CS 352 Computer Architecture

Midterm #1 – Prof. Mark Feb 17, 2005

Name: CHRIS'S ANSWER KEY & BILL'S (please print)

1.	15 points	_____
2.	25 points	_____
3.	15 points	_____
4.	20 points	_____
5.	25 points	_____
	TOTAL (100 pts)	_____

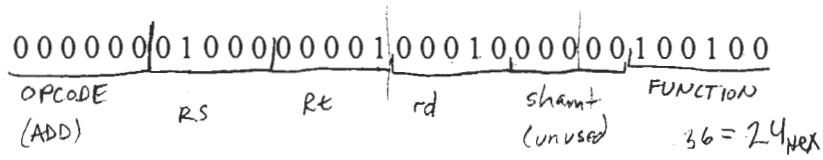
In recognition of University regulations regarding academic dishonesty, I certify that I have neither given nor received unpermitted aid on this examination. This exam is open notes (yours and lectures notes), open textbook. Calculators may be used, but only in non-programmable mode. No other materials may be used.

Signed: _____

General Instructions:
If there is an ambiguity in a problem, please either ask the instructor for clarification or clearly state a reasonable assumption that resolves the ambiguity, and proceed with the problem.

Problem #1 – Instruction and data representation [15 points]

Given the following bit pattern (written in the same order as the book does, MSB first):



What does it represent on a MIPS computer, assuming that it is a MIPS instruction?

R format

AND $2, $8, $1

AND $V0, $t0, $at

WRONG REGISTER ORDER -3, OR -2
WRONG OPCODE -5
WRONG REGISTERS -4
WRONG -9

What does it represent on a MIPS computer, assuming that it is a 32-bit integer?

$4 + 32 + 2^{12} + 2^{16} + 2^{24} =$

16,777,216
65,536
4096
32
4

16,846,884

MISSED A BIT, ERROR -2
WRONG -6

Problem #2 – Find and fix bugs in MIPS assembly code [25 points]

Consider the following fragment of C-like (or Java-like) code:

```
int x[1000];
int maxval ;

...

for (i=maxval; i>=0; i--) {
    x[i] = i;
}
```

When a buggy compiler was used to compile this C code to MIPS assembly code, it produced the following output. Assume that $r2 contains the variable 'maxval' on entry to this code fragment. Also assume that the program must work regardless of where the program and variables are stored in the MIPS address space.

Blackboard clarification → ori

```
        ...
        ; $r2 holds maxval
        ori $r9, $r9, #x        ; load base address for array 'x'
        addi $r4, $r2, #0       ; i = maxval
loop:   bltz $r4, done
        addu $r5, $r9, $r4      ; compute address of x[i]
        sw $r4, ($r5)           ; x[i] = i
        addi $r4, $r4, #-1      ; i--
        j loop
done:   ..
```

should be lui of high 16 bits followed by ori of low 16 bits

must multiply this by 4 to get byte offset into array

This code is valid MIPS assembly code, but it does not have the same behavior as the C/Java program, because the compiler made two simple mistakes. Describe what these mistakes are, and how to fix them. For the fix, specify what assembly code you would remove, and what the new assembly code would be.

Bug #1
Problem:

~~20~~ 25 for treating multiply by 4 as two separate bugs, (with good solutions for each)

Solution:

Bug #2
Problem:

13/25 for just multiply by 4 as one bug or just ori as one bug

Solution:

-4 for add not mult fix

23/25 for both bugs but only partial fix of mult by 4

Problem #3 – Execution Time [15 points]

Program A takes 9 seconds to run on a 2 GHz machine. Then, the compiler is changed so that it replaces all instances of multiplying a value by 4 (mult X, X, 4) with a pair of adds (add X, X; add X, X). We'll call this optimized program "Program B". The CPI of a multiply instruction is 4 and the CPI of an add is 1. Program B runs in 8 seconds on this same 2 GHz machine. How many multiplies were replaced by adds? Show your work.

A: $9 \text{ seconds} \cdot \frac{2 \times 10^9 \text{ cycles}}{\text{second}} = 1.8 \times 10^{10} \text{ cycles}$ 3 POINTS

B: $8 \text{ seconds} \cdot \frac{2 \times 10^9 \text{ cycles}}{\text{second}} = 1.6 \times 10^{10} \text{ cycles}$ 3 POINTS

DIFFERENCE IN CYCLES = 2×10^9 cycles 3 POINTS

EACH MULT → 2 ADDS REDUCES # cycles by 2, so 3 POINTS

$\frac{2 \times 10^9}{2} =$ 1×10^9 MULT → ADD, ADD 3 POINTS

Problem #4 – Compute speedup for change in division operation [20 points]

Your company is designing a new machine. The workload for your machine consists of two programs, A and B, with the following instruction usage statistics:

	Prog. A	**Prog. B**
Load/Store	22%	9%
Arithmetic	68%	66%
Branch	10%	25%

You have two competing design teams. Team #1 has a design which runs at 4 GHz, with the following CPI's:

Load/Store	3.0
Arithmetic	1.8
Branch	2.1

Team #2 has a design which runs at 2.5 GHz, with the following CPI's:

Load/Store	2.0
Arithmetic	1.5
Branch	1.2

a) For Program A, which team has designed the faster machine? By how much (give speedup)?

prog. A

TEAM 1 $(3.0 \text{ cycles/Ins.} \cdot .22) + (1.8 \cdot .68) + (2.1 \cdot 0.1) \Rightarrow 2.094$ CPI 2

$\frac{4\times10^9 \text{ cycles/second}}{2.094 \text{ cycles/instruction}} \approx 1.91\times10^9 \text{ instructions/second}$ 3 2

TEAM 2 $(2.0 \cdot .22) + (1.5 \cdot .68) + (2 \cdot .1) \Rightarrow 1.58$ CPI 2

$\frac{2.5\times10^9 \text{ cy/sec}}{1.58 \text{ CPI}} \approx 1.58\times10^9$ inst/sec 2

SPEEDUP OF $\frac{1.91}{1.58} \approx 1.21 = 21\%$ 2

b) If Program A consists of 500 instructions and Program B consists of 1000 instructions, and the user runs programs A and B equally often, which team has built the faster machine for this user? By how much (give speedup)?

1 UNIT OF WORK = A + B

TEAM 1: A: $\frac{500 \text{ instructions}}{1.91\times10^9 \text{ inst/sec}} = 2.61\times10^{-7}$ sec

B: $(.09 \cdot 2.0) + (1.8 \cdot .66) + (2.1 \cdot 2.5) = 1.98$ CPI

$\frac{4\text{GHz}}{1.98 \text{ CPI}} = 2.02\times10^9$ inst/sec

$\frac{1000}{2.02\times10^9} = 4.95\times10^{-7}$ sec

TOTAL $= 7.56\times10^{-7}$ sec 3 2+2

TEAM 2: A $\frac{500}{1.58\times10^9} = 3.16\times10^{-7}$ sec

B $(2.0 \cdot 0.09) + (1.5 \cdot .66) + (1.2 \cdot .25) = 1.47$

$\frac{2.5\times10^9}{1.47} = 1.7\times10^9$ inst/sec

$\frac{1000}{1.7\times10^9} = 5.88\times10^{-7}$ sec

TOTAL 9.04×10^{-7} sec 3 2+2

TEAM 1 FASTER,

1.20 SPEEDUP

1.196 2 4

Problem #5 – ISA modification [25 points]

Suppose that we decide that we want to modify the MIPS architecture to have access to 64 registers, but we keep the instruction size fixed at 32 bits. There aren't enough instruction bits to do this properly, so we're going to use a trick: the extra 32 registers that we've added will only be accessible by LW, SW, and a new MOV instruction. (This is actually more useful than it sounds – real architects have done similar things).

More precisely, for LW and SW the extra 32 registers are only accessible for the data register used by the instruction, and not by the address register. For the new MOV instruction, all 64 registers are accessible by both the source and destination.

All other MIPS instructions must continue to use their current opcodes and formats.

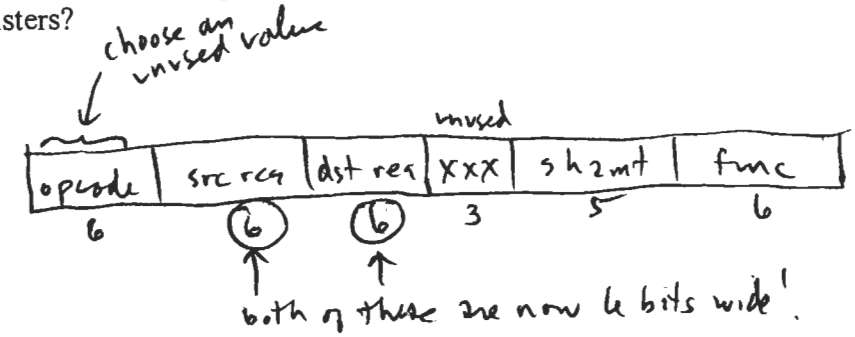
5

a) How many bits are needed to represent a register operand when all 64 registers can be accessed by the operand?

$\log_2(64) = \boxed{6}$

10

b) Show one possible instruction format for the new MOV instruction. Make a figure like the one in the lower-left hand corner of the green page at the front of the book. What compromise(s), if any, are made to support the additional registers?

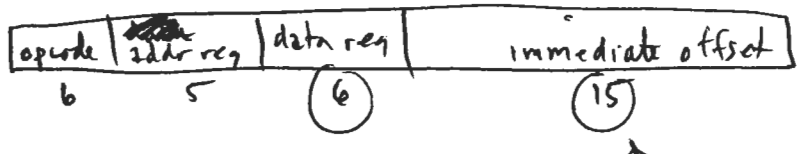


modified R-format
-2 for all 3

Compromise: register operands are no longer consistent across all formats (or "none" ok too)

10

c) Show one possible instruction format for the new LW and SW instructions. What compromise(s), if any, are made to support the additional registers?



modified I-format
-8 for major mistake
-2 for specifying both registers as 6 bits
-6 for not cutting into immediate (no compromise)

1 bit wider

compromise: one bit less for offset → offsets smaller.