CMPEN 331 – Computer Organization and Design,
Chapter 2-2 Review Questions

1. 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. The table below shows 32-bit values of an array stored in memory.

Address	Data
24	2
38	4
32	3
36	6
40	1

For the memory locations in the table above, the following C code is sorting the data from lowest to highest, placing the lowest value in the smallest memory location shown in the figure. The data shown represents the C variable called Array, which is an array of type int, and that the first number in the array shown is the first element in the array. Assume that this particular machine is a byte-addressable machine and a word consists of four bytes.

```
temp = Array[0];
temp2 = Array[1];
Array[0] = Array[4];
Array[1] = temp;
Array[4] = Array[3];
Array[3] = temp2;
```

a. For the memory locations in the table above, write MIPS code to sort the data from lowest to highest, placing the lowest value in the smallest memory location. Assume the base address of Array is stored in register $s6.

3. Translate function f into MIPS assembly language. If you need to use registers `$t0` through `$t7`, use the lower-numbered registers first. Assume the function declaration for `func` is `"int func(int a, int b);"`. The code for function f is as follows:

```
int f(int a, int b, int c, int d){
   return func(func(a, b), c + d);
}
```

4. Right before your function f from problem 12 returns, what do we know about contents of registers `$s3`, `$ra`, and `$sp`? Keep in mind that we know what the entire function `f` looks like, but for function `func` we only know its declaration.

Solutions

1. 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)
```

```
B[g] = A[f] + A[1+f];
```

2. The table below shows 32-bit values of an array stored in memory.

Address	Data
24	2
38	4
32	3
36	6
40	2

For the memory locations in the table above, the following C code is sorting the data from lowest to highest, placing the lowest value in the smallest memory location shown in the figure. The data shown represents the C variable called Array, which is an array of type int, and that the first number in the array shown is the first element in the array. Assume that this particular machine is a byte-addressable machine and a word consists of four bytes.

```
temp = Array[0];
```

```
temp2 = Array[1];
Array[0] = Array[4];
Array[1] = temp;
Array[4] = Array[3];
Array[3] = temp2;
```

a. For the memory locations in the table above, write MIPS code to sort the data from lowest to highest, placing the lowest value in the smallest memory location. Assume the base address of Array is stored in register $s6.

```
lw $t0, 0($s6)
lw $t1, 4($s6)
lw $t2, 16($s6)
sw $t2, 0($s6)
sw $t0, 4($s6)
lw $t0, 12($s6)
sw $t0, 16($s6)
sw $t1, 12($s6)
```

3 Translate function f into MIPS assembly language. If you need to use registers `$t0` through `$t7`, use the lower-numbered registers first. Assume the function declaration for `func` is "`int func(int a, int b);`". The code for function f is as follows:

```
int f(int a, int b, int c, int d){
   return func(func(a, b), c + d);
}
```

```
f: addi $sp,$sp,-12
sw $ra,8($sp)
sw $s1,4($sp)
sw $s0,0($sp)
move $s1,$a2
move $s0,$a3
jal func
move $a0,$v0
```

```
add $a1,$s0,$s1
jal func
lw $ra,8($sp)
lw $s1,4($sp)
lw $s0,0($sp)
addi $sp,$sp,12
jr $ra
```

4 Right before your function f from problem 3 returns, what do we know about contents of registers `$s3`, `$ra`, and `$sp`? Keep in mind that we know what the entire function `f` looks like, but for function `func` we only know its declaration.

Register $ra is equal to the return address in the caller function, registers $sp and $s3 have the same values they had when function f was called.