Professor: Hong Zhang

CMPUT 229 Computer Organization and Architecture I A3
2000-2001
Final Examination
(December 15, 2000)

Name: ______________ SID: ______________

Do all problems. Closed book and notes. No calculator. Instruction set sheet is provided.

Problem 1 (5 marks)

If $(34)_5$ is a base-5 number, what is it in decimal and binary, respectively? In decimal, what is the largest unsigned 4-digit base-5 number equal to?

Problem 2 (12 marks)

Assuming the data segment begins at 0x10000000, show the effect of the following assembler directives on memory by completing the memory map on the right. Show in **hexadecimal** both the addresses and contents of the memory locations affected, one word per line. When memory contents are unknown, use question marks (?'s). (The ASCII code for 'A' is 0x41.)

```
.data
.word 11, -5
.byte 10, 20
.align 2
.space 8
.ascii "ABCD"
.float 1.0
```

Address	Contents
10000000	0000000B

Problem 3 (8 marks)

`0x20a3001e` is a valid machine code of a MIPS assembly instruction. Determine what is this instruction.

Problem 4 (6 marks)

What is the range of a branch instruction such as **`beq`** in MIPS? Why is there a limit on where the destination address can be, relative to a branch instruction? If PC is currently at `0x4000`, and the destination address is PC=`0x20000`, can this be accomplished with a branch instruction? Explain why or why not.

Problem 5 (5 marks)

If the CPU is driven by a 500 MHz clock, and if its average CPI is 2.0, how much time will it take on average to execute a program of 400 instructions on this machine?

Problem 6 (6 marks)

If a "memory-add" or `madd` instruction is to be introduced as a new MIPS pseudo-instruction, where "`madd 8($a1), 0($a2), 4($a3)`" would perform "`mem[$a1+8] <- mem[$a2] + mem[$a3+4]`", show how it can be implemented with a minimal number of real MIPS instructions.

Problem 7 (8 marks)

A C function is given below, together with its partial implementation in MIPS assembler. Notice that local variables, `i` and `j`, reside on the stack. Assume the input parameters `a` and `b` are passed through `$a0` and `$a1`. Complete the MIPS implementation. Whenever a local variable is referenced, you **must** use the stack, i.e., you cannot use registers to implement the two local variables.

```
int foo(int a, int *b)
{
      int i, j;

      i = a + *b;
      j = a - *b;
      return (i>j)? 0:1;
}
```

```
foo:    addi  $sp, $sp, -12
        sw    $ra, 8($sp)
        # put your answer from here on




        lw    $ra, 8($sp)
        addi  $sp, $sp, 12
        jr    $ra
```

Problem 8 (8 marks)

In IEEE single-precision representation, exactly how many normalized numbers can be encoded? (Infinities, NaN's, and 0 are not considered normalized numbers.) Explain your answer. Encode $(0.1)_{10}$ in IEEE single-precision representation. Use rounding rather than truncation if it cannot be represented exactly. Show your result in `hexadecimal`.

Problem 9 (5 marks)

Interrupt-driven strategy is a more efficient way of handling I/O requests than polled I/O, if I/O devices are much slower than the CPU, a property that holds true for almost every I/O device. What would happen if the I/O devices could keep pace with the CPU? Why could a interrupt-driven strategy be less efficient than polling in that case?

Problem 10 (12 marks)

Suppose the following memory references are byte references. Assume a cache with 8 one-word (4 bytes each) entries that are initially empty. For simplicity, assume a 128-byte memory.

```
0, 1, 2, 3, 2, 3, 4, 5, 32, 33, 34, 35, 63, 62, 61, 7
```

1. With a direct-mapped cache organization, show the hits and misses, and the final cache contents including the tag field. Calculate the hit rate.

```
        0, 1, 2, 3, 2, 3, 4, 5, 32, 33, 34, 35, 63, 62, 61, 7
H/M:
```

2. If instead we use a 2-way set associative organization, show the hits and misses, the hit rate, and the final contents of the cache including the tag field.

```
        0, 1, 2, 3, 2, 3, 4, 5, 32, 33, 34, 35, 63, 62, 61, 7
H/M:
```