Fall 2003 Name: _____

Computer Organization Test 2

Question 1. (18 points) Translate the following high-level language code segment to MIPS assembly language. Use the registers indicated in the code.

a) for $4 = 0 to 100 by steps of size 10 do
    if ($3 < $4) OR ($2 >= 50) then
        $2 = $2 + $3
    end if
  end for

```
for:
      li  $4, 0
for_compare:
      bgt  $4, 100, end_for
if:
      blt  $3, $4, then
      blt  $2, 50, end_if
then:
      add  $2, $2, $3
end_if:
      addi $4, $4, 10
      j  for_compare
end_for:
```

b) while ($8 > 20) do
    if ($8 > 100) AND ($8 < 200) then
        $7 = $8
        $8 = $8 - 10
    else
        $8 = $8 - $7
    end if
    $7 = $6 + 4
  end while

```
while:
      ble  $8, 20, end_while
if:
      ble  $8, 100, else
      bge  $8, 200, else
      move $7, $8
      sub  $8, $8, 10
      j end_if
else:
      sub  $8, $8, 7
end_if:

      addi  $7, $6, 4
      j  while
end_while:
```

Question 2. (12 points) Suppose you have the following .data area in MIPS assembly language:

```
            .data
array:      .word  10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
```

For each of the following assembly language segments, what value is loaded into register $t2?

a)	b)	c)

li $t0, 5
la $t1, array
mul $t3, $t0, 4
add $t1, $t1, $t3
lw $t2, 0($t1)
 	
answer: 15
	la $t1, array
	lw $t2, 8($t1)
	
	answer: 12
		li $t0, 5
		la $t1, array
		sll $t3, $t0, 2 # shift left logical
		add $t1, $t1, $t3
		lw $t2, 0($t1)
		answer: 15

Question 3. (8 points) For the .data area in question 2, complete the translation of the following high-level code segment to MIPS assembly language.

```
for i = 0  to 14 do
    array[i+1] = array[i]
end for
```

```
        la   $t1, array
for:
        li   $t0, 0
        li   $t2, 14
for_compare:
        bgt  $t0, $t2, end_for

        mul  $t3, $t0, 4
        add  $t3, $t3, $t1
        lw   $t4, 0($t3)
        sw   $t4, 4($t3)



        j  for_compare
end_for:
```

Question 4. (8 points) Assume that a two-dimensional array A is stored in memory using column-major order, i.e., column 0 is followed by column 1, etc. Write a formula to calculate the address of element A[i][j].

addr. of A[i][j] = base addr. of A + [j * (# rows) + i] * (element size)

Question 5. Consider the following selection sort subprogram that utilizes a function Max to search for the largest element in the unsorted part of the array.

```
procedure selectionSort(numbers - array of integers, count - integer)
    local integer variables: lastUnsortedIndex, maxIndex, temp

    for lastUnsortedIndex = (count-1) downto 1 do
        maxIndex = Max(numbers, 0, lastUnsortedIndex)
        temp = numbers[lastUnsortedIndex]
        numbers[lastUnsortedIndex] = numbers[maxIndex]
        numbers[maxIndex] = temp
    end for
end selectionSort
```

a) (6 points) Using the MIPS register conventions ($a0-$a3, $t0-$t9, $s0-$s7, $v0-$v1, $sp, $ra), what registers would be used to pass each of the following parameters to selectionSort:

base address of "numbers" array	count
$a0	$a1

b) (6 points) Using the MIPS register conventions, which of these parameters ("numbers", "count", or both of them) should be moved into $s-registers?

numbers should get moved into $s0, but count is just used to initialize lastUnsortedIndex

c) (6 points) Using the MIPS register conventions, what registers should be used for each of the local variables:

lastUnsortedIndex	maxIndex	temp
$s1	$v0 (is best, but $t0 is okay)	$t1

d) (4 points) In addition to the above registers, the value of "numbers[maxIndex]" will need to be stored into a register. Using the MIPS register conventions, what register should be used to hold this value?

some $t register like $t2

e) (6 points) For the registers indicated above, write the MIPS instructions to set up the selectionSort call-frame on the run-time stack (i.e., move the stack pointer and save registers, etc.).

selectionSort:

```
sub  $sp, $sp, 12
sw   $ra, 4($sp)
sw   $s0, 8($sp)
sw   $s1, 12($sp)
```

f) (8 points) For the registers indicated above, write the assemble language code to call the Max function ("maxIndex = Max(numbers, 0, lastUnsortedIndex)"). Include the MIPS instructions to setup the parameters to Max and assigning "maxIndex" the value returned. (You do not need to write the Max function code just the code to call it)

```
move   $a0, $s0
li     $a1, 0
move   $a2, $s1
jal  Max
```

g) (8 points) Using the registers you indicated, write the MIPS assembly language statements to perform the statements:

```
temp = numbers[lastUnsortedIndex]
numbers[lastUnsortedIndex] = numbers[maxIndex]
numbers[maxIndex] = temp
```

```
mul   $t3, $s1, 4
add   $t3, $t3, $s0
lw    $t1, 0($t3)
mul   $t4, $v0, 4
add   $t4, $t4, $s0
lw   $t2, 0($t4)
sw   $t2, 0($t3)
sw   $t1, 0($t4)
```