CprE 381: Computer Organization and Assembly Level Programming

Exam 1 Review

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Administrative – Course Workload

- Qualitative Feedback:
  - Too much work
  - Liked “putting it together”
  - Liked and disliked seeing different ways to implement
  - Disliked amount of testing
- Going forward:
  - Reducing the number of instructions needed to support → reduce work on lower-value content
  - Reduce # of test programs individual teams are responsible for → common test pool created by class
  - Testing framework and processor interface/memory instantiation provided

Administrative – Course Workload

- Lab 2
  - Median reporting student: 8.2hrs outside
  - Median grade (all students): 100%
- Lab 3
  - Median reporting student: 9.8hrs outside
  - Most students had 0hrs in lab two weeks ago
- 4 week average from reporting students
  - ~5hrs on labs outside of class
  - Will keep updated throughout semester

Administrative

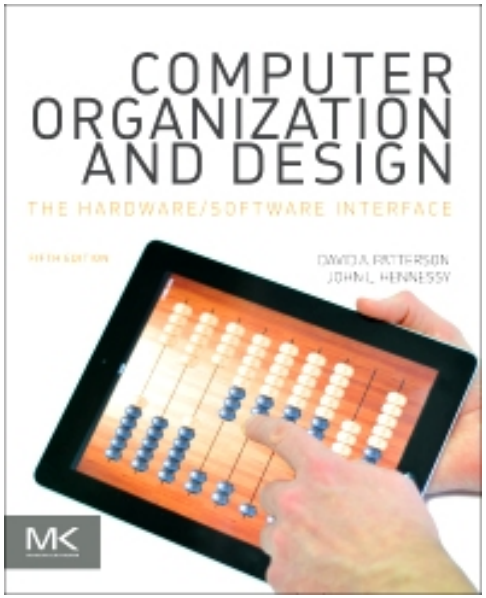
- Exam 1
  - When: Mon, Feb 18 In-Class (i.e., 50 minutes)
  - Where:
    - **A-L: 2105 Pearson (THIS ROOM)**
    - **M-Z: 0308 Elings**
  - What: Intro to computers + MIPS assembly
  - Will curve average up if necessary
  - Aids:
    - 1 8.5”x11” notes sheet; double-sided; self-generated; hand-written
    - No electronic devices, including calculators
    - Copy of green sheet will be provided on exam
  - TODOs:
    - Review in-class assessments, HWs, Lab high-level designs, P&H exercises, lecture slides, readings
    - Ask questions here, OH at 1:30pm, and on Canvas

Exam Focus

- General Concepts (textbook + lab + lecture)
  - Definitions, examples, trends
- MIPS ISA and architecture
  - ISA design
  - Register conventions – not emphasized
  - Translation from C to MIPS
  - Translation from MIPS assembly to machine code
  - MIPS assembly hand-simulation
  - Branches
  - Memory access (from arrays/pointers to load/store)
  - Procedure call basics (jal, jr $ra, $a0-a3)
  - SIMD instructions
- Basic HW design complexity as applies to ISA design
  - No VHDL-specific questions

Textbook References

- Patterson & Hennessy:
  - Chapter 1: Computer Abstractions and Technology
    - 1.1-1.5
  - Chapter 2: Instruction: Language of the Computer
    - 2.1-2.10,2.12-2.14



MIPS ISA – Things to Know

- List of instructions
  - How to use/read green card (also on 381 website)
    - `Add  add  R R[rd] = R[rs] + R[rt]`
    - `Add Imm  addi I R[rt] = R[rs] + SgnExtImm`
    - `Jump  j  J PC=jumpAddr`
  - Some frequently used ones on P&H p.64
  - SIMD instructions
- Instruction types/formats
  - R (register) arithmetic inst. format (`add $rd, $rs, $rt`)
  - I (immediate) data transfer format (`addi $rt, $rs, Imm`)
  - J (jump) jump format (`j label`)

MIPS ISA – Things to Know (cont.)

- Register usage conventions
  - P&H 2.8
  - Most of them relating to procedure calls

- Some (simple) sample problems:
  - How to implement `clear $t0` instruction?
  - How to implement `ble $t3, $t5, label` instruction?
    - Can use `slt` and `beq`
  - Translate the following C code to MIPS (and vice versa):

```
for (i = 0; i < 100; i++) {
  for (j = 0; j < 100; j++) {
    C[i] += A[i][j] * B[j];
  }
}
```

MIPS Procedure Call Convention Base

- Caller/Callee MIPS Conventions
- Caller
  - Load arguments in `$a0`-`$a3`, rest passed on stack
  - Execute `jal` instruction
- Callee Setup
  - Save return address register, `$ra`, if calls another function
- Callee Return
  - Place return value in `$v0` and `$v1`
  - Return by `jr $ra`

Pseudoinstructions

- Assembler expands pseudoinstructions

```
move $t0, $t1       # Copy $t1 to $t0
```

```
addu $t0, $zero, $t1 # Actual instruction
```

- Some pseudoinstructions need a temporary register:
  - Cannot use **`$t`**, **`$s`**, etc. since they may be in use
  - The **`$at`** register is reserved for the assembler

```
blt $t0, $t1, L1    # Goto L1 if $t0 < $t1
```

```
slt $at, $t0, $t1   # Set $at = 1 if $t0 < $t1
bne $at, $zero, L1 # Goto L1 if $at != 0
```

Some (Rejected) Questions

1. Consider the `bne` instruction:

   ```
   bne $t0, $t1, L1
   ```

   How far can L1 be away?

2. What are the advantages and disadvantages of using assembly language?

3. Give a MIPS instruction sequence to represent the pseudoinstruction

   ```
   rol rdest, rsrc1, rsrc2   # rotate left: msb→lsb
   ```

4. Write a MIPS implementation of the power function given below

```
char satAdd(char val1, char val2) {
   int sum;
   sum = (int)val1 + (int)val2;
   if(sum>=255) return 255;
   return (char)sum;
}
```

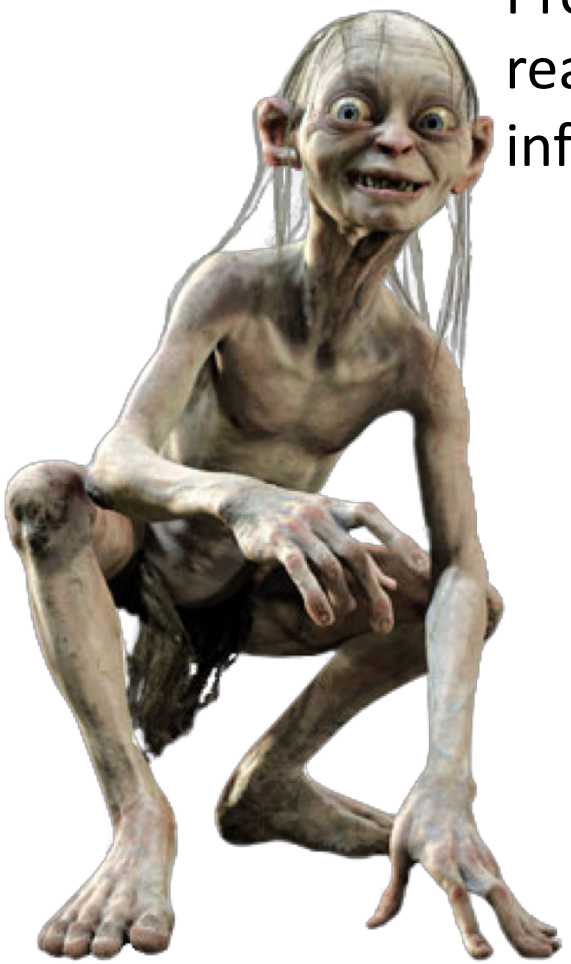
Some More (Rejected) Questions

5. The MIPS ISA has fixed length instructions (32 bits) and only three instruction encoding formats. Taking into account what you know of memory and hooking up register file addresses, briefly explain how the fixed-length encoding and the simple formats simplify the processor design. Briefly give a technical reason why complex instructions are not included in MIPS ISA.
   - Because it's RISC is not a technical reason

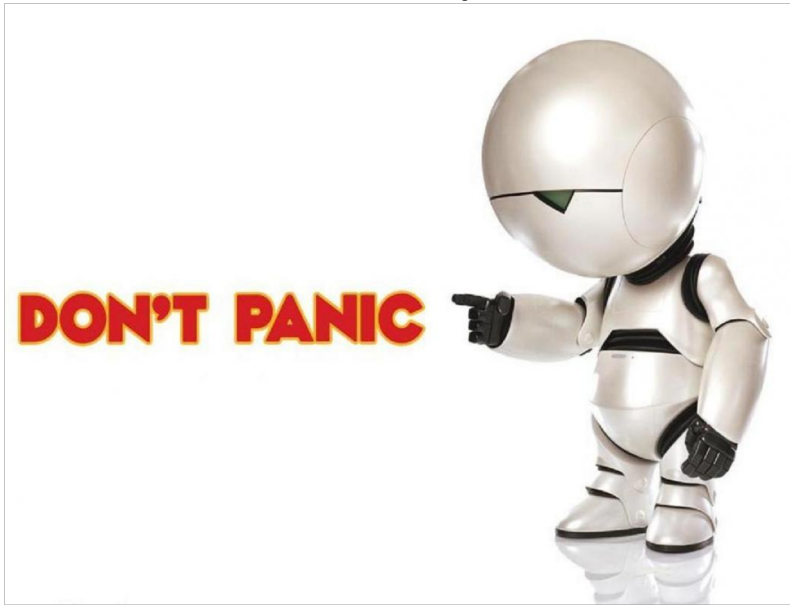
6. Given the processor's datapath and control, as discussed in the lecture, come up with a scheme to add the 'lui' instruction to it

7. Write the body of a function that returns the second least significant byte of a 32-bit integer using bit-wise logical operations and shifts.

Final Advice



Professors are (inadvertently) tricksy → read problems and discard superfulous information



You can do well on this exam; show me your knowledge. Don't get thrown by a hard or confusing problem.

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)