Name ______________________________

Computer Architecture
EE 4720
Midterm Examination

Wednesday, 22 October 2003, 10:40–11:30 CDT

Problem 1	______	(25 pts)
Problem 2	______	(15 pts)
Problem 3	______	(18 pts)
Problem 4	______	(18 pts)
Problem 5	______	(24 pts)
Exam Total	______	(100 pts)

Alias ______________________________

Good Luck!

Problem 1: The routine below is called with an unsigned integer in register `$a0` and the address of some allocated memory in register `$a1`. When it returns the memory at should `$a1` contain the hexadecimal representation of `$a0` as a null-terminated (C format) string. Complete the routine, follow the guidelines in the comments. (For **partial credit** write a routine that converts a string holding a hexadecimal number to an integer.) [25 pts]

```
################################################################################
# utoh: Convert unsigned integer to hexadecimal string.
#
        # $a0: Call Value: Unsigned integer to convert.
        # $a1: Call Value: Address of allocated memory.
        #      Write converted string to this address, assume there is enough.
        #      Sample strings: "1F3", "1", "0" written at $a1.

        # [ ] String should not have leading zeros. (Good: "123", Bad "00123".)
        # [ ] Fill as many delay slots as possible.
        # [ ] Registers $a0-$a3 and $t0-$t7 can be modified.

        # The ASCII value of '0' is 48,  the ASCII value of 'A' is 65

# Put solution here.
utoh:
































        jr $ra
        nop
```

Problem 2: Code similar to the histogram program presented in class appears below. MIPS instruction formats are shown below for reference. [15 pts]

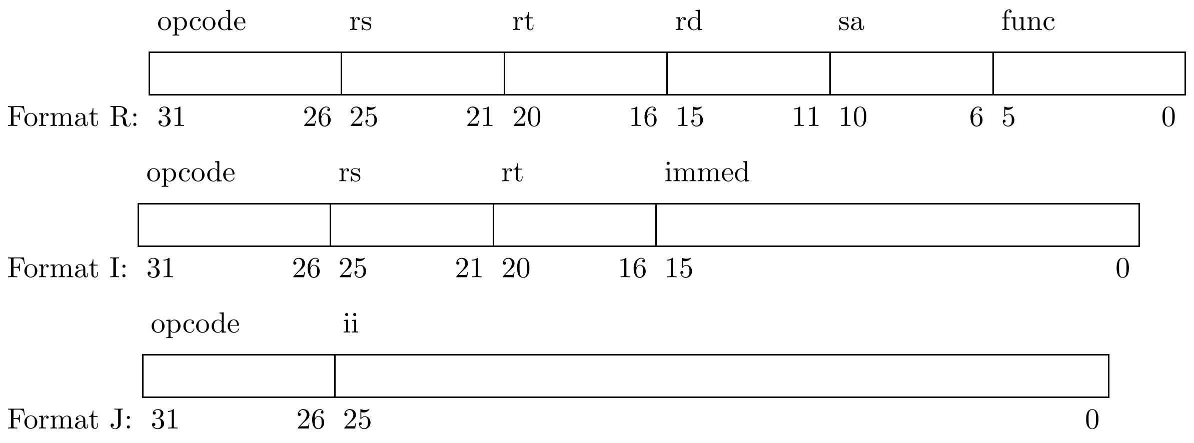
(*a*) Design three new MIPS instructions to reduce the size of the program fragment below

- Each new instruction should replace at least two related instructions in the program below. (Do not combine two *unrelated* instructions, such as `j` and `sw`.)

☐ Show the coding for the new instruction, making up the opcode and other field values as necessary. The coding should use one of MIPS' existing formats and should fit as naturally as possible.

- Do not worry whether the instruction is appropriate for a RISC ISA.

```
 addi $t7, $0, 26
LOOP:
 lbu $t1, 0($t0)
 addi $t0, $t0, 1
 beq $t1, $0, DONE
 addi $t1, $t1, -65
 sltu $t2, $t1, $t7
 beq $t2, $0, LOOP
 sll $t1, $t1, 2
 add $t3, $a1, $t1
 lw $t4, 0($t3)
 addi $t4, $t4, 1
 j LOOP
 sw $t4, 0($t3)
```


opcode rs rt rd sa func
Format R: 31 26 25 21 20 16 15 11 10 6 5 0
opcode rs rt immed
Format I: 31 26 25 21 20 16 15 0
opcode ii
Format J: 31 26 25 0

(*b*) Show an instruction that can be created by combining several instructions from the program but that would be impossible to code. Explain why it would be impossible to code.

Problem 3: Answer each question below.

(*a*) In MIPS (and similar ISAs) there is a `lb`, `lbu` (load byte unsigned), and a `sb` but there is no `sbu` (store byte unsigned). Why not? [6 pts]

(*b*) Explain why the MIPS instruction below won't work: [6 pts]

```
add.d $f1, $f2, $f3
```

(*c*) Show the result of each add instruction below. The instructions execute on a machine with 32-bit registers that is capable of BCD, and packed integer, and ordinary integer arithmetic. The packed integer operations all use saturating unsigned arithmetic. [6 pts]

```
# r1 = 0x8080888
# r2 = 0x1020999


# Ordinary Integer Add
add r3, r1, r2           r3 =

# BCD Add
add.bcd r3, r1, r2       r3 =

# Packed Integer (4 bits per int.)
add.p4 r3, r1, r2        r3 =

# Packed Integer (8 bits per int.)
add.p8 r3, r1, r2        r3 =
```

Problem 4: Answer each question below.

(*a*) A company compiles and runs the SPECint2000 benchmarks on its new system, complying with all rules except one: it refuses to divulge the steps it used to compile the programs. Nevermind that it's against the rules, is it in the company's interest to keep this information secret? Explain. [6 pts]

(*b*) A program is compiled two ways, one for ISA A, implementation x, the other also for ISA A, but for implementation y. Explain what would be common to the two executables and what would be different. Provide an example. [6 pts]

(*c*) Explain the dead-code elimination (DCE) optimization using an example. [6 pts]

Problem 5: Answer each question below.

(*a*) CISC ISAs have variable length instructions. What advantages over RISC ISAs does that enable? Name at least two and briefly explain each advantage. [6 pts]

(*b*) RISC ISAs have fixed length instructions. What advantages over CISC ISAs does that enable? Name at least one and briefly explain each advantage. [6 pts]

(*c*) Why are programs written in a stack ISA small? Be brief but as specific as possible. [6 pts]

(*d*) Listed below are several behaviors or features. For each, explain whether it is usually an ISA feature, an implementation feature, or an ABI (application binary interface) feature. **Briefly explain why.** If it can fit into more than one category (for example, ISA or implementation) say so and explain why. [6 pts]

☐ An `add` instruction raises an exception on an overflow.

☐ The unused field in an instruction must be set to zero.

☐ Procedure call saves return address in a particular register.

☐ Multiply instruction takes four cycles.