EE 3755

Spring 2005

HW# 1 ⓐ Solutions.

EE 3755,

HW#1ⓐ Solutions

[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:

$(\text{magnitude of } X) - (\text{magnitude of } Y) = (1011011)_2 - (1101001)_2$
$= (1011011) + 2\text{'s complement of } (1101001) = (1011011) + (0010111)$

$$\begin{array}{r} 1011011 \\ +)\ \ 0010111 \\ \hline 0\ 1110010 \end{array}$$

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

$\Rightarrow \text{magnitude of } X < \text{magnitude of } Y.$

So sign bit of result = sign bit of Y = 1 and magnitude of $(X+Y)$ = 2's compl. of $(1110010) = (0001110)$.

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

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

	C	B	A	
Initialization	0	00000	11001	↳ 1 ⇒ add multiplicand and shift
+)		11110		
result of addition	0	11110	11001	
result of 1st cycle	0	01111	01100	↳ 0 ⇒ shift
result of 2nd cycle	0	00111	10110	↳ 0 ⇒ shift
result of 3rd cycle	0	00011	11011	↳ 1 ⇒ add multiplicand and shift
+)		11110		
result of addition	1	00001	11011	
result of 4th cycle	0	10000	11101	↳ 1 ⇒ add mult/cand and shift
+)		11110		
result of addition	1	01110	11101	
result of 5th cycle	0	10111	01110	

↳ product $= (1011101110)_2 = 750$
$= 30 \times 25.$

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

	B	A	d	
Initialization	000000	100101	0	
+)	010010			↳ 1, 0 ⇒ subtr. mult/cand and then shift
	010010	100101	0	
result of 1st cycle	001001	010010	1	
+)	101110			↳ 0, 1 ⇒ add mult/cand and then shift
	110111	010010	1	
result of 2nd cycle	111011	101001	0	
+)	010010			↳ 1, 0 ⇒ subtr. mult/cand and then shift.
	001101	101001	0	
result of 3rd cycle	000110	110100	1	
+)	101110			↳ 0, 1 ⇒ add mult/cand and then shift
	110100	110100	1	
result of 4th cycle	111010	011010	0	
				↳ 0, 0 ⇒ shift
result of 5th cycle	111101	001101	0	
+)	010010			↳ 1, 0 ⇒ subtr. mult/cand and then shift
	001111	001101	0	
result of 6th cycle	000111	100110	1	

↳ product $= (000111\,100110)_2$
$= 486 = (-18) \times (-27)$.

[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

(3)

	B	A	d
Initialization	0000000	100101	0

↳ 010 ⇒ add 1 × mult/cand and then do 2-bit right shift

	B	A	d
+)	1101110		
	1101110	100101	0
result of 1st cycle	1111011	101001	0

↳ 010 ⇒ add 1 × mult/cand and then do 2-bit right shift

	B	A	d
+)	1101110		
	1101001	101001	0
result of 2nd cycle	1111010	011010	0

↳ 100 ⇒ add −2 × mult/cand and then do 2-bit right shift

	B	A	d
+)	0100100		
	0011110	011010	0
result of 3rd cycle	0000111	100110	1

↳ 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 0×mult/cand and then shift
1 0	add −1×mult/cand and then shift

a_0 d	
0 1	add 1×mult/cand and then shift
1 1	add 0×mult/cand and then shift

Obviously the version corresponding to d←1 creates 1×multiplicand more than the version corresponding to d←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 multiplicand × multiplier + multiplicand.

(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←0 and d←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 0×mult/cand and then 2-bit shift
0 1 0	add 1×mult/cand and then double shift
1 0 0	add −2×mult/cand and then double shift
1 1 0	add −1×mult/cand and then double shift

a_1 a_0 d	
0 0 1	add 1×mult/cand and then double shift
0 1 1	add 2×mult/cand and then double shift
1 0 1	add −1×multiplicand and then double shift
1 1 1	add 0×mult/cand and then double shift

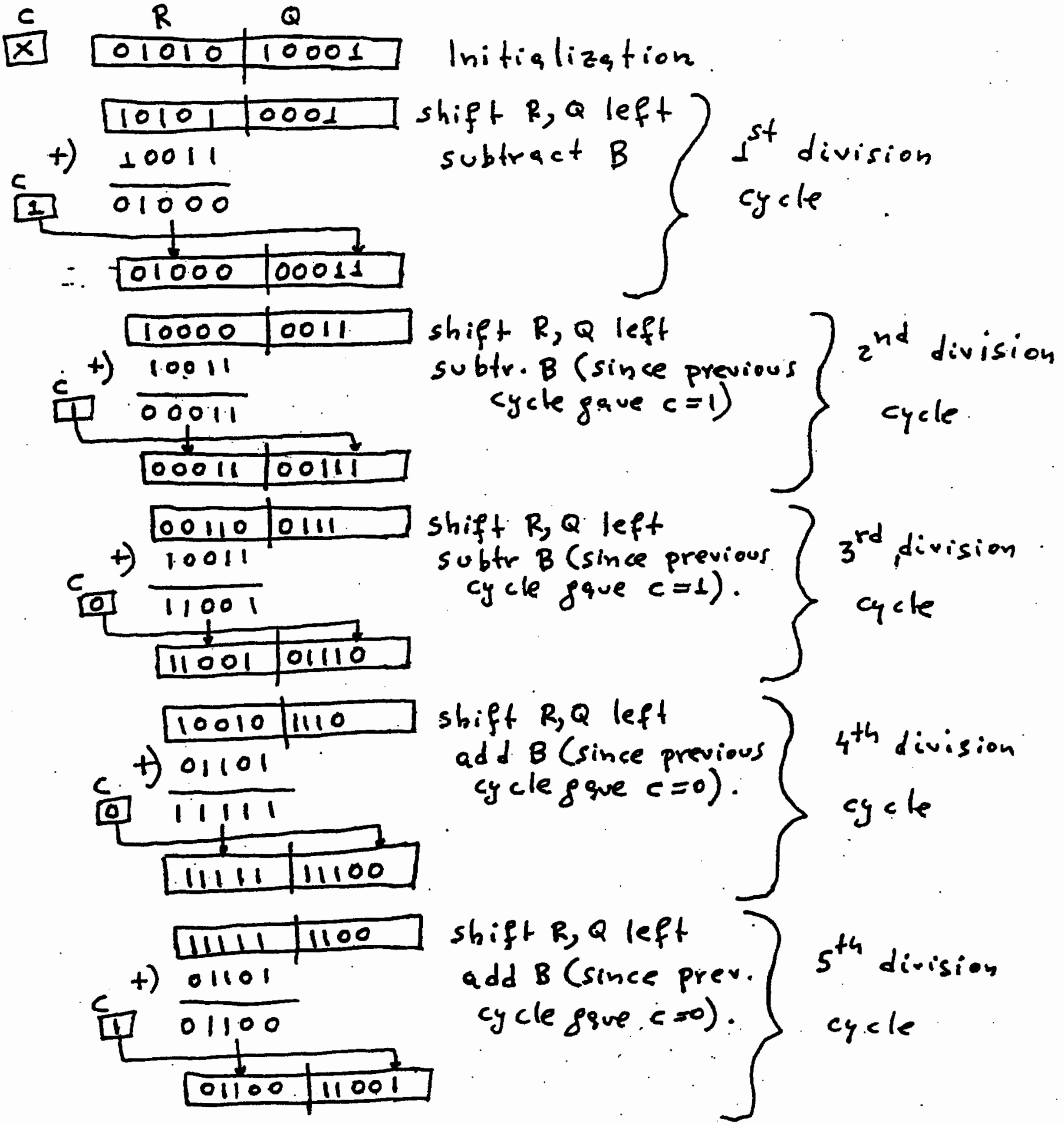
Again here, the version that uses d←1 (at initialization) creates in the first cycle 1×mult/cand more than the version that corresponds to d←0 (for initialization).

$\boxed{6}$ 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.

$\boxed{7}$ 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.

$\boxed{8}$ 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.

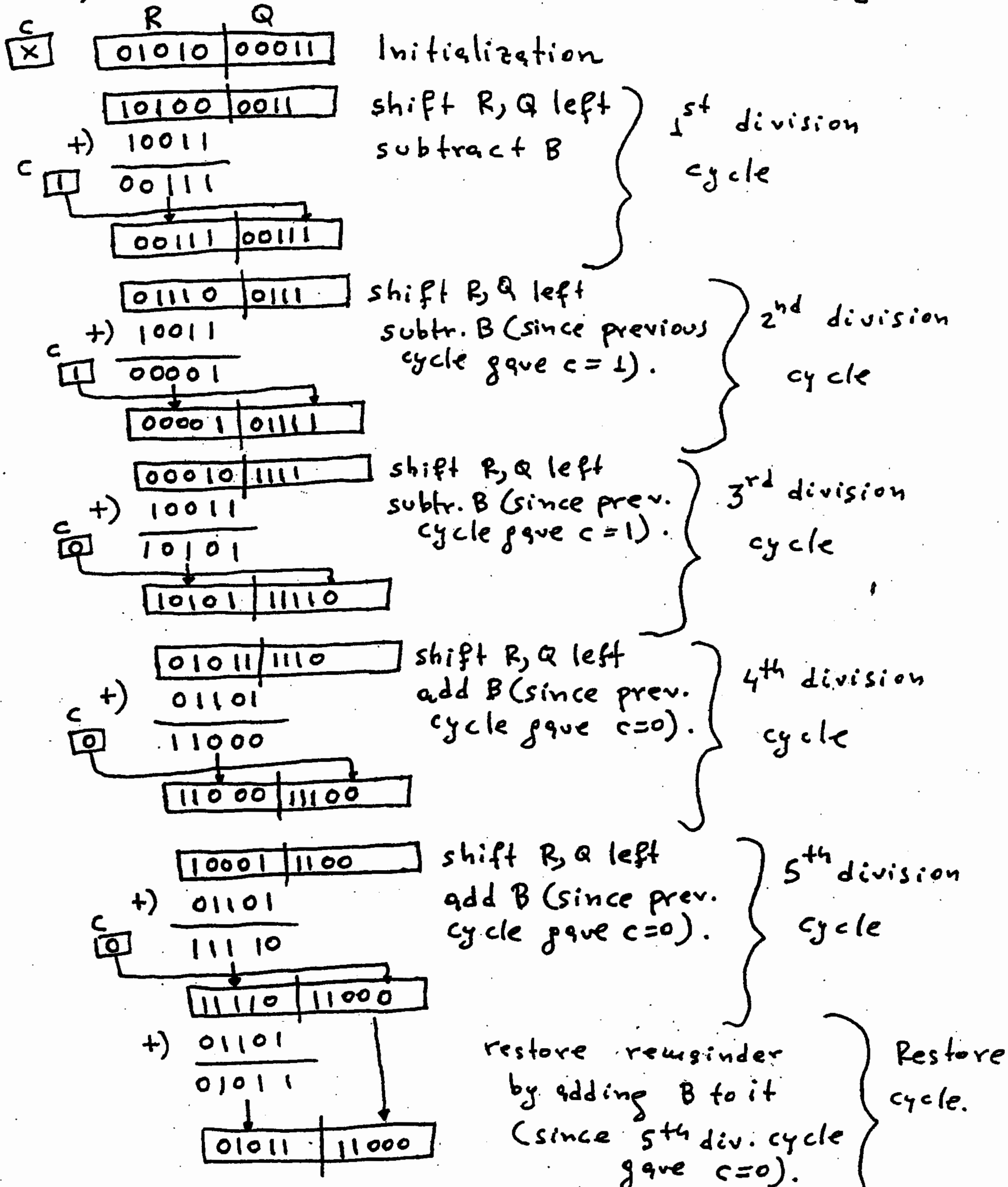
$\boxed{9}$ Here the dividend is $A=(01010\ 10001)_2 = 10\times 32+17 = 337$; the divisor is $B=(01101)_2 = 13$ and $n=5$. Also $-B$ = 2's compl. of $B = (10011)_2$



```
 C        R       Q
[X]    01010 | 10001     Initialization.

       10101 | 0001      shift R,Q left        } 1st division
  +)   10011             subtract B            }    cycle
 C
[1]    01000
       01000 | 00011

       10000 | 0011      shift R,Q left        } 2nd division
  +)   10011             subtr. B (since previous }
 C                         cycle gave c=1)     }    cycle
[1]    00011
       00011 | 00111

       00110 | 0111      shift R,Q left        } 3rd division
  +)   10011             subtr B (since previous }
 C                         cycle gave c=1).    }    cycle
[0]    11001
       11001 | 01110

       10010 | 1110      shift R,Q left        } 4th division
  +)   01101             add B (since previous }
 C                         cycle gave c=0).    }    cycle
[0]    11111
       11111 | 11100

       11111 | 1100      shift R,Q left        } 5th division
  +)   01101             add B (since prev.    }
 C                         cycle gave c=0).    }    cycle
[1]    01100
       01100 | 11001
```

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

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



Initialization

shift R, Q left
subtract B
1st division cycle

shift R, Q left
subtr. B (since previous cycle gave c = 1).
2nd division cycle

shift R, Q left
subtr. B (since prev. cycle gave c = 1).
3rd division cycle

shift R, Q left
add B (since prev. cycle gave c=0).
4th division cycle

shift R, Q left
add B (since prev. cycle gave c=0).
5th division cycle

restore remainder by adding B to it (since 5th div. cycle gave c=0).
Restore cycle.

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

$\boxed{11}$ 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

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

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

$\boxed{13}$ (a)

1. Align fractions/adjust smaller exp.

In order to find the larger exponent we can perform $e_1 - e_2 = e_1 + 2\text{'s compl. of } e_2 = (1010) + (1001)$

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

$\hookrightarrow c=1 \Rightarrow e_1 - e_2 > 0$ or $e_1 > e_2$. So e_1 is the larger exponent and the difference is $e_1 - e_2 = (011)_2 = 3$. The number A_2 now becomes

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

A fraction underflow occured 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

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

$\hookrightarrow$ fraction overflow.

3. <u>Postnormalize</u> : After postnormalization we get

$A_3 = A_1 + A_2$:

s_3	e_3	f_3
1	1011	10000111

A fraction underflow occured as a result of postnormal.

4. <u>Check for exp ovf</u> : No exp. ovf occured ($e_3 = (1011)_2 = 11$ 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. <u>Align/adjust</u> : 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. <u>Add fractions</u> : A true addition $f_1 + f_2'$ has to take place.

$$
\begin{array}{rr}
f_1 + f_2' = & .1111\,0101 \\
+) & .0001\,0110 \\
\hline
& 1.0000\,1011
\end{array}
$$

↳ fract. overflow (postnormalization needed)

3. <u>Postnormalize and check for exp ovf</u>

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

(d) 1. <u>Align/adjust</u> : 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 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 =

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

↳ $c = 0 \Rightarrow f_1' - f_2 < 0 \Rightarrow f_1' < f_2$. 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 $A_3 = A_1 + A_2$: $\boxed{1 \mid 1011 \mid 01010110}$

3. Postnormalize: After postnormalization we get

A_3 :

s_3	e_3	f_3
1	1010	10101100

4. Check for exp. underflow

No exp. underflow occured.

(e) 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 =

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

↳ $c = 1 \Rightarrow f_1 - f_2 > 0 \Rightarrow f_1 > f_2$. 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 : $\boxed{0 \mid 0010 \mid 00000100}$

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 occured an exp. underflow flag is set and the result is forced to the unique FLP zero or

A_3: $\overset{s_3}{\boxed{0}}\overset{e_3}{\boxed{0000}}\overset{f_3}{\boxed{00000000}}$.

(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 righmost 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 $A_3 = A_1 \times A_2$: $\overset{s_3}{\boxed{1}}\overset{e_3}{\boxed{1011}}\overset{f_3}{\boxed{11110100}}$. Observe that the result is normalized so no postnormalization is needed. No exp. ovf or exp. underflow occured.

$\boxed{14}$ (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. ovf 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 ovf nor exp. undf.

(c) If no postnormalization is needed, the product's exponent 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 worst if postnormalization is needed since the product's exp. will be $e_1+e_2-bias-1 = -7$ (exp. undf.)

$\boxed{15}$ (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 worst 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.

[16] Look in handouts ~~[illegible]~~

[17] Let the two numbers to be added or subtracted be $A = a_{n-1} a_{n-2} \cdots a_1 a_0$ and $B = b_{n-1} b_{n-2} \cdots b_1 b_0$, 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 ~~ep~~ different signs; (a positive and a negative).

Consider the following two cases:

$$\begin{array}{rr} & c_{in}=0 \\ & a_{n-1}=0 \\ +) & b_{n-1}=1 \\ \hline 01 = & C_{out}\, S_{n-1} \end{array}$$

$$\begin{array}{rr} & c_{in}=1 \\ & a_{n-1}=0 \\ +) & b_{n-1}=1 \\ \hline 10 = & C_{out}\, S_{n-1} \end{array}$$

As you see, in both cases $c_{in} = C_{out}$. 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_1) Case $a_{n-1} = b_{n-1} = 0$: In this case of adding two positive numbers, an overflow might sometimes occur. Consider the case $c_{in} = a_{n-1} = b_{n-1} = 0$. Then

$$\begin{array}{rr} & c_{in}=0 \\ & a_{n-1}=0 \\ +) & b_{n-1}=0 \\ \hline 00 = & C_{out}\, S_{n-1} \end{array}$$

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

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

$$\begin{array}{rr} & C_{in}=1 \\ & a_{n-1}=0 \\ & \underline{b_{n-1}=0 \quad +)} \\ 0\,1 = C_{out}\,S_{n-1} & \end{array}$$

Here, since $S_{n-1}=1$, we know that overflow occured. 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

$$\begin{array}{rr} & C_{in}=1 \\ & a_{n-1}=1 \\ & \underline{b_{n-1}=1 \quad +)} \\ 1\,1 = C_{out}\,S_{n-1} & \end{array}$$

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

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

$$\begin{array}{r} C_{in} = 0 \\ a_{n-1} = 1 \\ b_{n-1} = 1 \\ \hline 10 = C_{out}\, S_{n-1} \end{array}$$

Here, since $S_{n-1} = 0$, we know that underflow occured. 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 occured.
- If $C_{in} = 1$ and $C_{out} = 0$ an overflow has occured.
- If $C_{in} = 0$ and $C_{out} = 1$ an underflow has occured.

18 [redacted] (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 $\frac{R}{B} = \frac{2}{10} < 1/2$, $Q' = Q = 1011.$ Thus

$$\boxed{f_3 = \frac{f_1}{f_2} \cong .Q' = .1011}$$

I claim that the obtained result .1011 is the best 4-bit approximation of f_1/f_2. Observe that the actual f_1/f_2 is

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

What we got is $.1011 = \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{11}{16} = .6875$. The error is $.7 - .6875 = .0125$

The next higher value is $.1100 = = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} = .75$ while the error here is $.75 - .7 = .05 > .0125$

(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 $\boxed{f_3=\frac{f_1}{f_2}\cong .Q'=.1010}$

I claim that the obtained result $.1010$ is the best 4-bit approximation of f_1/f_2. Observe that the actual f_1/f_2 is

$$\frac{f_1}{f_2}=\frac{.0110}{.1010}=\frac{(.0110)\times 2^4}{(.1010)\times 2^4}=\frac{0110.}{1010.}$$

$$=\frac{6}{10}=.6.$$

What we got is $.1010=\frac{1}{2}+\frac{1}{8}=\frac{5}{8}=.625$

The error is $.625-.6=.025$.

The next lower value is $.1001=\frac{1}{2}+\frac{1}{16}$ $=\frac{9}{16}=.5625$ while the error here is

$.6-.5625=.0375>.025$