11259022

Computer Organization HW3-1

1. (a) C = $0.8125_{(10)}$ = 0.5 + 0.25 + 0.0625 ⇒ $0.1101_{(2)}$ #

(b) A + (-B)

```
  00111001
+ 01010100
----------
  10001101
```

Carry in 8th bit not equal carry out 8th bit

overflow #

(c) A + B

```
  00111001
+ 10101100
----------
  11100101
```

Carry in 8th bit equal carry out 8th bit

not overflow #

2.

operation	Result indicating overflow
addition (a)	negative number
addition (b)	positive number and 0
subtraction (c)	negative number
subtraction (d)	positive number and 0

3.

signA	signB	Carry in	Carry out	result of sign	correct sign	Carry in XOR Carry out	overflow
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0
1	0	1	1	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0

4. -41-87 = -128 not in z^8 represent $-(z^7-1) \sim z^7$, (-127 ~ 128)

underflow # ∨ -128

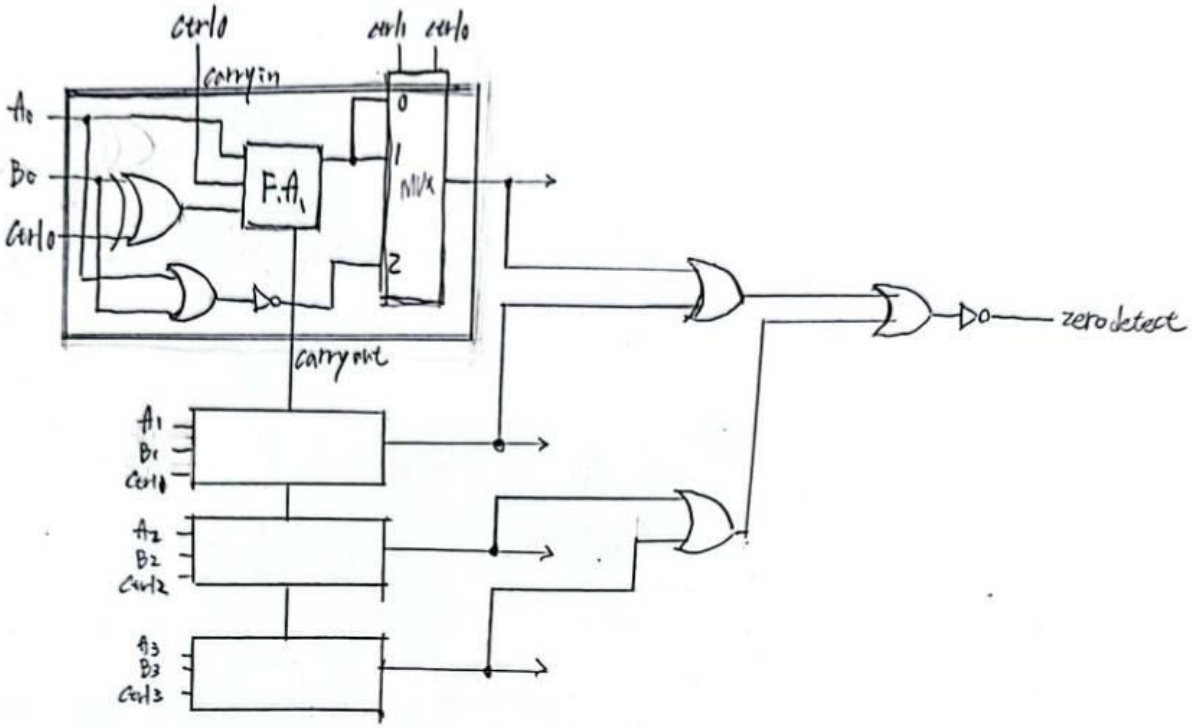
5×16^1 $16^1\times4$

5. 0101 0011 1111 1111 1111 1111 1111 1111 +1 = 0101 0100 0000 0000 0000 0000 0000 0000

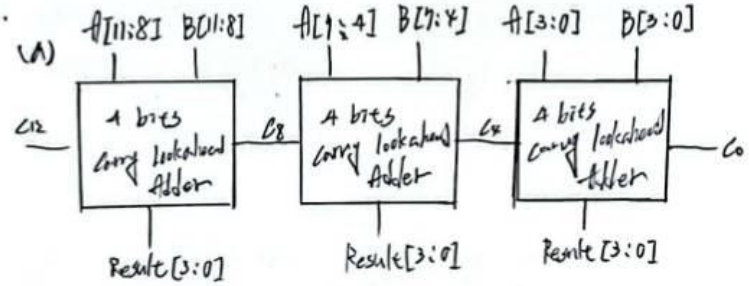
⇒ -1409286144 #

⇒ 1409286144 #

6.



7.



(b)

$C_1 = A_0B_0 + C_0(A_0+B_0) = g_0 + C_0P_0$

$C_2 = A_1B_1 + (g_0 + C_0P_0)(A_1+B_1) = g_1 + g_0P_1 + C_0P_0P_1$

$C_3 = g_2 + g_1P_2 + g_0P_1P_2 + C_0P_0P_1P_2$

$C_4 = g_3 + g_2P_3 + g_1P_3P_2 + g_0P_3P_2P_1 + C_0P_0P_1P_2P_3 = 1$

$\downarrow \quad 0 + 1 + \cdots$

$C_8 = g_7 + g_6P_7 + g_5P_7P_6 + g_4P_7P_6P_5 + C_4P_7P_6P_5P_4 = 1$

$\downarrow \quad 0 + 0 + 1 + \cdots$

$C_{12} = g_{11} + g_{10}P_{11} + g_9P_{11}P_{10} + g_8P_{11}P_{10}P_9 + C_8P_{11}P_{10}P_9P_8 = 1$

$0 + 0 + 0 + 0 + 1$

X: 1010 0011 0011
Y: 0101 1110 1011
+ P: 1111 1111 1011
· g: 0000 0000 0011
: 11 10 9 8 7 6 5 4 3 2 1 0

8.

Multiplicand ⇒ M

	Booth's Algorithm			Original method	
if multiplier is 0101 ⇒	M × 0101: +M −M +M −M	two addition & two subtraction	>	M × 0101: +M +M	two addition
if multiplier is 0111 ⇒	M × 0111: +M −M	one addition & one subtraction	<	M × 0111: +M +M +M	three addition

Booth's Algorithm not always faster than original method #

9.

$78_{(10)} = 01001110_{(2)}$

$27_{(10)} = 00011011_{(2)}$

① if multiplier[0] is 1 then product += multiplicand

② multiplicand shift left 1 bit, multiplier shift right 1 bit

		multiplicand	multiplier	product
1st	①	0000 0000 0100 1110	0001 1011	~~0000 0000 0000 0000~~ 0000 0000 0100 1110
	②	0000 0000 1001 1100	0000 1101	0000 0000 0100 1110
2nd	①	0000 0000 1001 1100	0000 1101	~~0000 0000 0100 1110~~ 0000 0000 1110 1010
	②	0000 0001 0011 1000	0000 0110	0000 0000 1110 1010
3rd	①	0000 0001 0011 1000	0000 0110	0000 0000 1110 1010
	②	0000 0010 0111 0000	0000 0011	0000 0000 1110 1010
4th	①	0000 0010 0111 0000	0000 0011	~~0000 0000 1110 1010~~ 0000 0010 0111 0000
	②	0000 0100 1110 0000	0000 0001	0000 0010 0111 0000
5th	①	0000 0100 1110 0000	0000 0001	~~0000 0010 0111 0000~~ 0000 1000 0011 1010
	②	0000 1001 1100 0000	0000 0000	0000 1000 0011 1010
6			⋮	
7			⋮	
8th		0100 1110 0000 0000	0000 0000	0000 1000 0011 1010

0000 1000 0011 1010 → $2106_{(10)}$ #