Computer Architecture

EE 4720

Midterm Examination

Wednesday, 8 November 2007, 10:40–11:30 CST

Problem 1  ________ (50 pts)
Problem 2  ________ (20 pts)
Problem 3  ________ (15 pts)
Problem 4  ________ (15 pts)

Alias  ____________________________  Exam Total  ________ (100 pts)

Good Luck!
Problem 1: In the MIPS implementation below some wires are labeled with cycle numbers and values that will then be present. For example, \[ c2:9 \] indicates that at cycle 2 the wire will hold a 9. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. If a value on any labeled wire is changed the code would execute incorrectly. Note that the fourth instruction has been provided. [50 pts]

- Finish a program consistent with these labels.
- All register numbers and immediate values can be determined.
- Be sure to fill the block marked Fill In.
Problem 2: The VAX locc instruction (from Homework 3) appears below, along with its encoding (taken from the Homework 3 solution).[20 pts]

locc #65, r2, (r3)

Instruction Encoding:

-opcode- -- 1st operand ---- -- 2nd op - -- 3rd op -

locc imm PC* immed reg r2 reg-d r3
mode value mode mode

0x3a 0x8 0xf 0x41 0x5 0x2 0x6 0x3 <- Encoded value.
7 0 7 4 3 0 7 0 7 4 3 0 <- Bit position.

(a) A MIPS implementation can easily retrieve its two source register values in one clock cycle.

What about the VAX instruction formats makes one-cycle source register value retrieval more difficult in a VAX implementation (without lowering clock frequency)? Consider instructions with at most two source register operands.

(b) The constant 65 (0x41) is encoded in immediate mode, taking a total of two bytes, rather than literal mode, which would take only one byte if 65 were small enough for literal mode.

Given the way VAX encodes operands one might expect the maximum literal size to be only four bits. Why?

In fact, the maximum literal size is six bits. How is that accomplished? (If you don’t know or remember the details then make up something reasonable.)
Problem 3: Under SPECcpu2006 base rules at most four compiler optimization switches can be used per language. [15 pts]

☐ What is the rationale for that restriction?

Suppose a tester would like to use five options:

```
cc -O4 --optimization-a --optimization-b --optimization-c --optimization-d
```

The tester modifies the compiler by combining options a and b, the modified compiler is made available as a product. Now compilation can be done consistent with the rules:

```
cc -O4 --optimization-ab --optimization-c --optimization-d
```

☐ Should that be considered cheating? Explain why or why not.
Problem 4: Answer each question below.[15 pts]

(a) In MIPS the branch instruction uses a 16-bit immediate to specify a displacement target, and the jal instruction uses a 26-bit immediate. Why does it make sense to choose a coding for jal that has a larger immediate?

(b) An ISA should be designed to support decades of implementations but invariably ISA designers are biased towards a first implementation at the expense of later ones. A branch delay slot (as present in MIPS and SPARC, for example) is a good example of such a phenomenon.

☐ Explain why the branch delay slot is a good example of a shortsighted ISA feature.