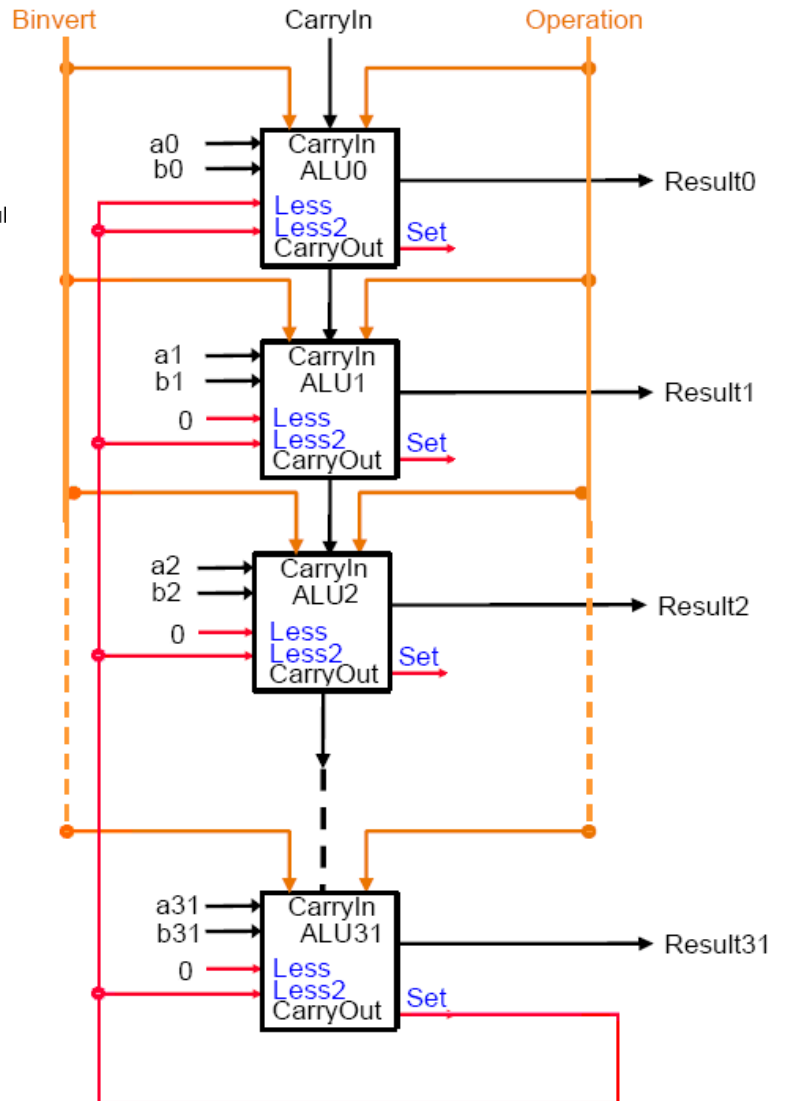
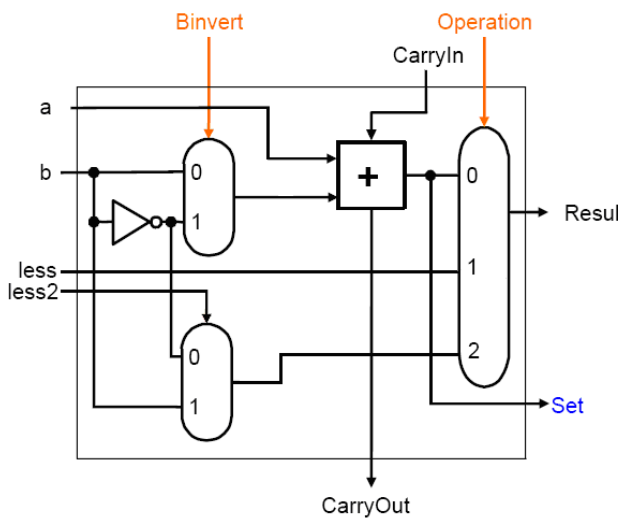


CSC3420 Computer System Architectures  
 2<sup>nd</sup> Term, Year 2008-2009 Quiz 4  
Suggested Solution

**Q1) Block Diagram design for SEQ pseudo-instruction**

[25marks]

Value of Binvert should be 1, and CarryIn should be 1.



[3 marks for Binvert signal (equal to 1); 2 marks for CarryIn (equal to 1); 10 marks for correct 1-bit ALU, use output as input control signal; 10 marks for correct 32-bit ALU connections]

Q2) -391.203125

[10 marks]

$$= 110000111.001101$$

$$= 1.10000111001101 \times 2^{1000}$$

$$= 1\ 1000\ 011\ 1100\ 0011\ 1001\ 1010\ 0000\ 0000$$

$$= \underline{\underline{C3C39A00}}$$

[2 marks for correct binary number; 2 marks for correct normalization; 1 marks for sign bit; 2 marks for exponent; 2 marks for mantissa; 1 marks for hexadecimal representation]

Q3) Signed Multiplication with Negative Multiplier

[30 marks]

3a)

$$A_i = 1 \Leftrightarrow \hat{A}_i = 0$$

$$A_i = 0 \Leftrightarrow \hat{A}_i = 1 \quad [3 \text{ marks}]$$

$$\therefore A_i + \hat{A}_i = 1 \quad (\text{always}) \quad [2 \text{ marks}]$$

$$\text{i.e. } A + \hat{A} = 11111\dots1 = 2^n - 1$$

$$\hat{A} + 1 = 2^n - A \quad [3 \text{ marks}]$$

where  $A_i$  is the  $i$ -th bit of  $A$

3b)

Step 1. Convert multiplier to a positive number:  $2^n - B$  (by part a)

Step 2. Perform multiplication with the original Signed Multiplication algorithm:  $A \times (2^n - B)$

Step 3. Convert result back to the expected polarity:  $2^{2n} - [A \times (2^n - B)]$  (as the product has  $2n$  bits)

[4 marks for each step; 2 marks for explanation and 2 marks for the calculation]

Therefore Result:

$$2^{2n} - [A \times (2^n - B)]$$

$$= 2^{2n} + A \times B - 2^n \times A \quad [5 \text{ marks}]$$

$$= \underline{\underline{A \times B - 2^n \times A}} \quad (2^{2n} \text{ will always overflow}) \quad [5 \text{ marks}]$$

**Q4) Unsigned Division****[35 marks]**4a) Steps of  $1011011_2 \div 1100_2$ 

Step	Comment	Register Hi	Register Lo
0	Initial	0101	1011
	Shift Left 1 bit	1011	0110
1	Subtract 1100	1111	0110
	Shift Left 1 bit, set rightmost bit to 0	1110	1100
2	Add 1100	1010	1100
	Shift Left 1 bit, set rightmost bit to 0	0101	1000
3	Add 1100	0001	1000
	Shift Left 1 bit, set rightmost bit to 1	0011	0001
4	Subtract 1100	0111	0001
	Shift Left 1 bit, set rightmost bit to 1	1110	0011
5	Shift Left Half (remainder) right 1 bit	0111	0011

Quotient is  $0011_2$ , Remainder is  $0111_2$  **[12 marks, 2 marks for each step]**4b) Yes. The answer is wrong. **[1 marks]**

In step 2, a significant bit (carrying data) is being left shifted. According to the non-restoring algorithm:  $(\text{remainder} + \text{divisor} \times 2^{-1})$  instead of  $(-2^{-1} \times \text{Divisor})$ , the Register Hi's content is restored wrongly in step 3. **[9 marks, 4 marks for step 2 error, 5 marks for step 3 failing to restore]**

4c) Usually, in proper (correct) usage, the bit length of the dividends should be no more than half of that of the Remainder Register. So the above situation cannot be arisen. **[8 marks]**

Possible Solutions: **[5 marks]**

1. Add more bits to the left hand side of the Remainder Register and the ALU
2. Add a carry bit to the Remainder Register