(5 pts) Exercise 3-1

- **Assume we have 4 bits. Convert the given decimal numbers to the stated binary representations.**

	5	-7
Unsigned		
Sign Magnitude		
One's Comp.		
Two's Comp.		

(5 pts) Exercise 3-2

- **Convert the given decimal numbers to the stated binary representations.**

	-3 (using 4 bits)	-3 (using 6 bits)
Sign Magnitude		
One's Comp		
Two's Comp.		

(5 pts) Exercise 3-3

- **Assume the following is in binary two's complement form.**
  What do they represent in decimal?

  `001011`

  `111011`

- **Now negate these numbers and show the new binary form:**

  `-(001011) =`

  `-(111011) =`

(10 pts) Exercise 3-6

- **Suppose we use 8 bits to represent a two's complement binary number. What is the largest number that can be represented (give answer in binary AND decimal)**

- **What is the smallest number that can be represented (give binary AND decimal)**

(5 pts) Exercise 3-11

- **Do the following assuming <u>6 bit, two's complement numbers</u>. Circle YES if overflow occurs or NO if not**

```
  010101   YES
+ 001101   NO
```

```
  111111   YES
+ 111101   NO
```

```
  010011   YES
+ 001110   NO
```

```
  010011   YES
+ 111110   NO
```

(5 pts) Exercise 3-12

- **Do the following assuming <u>6 bit, two's complement numbers</u>. When does overflow occur? (<u>note subtraction here</u>)**

```
  011101   YES
- 100101   NO
```

```
  111111   YES
+ 111101   NO
```

```
  010011   YES
- 001110   NO
```

```
  010011   YES
+ 111110   NO
```

(10 pts) Exercise 3-16

(You COULD use a calculator for these. But recommended not – you should be able to do this by hand on an exam, where calculators are not permitted).

Convert 257_{ten} into a 32-bit two's complement binary number.

- **Convert -37_{ten} into a 32-bit two's complement binary number.**

(10 pts) Exercise 3-17

(You COULD use a calculator for these. But recommended not – you should be able to do this by hand on an exam, where calculators are not permitted).

What decimal number does this two's complement binary number represent?

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000\ 0110_{two}$

What decimal number does this two's complement binary number represent?

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0110_{two}$

(5 pts) Exercise 3-21

- **Convert the following C code to MIPS:**

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

(5 pts) Exercise 3-22

- **Convert the following C code to MIPS:**

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

(5 pts) Exercise 3-23

- **Convert the following C code to MIPS:**

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

(10 pts) Exercise 3-25

- **Convert the following C code into MIPS.**

```
float function2 (float x, float y) {
  if (x > y)
    return x + y;
  else
    return x - y;
}
```

(20 pts) Exercise 3-26

- **Convert the following C code into MIPS. A C float is stored as a MIPS single precision floating point value.**

```
float dotproduct (float A[], float B[]) {
   float sum = 0;
   int ii;
   for (ii = 0; ii < 20; ii++) {
       sum = sum + A[ii] * B[ii];
   }
   return sum;
}
```

(10 pts) Exercise 3-27

- **Convert the following C code into MIPS. ASSUME that the result of multiplying g by h will always fit in just 32 bits.**

NOTE 1: using integers, not floats, here!

NOTE 2: Use the integer "mult" instruction that we learned in class (that takes just 2 arguments) NOT a pseudo-instruction that takes 3 arguments

```
int function6 (int g, int h) {
  int prod = g * h;
  if (prod < 0)
    prod *= -1;
  return prod;
}
```

(3 pts EXTRA CREDIT) Exercise 3-31

- **Convert the following C code to MIPS:**

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