CSE 321b: Computer Organization (II) Third Year, Computer & Systems Engineering

Solution to Assignment #3

1. Apply **Booth's** algorithm to multiply -12 (multiplicand) by +6 (multiplier). Represent the numbers using the least number of bits.

<u>A</u>	\mathbf{Q}	<u>Q-1</u>	$\underline{\mathbf{M}}$	
00000	00110	O	10100	Initial values
00000	00011	O		Shift
01100	00011	0		$A \leftarrow A - M$
00110	00001	1		Shift
00011	0000	1		Shift
10111	00000	1		$A \leftarrow A + M$
11011	1000 <mark>0</mark>	O		Shift
11101	11000	0		Shift

2. Show all the steps required to divide +14 (dividend) by -5 (divisor) using the **non-restoring division** algorithm. Represent the numbers using the least number of bits.

<u>A</u>	Q	<u>M</u>	
00000	$01\overline{1}10$	11011	Initial values
<mark>0</mark> 0000	01110	00101	Take absolute of Q & M
00000	1110?		Shift
<mark>1</mark> 1011			Subtract
<mark>1</mark> 1011	1110 0		$Q_0 \leftarrow 0$
10111	1100?		Shift
11100			Add
<mark>1</mark> 1100	1100 0		$Q_0 \leftarrow 0$
11001	1000?		Shift
<mark>1</mark> 1110			Add
<mark>1</mark> 1110	1000 0		$Q_0 \leftarrow 0$
11101	0000?		Shift
<mark>0</mark> 0010			Add
<mark>0</mark> 0010	00001		$Q_0 \leftarrow 1$
00100	0001?		Shift
<mark>1</mark> 1111			Subtract
<mark>1</mark> 1111	00010		$Q_0 \leftarrow 0$
00100	00010		Add
00100	11110		Adjust signs of A & Q

3. Consider the IEEE 754 **half-precision** format (which is also known as: **binary16**) in which floating point numbers are represented using 16 bits: 1 sign bit, 5-bit biased exponent, and 10-bit fraction. Convert the following numbers to their IEEE half-precision counterparts:

(a) 4.57763671875*10⁻⁵

```
4.57763671875*10^{-5} = 2^{-14.405} = 0.75*2^{-14} \implies 0.0000011000000000
```

(b) -217.375

```
-217.375 = -2^{7.764} = -1.6982421875 * 2^7 \rightarrow 1 10110 1011001011
```

- 4. Suppose the IEEE 754 Standard has a **binary14** format that uses: 1 sign bit, 6-bit biased exponent, and 7-bit fraction. Perform the following calculations while interpreting each of the given binary values as a binary16 floating-point number. Use two guard bits and round the result to the **nearest** binary16 number whenever is necessary.
 - (a) $1\ 101001\ 1110000 + 0\ 100111\ 1000010$
 - i) Check for special cases
 - → No special cases
 - ii) Transform subtraction to addition and negate second number
 - → Not needed
 - iii) Align
 - → Second number has a smaller exponent
 - → Add 2 to its exponent and shift its fraction to the right twice
 - \rightarrow Exponent of second number = 101001
 - \rightarrow Significand of second number = 0.011000010
 - iv) Add significands (taking signs into consideration)
 - \rightarrow Significand of result = -1.111000000 + 0.011000010 = -1.0111111110 (no overflow!)
 - → Fraction of result = 0111111110
 - \rightarrow Sign bit of result = 1
 - v) Normalize
 - → Not needed
 - \rightarrow Exponent of result = 101001
 - vi) Round
 - → Candidate fractions are 0111111 and 1000000
 - \rightarrow Guard bits = 10
 - → Round to the even candidate
 - \rightarrow Fraction of result = 1000000

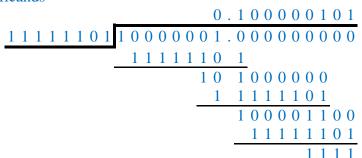
Result = 1 101001 1000000

- (b) $1\ 1111111\ 00000000\ \times\ 1\ 000000\ 00000000$
 - i) Check for special cases
 - \rightarrow First number represents $-\infty$, Second number represents -0
 - → Result is qNaN

Result = x 11111 1xxxxx

- (c) $1\ 011101\ 0000001$ \div $0\ 100010\ 1111101$
 - i) Check for special cases
 - → No special cases
 - ii) Subtract exponents (and add the bias)

- \rightarrow Exponent of result = 011101 100010 + 011111 = 011010 (no overflow/underflow!)
- iii) Calculate sign of result
 - → Numbers have different signs
 - \rightarrow Sign bit of result = 1
- iv) Divide significands



- \rightarrow Significand of result = 0.100000101
- v) Normalize
 - → Subtract 1 from exponent and shift significant (one position to the) left
 - \rightarrow Exponent of result = 011001 (no underflow)
 - \rightarrow Fraction of result = 000001010
- vi) Round
 - → Candidate fractions are 0000010 and 0000011
 - \rightarrow Guard bits = 10
 - → Round to the even candidate
 - \rightarrow Fraction of result = $\frac{0000010}{1}$

Result = 1 011001 0000010