2.3 [5] <§§2.2, 2.3> For the following C statement, what is the corresponding MIPS assembly code? 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.

```
B[8] = A[i-j];
```

2.4 [5] <§§2.2, 2.3> 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
lw    $t0, 0($t2)
add   $t0, $t0, $s0
sw    $t0, 0($t1)
```

2.7 [5] <§2.3> Show how the value 0xabcdef12 would be arranged in memory of a little-endian and a big-endian machine. Assume the data is stored starting at address 0.

2.12 Assume that registers $s0 and $s1 hold the values 0x80000000 and 0xD0000000, respectively.

2.12.1 [5] <§2.4> What is the value of $t0 for the following assembly code?

```
add $t0, $s0, $s1
```

2.12.2 [5] <§2.4> Is the result in $t0 the desired result, or has there been overflow?

2.12.3 [5] <§2.4> For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code?

```
sub $t0, $s0, $s1
```

2.12.4 [5] <§2.4> Is the result in $t0 the desired result, or has there been overflow?

2.12.5 [5] <§2.4> For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code?

```
add $t0, $s0, $s1
add $t0, $t0, $s0
```

2.12.6 [5] <§2.4> Is the result in $t0 the desired result, or has there been overflow?

2.19 Assume the following register contents:

```
$t0 = 0xAAAAAAAA, $t1 = 0x12345678
```

2.19.1 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll $t2, $t0, 44
or  $t2, $t2, $t1
```

2.19.2 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll  $t2, $t0, 4
andi $t2, $t2, -1
```

2.19.3 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
srl  $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

2.20 [5] <§2.6> Find the shortest sequence of MIPS instructions that extracts bits 16 down to 11 from register $t0 and uses the value of this field to replace bits 31 down to 26 in register $t1 without changing the other 26 bits of register $t1.

2.27 [5] <§2.7> Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers $s0, $s1, $t0, and $t1, respectively. Also, assume that register $s2 holds the base address of the array D.

```
for(i=0; i<a; i++)
    for(j=0; j<b; j++)
        D[4*j] = i + j;
```

2.39 [5] <§2.10> Write the MIPS assembly code that creates the 32-bit constant 0010 0000 0000 0001 0100 1001 0010 0100$_{two}$ and stores that value to register $t1.