Unit-6
Computer Arithmetic

Marwadi University

Department of
Computer Engineering

Computer
Organization and
Architecture
01CE1402

Prof. Kishan Makadiya

Introduction

- The four basic arithmetic operations includes:-
  - Addition
  - Subtraction
  - Multiplication
  - Division

Addition & Subtraction

- Addition and subtraction with Signed-Magnitude Data:-

Operation	**Add Magnitudes**	**Subtract Magnitudes**		
		When A > B	**When A < B**	**When A = B**
$(+A) + (+B)$				
$(+A) + (-B)$				
$(-A) + (+B)$				
$(-A) + (-B)$				
$(+A) - (+B)$				
$(+A) - (-B)$				
$(-A) - (+B)$				
$(-A) - (-B)$				

Addition & Subtraction

- Addition and subtraction with Signed-Magnitude Data:-

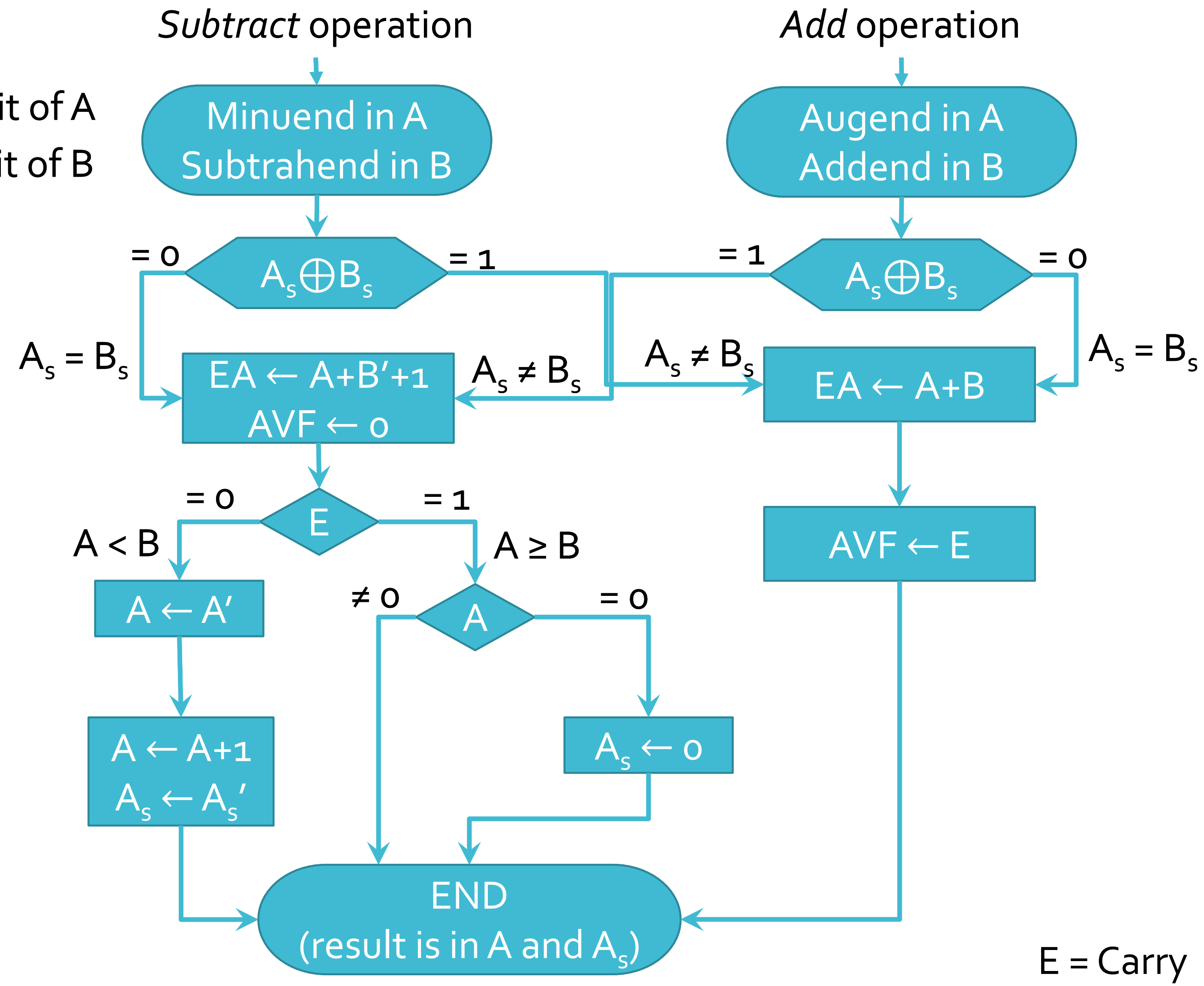
Operation	Add Magnitudes	Subtract Magnitudes		
		When A > B	When A < B	When A = B
(+*A*) + (+*B*)	+ (*A* + *B*)			
(+*A*) + (-*B*)		+ (*A* - *B*)	- (*B* - *A*)	+ (*A* - *B*)
(-*A*) + (+*B*)		- (*A* - *B*)	+ (*B* - *A*)	+ (*A* - *B*)
(-*A*) + (-*B*)	- (*A* + *B*)			
(+*A*) - (+*B*)		+ (*A* - *B*)	- (*B*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

Addition & Subtraction

- Golden Rule
- When sign of magnitude A and B are identical, add the two magnitudes and attach the sign of A to the result.
- When sign of A and B are different, compare the magnitudes using conditions and subtract smaller no from larger.
- Conditions:-
  - If $A > B$ -> Then Choose sign of the result to be same as A.
  - If $A < B$ -> Then Choose sign of complement of sign of A.
  - If $A = B$ -> Then subtract B from A and make sign of result Positive

Flowchart for Addition & Subtraction

A_s = Sign bit of A
B_s = Sign bit of B



E = Carry

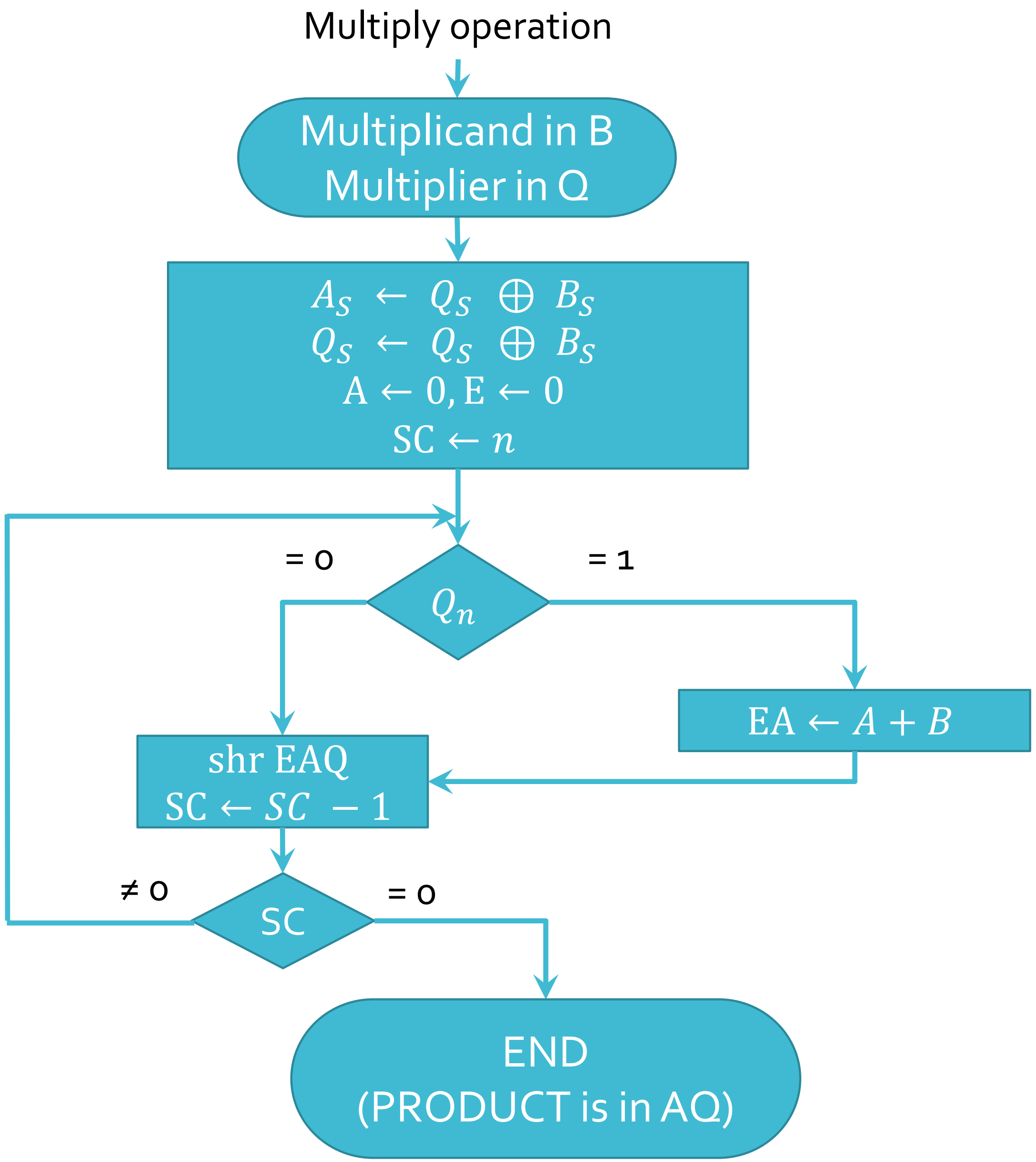
AVF = add overflow flip-flop

Multiplication Algorithms

- Multiplication of two floating-point binary numbers in signed-magnitude representation is done by a process of successive shift and add operations.

```
   23        10111
  x19      x 10011
  ----     -------
             10111
            10111
           00000
          00000
         10111
         ---------
  437    110110101
```

Flow chart for multiply operation




```
  B = 23        10111
x Q= 19       x 10011
-------       -------
                10111
               10111
              00000
             00000
            10111
          ---------
          110110101
```

Perform
23 X 19

Multiplicand B = 10111	**E**	**A**	**Q**	**SC**
Multiplier in Q	0	00000	10011	101
Q_n = 1; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
Q_n = 1; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
Q_n = 0; shift right EAQ	0	01000	10110	010
Q_n = 0; shift right EAQ	0	00100	01011	001
Q_n = 1; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

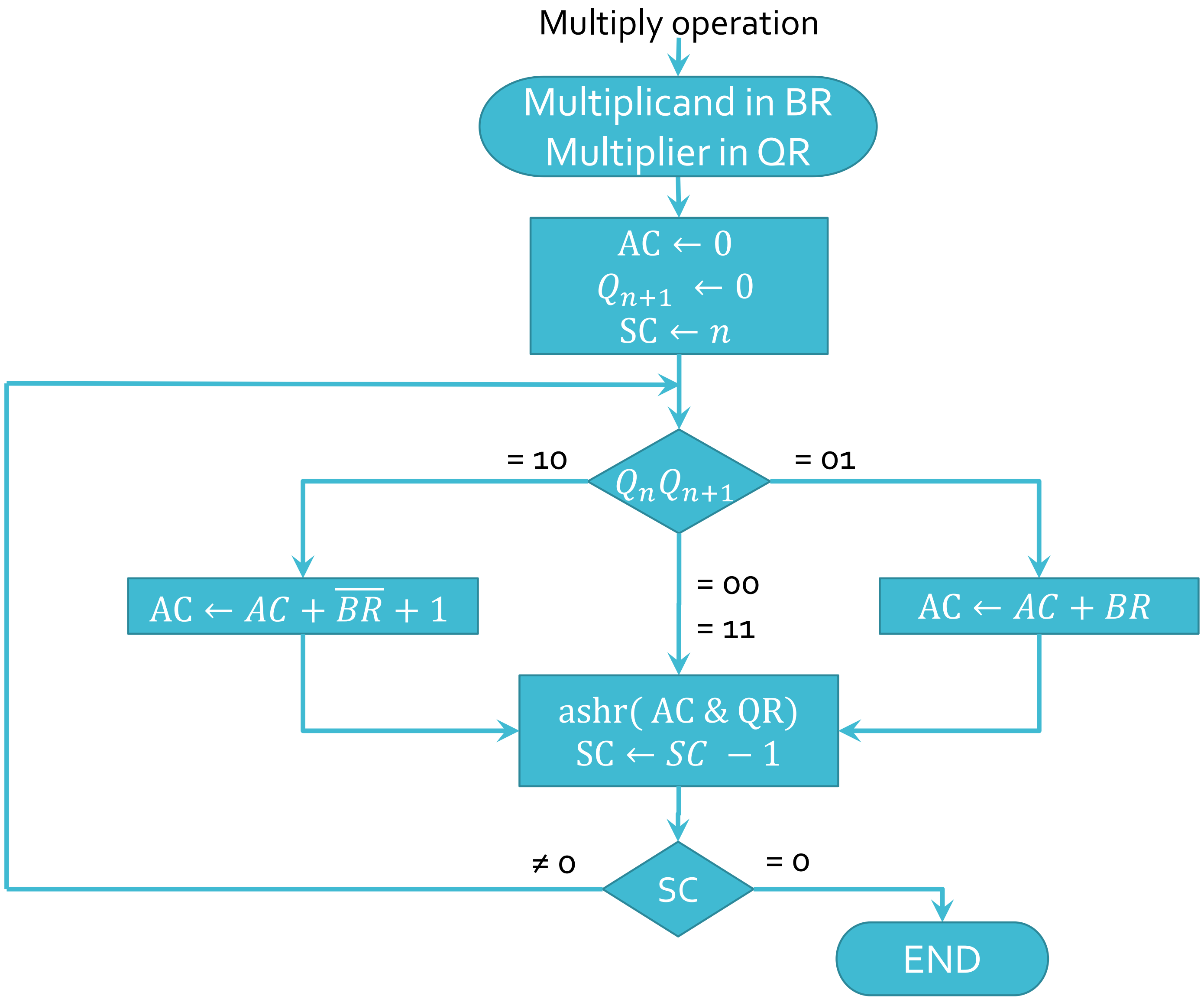
Perform
23 X 10

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	01010	101

Booth's Algorithm for Multiplication

- Prior to shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following rules:
  1. The multiplicand is **subtracted** from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
  2. The multiplicand is **added** to the partial product upon encountering the first 0 (provided that there was a previous 1) in a string of 0's in the multiplier.
  3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Booth Multiplication Algorithm



Multiply (-9) x (-13) using Booth Multiplication Algorithm

Q_n	Q_{n+1}	BR = 10111 $\overline{BR}+1$ = 01001	AC	QR	Q_{n+1}	SC
		Initial	00000	10011	0	101
1	0	Subtract BR	01001			
			01001			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add BR	10111			
			11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	Subtract BR	01001			
			00111			
		ashr	00011	10101	1	000

Multiply (+14) x (-11) using Booth Algorithm

Q_n	Q_{n+1}	$BR = 01110$ $\overline{BR} + 1 = 10010$	AC	QR	Q_{n+1}	SC
		Initial	00000	10101	0	5
1	0	Sub BR	10010			
			10010			
		ashr	11001	01010	1	4
0	1	Add BR	01110			
			00111			
		ashr	00011	10101	0	3
1	0	Sub BR	10010			
			10101			
		ashr	11010	11010	1	2
0	1	Add BR	01110			
			01000			
		ashr	00100	01101	0	1
1	0	Sub BR	10010			
			10110			
		ashr	11011	00110	1	0

Division Algorithm

- Division of two fixed-point binary numbers in signed-magnitude representation is done by a process of successive compare, shift and subtract operations.

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	6 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

Example

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

	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	

Decimal Arithmetic Unit - BCD Adder

- Two BCD digits are applied to 4-bit binary adder which produce result ranging from 0 to 19 i.e. 9 + 9 + 1 = 19
- Output sum of two decimal numbers must be represented in BCD.
- Problem is to find rule by which binary number is to be converted to correct BCD

		1		
97		1001	0111	
99	+	1001	1001	
196	1	0011	0000	Both groups generate carry
		+0110	+0110	Add 0110 to each
	1	1001	0110	

BCD Adder

Binary Sum					BCD Sum					
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	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

BCD Adder

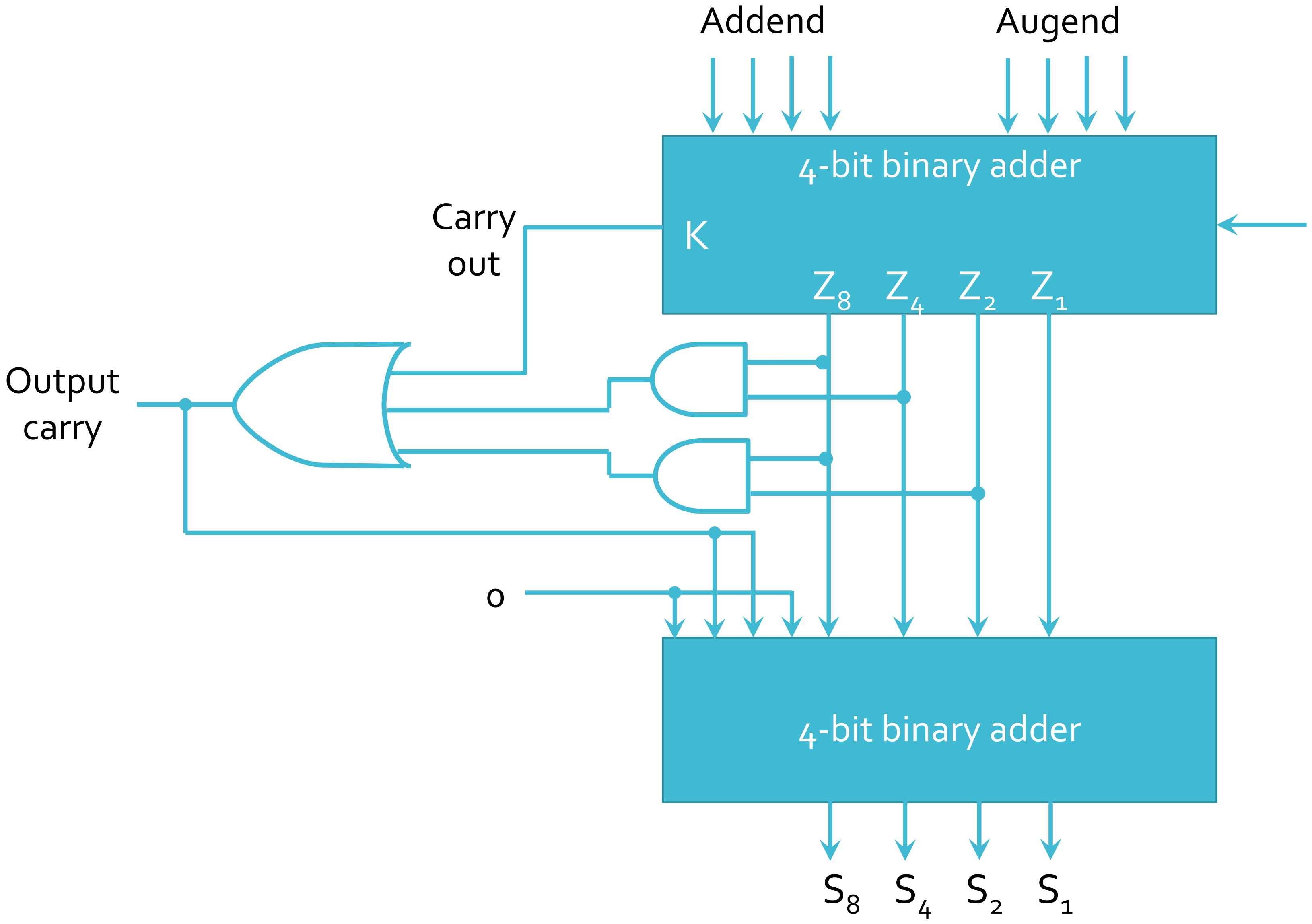
Binary Sum					BCD Sum					
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	Decimal
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

BCD Adder

- Correction from binary to BCD is needed in following conditions
  - K = 1
  - Z_8 and Z_4 or Z_8 and Z_2 must have 1

$$\boxed{C = K + Z_8 Z_4 + Z_8 Z_2}$$

BCD Adder



Addend
Augend
4-bit binary adder
Carry out
K
Z_8 Z_4 Z_2 Z_1
Output carry
0
4-bit binary adder
S_8 S_4 S_2 S_1

Thank You

Extra Example

- Step by step multiplication process using booth's

1) (+15) * (+13)
2) (+15) * (-13)

Extra Example

$(+15) \times (+13) = +195 = (0\ 011000011)_2$

$BR = 01111\ (+15);\ \overline{BR} + 1 = 10001\ (-15);\ QR = 01101\ (+13)$

$Q_n\ Q_{n+1}$		AC	QR	Q_{n+1}	SC
	Initial	00000	01101	0	101
1 0	Subtract BR	10001			
		10001			
	ashr	11000	10110	1	100
0 1	Add BR	01111			
		00111			
	ashr	00011	11011	0	011
1 0	Subtract BR	10001			
		10100			
	ashr	11010	01101	1	010
1 1	ashr	11101	00110	1	001
0 1	Add BR	01111			
		01100			
	ashr	00110	00011	0	000
		+195			

Extra Example

$(+15) \times (-13) = -195 \quad = (1100\ 111101)_{2\text{'s comp.}}$

$BR = 0\ 11111\ (+15); \quad \overline{BR} + 1 = 10001\ (-15); \quad QR = 10011\ (-13)$

Q_nQ_{n+1}		AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	10001			
		10001			
	ashr ——	11000	11001	1	100
1 1	ashr ——	11100	01100	1	011
0 1	add BR	01111			
		01011			
	ashr ——	00101	10110	0	010
0 0	ashr ——	00010	11011	0	001
1 0	Subtract BR	10001			
		10011			
	ashr ——	11001	11101	1	000
		−195			