Name ________________________________

Computer Architecture

EE 4720

Midterm Examination

Monday, 24 October 2005, 12:40–13:30 CDT

Problem 1	________	(50 pts)
Problem 2	________	(50 pts)
Exam Total	________	(100 pts)

Alias ________________________________

Good Luck!

Problem 1: The partially completed routine on the next page is called with the address of a string which may contain characters that look like letters, for example, “`g0!1y`.” The routine should replace those characters with the letters they look like; the example would be converted to “`golly`.”

The string DMAP (see the next page) specifies the look-alikes. It specifies a letter (always lower case) followed by one or more look-alike characters followed by a comma (possibly followed by another group). For example, “`o0,l1!,`” indicates that the letter oh’s look-alike is the digit zero and the letter el’s look-alikes are the digit 1 and an exclamation point.

The routine on the next page is almost finished. The part at the end uses a look-up table to translate the string, and the part at the beginning has started setting up the look-up table. Write the code that finishes setting up the look-up table based on DMAP. [50 pts]

□ Write the code that sets up the look-up table.

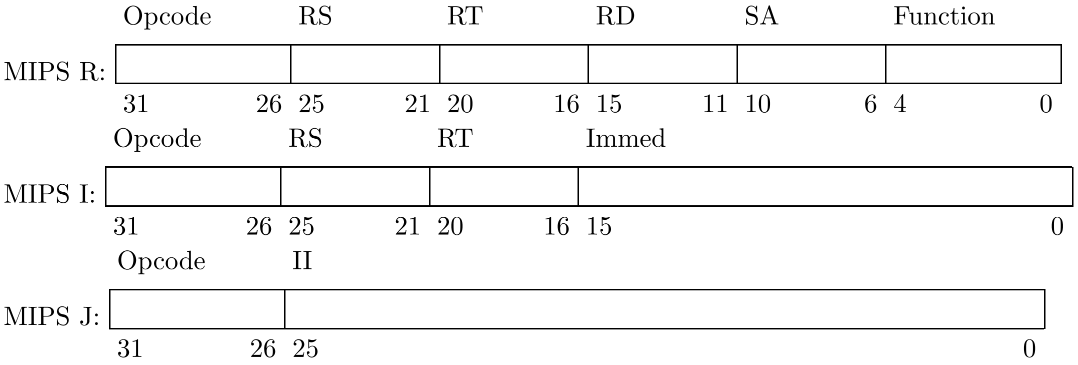
□ The code must be reasonably efficient.

□ The only synthetic instructions that can be used are `nop` and `la`.

Use the next page for the solution.

```
        ## Register Usage
        #
        # $a0: Procedure call argument. Address of string to translate.
        # There are no return values.

DMAP:   .asciiz "o0,l1!,c([,t+,s$,"  # Translate 0->o, 1->l, !->l, (->c, etc.

LUT:    .space 256

        la $t0, LUT
        addi $t1, $0, 255

        # First, initialize look-up table so every character is mapped to itself.
        #
LOOP0:  add $t3, $t0, $t1         # Compute address of LUT entry.
        sb $t1, 0($t3)            # Write default entry. (A->A, B->B, etc.)
        bne $t1, $0  LOOP0
        addi $t1, $t1, -1

        # Next, set up look-up table based on DMAP string. (Finish in solution.)
        #
        la $t0, DMAP
        la $t1, LUT
        addi $t5, $0, 44   # ’,’  Comma character separates groups.

        # Start solution here. (Can be done in 9 insn.)




        # Use the look-up table to translate the string.
        la $t1, LUT
LOOP3:  lb $t2, 0($a0)       # Load character of string.
        beq $t2, $0, EXIT2
        add $t2, $t2, $t1    # Compute address of look-up table entry.
        lb $t2, 0($t2)       # Load translated character.
        sb $t2, 0($a0)       # Write translated character to string.
        j LOOP3
        addi $a0, $a0, 1

EXIT2:  jr $ra
        nop
```

Problem 2: Answer each question below. The instruction format descriptions are provided for reference.



(*a*) A proposed new MIPS branch instruction compares a register value to a constant to determine if the branch should be taken. For example, the branch below is taken if the contents of `t1` is 123. [10 pts]

```
beqi $t1, 123,  TARGET
nop
```

□ Why isn't it feasible to code such an instruction using any of the existing MIPS formats?

□ How could a conditional control transfer instruction that compared a register to an immediate be coded using the MIPS formats? (The instruction would be different than `beqi`.)

□ Show the instruction in assembly language.

(*b*) Show the results of addition for the 32-bit data types shown below. [10 pts]

□ Unsigned Integer:

```
  0x0999
+ 0x0109
```

□ BCD:

```
  0x0999
+ 0x0109
```

□ Packed 4-bit integers with saturating arithmetic.:

```
  0x0999
+ 0x0109
```

(*c*) The MIPS ISA specifies that the sa (shift amount) field in the `add` instruction must be zero (an `add` instruction with a non-zero sa field value should cause an execution error). [10 pts]

□ Describe a difficulty that might have arisen if the MIPS ISA had specified that implementations should ignore the sa field in an `add` instruction (and so the `sa` field could contain any value).

□ Describe a difficulty that might have arisen if the MIPS ISA said nothing about the `sa` field in an add instruction.

□ Describe a difficulty that might have arisen if the MIPS ISA did specify that sa must be zero but did not say what should happen if the field were non-zero.

(*d*) Consider a benchmark suite that's similar to SPECcpu in that the tester is responsible for compiling the programs but that unlike SPEC, specifies that the tester should not use any optimizations when compiling the benchmarks. [10 pts]

□ Compared to SPECcpu base and peak (result) scores, how useful would such results be? Explain.

(*e*) Using a new compiler optimization a program's dynamic instruction count is cut in half. The values for CPI, IC, and ϕ are available for a run of the program compiled without the new optimization. Other than the instruction count, nothing has been measured for the program using the new optimization. How useful is the CPU performance equation for estimating the run time with the new optimization on the same system in this situation? Explain. [10 pts]