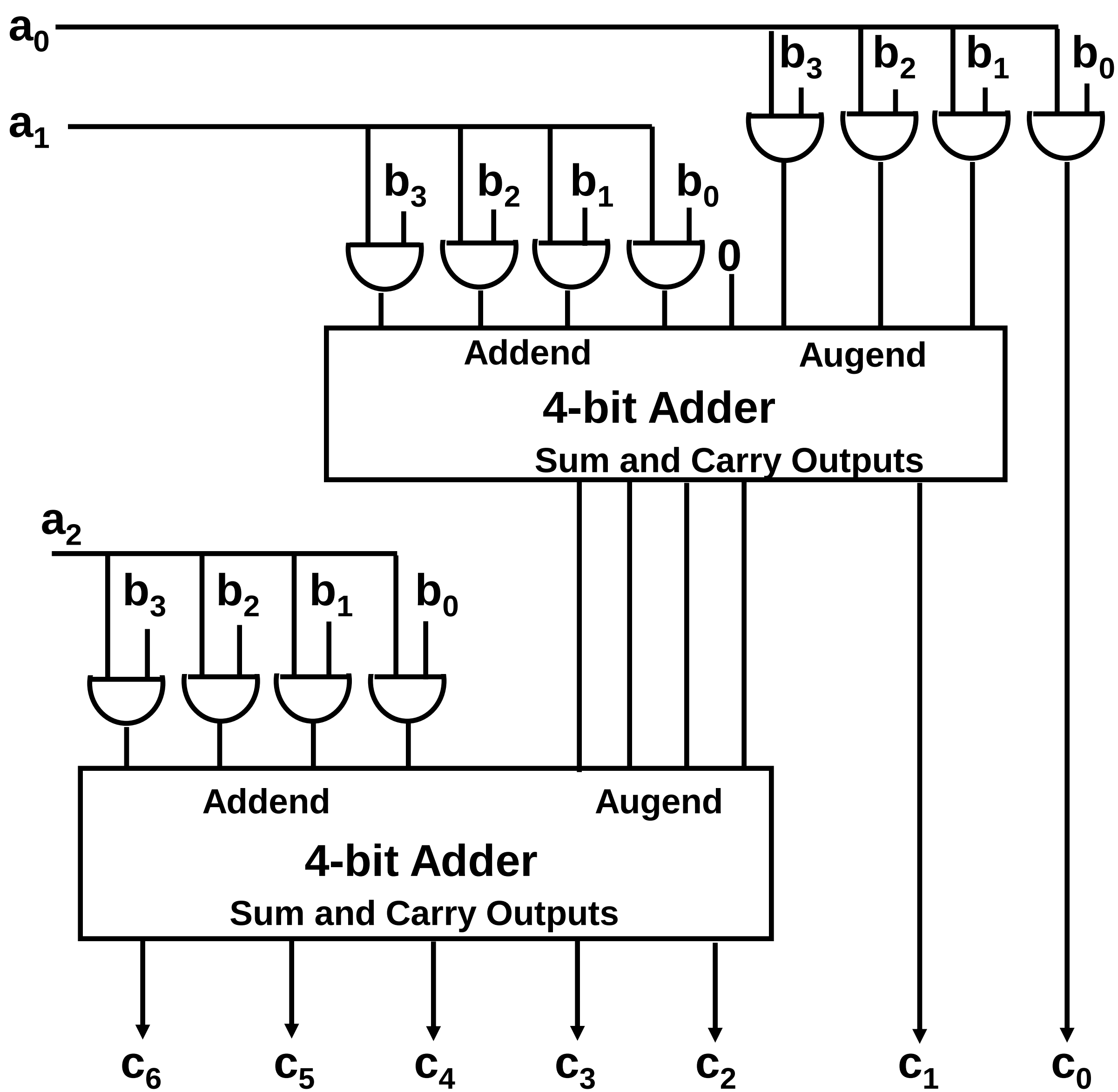
$A = a_1 a_0$: Multiplier
$B = b_1 b_0$: Multiplicand

$C = B * A = c_3 c_2 c_1 c_0$

$$
\begin{array}{cccc}
 & & b_1 & b_0 \\
 & & a_1 & a_0 \\
\hline
 & & a_0 b_1 & a_0 b_0 \\
 & a_1 b_1 & a_1 b_0 & \\
\hline
c_3 & c_2 & c_1 & c_0
\end{array}
$$



ARRAY MULTIPLIER 4-BIT X 3-BIT



DIVISION

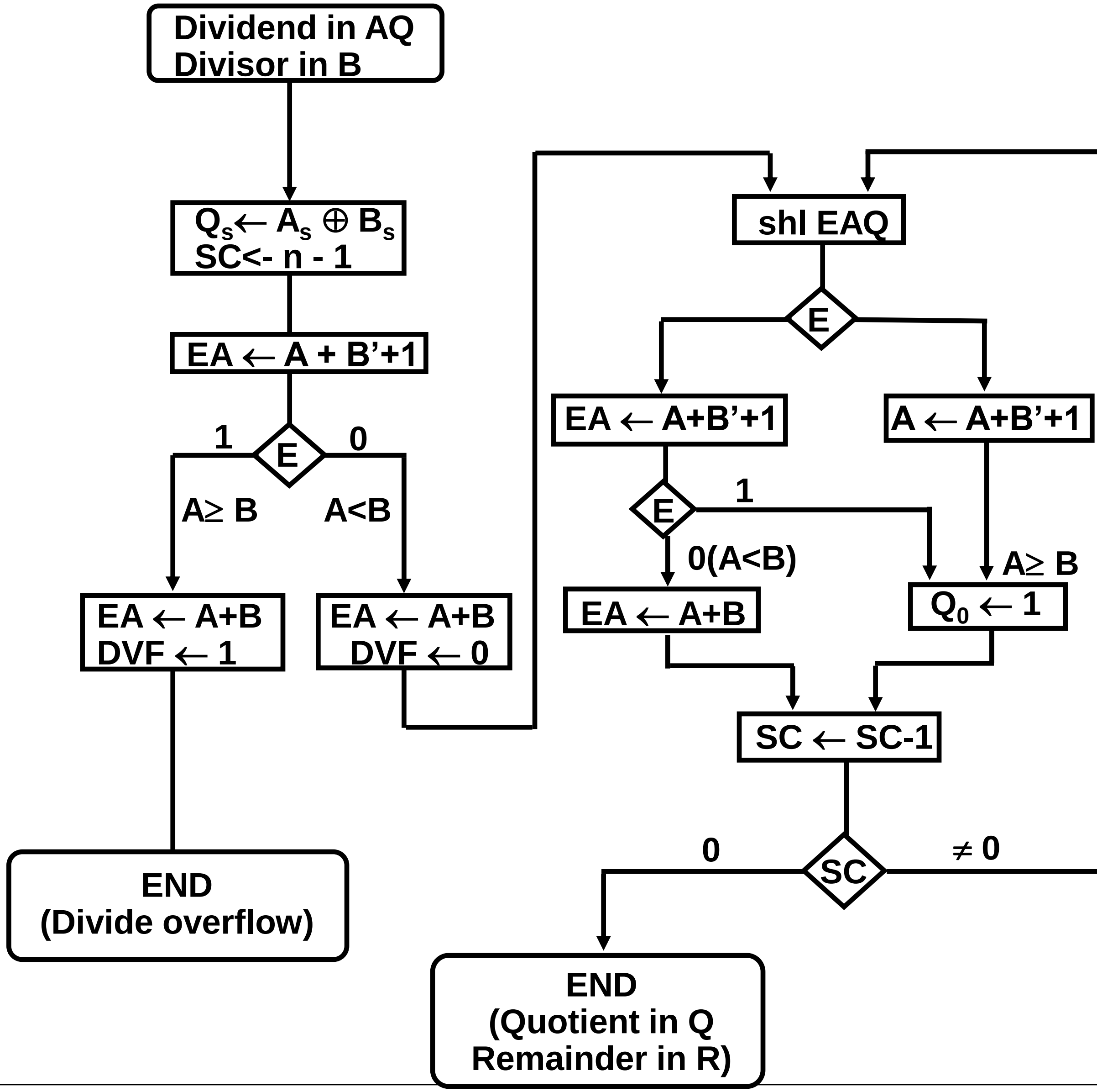
A / B = Q + R

A: Dividend; B: Divisor; Q: Quotient; R: Remainder

Divisor B = 10001, B'+ 1 = 01111

	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add B'+1		01111		
E=1	1	01011		
Set Q_0=1	1	01011	00001	4
shl EAQ	0	10110	00010	
Add B'+1		01111		
E=1	1	00101		
Set Q_0=1	1	00101	00011	3
shl EAQ	0	01010	00110	
add B'+1		01111		
E=0; Q_0=0	0	11001	00110	
add B		10001		
restore remainder	1	01010		2
shl EAQ	0	10100	01100	
add B'+1		01111		
E=1	1	00011		
Set Q_0=1	1	00011	01101	1
shl EAQ	0	00110	11010	
add B'+1		01111		
E=0; Q_0=0	0	10101	11010	
add B		10001		
restore remainder	1	00110	11010	0
neglect E				
remainder in A		00110		
quotient in Q			11010	

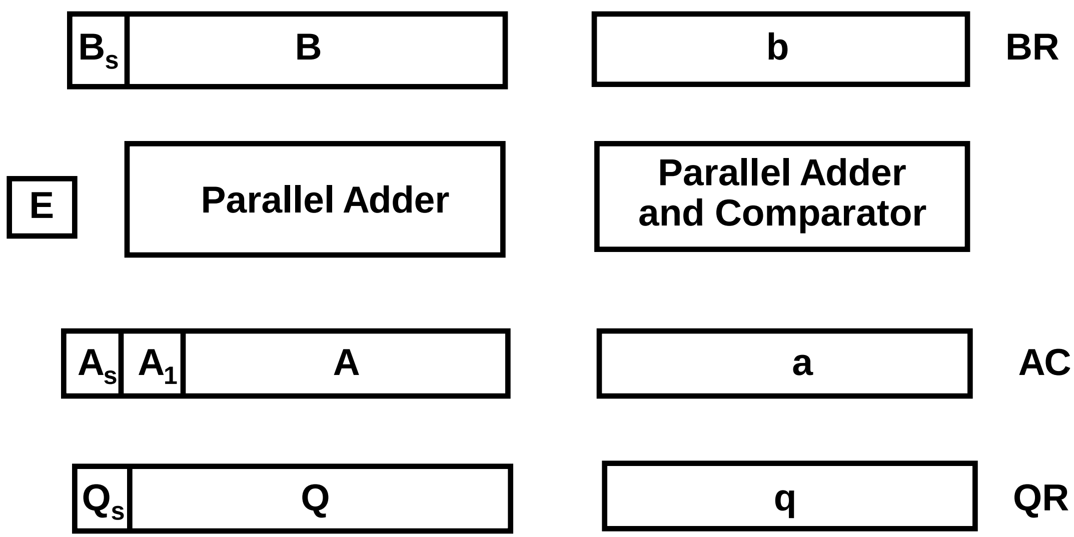
FLOWCHART OF DIVIDE OPERATION



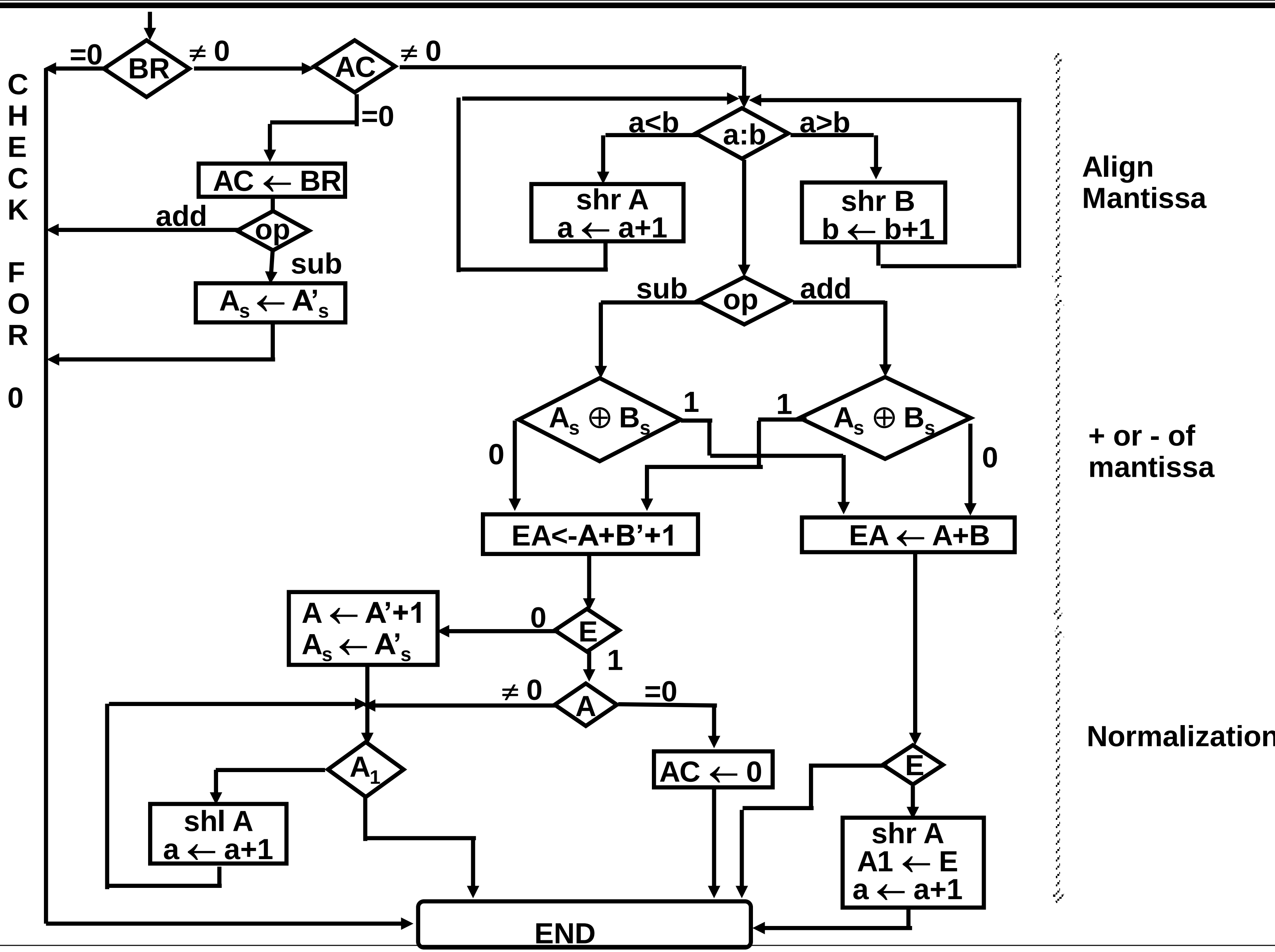
FLOATING POINT ARITHMETIC OPERATIONS

$F = m \times r^e$
where m: Mantissa
r: Radix
e: Exponent

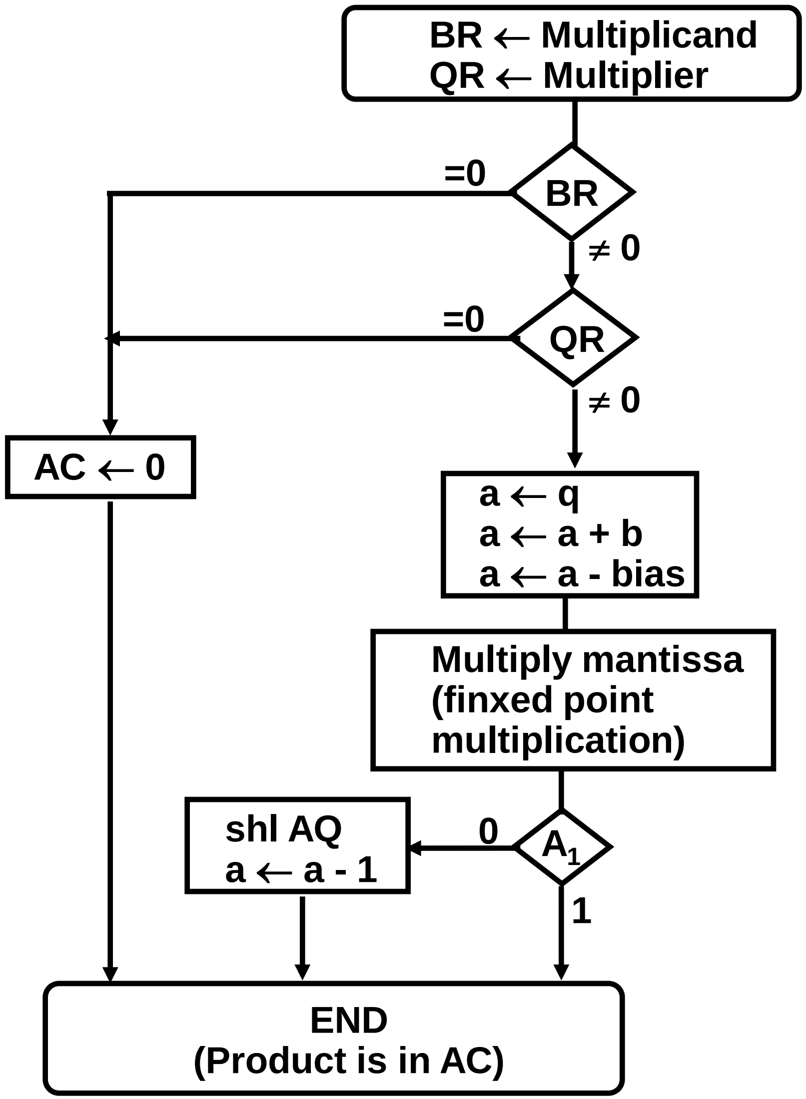
Registers for Floating Point Arithmetic



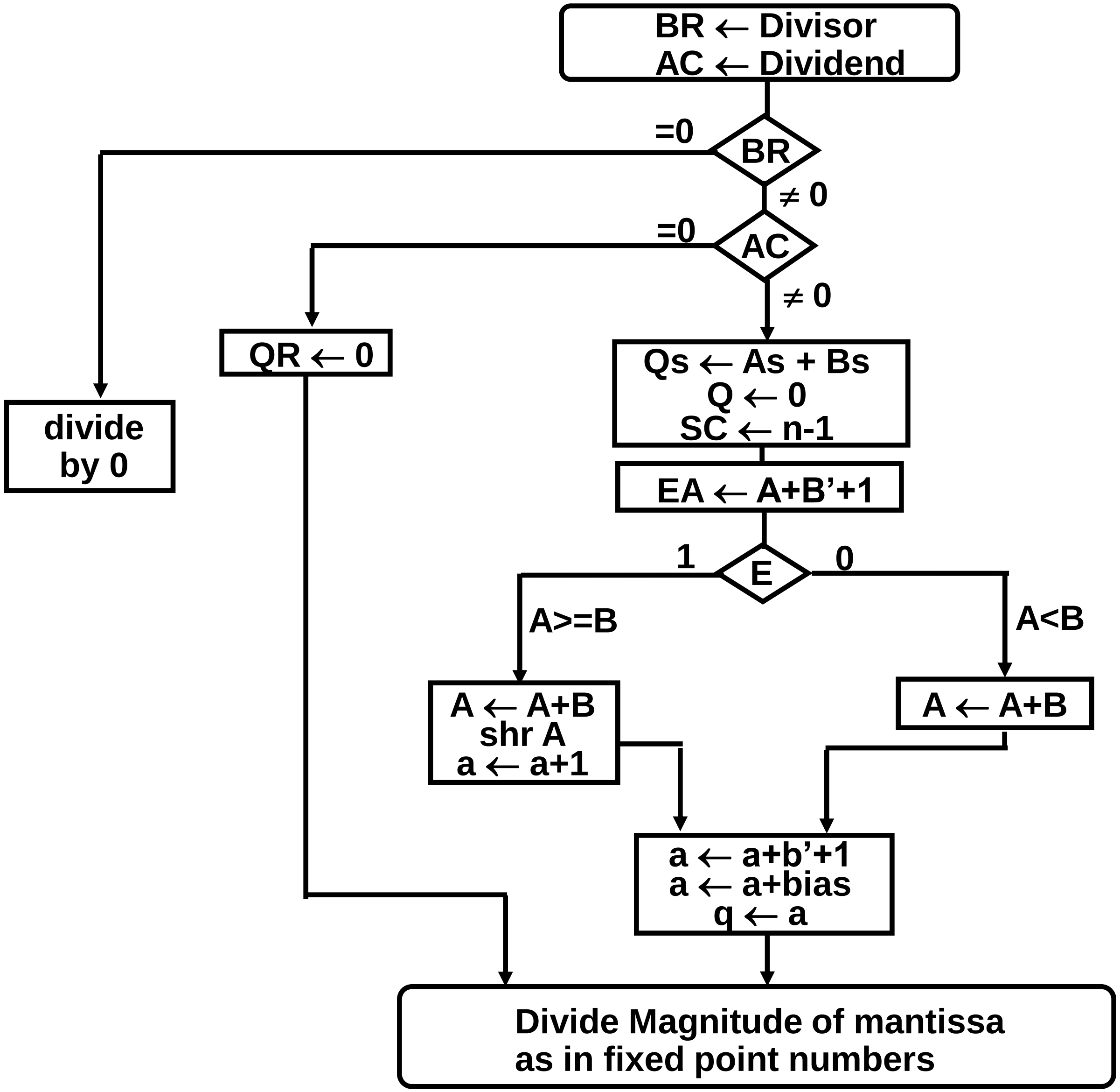
FLOATING POINT ADD AND AUBTRACT



FLOATING POINT MULTIPLICATION



FLOATING POINT DIVISION



BCD ADD

BCD digit < 10
BCD digit + BCD digit + carry =< 19

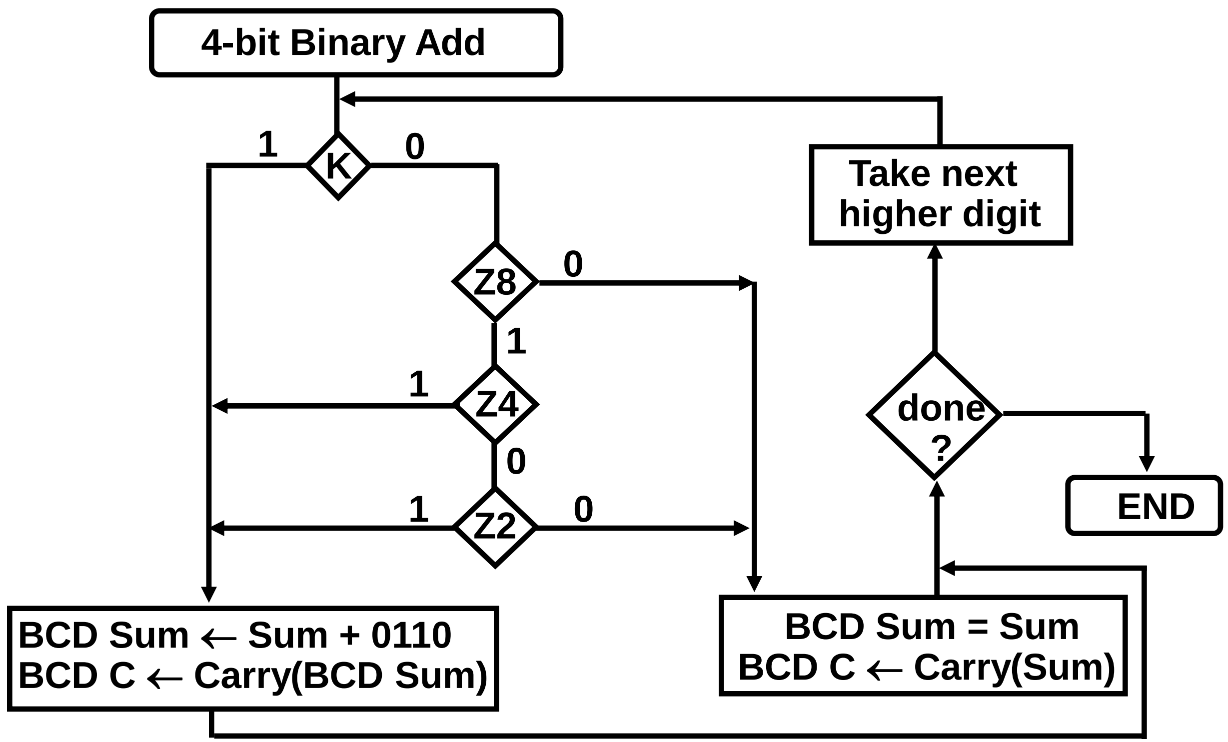
Binary Sum					BCD Sum					
K	Z8	Z4	Z2	Z1	C	S8	S4	S2	S1	Decimal
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

BCD ADDER

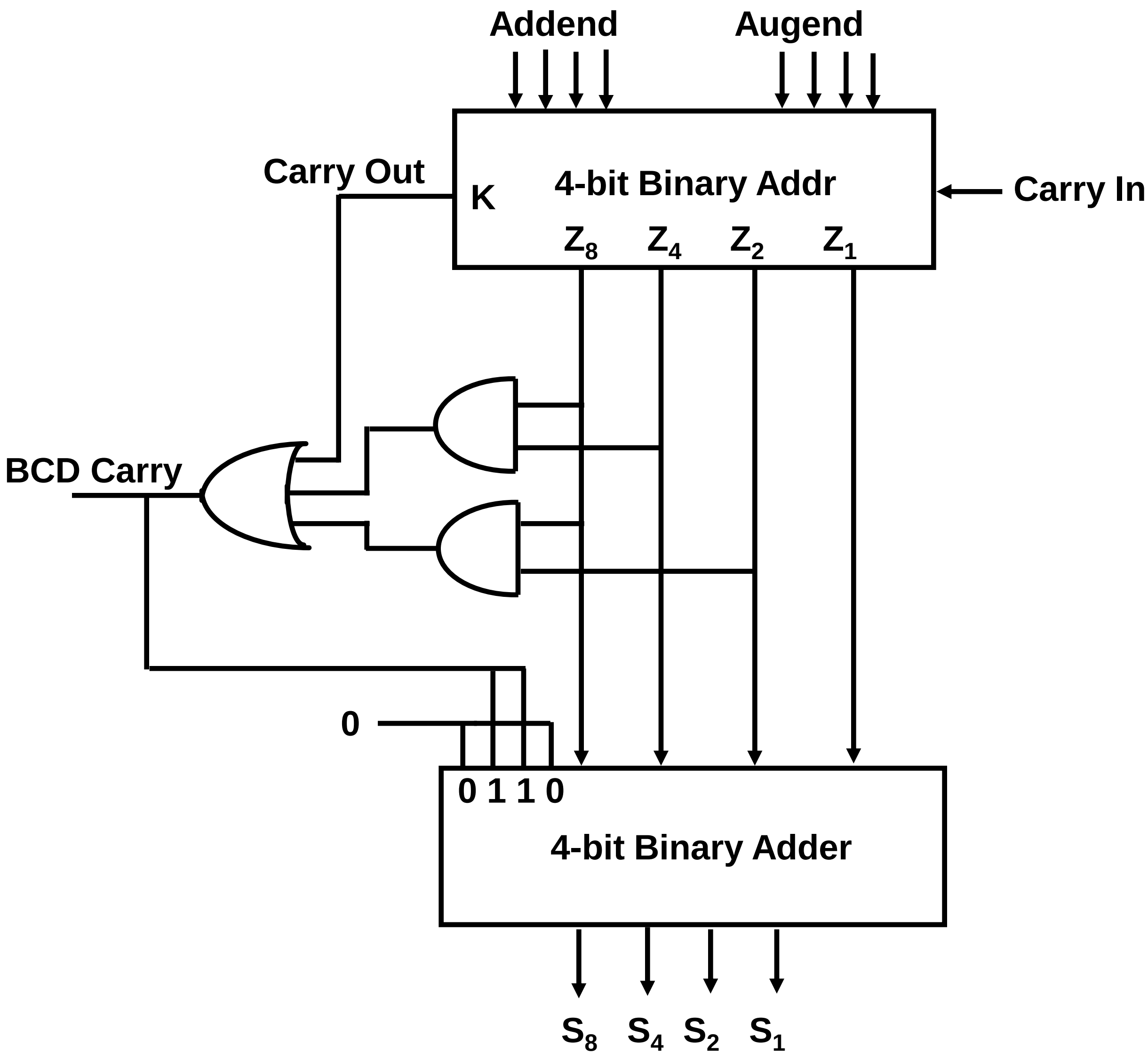
If we can convert *Binary Sums* to *BCD Sum* ,
we can use a binary adder to add two BCD numbers

SUM =< 9
BCD Sum = Binary Sum
BCD Carry = Binary Carry

19 >= SUM > 9
BCD Sum = Binary Sum + 0110
BCD Carry = Carry(Binary Sum + 0110)



BCD ADDER HARDWARE



DECIMAL ARITHMETIC OPERATIONS

Addition

- **Identical to the BCD addition**
- **9's complement and 10's complement are identical to 1's complement and 10's complement, respectively**