

### (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 3-11

---

- Do the following assuming 6 bit, two's complement numbers.  
When does overflow occur?

$$\begin{array}{r} 010101 \\ + 001101 \\ \hline \end{array}$$

$$\begin{array}{r} 111111 \\ + 111101 \\ \hline \end{array}$$

$$\begin{array}{r} 010011 \\ + 001110 \\ \hline \end{array}$$

$$\begin{array}{r} 010011 \\ + 111110 \\ \hline \end{array}$$

### (5 pts) Exercise 3-12

---

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

$$\begin{array}{r} 011101 \\ - 100101 \\ \hline \end{array}$$

$$\begin{array}{r} 111111 \\ - 111101 \\ \hline \end{array}$$

$$\begin{array}{r} 010011 \\ - 001110 \\ \hline \end{array}$$

$$\begin{array}{r} 010011 \\ - 111110 \\ \hline \end{array}$$

## (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  $257_{\text{ten}}$  into a 32-bit two's complement binary number.

- Convert  $-37_{\text{ten}}$  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<sub>two</sub>

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

0000 0000 0000 0000 0000 0000 0001 0110<sub>two</sub>

## (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: do NOT use use a “mult” pseudo-instruction – use what we did in class instead

```
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;
}
```