EE108B Review Session #3

Drew Hall

Stanford University

Agenda

- Introduction
- Announcements
- Homework Hints
- Midterm Solutions

Introduction

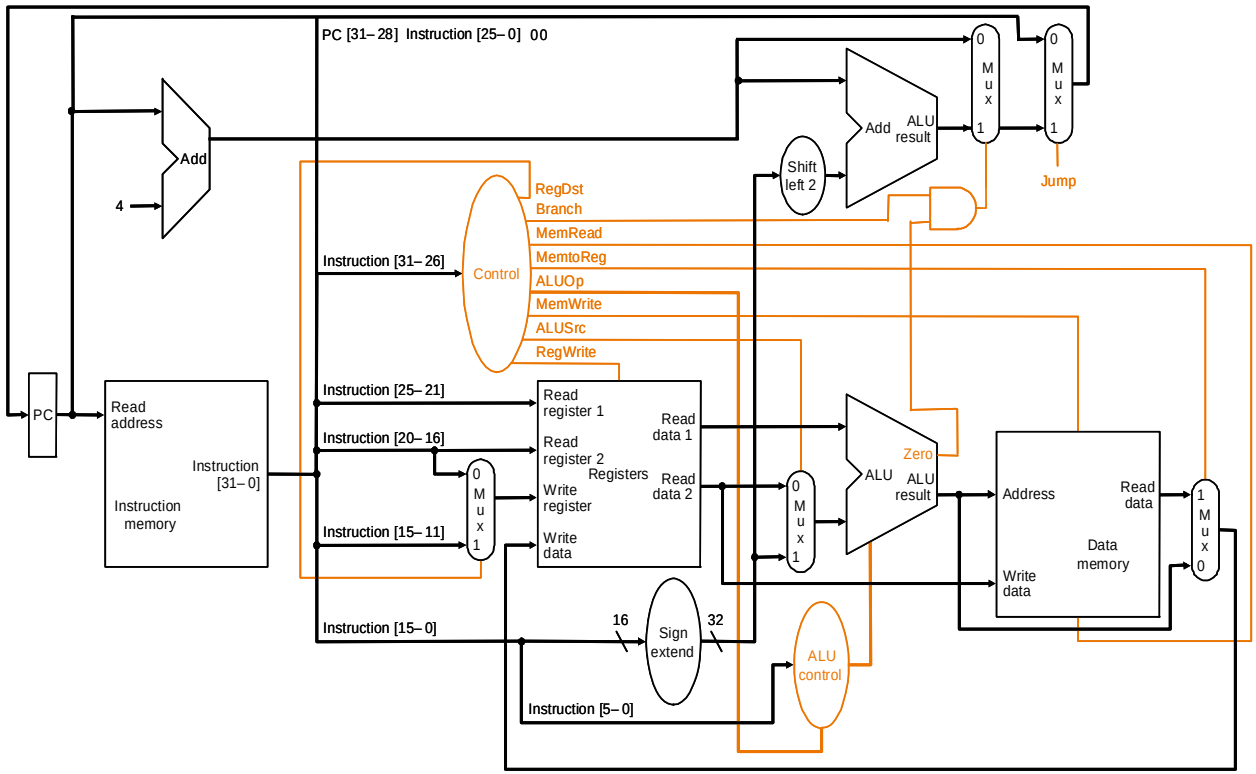
- MS/Ph.D student in Electrical Engineering
  - Specialties: Digital System Design, Computer Architecture
  - 1st time TA'ing 108B
- Office Hours:
  - Monday: 5:30 PM – 7:30 PM
  - Tuesday: 11:00 AM – 1:00 PM
  - Packard 129 (the 108B lab)
  - Preference is given to the lab students but all students welcome

Announcements

- Homework 2 Due Oct 25
  - NO LATE DAY
- Lab 2 Due Nov 1
- Midterm Oct 27
  - 7:00 PM – 9:00 PM
  - Gates B01
- Halloween October 31st

Homework Hints 1

Using the figure below, show all the necessary data and control path for instruction “jalr rd, rs” in the single-cycle MIPS processor. Hint: Think about how the instruction will be encoded and what extra control lines will be necessary to implement this design.



Homework Hints 2

Problem 6 – Be patient… we will discuss this one a little later on!

Midterm

- Format will be similar to sample midterms
- Be prepared!!
  - Read the text book
  - Read the lecture notes
- Be comfortable and familiar with the MIPS ISA
- Ask questions as we go along if you don't understand
- If you have questions after the review session, go to office hours
- If you need to write code, add comments! Yes I know… comments don't matter, except when it comes to understanding and grading your midterm!

Midterm Q1

The following program tries to copy words from the address in register $a0 to the address in register $a1, counting the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers $v1, $a0, and $a1. This terminating word should be copied but not counted.

```
loop:
    lw $v1, 0($a0)          # Read next word from source
    addi $v0, $v0, 1        # Increment count words copied
    sw $v1, 0($a1)          # Write to destination
    addi $a0, $a0, 1        # Advance pointer to next source
    addi $a1, $a1, 1        # Advance pointer to next dest
    beq $v1, $zero, loop    # Loop if word copied != zero
```

There are multiple bugs in this MIPS program. Fix them and write a bug-free version.

Midterm Q1 – cont'd

- Assuming that you loop five times, how many instructions are executed during the running of your new code?

- Assuming that you loop five times, how many memory data references will be made during execution?

Midterm Q2

```
func: addiu $sp, $sp, -16
      sw $ra, 8($sp)
      sw $s0, 12($sp)
      beq $a0, $0, here
      addi $s0, $0, 1
      add $t1, $a0, $0
      j test
mul:  mult $s0, $a1
      mflo $s0
      addi $t1, $t1, -1
test: bne $t1, $0, mul
      addiu $a0, $a0, -1
      jal func
      add $v0, $v0, $s0
      j exit
here: addiu $v0, $0, 1
exit: lw $ra 8($sp)
      lw $s0, 12($sp)
      addiu $sp, $sp,  16
      jr $ra
```

- Describe in one sentence what this code does.

Midterm Q3

- You have decided to increase the number of general purpose registers in the MIPS ISA from 32 to 64 to reduce the number of loads and stores. What impact will this have on assembly language programming? What impact will this have on processor implementation?

- What is the benefit of improving CPU performance by reducing CPI as opposed to increasing clock frequency?

Midterm Q3 – cont'd

- If you want to call a function using a function pointer it is useful to have an instruction called jalr (jump and link register). Write a minimal sequence of MIPS instructions for the following:

  jalr $s0; go to $s0, $ra = PC + 4

- Let CONST_A and CONST_B represent 16-bit constant values. For the following MIPS code fragment, please specify if the resulting values in $s0 and $s1 will ALWAYS BE THE SAME, NEVER BE THE SAME, or SOMETIMES BE THE SAME. To receive credit, you must explain your reasoning.

  lui $t0, CONST_A

  addi $s0, $t0, CONST_B

  ori $s1, $t0, CONST_B

Midterm Q4

- What is the effective CPI if we consider **lui** to be an ALU operation?

Instruction Type	CPI
Load/Store	2
ALU Operations	1
Branches	2
Jumps	2

Core MIPS	Name	Freq.
add immediate unsigned	addiu	18%
add unsigned	addu	12%
And	and	2%
and immediate	andi	1%
branch greater than or equal zero	bgez	1%
branch less than zero	bltz	1%
branch on equal (zero)	beq	3%
branch on not equal (zero)	bne	2%
jump and link	jal	2%
jump register	jr	1%
load byte	lb	2%
load half	lh	2%
load upper immediate	lui	6%
load word	lw	17%
set less than	slt	1%
set less than immediate	slti	3%
set less than immediate unsigned	sltiu	1%
set less than unsigned	sltu	1%
shift left logical	sll	5%
shift right arithmetic	sra	3%
store byte	sb	2%
store word	sw	14%