EE/CSE 469: Computer Design and Organization

Professor Scott Hauck, 307Q, hauck@uw.edu
Office hours: email w/schedule

TA: Hongwei Ding (hwding@uw.edu)
    Ailing Piao (alpiao@uw.edu)

Office hours: (EEB-361) up-to-date times on website

Book:

Grading (approximate):
20% - Homeworks 35% - Design Project 20% - Midterm 25% - Final Exam

Prerequisites

Basic Logic Design and Boolean Algebra
   AND, OR, NAND, NOR gates
   Boolean Algebra
   D flip-flops, registers, and memories
   Binary numbers, 2’s complement, negation, overflows

Verilog

C/C++/Java programming

If you don’t know this material, **DO NOT TAKE THE CLASS**

If you don’t remember this material, **REVIEW NOW.**
Joint Work Policy

The processor design and homeworks will be done in groups of 1-2.
Groups may not collaborate on the specifics of homework or on the projects.
Let me know if you need help forming groups.

OK:
- Studying together for exams
- Discussing lectures or readings
- Talking about general approaches
- Help in debugging, CAD tools peculiarities, etc.

Not OK:
- Developing a design between groups
- Implementing the CPU between groups
- Checking homework answers between groups

Violation of these rules is at minimum:
- Loss of twice the points of that assignment.
- Report of Academic Misconduct to Dean’s Level.
- Potentially fail class, be expelled from UW.

Late Policy

All assignments due by the end of the class period

Late penalties:
-10% for the first 24 hours
-20% for the second 24 hours (total ~30%)
-30% for the third 24 hours (total ~60%)
-40% for all additional hours (total ~100%)
Computer Architecture

Readings: 1.1-1.4

Interaction between hardware and software
Hardware sets realities, requirements
Area, power, performance
Software places demands on hardware
Processor only as good as software it runs

Implementing Software – The Compilation Process

/* Swap the i-th and (i+1)th element of an array */
swap(int v[], int k) {
    int temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}

SWAP:
LSL X9, X1, #3
ADD X9, X0, X9  // Compute address of v[k]
LDUR X10, [X9, #0]  // get v[k]
LDUR X11, [X9, #8]  // get v[k+1]
STUR X11, [X9,#0]  // save new value to v[k]
STUR X10, [X9, #8]  // save new value to v[k+1]
BR X30  // return from subroutine

C, C++, Java, …

Compiler

Assembly Language

Assembler

Machine Language

CPU ↔ Memory

<table>
<thead>
<tr>
<th>Instruction Set Architecture</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler</td>
<td>Operating System</td>
</tr>
<tr>
<td>Firmware</td>
<td>I/O system</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital Design</th>
<th>Circuit Design</th>
<th>Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instr. Set Proc.</td>
<td>I/O system</td>
<td></td>
</tr>
</tbody>
</table>
Computer Organization

Five classic components

- **Computer**
  - Processor
    - Control
    - Datapath
  - Memory
  - Devices
    - Input
    - Output

- **Memory**: Store instructions, data
- **Datapath**: Perform operations (Add, subtract, …)
- **Control**: Orchestrate operations (who does what when)
- **Input**: Get information from the outside world
- **Output**: Provide results

Execution cycle

1. **Instruction Fetch**: Obtain instruction from program storage
2. **Instruction Decode**: Determine required actions and instruction size
3. **Operand Fetch**: Locate and obtain operand data
4. **Execute**: Compute result value or status
5. **Result Store**: Deposit results in storage for later use
6. **Next Instruction**: Determine successor instruction