IC220: HW 4

Due: 13 Feb 2019

Full Name: ____________________ **Alpha:** ____________________

Circle Your Section: Aviv/1001 Aviv/2001 Aviv/4001 Choi/5001 Missler/5002

Total Points: 110

Preliminary: Carefully do the assigned reading for Chapter 2 (2.1-2.3,2.5-2.10,2.12)

---

1. Convert the given decimal numbers to their binary representation

   (a) **[5 points]**

	5 (4-bits)	-7 (4-bits)
Unsigned		
Sign Magnitude		
One's Compliment		
Two's Compliment		

   (b) **[5 points]**

	-3 (4-bits)	-3 (6-bits)
Sign Magnitude		
One's Compliment		
Two's Compliment		

2. Assume the following is in binary two's complement form:
   (a) **[1 point]** 001011

   (b) **[2 points]** 111011

3. Apply the negation operator to the binary values, and show the resulting binary value, in two's complement.
   (a) **[1 point]** -(001011)

   (b) **[1 point]** -(111011)

4. Suppose we use 8-bits to represent a two's complement binary number.
   (a) **[5 points]** What is the largest number that can be presented? (Give answer in binary **and** decimal)

   (b) **[5 points]** What is the smalles number that can be presented? (Give answer in binary **and** decimal)

5. **[10 points]** Complete the following 6-bit, two's complement additions. Indicate if there is an overflow or not.

   (a) [**points**]

   ```
         010101
   +     001101
   ------------
   ```

   (b) [**points**]

   ```
         111111
   +     111101
   ------------
   ```

   (c) [**points**]

   ```
         010011
   +     001110
   ------------
   ```

   (d) [**points**]

   ```
         010011
   +     111110
   ------------
   ```

6. **[10 points]** Complete the following 6-bit, two's complement **subtraction**. Indicate if there is an overflow or not.

   (a) [**points**]

```
      011101
-     100101
------------
```

   (b) [**points**]

```
      111111
-     111101
------------
```

   (c) [**points**]

```
      010011
-     001110
------------
```

   (d) [**points**]

```
      010011
+     111110
------------
```

7. **[5 points]** Convert the (decimal) 269 into a 32-bit two's complement binary number. *(Note, you can use a calculator for this, but you'd be expected to do this by hand, without a calculator, on a exam.)*

8. **[5 points]** Convert the (decimal) -45 into a 32-bit two's complement binary number. *(Note, you can use a calculator for this, but you'd be expected to do this by hand, without a calculator, on a exam.)*

9. Convert the following 32-bit binary, two's complement number into decimal. *(Note, you can use a calculator for this, but you'd be expected to do this by hand, without a calculator, on a exam.)*

   (a) **[5 points]**

   `1111 1111 1111 1111 1111 1111 1000 0110`

   (b) **[5 points]**

   `0000 0000 0000 0000 0000 0101 0110`

10. Convert the following 32-bit, single precision float into a decimal (base 10) number. You can leave your answer in reduced fraction form or in decimal. For convenience, the number is broken with hyphens for different segments of the encoding. *(Note, you can use a calculator for this, but you'd be expected to do this by hand, without a calculator, on a exam.)*

    (a) [**5 points**]

    `1 - 0 1 1 1 1 1 0 0 - 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`

    (b) [**5 points**]

    `0 - 1 0 0 0 0 0 1 0 - 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`

11. [**5 points**] Show the encoding of 12.6875 in 32-bit, single precision float. *(Note, you can use a calculator for this, but you'd be expected to do this by hand, without a calculator, on a exam.)*

12. [**5 points**] Convert the following C code to MIPS. You can assume single precision floats, and use pseudo instruction `li.s`.

```
float pick (float G[], int index){
   return G[index];
}
```

13. [**5 points**] Convert the following C code to MIPS. You can assume single precision floats, and use pseudo instruction `li.s`.

```
float maxdiv(float A, float B){
  if(A> B) return A/B;
  else     return B/A;
}
```

14. [**5 points**] Convert the following C code to MIPS. You can assume single precision floats, and use pseudo instruction `li.s`.

```
float sum(float A[], int N){
   int j;
   float sum = 0.0;
   for (j=0; j<N; j++){
      sum = sum + A[j];
   return sum;
}
```

15. [**5 points**] Convert the following C code to MIPS. You can assume single precision floats, and use pseudo instruction `li.s`.

```
float foo(float x, float y){
  if (x > y)
    return x + y;
  else
    return x - y;
}
```

16. [**5 points**] Convert the following C code to MIPS. Note: use **integers** not floats here! Also, use **mult** instruction that we learned in class that takes just 2 arguments.

```
int muskrat(int g, int h){
   int prod = g * h;
   if (prod < 0)
      prod *= -1;
   return prod;
}
```

17. [**5 points**] Convert the following C code to MIPS. Note: use **integers** not floats here! Also, use **mult** instruction that we learned in class that takes just 2 arguments.

```
int log(int x, int b){
   int r = 0;
   while (x < b){
      x = x*x;
      r+=1;
   }
   return r;
}
```