CSE 141L
Computer Architecture Lab
Spring 2005

Lecture 4
Pramod V. Argade
May 17th, 2005

CSE141: Computer Architecture Lab

Instructor: Pramod V. Argade (p2argade@cs.ucsd.edu)
Office Hour:
Mon. 5:00 - 6:00 PM (AP&M 5218)

TAs:
Anjum Gupta: a3gupta@cs.ucsd.edu
Baris Arslan: barslan@cs.ucsd.edu
Kevin Hu: huhu@cs.ucsd.edu
Raid Ayoub: rayoub@cs.ucsd.edu

Textbook: LogicWorks 5, Interactive Circuit Design Software
Capilano Computing Systems, 2004
(Available in UCSD Bookstore.)

Web-page: http://www.cse.ucsd.edu/classes/sp05/cse141L
### Spring 2005 CSE 141L Course Schedule

<table>
<thead>
<tr>
<th>Lecture #</th>
<th>Date</th>
<th>Day</th>
<th>Lecture Topic</th>
<th>Lab Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3/29</td>
<td>Tuesday</td>
<td>No Class</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4/5</td>
<td>Tuesday</td>
<td>Lab 1: 8-bit Processor ISA</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>4/12</td>
<td>Tuesday</td>
<td>Lab 1 Discussion</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4/19</td>
<td>Tuesday</td>
<td>Lab 2: Assembler, ISS</td>
<td>Lab1</td>
</tr>
<tr>
<td>5</td>
<td>4/26</td>
<td>Tuesday</td>
<td>LogicWorks 5 Review</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>5/3</td>
<td>Tuesday</td>
<td>Lab 3: Data Path</td>
<td>Lab2</td>
</tr>
<tr>
<td>7</td>
<td>5/10</td>
<td>Tuesday</td>
<td>Lab 3 Discussion</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>5/17</td>
<td>Tuesday</td>
<td>Lab 4: Full CPU</td>
<td>Lab3</td>
</tr>
<tr>
<td>9</td>
<td>5/24</td>
<td>Tuesday</td>
<td>Lab 4 Discussion</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>6/1 - 6/3</td>
<td></td>
<td>Lab 4 Demo by Students</td>
<td>Lab4</td>
</tr>
</tbody>
</table>

### Lab Due Dates

- **Lab 4 Due:**
  - ⇒ Before 6:30 PM Tuesday, May 31<sup>st</sup>
  - ⇒ You must make an appointment to demonstrate your CPU June 1 - June 3
- You cannot make any changes to the design between the time you submit to your demo to TA
- No late submissions!
Microprocessor Design Steps

- Design Instruction Set Architecture (ISA)
- Develop software generation tools
- Code applications
- Develop instruction set simulator (ISS)
- Design datapath, verify it
  - Design the Processor, simulate logic
  - Verify the processor
  - (Fabricate the chip: not in this class!)

Lab 4 Assignment

- Design logic for complete 8-bit processor architecture
- Use LogicWorks 4 to design logic
- Simulate the operation executing 3 programs from Lab 1
What you must include

- Reset logic
- Program Counter Logic
  - Non-control transfer instructions
  - Control transfer instructions
  - Halt instruction
- Data path modified for Load/Store instructions
- Control logic
- Instruction counter
  - Initialize on reset
  - Freeze on HALT instruction
What you will turn in for this Lab

- Summary of your ISA from Lab 1 and assembly code with machine code for 3 programs.
- Printed schematics for the top level CPU as well as all the lower level modules you designed in LogicWorks 4.
- All the LogicWorks 4 files you created (to be submitted via electronic submission).
- Answers to following questions.

Questions

- What changes did you make in your original ISA and why?
- What is instruction count for each one of the three programs? How do the numbers compare with those for the ISS. If the numbers are different, why?
- What are the strengths of your design?
- What are the deficiencies of your design?
- Which instruction is most responsible for the length of the critical path?
- Which instruction is most expensive in terms of the number of gates required?
Question Continued

• Having gone through a complete CPU design experience, what would you do differently in your ISA to:
  – Decrease static and dynamic instruction count
  – Simplify data path design
  – Simplify CPU design
• If you were to pipeline the execution, what would the pipeline stages be? Give at least three issues that will complicate the design of your processor.

Lab 4 Grading

• It is your responsibility to make an appointment with one of the TAs before 5/31/05
• Show TA that your CPU design works before end of Friday, June 3rd.
• You should test the programs using the data patterns given in Lab1.
• The TAs may test your design for correct functionality using their own data files that satisfy the constraints outlined in Lab 1.
Useful Hints

• Avoid propagation of unknowns
  – Initialize on reset
• Build hierarchical design
• Test thoroughly at every level of hierarchy
  – Connect binary switches and hex keypads to provide inputs
  – Connect binary and hex displays to observe behavior
• Write an assembly program to test individual instructions in your CPU
  – Self-checking programs are ideal!

What is Functional Verification?

• Making sure that your design is functionally correct!
  – Reset behavior
  – Instructions
    • Addressing modes
    • Algorithms in hardware, e.g. setting carry
    • Corner cases
    • Control transfer: branch/jump
  – Memory access
  – Special features
    • e.g. HALT in our case
Importance of Verification

- General purpose processor must run without a flaw any application running on it!
- Programmers will use the CPU in ways you never imagined!
- Processor may be used in mission critical application
- It is costly to fix bugs in processors
  - Chip mask and fabrication costs
  - System HW and SW redesign
  - Lost market opportunity

A Program to Test 8-bit CPU
Before you leave

• Discussion on Lab 4 next Tuesday
• Remember to make appointment with the TAs to show your operational design