

EE 3755, Fall 03

HW# 1(a) Solutions

EE 3755,  
HW#1@ Solutions

(1)

[1] Here the two numbers X and Y are of different signs and the addition  $X+Y$  needs to be performed. We thus have to perform the following subtraction:

$$\begin{aligned} (\text{magnitude of } X) - (\text{magnitude of } Y) &= (1011011)_2 - (1101001)_2 \\ &= (1011011) + 2^2\text{s complement of } (1101001) = (1011011) + (0010111) \end{aligned}$$

$$\begin{array}{r} 1011011 \\ +) 0010111 \\ \hline 01110010 \end{array}$$

$\hookrightarrow c=0 \Rightarrow \text{result} < 0 \Rightarrow (\text{magnit. of } X) - (\text{magnitude of } Y) < 0$   
 $\Rightarrow \text{magnitude of } X < \text{magnitude of } Y.$

So sign bit of result = sign bit of  $Y = 1$  and  
 magnitude of  $(X+Y) = 2^2\text{s compl. of } (1110010) = (0001110).$

Thus  $X+Y = (10001110)_2 = (-14)_{10}.$

[2] Here multiplier  $= (25)_{10} = (11001)_2$ ; multiplicand  $= (30)_{10} = (11110)_2$ ;  
 $n=5$

Initialization 

C	B	A
0	00000	11001

 $\hookrightarrow 1 \Rightarrow \text{add multiplicand and shift}$

+)  
 result of addition 

	11110	
0	11110	11001

result of 1<sup>st</sup> cycle 

	01111	01100
--	-------	-------

 $\hookrightarrow 0 \Rightarrow \text{shift}$

result of 2<sup>nd</sup> cycle 

	00011	10110
--	-------	-------

 $\hookrightarrow 0 \Rightarrow \text{shift}$

result of 3<sup>rd</sup> cycle 

	00011	11011
--	-------	-------

 +)  
 $\hookrightarrow 1 \Rightarrow \text{add multiplicand and shift}$

result of addition 

1	00001	11011
---	-------	-------

result of 4<sup>th</sup> cycle 

	010000	11101
--	--------	-------

 +)  
 $\hookrightarrow 1 \Rightarrow \text{add mult/cand and shift}$

result of addition 

1	01110	11101
---	-------	-------

result of 5<sup>th</sup> cycle 

	010111	01110
--	--------	-------

$\hookrightarrow \text{product} = (1011101110)_2 = 750$   
 $= 30 \times 25.$

3] Here  $n=6$ ; multiplier =  $(-27)_{10} = (100101)_2$ ; ②  
 multiplicand =  $(-18)_{10} = (101110)_2$

Initialization 

B	A	d
000000	100101	0

+)  

010010		
--------	--	--

↘ 1,0 ⇒ subtr. mult/cand and then shift

010010	100101	0
--------	--------	---

result of 1<sup>st</sup> cycle 

001001	010010	1
--------	--------	---

+)  

101110		
--------	--	--

↘ 0,1 ⇒ add mult/cand and then shift

110111	010010	1
--------	--------	---

result of 2<sup>nd</sup> cycle 

111011	101001	0
--------	--------	---

+)  

010010		
--------	--	--

↘ 1,0 ⇒ subtr. mult/cand and then shift.

001101	101001	0
--------	--------	---

result of 3<sup>rd</sup> cycle 

000110	110100	1
--------	--------	---

+)  

101110		
--------	--	--

↘ 0,1 ⇒ add mult/cand and then shift

110100	110100	1
--------	--------	---

result of 4<sup>th</sup> cycle 

111010	011010	0
--------	--------	---

↘ 0,0 ⇒ shift

result of 5<sup>th</sup> cycle 

111101	001101	0
--------	--------	---

+)  

010010		
--------	--	--

↘ 1,0 ⇒ subtr. mult/cand, and then shift

001111	001101	0
--------	--------	---

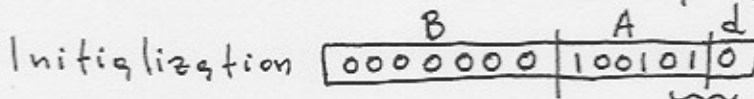
result of 6<sup>th</sup> cycle 

000111	100110	1
--------	--------	---

↘ product =  $(000111100110)_2$   
 $= 486 = (-18) \times (-27)$ .

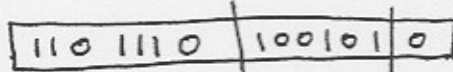
3

4 Here  $n=6$ ; multiplier  $= (-27)_{10} = (100101)_2$ ;  
 multiplicand  $= (-18)_{10} = (101110)_2$ . Since three bits are  
 to be examined at a time the field B (left of  
 multiplier field) should be of length  $6+1=7$  bits

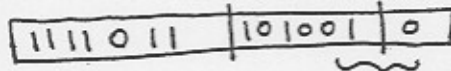


+ ) 1101110

010  $\Rightarrow$  add  $1 \times \text{mult/csnd}$   
 and then do 2-bit right  
 shift

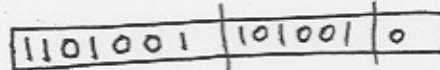


result of 1<sup>st</sup> cycle

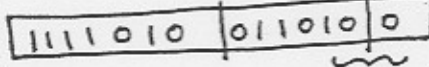


+ ) 1101110

010  $\Rightarrow$  add  $1 \times \text{mult/csnd}$   
 and then do 2-bit  
 right shift

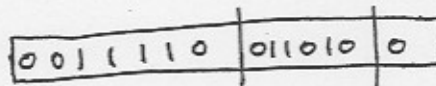


result of 2<sup>nd</sup> cycle

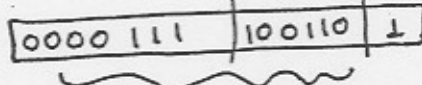


+ ) 0100100

100  $\Rightarrow$  add  $-2 \times \text{mult/csnd}$   
 and then do 2-bit  
 right shift



result of 3<sup>rd</sup> cycle



$\hookrightarrow$  product  $= (000111100110)_2$   
 $= 486 = (-18) \times (-27)$ .

5

(i) Case of examining two bits at a time:

The two versions (the one initialized with  $d \leftarrow 0$   
 and the one initialized with  $d \leftarrow 1$ ) differ only in their  
 first cycles of operation. The rest of the cycles are  
 the same. The following comparative tables show the  
 differences of the first cycle for the two cases of  
 $d \leftarrow 0$  and  $d \leftarrow 1$ . The rightmost bit of the multiplier  
 field is  $a_0$

$a_0 d$	
0 0	add $0x \text{mult}/c_{\text{end}}$ and then shift
1 0	add $-1x \text{mult}/c_{\text{end}}$ and then shift

$a_0 d$	
0 1	add $1x \text{mult}/c_{\text{end}}$ and then shift
1 1	add $0x \text{mult}/c_{\text{end}}$ and then shift

Obviously the version corresponding to  $d \leftarrow 1$  creates  $1x \text{mult}/c_{\text{end}}$  more than the version corresponding to  $d \leftarrow 0$ . This holds true only for the first cycle and since the remaining cycles are the same the version that corresponds to initializing  $d$  with one will compute  $\text{mult}/c_{\text{end}} \times \text{multiplier} + \text{mult}/c_{\text{end}}$ .

(ii) Case of examining three bits at a time:

The following tables show the differences of the first cycle for the two different initializations of  $d$  ( $d \leftarrow 0$  and  $d \leftarrow 1$ ). In the tables below,  $a_1$  and  $a_0$  are the two rightmost bits of the multiplier field.

$a_1 a_0 d$	
0 0 0	add $0x \text{mult}/c_{\text{end}}$ and then 2-bit shift
0 1 0	add $1x \text{mult}/c_{\text{end}}$ and then double shift
1 0 0	add $-2x \text{mult}/c_{\text{end}}$ and then double shift
1 1 0	add $-1x \text{mult}/c_{\text{end}}$ and then double shift

$a_1 a_0 d$	
0 0 1	add $1x \text{mult}/c_{\text{end}}$ and then double shift
0 1 1	add $2x \text{mult}/c_{\text{end}}$ and then double shift
1 0 1	add $-1x \text{mult}/c_{\text{end}}$ and then double shift
1 1 1	add $0x \text{mult}/c_{\text{end}}$ and then double shift

Again here, the version that uses  $d \leftarrow 1$  (at initialization) creates in the first cycle  $1x \text{mult}/c_{\text{end}}$  more than the version that corresponds to  $d \leftarrow 0$  (for initialization).

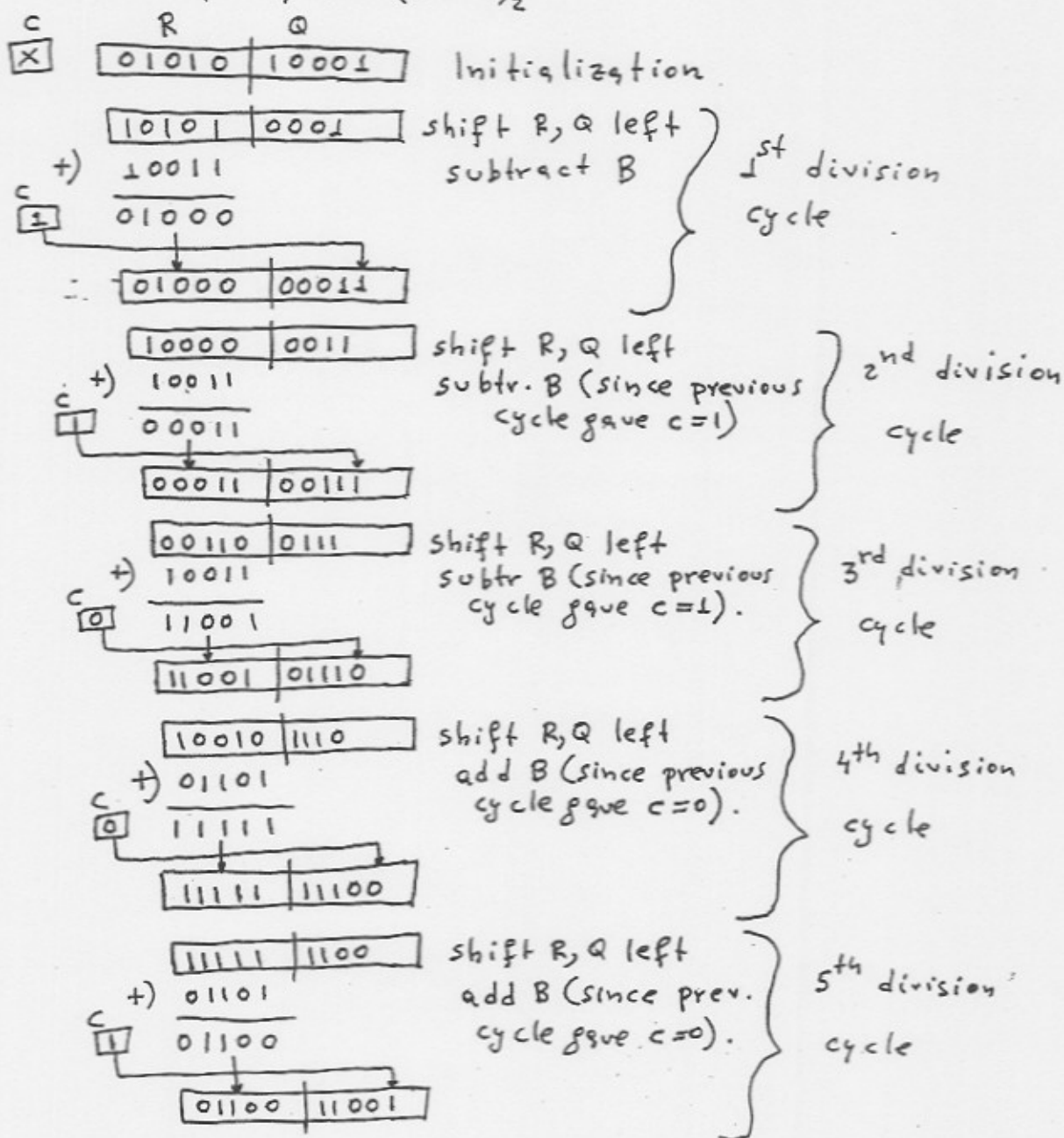
(5)

[6] Here the leftmost 5-bit part of the dividend is  $A_1 = (01010)_2 = 10$  while the divisor is  $B = (01001)_2 = 9$ .  
Since  $A_1 > B$  a division overflow will occur.

[7] Here the leftmost 5-bit part of the dividend is  $A_1 = (01011)_2 = 11$  and the divisor is  $B = (01011)_2 = 11$ . Since  $A_1 = B$  a division overflow will occur.

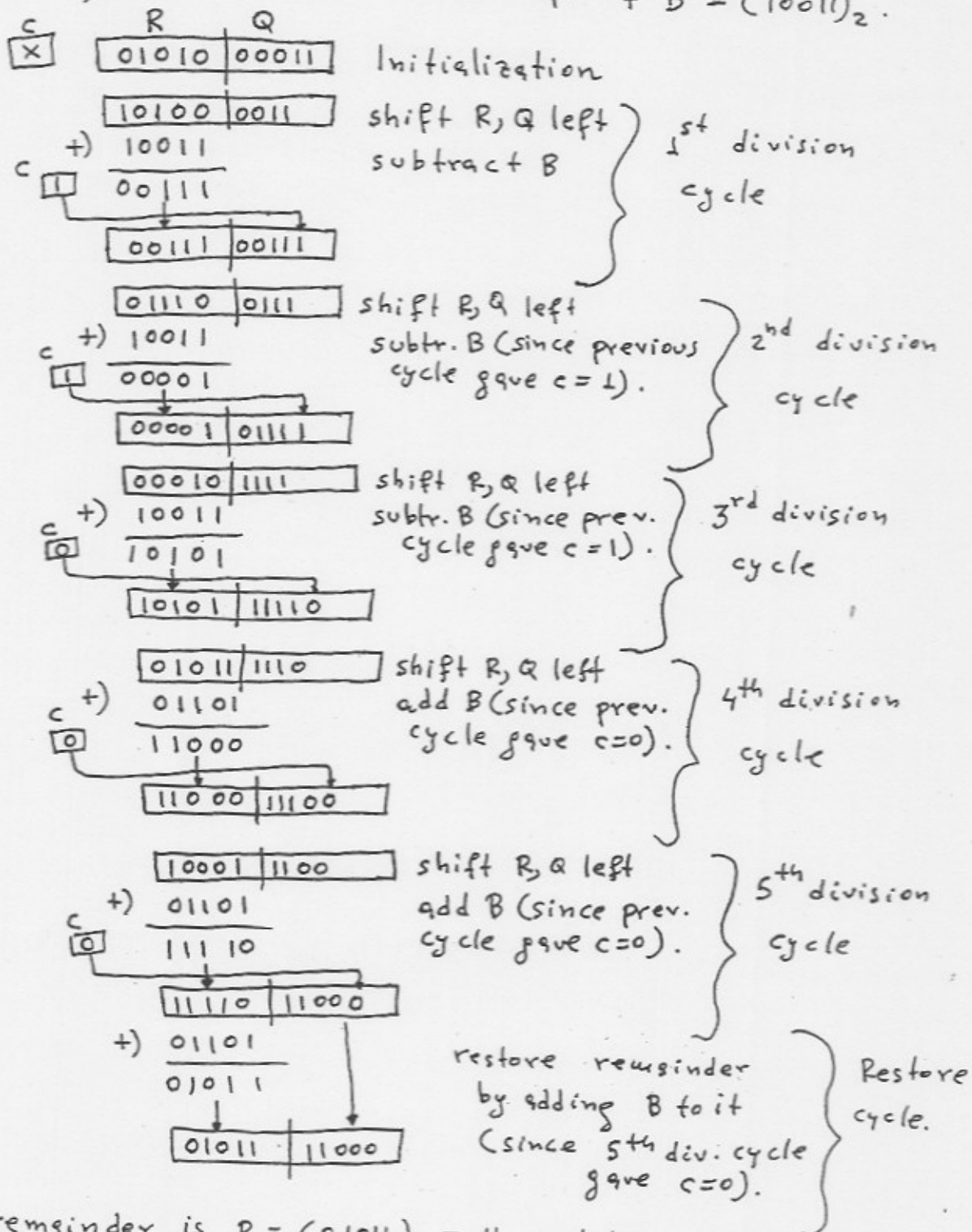
[8] Here the leftmost 5-bit part of the dividend is  $A_1 = (01100)_2 = 12$  while the divisor is  $B = (01101)_2 = 13$ .  
Since  $A_1 < B$  division overflow will not occur.

9 Here the dividend is  $A = (0101010001)_2 = 10 \times 32 + 17 = 337$ ; the divisor is  $B = (01101)_2 = 13$  and  $n = 5$ . Also  $-B = 2$ 's compl. of  $B = (10011)_2$  6



No restore cycle is necessary since the carry out of the 5<sup>th</sup> division cycle is 1. Thus the final R contains the correct remainder  $R = (01100)_2 = 12$  while the quotient is  $Q = (11001)_2 = 25$ . Double check to see that  $B \times Q + R = 13 \times 25 + 12 = 337 = A$ .

10 Here  $A = (0101000011)_2 = 10 \times 32 + 3 = 323$ ;  $B = (01101)_2 = 13$ ;  $n = 5$ . Also  $-B = 2^5 \text{ compl. of } B = (10011)_2$ . ⑦



So remainder is  $R = (01011)_2 = 11$  while quotient is  $Q = (11000)_2 = 24$ . Double check to see that  $B \times Q + R = 13 \times 24 + 11 = 323 = A$ .



11 The range of the fraction is  $0.5 \leq f \leq 1 - 2^{-48}$ . The range of the exponent is  $-2^{10} \leq e \leq 2^{10} - 1$  or  $-1024 \leq e \leq 1023$ . So the positive dynamic range is

$$0.5 \times 2^{-1024} \leq A^+ \leq (1 - 2^{-48}) \times 2^{1023}$$

while the negative dynamic range is

$$-(1 - 2^{-48}) \times 2^{1023} \leq A^- \leq -0.5 \times 2^{-1024}$$

12 Here  $e_{\text{biased}} = (10100)_2 = 20$  while  $\text{bias} = 2^4 = 16$ . So  $e_{\text{unbiased}} = e_{\text{biased}} - \text{bias} = 20 - 16 = 4$ . Then the value of  $A$  is  $A = -0.1101100000 \times 2^4 = (-1101.100000)_2 = (-13.5)_{10}$ .

13 a

1. Align fractions/adjust smaller exp.

In order to find the larger exponent we can perform

$$e_1 - e_2 = e_1 + 2^5 \text{ compl. of } e_2 = (1010) + (1001)$$

$$\begin{array}{r} 1010 \\ +) 1001 \\ \hline 10011 \end{array}$$

$\rightarrow c=1 \Rightarrow e_1 - e_2 > 0$  or  $e_1 > e_2$ . So  $e_1$  is the larger exponent and the difference is  $e_1 - e_2 = (0011)_2 = 3$ . The number

$A_2$  now becomes

$$A_2: \begin{array}{c|c|c} s_2 & e_1 & f_2' \\ \hline 1 & 1010 & 00011010 \end{array}$$

A fraction underflow occurred as the result of alignment.

2. Add fractions: Here since  $A_1$  and  $A_2$  are of the same sign and the operation is addition ( $A_3 = A_1 + A_2$  needs to be computed) a true addition between  $f_1$  and  $f_2'$  has to take place

$$\begin{array}{r} f_1 + f_2' = .11110101 \\ +) .00011010 \\ \hline 1.00001111 \end{array}$$

$\rightarrow$  fraction overflow.

(9)

3. Postnormalize: After postnormalization we get

$A_3 = A_1 + A_2$ : 

$s_3$	$e_3$	$f_3$
1	1011	10000111

 A fraction underflow occurred as a result of postnorml.

4. Check for exp overflow: No exp. overflow occurred ( $e_3 = (1011)_2 = 11$  is within range  $[0, 15]$ ).

(b) Here the larger exponent is  $e_1$ , the smaller exponent is  $e_2$  and the difference  $e_1 - e_2 = 12 - 3 = 9 > 8$  (8 is the frac. length). So  $A_3 = A_1 + A_2 = A_1$ .

(c) 1. Align/adjust: Here the larger exponent is  $e_1$ , the smaller exp. is  $e_2$  and  $e_1 - e_2 = 15 - 12 = 3$ . The number  $A_2$  now becomes  $A_2$ : 

$s_2$	$e_1$	$f_2'$
0	1111	00010110

2. Add fractions: A true addition  $f_1 + f_2'$  has to take place.

$$\begin{array}{r} f_1 + f_2' = .11110101 \\ +) .00010110 \\ \hline 1.00001011 \end{array}$$

↳ fract. overflow (postnormalization needed)

3. Postnormalize and check for exp overflow

Here postnormalization of the fraction will result in exponent overflow (observe that 1111 is the largest 4-bit biased exponent). An exp. overflow flag has to be set.

(d) 1. Align/adjust: Here the larger exp. is  $e_2$ , the smaller exp. is  $e_1$  and  $e_2 - e_1 = 11 - 9 = 2$ . The number  $A_1$  now becomes  $A_1$ : 

$s_1$	$e_2$	$f_1'$
0	1011	00111100

2. Subtract fractions: Since  $A_1$  and  $A_2$  are of 10 different signs the true subtraction  $f_1' - f_2$  has to take place.  $f_1' - f_2 = f_1' + 2$ 's complement of  $f_2 =$

$$\begin{array}{r} .00111100 \\ +) .01101110 \\ \hline 0.10101010 \end{array}$$

$\rightarrow c=0 \Rightarrow f_1' - f_2 < 0 \Rightarrow f_1' < f_2$ . Since  $f_2$  is the larger fraction the result  $A_3 = A_1 + A_2$  must have as a sign bit the sign bit of  $A_2$  (negative sign). The fraction of  $A_3$  will be 2's compl. of  $(f_1' - f_2) = 2$ 's compl. of  $(10101010) = (01010110)_2$ .

So  $A_3 = A_1 + A_2$ : 

1	1011	01010110
---	------	----------

3. Postnormalize: After postnormalization we get

$$A_3: \begin{array}{c} s_3 \quad e_3 \quad f_3 \\ \hline \boxed{1 \mid 1010 \mid 10101100} \end{array}$$

4. Check for exp. underflow

No exp. underflow occurred.

② 1. Align/adjust: Here  $e_1 = e_2$  and no alignment/adjustment is needed.

2. Subtract fractions: Since  $A_1$  and  $A_2$  are of different signs the subtraction  $f_1 - f_2$  has to take place.

$$f_1 - f_2 = f_1 + 2$$
's compl. of  $f_2 =$

$$\begin{array}{r} .11111100 \\ +) .00001000 \\ \hline 1.00000100 \end{array}$$

$\rightarrow c=1 \Rightarrow f_1 - f_2 > 0 \Rightarrow f_1 > f_2$ . Since  $f_1$  is the larger fraction the result  $A_3 = A_1 + A_2$  must have as a sign bit the sign bit of  $A_1$  (sign bit of zero).

The fraction of  $A_3$  will be  $f_1 - f_2 = .00000100$

Thus  $A_3$ : 

0	0010	00000100
---	------	----------

### 3. Postnormalize and check for exp. underflow:

The resulting fraction  $.00000100$  needs to be shifted to the left by 5 bits. This will create an exp. underflow since the exponent will have to be decremented by 5 and  $2-5 = -3 < 0$ . Recall that the range of 4-bit biased exponents is  $[0, 15]$ . Since an exponent underflow occurred an exp. underflow flag is set and the result is forced to the unique FLP zero or

$$A_3: \begin{array}{c|c|c} s_3 & e_3 & f_3 \\ \hline 0 & 0000 & 00000000 \end{array}$$

(f) The sign of the product is  $s_3 = s_1 \oplus s_2 = 1$ . The product of the fractions is  $(.11111100) \times (.11111000) = .1111010000100000$ . Truncating the rightmost 8-bit part we get product of fractions =  $.11110100$ . Also  $e_1 + e_2 - \text{bias} = 10 + 9 - 8 = (11)_{10} = (1011)_2$ . The product is  $A_3 = A_1 \times A_2: \begin{array}{c|c|c} s_3 & e_3 & f_3 \\ \hline 1 & 1011 & 11110100 \end{array}$ . Observe that the result is normalized so no postnormalization is needed. No exp. overflow or exp. underflow occurred.

14 (a) The exponent of the product will be  $e_1 + e_2 - \text{bias}$  if no postnormalization is needed, while it will be  $e_1 + e_2 - \text{bias} - 1$  if postnormalization is needed. Here  $e_1 = (10100)_2 = 20$ ,  $e_2 = (01010)_2 = 10$ ,  $\text{bias} = 2^4 = 16$  while the dynamic range of 5-bit biased exponents is  $[0, 31]$ . So if postnormalization is not needed, the product's exponent will be  $e_1 + e_2 - \text{bias} = 20 + 10 - 16 = 14$  and neither exp. overflow nor exp. underflow occurs. In the case that postnormalization is needed the product's exp. will be  $e_1 + e_2 - \text{bias} - 1 = 13$  (again safe exp.).

(b) Here since  $f_1 \times f_2 = (.100) \times (.100) = .010000$  postnormalization will be necessary and the product's exp. will be  $e_1 + e_2 - \text{bias} - 1 = 24 + 24 - 16 - 1 = 31$ . So no exp. overflow nor exp. underflow.

(c) If no postnormalization is needed, the product's exponent (12) will be  $e_1 + e_2 - bias = 30 + 20 - 16 = 34 > 31$  and an exp. ovf. occurs. Even if postnormalization is needed, the product's exp. will be  $e_1 + e_2 - bias - 1 = 33 > 31$  and we will still have exp. ovf.

(d) If no postnormalization is needed, the product's exp. will be  $e_1 + e_2 - bias = 7 + 3 - 16 = -6 < 0$  and an exp. undf. occurs. The situation is even worse if postnormalization is needed since the product's exp. will be  $e_1 + e_2 - bias - 1 = -7$  (exp. undf.)

15 (a) The quotient's exponent will be  $e_1 - e_2 + bias$  if alignment of dividend is not needed while it will be  $e_1 + 1 - e_2 + bias$  if alignment of dividend is needed. Observe that  $e_1 - e_2 + bias = 20 - 23 + 16 = 13 \in [0, 31]$  while  $e_1 + 1 - e_2 + bias = 14 \in [0, 31]$ . So we will never have exp. ovf nor exp. undf.

(b) Here since  $f_1 > f_2$  dividend alignment is needed. This way the quotient's exp. will be  $e_1 + 1 - e_2 + bias = 2 + 1 - 19 + 16 = 0 \in [0, 31]$ . So no exp. ovf or exp. undf. occurs.

(c) If no dividend alignment is needed, the quotient's exp. will be  $e_1 - e_2 + bias = 30 - 2 + 16 = 44$  and an exp. ovf. occurs. The situation is even worse if alignment of dividend is needed since the quotient's exp. will be  $e_1 + 1 - e_2 + bias = 45$  (exp. ovf.).

(d) Here, if no dividend alignment is needed, the exp. of the quotient will be  $e_1 - e_2 + bias = 2 - 30 + 16 = -12 < 0$ , so exp. undf. occurs. Even if alignment of dividend is needed, the quotient's exp. will be  $e_1 + 1 - e_2 + bias = -11$  and still an exp. undf. occurs.

(13)

[16] Look in handouts as well as in your notes that you copied from the blackboard

[17] Let the two numbers to be added or subtracted be  $A = a_{n-1} a_{n-2} \dots a_1 a_0$  and  $B = b_{n-1} b_{n-2} \dots b_1 b_0$ , where  $a_{n-1}$  and  $b_{n-1}$  are the sign bits of  $A$  and  $B$  respectively. Let  $c_{in}$  and  $c_{out}$  denote the carry into the sign location and carry out of the sign location respectively. Let the summation of  $c_{in}$ ,  $a_{n-1}$  and  $b_{n-1}$  produce the 2-bit result  $c_{out} s_{n-1}$ .

(a) Consider the case where  $a_{n-1} \neq b_{n-1}$  ( $a_{n-1}, b_{n-1} = 0, 1$  or  $1, 0$ ). Here, neither overflow nor underflow can occur since the numbers are of ~~op~~ different signs; ( $a$  positive and  $a$  negative).

Consider the following two cases:

$$\begin{array}{r} c_{in} = 0 \\ a_{n-1} = 0 \\ \underline{b_{n-1} = 1} \quad +) \\ 01 = \text{Carry } S_{n-1} \end{array}$$

$$\begin{array}{r} c_{in} = 1 \\ a_{n-1} = 0 \\ \underline{b_{n-1} = 1} \quad +) \\ 10 = \text{Carry } S_{n-1} \end{array}$$

(14)

As you see, in both cases  $c_{in} = \text{Carry}$ .  
The scenario  $a_{n-1}, b_{n-1} = 1, 0$  is the same  
as the above.

(b) Consider the case where  $a_{n-1} = b_{n-1}$ ;  
( $a_{n-1}, b_{n-1} = 0, 0$  or  $1, 1$ ).

(b<sub>1</sub>) Case  $a_{n-1} = b_{n-1} = 0$ : In this case of  
adding two positive numbers, an  
overflow might sometimes occur. Consi-  
der the case  $c_{in} = a_{n-1} = b_{n-1} = 0$ . Then

$$\begin{array}{r} c_{in} = 0 \\ a_{n-1} = 0 \\ \underline{b_{n-1} = 0} \quad +) \\ 00 = \text{Carry } S_{n-1} \end{array}$$

Here, since  $S_{n-1} = 0$ , overflow did not  
occur. Observe that  $c_{in} = \text{Carry}$ .

(15)

Consider now the case  $c_{in} = 1$  and  $a_{n-1} = b_{n-1} = 0$ . Then

$$\begin{array}{r} c_{in} = 1 \\ a_{n-1} = 0 \\ b_{n-1} = 0 \quad +) \\ \hline \end{array}$$

$$01 = c_{out} s_{n-1}$$

Here, since  $s_{n-1} = 1$ , we know that overflow occurred. Also observe that  $c_{in} \neq c_{out}$ ; (more specifically  $c_{in} = 1$  and  $c_{out} = 0$ ).

(b2) Case  $a_{n-1} = b_{n-1} = 1$ : In this case of adding two negative numbers, an underflow might sometimes occur. Consider the case  $c_{in} = a_{n-1} = b_{n-1} = 1$ . Then

$$\begin{array}{r} c_{in} = 1 \\ a_{n-1} = 1 \\ b_{n-1} = 1 \quad +) \\ \hline \end{array}$$

$$11 = c_{out} s_{n-1}$$

Since  $s_{n-1} = 1$ , underflow did not occur. Also observe that  $c_{in} = c_{out}$ .



(16)

Finally consider the case  $c_{in} = 0$  and  $a_{n-1} = b_{n-1} = 1$ . Then

$$\begin{array}{r} c_{in} = 0 \\ a_{n-1} = 1 \\ b_{n-1} = 1 \\ \hline \end{array}$$

$$10 = c_{out} s_{n-1}$$

Here, since  $s_{n-1} = 0$ , we know that underflow occurred. Also observe that  $c_{in} \neq c_{out}$ ; (more specifically,  $c_{in} = 0$  and  $c_{out} = 1$ ).

In conclusion:

- If  $c_{in} = c_{out}$  neither overflow nor underflow has occurred.
- If  $c_{in} = 1$  and  $c_{out} = 0$  an overflow has occurred.
- If  $c_{in} = 0$  and  $c_{out} = 1$  an underflow has occurred.

18 (a) Consider the integers A and B where  $A = (01110000)_2 = (112)_{10}$  and

$B = (1010)_2 = (10)_{10}$ . Dividing A by B one gets quotient  $Q = 11 = (1011)_2$  and remainder  $R = 2$ . Since  $\frac{R}{B} = \frac{2}{10} < \frac{1}{2}$ ,  $Q' = Q = 1011$ . Thus

$$\boxed{f_3 = \frac{f_1}{f_2} \approx \cdot Q' = \cdot 1011}$$

I claim that the obtained result  $\cdot 1011$  is the best 4-bit approximation of  $\frac{f_1}{f_2}$ . Observe that the actual  $\frac{f_1}{f_2}$  is

$$\frac{f_1}{f_2} = \frac{\cdot 0111}{\cdot 1010} = \frac{(\cdot 0111) \times 2^4}{(\cdot 1010) \times 2^4} = \frac{0111}{1010} = \frac{7}{10} = \cdot 7.$$

What we got is  $\cdot 1011 = \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{11}{16} = \cdot 6875$ . The error is  $\cdot 7 - \cdot 6875 = \cdot 0125$ . The next higher value is  $\cdot 1100 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} = \cdot 75$  while the error here is  $\cdot 75 - \cdot 7 = \cdot 05 > \cdot 0125$ .

(b) Consider here the integers A and B where  $A = (01100000)_2 = (96)_{10}$  and  $B = (1010)_2 = (10)_{10}$ . Dividing A by B one gets quotient  $Q = 9 = (1001)_2$  and remainder  $R = 6$ . Since  $\frac{R}{B} = \frac{6}{10} > \frac{1}{2}$ , then  $Q' = Q + 1 = 9 + 1 = 10 = (1010)_2$ .

Thus  $f_3 = \frac{f_1}{f_2} \approx \cdot Q' = \cdot 1010$

I claim that the obtained result  $\cdot 1010$  is the best 4-bit approximation of  $f_1/f_2$ . Observe that the actual  $f_1/f_2$  is  $\frac{f_1}{f_2} = \frac{\cdot 0110}{\cdot 1010} = \frac{(\cdot 0110) \times 2^4}{(\cdot 1010) \times 2^4} = \frac{0110}{1010} = \frac{6}{10} = \cdot 6$ .

What we got is  $\cdot 1010 = \frac{1}{2} + \frac{1}{8} = \frac{5}{8} = \cdot 625$ . The error is  $\cdot 625 - \cdot 6 = \cdot 025$ .

The next lower value is  $\cdot 1001 = \frac{1}{2} + \frac{1}{16} = \frac{9}{16} = \cdot 5625$  while the error here is  $\cdot 6 - \cdot 5625 = \cdot 0375 > \cdot 025$