

CS232 Midterm Exam 1

February 19, 2001

Name: _____

- This exam has 6 pages, including this cover and the cheat sheet on the next page.
- There are four questions, each worth 25 points.
- You have 50 minutes; 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	25	
2	25	
3	25	
4	25	
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] = \$t0
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:

$$\text{CPI} = \text{Number of cycles needed} / \text{Number of instructions executed}$$

Question 1: Performance

Homework 1 presented two different MIPS code fragments for adding 64-element vectors:

Program 1:

```
# a[0] = b[0] + c[0]
lw    $t3, 0($t1)
lw    $t4, 0($t2)
add   $t4, $t3, $t4
sw    $t4, 0($t0)

# a[1] = b[1] + c[1]
lw    $t3, 4($t1)
lw    $t4, 4($t2)
add   $t4, $t3, $t4
sw    $t4, 4($t0)

# ...Repeat for elements
#    2 through 62...

# a[63] = b[63] + c[63]
lw    $t3, 252($t1)
lw    $t4, 252($t2)
add   $t4, $t3, $t4
sw    $t4, 252($t0)
```

Program 2:

```
add   $t4, $0, $0
Loop: add   $t5, $t4, $t1
      lw    $t6, 0($t5)
      add   $t5, $t4, $t2
      lw    $t7, 0($t5)
      add   $t6, $t6, $t7
      add   $t5, $t4, $t0
      sw    $t6, 0($t5)
      addi  $t4, $t4, 4
      slti  $t5, $t4, 256
      bne  $t5, $0, Loop
```

Assume that data transfers require 3 clock cycles, and all other instructions need just 1 clock cycle. Also, assume the clock cycle time is 2ns.

Part (a)

What is the CPI for each program? You may leave your answers as fractions. (12 points)

Part (b)

How much CPU time is needed to execute each program? This time, give an exact answer. (13 points)

Question 2: Finding and fixing MIPS logical errors

Little Howie is having a bad day. He has a C function which returns the smallest integer in an array V of n elements. However, when Little Howie translated the function into MIPS assembly language, he made several logical (not syntax) errors. Help Little Howie by *finding and fixing at least four distinct errors in his function*. Both the correct C code and the buggy MIPS code are shown below. Indicate the bugs and fixes directly on the assembly program. (6 points each)

```
int minimum(int V[], int n) // Find smallest integer in an array V with n elements
{
    int min, i;
    min = V[0];              // min is initialized with the first element of V
    for (i=1; i<n; i++)
        if (V[i] < min)      // Found an element smaller than the current min
            min = V[i];
    return min;
}
```

```
minimum:                                # $a0 contains "V"

    addi    $sp, $sp, -4

    sw      $t0, 0($sp)

    lw      $s0, 0($a0)                # $s0 is "min"

    li      $s1, 1                    # $s1 is "i"

loop:

    bgt     $s1, $a1, exit

    add     $t0, $s1, $a0

    lw      $t0, 0($t0)                # Load element of array

    bgt     $t0, $s0, next

    move    $s0, $t0                  # New "min" value

next:

    addi    $s1, $s1, 1

    j       loop

exit:

    addi    $sp, $sp, 4

    lw      $t0, 0($sp)

    jr      $ra
```

Question 3: Translating MIPS code

Here is a mystery function. It expects one non-negative integer argument and returns one integer result.

```
yuck:
    li    $t0, 1
    li    $t1, 1
loop:
    blt    $a0, 2, exit
    add    $t2, $t0, $t1
    move   $t0, $t1
    move   $t1, $t2
    addi   $a0, $a0, -1
    j      loop
exit:
    move   $v0, $t1
    jr     $ra
```

Part (a)

Translate the *yuck* function into C. Do not use “goto.” We will not deduct points for syntax errors *unless* they are significant enough to alter the meaning of your code. (20 points)

Part (b)

What will this function return if it is called with an argument (\$a0) of 4? (3 points)

Part (c)

Describe, in English, what this function computes. (2 points)

Question 4: Writing a nested function

Here is a C function *primes* which takes an integer argument *n* and prints the first *n* prime numbers. Translate it to MIPS assembly. (25 points)

- Assume that you already have a MIPS function *isPrime*, which takes an integer argument (in \$a0) and returns (in \$v0) 1 if the argument is prime, or 0 if the argument is not prime.
- To print an integer, you can put it in \$a0, load \$v0 with 1, and do a “syscall.”
- You will *not* be graded on the efficiency of your code, but you *should* follow all MIPS conventions.

```
void primes(int n)
{
    int candidate = 1;
    while (n >= 1) {
        do {
            candidate++;
        } while (isPrime(candidate) == 0);
        printf("%d", candidate);
        n--;
    }
}
```