CSE141L: Computer Architecture Lab

**Instructor:** Pramod V. Argade (p2argade@cs.ucsd.edu)
Office Hours:
Tue. 7:30 - 8:30 PM (AP&M 4141)
Wed. 4:30 - 5:30 PM (AP&M 4141)

**TA:**
Hua (Kevin) Yang (huhu@cs.ucsd.edu)
Office Hour: Mon/Tue 3 - 4:30 PM
Dianne Shaw (dshaw@ucsd.edu)
Office Hour: Tue 4 - 5:30 PM, Thu 4:30 - 6
Anthony Choi (buchoi@ucsd.edu)
Office Hour: Tue/Wed 1:30 - 3 PM
Hou Choi (hochoi@ucsd.edu)
Office Hour: Mon 1:30 - 3 PM, Fri. 3 - 4:30

**Lecture:** Tue. 6 - 7:15 PM, Center 109

**Reference Book:** LogicWorks4, Capilano Computing Systems
Available in Bookstore

**Web-page:** [http://www.cse.ucsd.edu/classes/su04/cse141L](http://www.cse.ucsd.edu/classes/su04/cse141L)
## CSE141L Course Schedule

<table>
<thead>
<tr>
<th>Lecture #</th>
<th>Date</th>
<th>Time</th>
<th>Room</th>
<th>Topic</th>
<th>Assignment Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tue. 6/29</td>
<td>6 - 7:15 PM</td>
<td>Center 109</td>
<td>Assignment 1: ISA</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Tue. 7/6</td>
<td>6 - 7:15 PM</td>
<td>Center 109</td>
<td>Assignment 2: ISS</td>
<td>#1</td>
</tr>
<tr>
<td>3</td>
<td>Tue. 7/13</td>
<td>6 - 7:15 PM</td>
<td>Center 109</td>
<td>Assignment 3: Datapath</td>
<td>#2</td>
</tr>
<tr>
<td>4</td>
<td>Tue. 7/20</td>
<td>6 - 7:15 PM</td>
<td>Center 109</td>
<td>Assignment 4: CPU</td>
<td>#3</td>
</tr>
<tr>
<td>-</td>
<td>Tue. 7/27</td>
<td></td>
<td>No Lecture</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

July 26 - 29: Students demonstrate their working CPU Design to TAs by appointment. Assignment #4 reports are due during the demonstration.
Prerequisites

- Completed CSE 140 & 140L
- Completed CSE 141 or currently enrolled in
- Working knowledge of LogicWorks 4
- You may use Xilinx tools, no TA support
- C/C++ or Java and assembly programming
Administrative

• From groups of preferrably 3 students
  – Don’t leave without forming a group
  – Can’t change group w/o approval
• Computer accounts distributed in class
• Lab: AP&M 2444 (PCs have LogicWorks)
• Unix workstations: AP&M B402
Grading

• All Labs are worth 100 points
• Weight of each Lab towards final grade
  Lab1: 20%
  Lab2: 20%
  Lab3: 25%
  Lab4: 35%
• Reports beginning of the class in one week
• No late submissions!
• Everyone in the group gets the same grade
• Appeal grade on Lab to TA in 2 days!
Academic Honesty

• **Do**
  – Work within your own group

• **Don’t**
  – Discuss solutions between groups
  – Discuss with some who took the course before
  – Look at completed project of other group
  – Find solutions on the Web or in textbooks
  – Copy design from someone outside the group
  – Alter LogicWorks data
Microprocessor Design Steps

• Design Instruction Set Architecture (ISA)
• Develop software generation tools
  – Compiler, Assembler, Linker, Debugger, Libraries, ...
• Code applications
• Develop Instruction Set Simulator (ISS)
• Run applications, gather run-time statistics, tune ISA
• Design the processor, tune implementation
• Verify the processor
• Fabricate the chip
Lab Course Assignment

Design an 8-bit Processor
Background

• **String Search**
  – Look for the first occurrence of a string in another string
  – Start address and size of each string is given
  – Make provision for a failed search (return -1)

• **Median**
  – At least half the numbers in the list are less than median
  – Sort the list in ascending order
    • If odd number of elements, median is the middle value
    • If even number of elements, median is average of the two middle values

• **Mean**
  – Average of all elements
  – Round average up if fractional part >= 0.5
Description of the Course

• **Lab 1:** Design Instruction Set Architecture (ISA) and code 3 programs in assembly.

• **Lab 2:** Develop an instruction set simulator (ISS) and test your programs from Lab 1.

• **Lab 3:** Develop datapath for your architecture and test it.

• **Lab 4:** Develop hardware for your processor and test your programs from Lab 1.
Lab 1

- Construct 8-bit ISA
- Optimize only for 3 given programs
- Lab 1 due in the beginning of class July 7th
- What you will turn in for this Lab
  - Lab report covering all the issues outlined below.
  - Assembly code and hand-assembled machine code for three programs in your ISA.
  - Instruction and data files for the three programs.
Program 1: Search a string

- D-Mem[ 0 ] = Start address of string A
- D-Mem[ 1 ] = # of elements in string A
- D-Mem[ 2 ] = Start address of string B
- D-Mem[ 3 ] = # of elements in string B
- Find the first occurrence of string A in string B and return the address in D-Mem[4], if successful and -1 otherwise
- See lab write up 1 for sample D-Mem data
Program 2: Median

- D-Mem[ 0 ] = # of elements of an array of unsigned bytes
- D-Mem[ 1 ] = starting address of array a[]
- Find Median value and save it at D-Mem[ 2 ]
- Initially, D-Mem[] = \{0x09, 0x07, 0x10, 0xdd, 0xdd, 0xdd, 0xdd, 0xdd, 0x78, 0xff, 0x65, 0x55, 0xea, 0x23, 0x9d, 0x09, 0x44\}
Bubble Sort: Early Out

```java
limit = num - 1;
flag = true;
while( flag ) {
    flag = false;
    for( int i = 0; i < limit; i++ ) {
        if( a[ i ] < a[ i+1 ] ) {
            temp = a[ i ];
            a[ i ] = a[ i+1 ];
            a[ i+1 ] = temp;
            flag = true;
        }
    }
    limit = limit--;
}
```
Program 3: Mean

- D-Mem[ 0 ] = starting address of array a[]
- D-Mem[ 1 ] = 16 (# of elements of array a[] of unsigned bytes)
- Find the mean (average value) for the elements of a[]
- Your program may be specific to mean of 16 elements
- You must round the mean to nearest unsigned byte
- Initially, D-Mem[] = \{0x08, 0x10, 0xde, 0xad, 0xf0, 0x0d, 0xdd, 0xdd, 0xff, 0x12, 0x80, 0x27, 0x34, 0x87, 0xce, 0x46, 0x01, 0xdd, 0x5d, 0xd8, 0x76, 0x49, 0xbf, 0x06\}
Tips for Mean

- Design specifically for count = 16
- No general divide instruction needed!
- Is there an easy way to divide by 16?
- Addition of 2 unsigned bytes produces a 9-bit result
- How many bits are required for upper result of a sum of 16 unsigned bytes?
- How to determine whether the fraction is $\geq 0.5$?
Design Goals

• Minimize dynamic instruction count
• Distinguish dynamic instruction count from static instruction count: what are the implications of both counts?
• Simplify your processor hardware design
  – You will design hardware in Lab 3 & 4
Requirements

- Instruction format should be fixed-length 8-bit instructions.
- Instruction memory (I-Mem) and data memory (D-Mem) are separate.
- The memory operations must be explicit load and store instructions in your ISA.
- Memory is byte addressable, and loads/stores read and write exactly 8 bits.
- This is a 8-bit machine for every aspect. All registers and data types are maximum 8 bits.
- Single-ported instruction and data memory (a maximum of one read or one write per cycle, not both) must be used.
- A register file with only one write port. May have multiple read ports may be used.
- When the processor is reset, execution starts at instruction memory location 0.
- A “halt” instruction that would halt the execution must be supported.
General Tips

- Instruction types: triadic, diadic, monadic, niladic
- Code and test programs in C/C++/Java
- Assume that all resources are available
- An instruction may have multiple purposes
- Use iterative design process
  - Design ISA ➔ Code programs ➔ Analyze

Pramod Argade
UCSD CSE 141L, Fall ‘03
ISA Specification

- How many instruction formats are supported and what they are.
- Instructions supported and their opcodes. Include a brief description for each instruction, including the operation it performs and registers affected.
- Number of general purpose registers, any other internal state registers.
- Size of the instruction and data memory supported
- Addressing modes supported, including how the addresses are computed.
Lab Report

• Describe your design goals.
• Classify your processor in any one of the classical ways (e.g. stack machine, accumulator, register, load-store, etc.) If your architecture is different from classical ways, propose a name for you architecture.
• What instruction formats are supported? Give all opcodes.
• How many registers are supported? What are the special features of these registers?
• What addressing modes are supported? How are the addresses calculated?
• What types of branch and jump instructions are supported? How are the target addresses calculated? What maximum branch distance is supported?
Lab Report Continued

- Give a brief description of each instruction and its side effects.
- How large are instruction and data memories?
- In what ways did you optimize for dynamic instruction count? Are there any special instructions for each one of the three programs that significantly reduced the instruction count?
- How did you optimize for ease of implementation?
- What would you have done differently if you had 4 more bits for instruction?
- What are the bottlenecks in your design, i.e. what resources will you run out of the most quickly for bigger, more complex programs?
Guidelines for Assembly Code

• Make it easy for us to understand it!
• Provide comment for every line
• Example:

    // mean.s
    // Temporary memory usage
    ...
    // Register usage
    // r0 used to index array
    ...
    0x0c 000 start: mov r3, $0 // r3 = 0, has address of a[ 0 ]
    0xe3 001 ld r0, *r3 // r0 points to a[ 0 ]
    0x08 002 mov r2, $0 // r2 = sum_low = 0
ISA Considerations

• Number of general purpose registers
• Accumulator(s)
• Special registers, internal state (e.g. PSW)
• Branch and jump instructions
  – Types: conditional, unconditional
  – PC-relative, absolute, register-indirect
Before you leave...

- Get computer account or form
- Form a group
  - Give information about your group
    - Names, email address

- Questions?