CSE 141L
Computer Architecture Lab
Fall, 2005

Lecture 1
Pramod V. Argade
Prerequisites

- Completed CSE 140 & 140L
- Completed CSE 141 or currently enrolled in
- Working knowledge of LogicWorks 4/5
- Experience in C/C++ or Java and assembly programming

Evolution of Intel Processors*

<table>
<thead>
<tr>
<th>Processor</th>
<th>Year of Introduction</th>
<th># Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>4004</td>
<td>1971</td>
<td>2,250</td>
</tr>
<tr>
<td>8008</td>
<td>1972</td>
<td>2,500</td>
</tr>
<tr>
<td>8080</td>
<td>1974</td>
<td>5,000</td>
</tr>
<tr>
<td>8086</td>
<td>1978</td>
<td>29,000</td>
</tr>
<tr>
<td>286</td>
<td>1982</td>
<td>120,000</td>
</tr>
<tr>
<td>Intel I386</td>
<td>1985</td>
<td>275,000</td>
</tr>
<tr>
<td>Intel I486</td>
<td>1989</td>
<td>1,180,000</td>
</tr>
<tr>
<td>Intel Pentium</td>
<td>1993</td>
<td>3,100,000</td>
</tr>
<tr>
<td>Intel Pentium II</td>
<td>1997</td>
<td>7,500,000</td>
</tr>
<tr>
<td>Intel Pentium III</td>
<td>1999</td>
<td>24,000,000</td>
</tr>
<tr>
<td>Intel Pentium 4</td>
<td>2000</td>
<td>42,000,000</td>
</tr>
<tr>
<td>Intel Itanium</td>
<td>2002</td>
<td>220,000,000</td>
</tr>
<tr>
<td>Intel Itanium 2</td>
<td>2003</td>
<td>410,000,000</td>
</tr>
</tbody>
</table>

*http://www.intel.com/research/silicon/mooreslaw.htm

1 Bit Memory: 6 Transistors
1 Inverter: 2 Transistors
1 NAND gate: 4 Transistors
1 NOR gate: 4 Transistors
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 a “Statistics Processor”

Statistics Background

- Mode
  - Most frequent value
  - Can have multiple modes
  - You can assume that there is only one mode for your data sets
- 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 middle two values
- Mean
  - Average of all elements
  - Round average up if fractional part $\geq 0.5$
Description of the Course

- **Lab1**: Design Instruction Set Architecture (ISA) and code 3 programs in assembly for your machine.

- **Lab2**: Develop assembler and an instruction set simulator (ISS) and test your programs from Lab 1.

- **Lab3**: Develop datapath for your architecture and test it.

- **Lab4**: Develop hardware for your processor and test your programs from Lab 1.

---

Fall 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>9/27</td>
<td>Tuesday</td>
<td>No Class</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10/4</td>
<td>Tuesday</td>
<td>Lab 1: 8-bit Process ISA</td>
<td>Lab 1</td>
</tr>
<tr>
<td>3</td>
<td>10/11</td>
<td>Tuesday</td>
<td>Lab 1 Discussion</td>
<td>Lab 1</td>
</tr>
<tr>
<td>4</td>
<td>10/18</td>
<td>Tuesday</td>
<td>Lab 2: Assembler, ISS</td>
<td>Lab 1</td>
</tr>
<tr>
<td>5</td>
<td>10/25</td>
<td>Tuesday</td>
<td>LogicWorks 5 Review</td>
<td>Lab 2</td>
</tr>
<tr>
<td>6</td>
<td>11/1</td>
<td>Tuesday</td>
<td>Lab 3: Data Path</td>
<td>Lab 2</td>
</tr>
<tr>
<td>7</td>
<td>11/8</td>
<td>Tuesday</td>
<td>Lab 3 Discussion</td>
<td>Lab 3</td>
</tr>
<tr>
<td>8</td>
<td>11/15</td>
<td>Tuesday</td>
<td>Lab 4: Full CPU</td>
<td>Lab 3</td>
</tr>
<tr>
<td>9</td>
<td>11/22</td>
<td>Tuesday</td>
<td>Lab 4 Discussion</td>
<td></td>
</tr>
<tr>
<td>11/29 - 12/2</td>
<td>Lab 4 Demo by Students</td>
<td>Lab 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 two weeks
- No late submissions!
- Everyone in the group gets the same grade
- Appeal grade on Lab to TA within 1 week

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: EBU3-B270 (PCs have LogicWorks)
- Unix workstations: AP&M B402
Academic Honesty

- Do
  - Work within your own group

- Don’t
  - Discuss solutions between groups
  - Discuss with someone 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

Lab 1

- Construct 8-bit ISA
- Optimize only for 3 given programs
- Lab 1 due in the beginning of class October 18th
- 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: Mode

- D-Mem[0] = # of elements of an array of unsigned bytes
- D-Mem[1] = starting address of array a[]
- Find the mode, i.e. most frequent value and save it in D-Mem[2]
- Initially, D-Mem[] = {0xb, 0x0a, 0xde, 0xad, 0xf0, 0x0d, 0xde, 0xad, 0xf0, 0x0d, 0x4, 0x7, 0x4, 0x5, 0x7, 0x4, 0x5, 0x6, 0x4, 0x7, 0x5}

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]
- Number of elements will always be odd
- Initially, D-Mem[] = {0x09, 0x0b, 0x00, 0xde, 0xad, 0xf0, 0x0d, 0xde, 0xad, 0xf0, 0x0d, 0x78, 0xff, 0x65, 0x55, 0xea, 0x23, 0x9d, 0x09, 0x44}
- You must implement bubble sort, early out algorithm!
Bubble Sort: Early Out

```c
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[] = {0x0b, 0x10, 0x0, 0xde, 0xad, 0xf0, 0x0d, 0x0, 0xde, 0xad, 0xf0, 0x0d, 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 >= 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
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
...
0xc00 start: mov r3, $0    // r3 = 0, has address of a[0]
0xe001 ldi r0, *r3  // r0 points to a[0]
0x02 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?