CSCI 516 - Fundamental Concepts in Computing and Machine Organization
Homework Assignment Solution

1. Describe with an example how addition, subtraction, multiplication, and division is done in CPU using the binary number system.

Addition: Digits are added bit by bit from right to left, with carries passed to the next digit to the left, just as you would do by hand.

$$
\begin{array}{rl}
 & 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000111_{two} = 7_{ten} \\
+ & 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000110_{two} = 6_{ten} \\
\hline
= & 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00001101_{two} = 13_{ten}
\end{array}
$$

Figure 1: Binary Addition

Subtraction uses addition: the appropriate operand is simply negated before being added.

$$
\begin{array}{rl}
 & 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000111_{two} = 7_{ten} \\
+ & 11111111\ 11111111\ 11111111\ 11111111\ 11111111\ 11111111\ 11111111\ 11111010_{two} = -6_{ten} \\
\hline
= & 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000001_{two} = 1_{ten}
\end{array}
$$

Figure 2: Binary Subtraction

Multiplication: The first operand is called the multiplicand and the second the multiplier. The final result is called the product. From right to left, multiplying the multiplicand by the single digit of the multiplier, and shifting the intermediate product one digit to the left of the earlier intermediate products. Each step of the multiplication is simple:

- Just place a copy of the multiplicand (1 multiplicand) in the proper place if the multiplier digit is a 1, or
- Place 0 (0 multiplicand) in the proper place if the digit is 0.

```
Multiplicand        1000
Multiplier     x    1001
                    ----
                    1000
                   0000
                  0000
                 1000
                 -------
Product          1001000
```

Figure 3: Example of binary multiplication

Division: Divides two operands, called the dividend and divisor, and the result, called the quotient, are accompanied by a second result, called the remainder. The basic division algorithm tries to see how big a number can be subtracted, creating a digit of the quotient on each attempt.

```
                  1001      Quotient
              --------
Divisor 1000  |1001010      Dividend
              -1000
              -----
                  10
                  101
                  1010
                 -1000
                 -----
                    10      Remainder
```

Figure 4: Example of binary division

2. Convert each of the following decimal values to IEEE-754 single and IEEE-754 double precision representation. Write your converted result in hexadecimal format. Clearly show all the steps.

- 3.75
- -12.5

a. 3.75

$3.75 = 15/4 = 15/2^2 = 1111_{two}/2^2_{ten} = 11.11_{two} = 1.111 * 2^1$

Subtracting the bias 127 from the exponent of $1.111 * 2^1$:

$(-1)^0 * (1 + .1110\ 0000\ 0000\ 0000\ 0000\ 000_{two}) * 2^{(128-127)}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 bit				8 bits														23 bits													

Figure 5: 3.75 IEEE-754 single precision representation

The Hexadecimal format of the single precision representation is: 0x40700000

Subtracting the bias 1023 from the exponent of $1.111 * 2^1$:

$(-1)^0 * (1 + 0.1110\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000) * 2^{(1024-1023)}$

The Hexadecimal format of the double precision representation is: 0x400E000000000000

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1bit				11 bits																20 bits											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											32 bits																				

b. -12.5

$-12.5 = -25/2 = -11001_{two}/2^1_{ten} = -1100.1 = -1.1001 * 2^3$

Subtracting the bias 127 from the exponent of $-1.1001 * 2^3$:

$(-1)^1 * (1 + .1001\ 0000\ 0000\ 0000\ 0000\ 000_{two}) * 2^{(130-127)}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 bit				8 bits														23 bits													

Figure 7: -12.5 IEEE-754 single precision representation

The Hexadecimal format of the single precision representation is: 0xC1480000

Subtracting the bias 1023 from the exponent of $-1.1001 * 2^3$:

$(-1)^1 * (1 + 0.1001\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000) * 2^{(1026-1023)}$

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1bit				11 bits																20 bits											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											32 bits																				

Figure 8: -12.5 IEEE-754 single precision representation

The Hexadecimal format of the double precision representation is: 0xC029000000000000

3. Convert each of the following IEEE-754 floating point representation to decimal values. Clearly show all the steps.

- 0x40200000
- 0xC1080000

a. 0x40200000

$$\begin{aligned} 0x40200000 &= 0100\ 0000\ 0010\ 0000\ 0000\ 0000\ 0000\ 0000 \\ &= (-1)^0 \ * \ (1 \ + \ 1 \ * \ 2^{-2}) \ * \ 2^{(128-127)} \\ &= 1 \ * \ (1 \ + \ 0.25) \ * 2 \\ &= 2.5 \end{aligned} \tag{1}$$

b. 0xC1080000

$$\begin{aligned} 0xC1080000 &= 1100\ 0001\ 0000\ 1000\ 0000\ 0000\ 0000\ 0000 \\ &= (-1)^1 \ * \ (1 \ + \ 1 \ * \ 2^{-4}) \ * \ 2^{(130-127)} \\ &= -1 \ * \ (1 \ + \ 0.0625) \ * 2^3 \\ &= -8.5 \end{aligned} \tag{2}$$