Mark Mocek
Week 4

2.1

For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, and i are given and could be considered 32-bit integers as declared in a C program. Use a minimal of MIPS assembly instructions. f = g + (h - 5);

```
sub     i, h, 5
add     f, g, i
```

2.2

For the following MIPS assembly instructions, what is a corresponding C statement?

```
add f, g, h             add f, i, f
f = g + h;              f = i + f;
```

2.4

For the MIPS assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

```
sll     $t0, $s0, 2         # $t0 = f * 4
add     $t0, $s6, $t0       # $t0 = &A[f]
sll     $t1, $s1, 2         # $t1 = g * 4
add     $t1, $s7, $t1       # $t1 = &B[g]
lw      $s0, 0($t0)         # f = A[f]
addi    $t2, $t0, 4         # $t2 = f + 4
lw      $t0, 0($t2)         # $t0 = A[0]
add     $t0, $t0, $s0       # $t0 = A[0] + f
sw      $t0, 0($t1)         # $t0 = B[0]
```

2.8

Translate 0xABCDEF12 into decimal.

- 2,882,400,018

2.19.1

For the register values shown, what is the value of $t2 for the following sequence of instructions?

```
        $t0 = 0xAAAAAAAA, $t1 = 0x12345678
        sll     $t2, $t0, 4             # makes $t2 0xAAAAAAA0
        or      $t2, $t2, $t1           # logical 'or' of $t1 paired with $t2
$t1     0001 0010 0011 0100 0101 0110 0111 1000
$t2     1010 1010 1010 1010 1010 1010 1010 0000
-------------------------------------------------------------
$t2     1011 1010 1011 1110 1111 1110 1111 1000
```

2.19.2

For the register values shown, what is the value of $t2 for the following sequence of instruction?

```
        $t0 = 0xAAAAAAAA, $t1 = 0x12345678
        sll     $t2, $t0, 4             # makes $t2 0xAAAAAAA0
        andi    $t2, $t2, -1            # logical 'and' of $t2 and -1
$t2     1010 1010 1010 1010 1010 1010 1010 0000
-1      0000 0000 0000 0000 0000 0000 0000 1111
-------------------------------------------------------------
$t2     0000 0000 0000 0000 0000 0000 0000 0000
```

2.29

Translate the following loop into C. Assume that the C-level integer i is held in register $t1, $s2 holds the C-level integer called *result,* and $s0 holds the base address of the integer *MemArray*.

```
        addi    $t1, $0, $0             i = 0;
LOOP:   lw      $s1, 0($s0)
        add     $s2, $s2, $s1           for (1 < 100; i=i+1)
        addi    $s0, $s0, 4             {
        addi    $t1, $t1, 1             result = result + MemArray[i];
        slti    $t2, $t1, 100           }
        bne     $t2, $s0, LOOP
```