Com S 321 Homework 6 Due: Thursday, Oct. 12 by Midnight

1. Add comments to the following MIPS code and describe in one sentence what it computes. Assume that *$a0* is used for the input and initially contains *n*, a positive integer. Assume that *$v0* is used for the output.

```
begin:  addi    $t0,  $zero,  0
        addi    $t1,  $zero,  1
loop:   slt     $t2,  $a0,  $t1
        bne     $t2,  $zero,  finish
        add     $t0,  $t0,  $t1
        addi    $t1,  $t1,  2
        j       loop
finish: add     $v0,  $t0,  $zero
```

2. Write MIPS code for the following high-level language program fragment:

```
Loop:  g = g + A[i]; // Assume g is in $s1; i is in $s3; base address of A is in $s5
       i = i + j;       // Assume j is in register $s4; h is in register $s2
       if (i != h) go to Loop
```

3. Describe the contents of registers *$t0* and *$t2* when the following MIPS program reaches the label marked "done".

Register *$s0* contains the base address of an int array B of size 100. Assume that integer values have been inputted into the array already.

```
        lw      $t0,  0($s0)
        addi    $t1,  $s0,  40
        addi    $s0,  $s0,  4
loop:   beq     $s0,  $t1, done
        lw      $t2,  0($s0)
        blt     $t2,  $t0,  skip        #  If $t2 is less than $t0, go to skip
        addi    $t0,  $t2,  0
skip:   addi    $s0,  $s0,  4
        j       loop
done:   . . . .
```

4. The following program tries to copy words from the address in register *$a0* to the address in register *$a1*, counting the number of words copied in register *$v0*. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers *$v1, $a0,* and *$a1*. This terminating word should be copied but not counted.

```
Loop:  lw      $v1,  0($a0)        # Read next word from source
       addi    $v0,  $v0,  1       # Increment count of words copied
       sw      $v1,  0($a1)        # Write to destination
```

```
        addi    $a0,  $a0,  1           # Advance pointer to next source
        addi    $a1,  $a1,  1           # Advance pointer to next destination
        bne     $v1,  $zero, Loop       # Loop if word copied is ≠ 0
```

There are multiple bugs in this MIPS program. You should fix all the bugs and turn this program into a bug-free version.

5. Consider the following fragment of high-level language code:

```
for (i = 0; i <= 100; i = i + 1)
{
  a [i] = b[i] + c;
} // end for i
```

Assume that a and b are integer arrays (each element is stored in 4 bytes) and the base address of a is in *$a0* and the base address of b is in *$a1*. The loop variable i is in register *$t0* and the constant c is in register *$s0*.

(a) Write MIPS code for this high-level language program fragment.
(b) How many MIPS instructions are executed during the running of your code?
(c) How many memory data references will be made during execution of your code?

6. Consider the following **while** loop in a high-level language:

```
        while ( k == a [i] )    // k is in register $s5
        {                       // i is in register $s3
                i = i + j ;     // j is in register $s4
                                // base address of integer array a is in register $s6
        } // end while
Exit:  . . .
```

The MIPS assembly language program given below implements the above loop.

```
Loop:   add     $t1,  $s3,  $s3         # t1 ← i + i
        add     $t1,  $t1,  $t1         # t1 ← 4*i
        add     $t1,  $t1,  $s6         # t1 ← address of a [i]
        lw      $t0,  0 ($t1)           # t0 ← a [i]
        bne     $t0,  $s5,  Exit
        add     $s3,  $s3,  $s4         # i ← i + j
        j       Loop
Exit:   . . .
```

Assume that in the while loop, a [i + m*j] is equal to **k** for values of 0 <= m <= 9 and is not equal to k when m = 10. Therefore, 10 iterations of the loop are executed.

(a) How many assembly language instructions are executed in the MIPS program given above for 10 iterations of the while loop?

(b) Write a semantically equivalent MIPS assembly language program to reduce by more than half the total number of instructions executed for 10 iterations of the while loop.