(5 pts) Exercise 3-1

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

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>-7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sign Magnitude</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One’s Comp.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two’s Comp.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5 pts) Exercise 3-2

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

<table>
<thead>
<tr>
<th></th>
<th>-3 (using 4 bits)</th>
<th>-3 (using 6 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign Magnitude</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One’s Comp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two’s Comp.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(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 6 bit, two’s complement numbers. Circle YES if overflow occurs or NO if not.

<table>
<thead>
<tr>
<th>Addends</th>
<th>Overflow</th>
<th>Addends</th>
<th>Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>010101</td>
<td>YES</td>
<td>111111</td>
<td>YES</td>
</tr>
<tr>
<td>+ 001101</td>
<td>NO</td>
<td>+ 111101</td>
<td>NO</td>
</tr>
<tr>
<td>010011</td>
<td>YES</td>
<td>010011</td>
<td>YES</td>
</tr>
<tr>
<td>+ 001110</td>
<td>NO</td>
<td>+ 111110</td>
<td>NO</td>
</tr>
</tbody>
</table>

(5 pts) Exercise 3-12

- Do the following assuming 6 bit, two’s complement numbers. When does overflow occur? (note subtraction here).

<table>
<thead>
<tr>
<th>Subtrahend</th>
<th>Minuend</th>
<th>Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>011101</td>
<td>11111</td>
<td>YES</td>
</tr>
<tr>
<td>- 100101</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>010011</td>
<td>11111</td>
<td>YES</td>
</tr>
<tr>
<td>- 111101</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>010011</td>
<td>11111</td>
<td>YES</td>
</tr>
<tr>
<td>- 001110</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>010011</td>
<td>11111</td>
<td>YES</td>
</tr>
<tr>
<td>- 111110</td>
<td></td>
<td>NO</td>
</tr>
</tbody>
</table>
(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_{10} into a 32-bit two's complement binary number.

- Convert -37_{10} 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?

\[1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000\ 0110\_\text{two}\]

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

\[0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0110\_\text{two}\]
(5 pts) Exercise 3-21

- Convert the following C code to MIPS:
  ```c
  float pick (float G[], int index) {
    return G[index];
  }
  ```

(5 pts) Exercise 3-22

- Convert the following C code to MIPS:
  ```c
  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:
  ```c
  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.

```c
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.

```c
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

```c
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:

```c
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;
}
```