CS232 Midterm Exam 1
February 18, 2002

Name: _____

- This exam has 6 pages, including this cover and the cheat sheet on the next page.
- There are three questions worth a total of 100 points.
- You have only 50 minutes, so budget your time!
- No written references or calculators are allowed.
- To make sure you receive full credit, please write clearly and show your work.
- We will not answer questions regarding course material.

Question	Maximum	Your Score
1	30	
2	40	
3	30	
Total	100	

MIPS Instructions

These are some of the most common MIPS instructions and pseudo-instructions, and should be all you need. However, you are free to use *any* valid MIPS instructions or pseudo-instruction in your programs.

Category	Example	Meaning
Arithmetic	add $t0, $t1, $t2	$t0 = $t1 + $t2
	sub $t0, $t1, $t2	$t0 = $t1 - $t2
	addi $t0, $t1, 100	$t0 = $t1 + 100
	mul $t0, $t1, $t2	$t0 = $t1 * $t2
	move $t0, $t1	$t0 = $t1
	li $t0, 100	$t0 = 100
Data Transfer	lw $t0, 100($t1)	$t0 = Mem[100 + $t1]
	sw $t0, 100($t1)	Mem[100 + $t1] = $t1
Branch	beq $t0, $t1, Label	if ($t0 == $t1) go to Label
	bne $t0, $t1, Label	if ($t0 != $t1) go to Label
	bge $t0, $t1, Label	if ($t0 >= $t1) go to Label
	bgt $t0, $t1, Label	if ($t0 > $t1) go to Label
	ble $t0, $t1, Label	if ($t0 <= $t1) go to Label
	blt $t0, $t1, Label	if ($t0 < $t1) go to Label
Set	slt $t0, $t1, $t2	if ($t1 < $t2) then $t0 = 1; else $t0 = 0
	slti $t0, $t1, 100	if ($t1 < 100) then $t0 = 1; else $t0 = 0
Jump	j Label	go to Label
	jr $ra	go to address in $ra
	jal Label	$ra = PC + 4; go to Label

The second source operand of *sub*, *mul*, and all the branch instructions may be a constant.

Register Conventions

The *caller* is responsible for saving any of the following registers that it needs, before invoking a function:

$t0-$t9 $a0-$a3 $v0-$v1

The *callee* is responsible for saving and restoring any of the following registers that it uses:

$s0-$s7 $ra

Performance

Formula for computing the CPU time of a program P running on a machine X:

$$\text{CPU time}_{X,P} = \text{Number of instructions executed}_P \times \text{CPI}_{X,P} \times \text{Clock cycle time}_X$$

CPI is the average number of clock cycles per instruction, or:

CPI = Number of cycles needed / Number of instructions executed

Question 1: Understanding MIPS programs (30 points)

```
Ostrich:
      bge   $a1, $a2, Duck
      mul   $t1, $a1, 4
      add   $t1, $a0, $t1
      mul   $t2, $a2, 4
      add   $t2, $a0, $t2
      lw    $t3, 0($t1)
      lw    $t4, 0($t2)
      sw    $t3, 0($t2)
      sw    $t4, 0($t1)
      addi  $a1, $a1, 1
      sub   $a2, $a2, 1
      j     Ostrich
Duck:
      jr    $ra
```

Part (a)
Translate *Ostrich* into a high-level language like C or Java. Be sure to describe the number and types of any arguments and return values. We will not deduct points for syntax errors *unless* they are significant enough to alter the meaning of your code. (25 points)

Part (b)
Describe briefly, in English, what this function does. (5 points)

Question 2: Performance (40 points)

Let's study the performance of this function. Assume that we have a 500MHz processor with a clock cycle time of 2ns. The table below shows CPIs for some various classes of MIPS instructions. (You can assume that pseudoinstructions are also accounted for in this table.)

Instruction class	CPI
mul	5
branches and jumps	3
data transfers	3
all others	1

Part (a)
What is the CPU time of this function in nanoseconds, assuming that the loop is executed 100 times? In other words, the "bge" test fails 100 times and then succeeds on the 101st test. (10 points)

```
Ostrich:
      bge   $a1, $a2, Duck
      mul   $t1, $a1, 4
      add   $t1, $a0, $t1
      mul   $t2, $a2, 4
      add   $t2, $a0, $t2
      lw    $t3, 0($t1)
      lw    $t4, 0($t2)
      sw    $t3, 0($t2)
      sw    $t4, 0($t1)
      addi  $a1, $a1, 1
      sub   $a2, $a2, 1
      j     Ostrich
Duck:
      jr    $ra
```

Question 2 continued

Part (b)
Our code is somewhat slow, since expensive multiplication operations are executed repeatedly. Rewrite the function to eliminate both multiplications from the main body of the loop. (20 points)

```
Ostrich:
      bge   $a1, $a2, Duck
      mul   $t1, $a1, 4
      add   $t1, $a0, $t1
      mul   $t2, $a2, 4
      add   $t2, $a0, $t2
      lw    $t3, 0($t1)
      lw    $t4, 0($t2)
      sw    $t3, 0($t2)
      sw    $t4, 0($t1)
      addi  $a1, $a1, 1
      sub   $a2, $a2, 1
      j     Ostrich
Duck:
      jr    $ra
```

Part (c)
What is the CPU time in nanoseconds for 100 loop iterations of your revised function? (10 points)

Instruction class	CPI
mul	5
branches and jumps	3
data transfers	3
all others	1

Question 3: Writing a nested function (30 points)

Show how to translate the pseudocode below for *SelectionSort* into a MIPS assembly language function. This algorithm first finds the largest element in A[0]..A[n] and moves it to position n, then finds the largest element in A[0]..A[n-1] and puts that in position n-1, and so forth.

- You will not be graded on the efficiency of your code, but you *must* follow all MIPS conventions.
- Assume that you already have a MIPS function *MaxIndex*, which takes two arguments A and n, and returns the index of the largest element of A[0]..A[n]. The arguments and return values are passed in registers $a0, $a1 and $v0 respectively.
- Remember that integers are 32-bit, or 4-byte, MIPS quantities.

```
SelectionSort(A, n)
    for i = n downto 1
        m = MaxIndex(A, i)              // Find the largest element in A[0]..A[i]
        exchange A[i] and A[m]          // Move it to position i
```