EE/CSE 469: Computer Design and Organization

Professor Mark Oskin, mhoskin@uw.edu
   Office hours: email w/schedule

TA: Eric Mullen and Xinyu Sui

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**.
Who are you?

53 in EE
2 in CSE

1 sophomore
~ 25 junior, and ~25 senior

Hawaii

Hawaii > LA > Olympia > Whidbey Island
Who am I?
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%)
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

C, C++, Java, ...

Compiler

/* Swap the ith 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

11010110000 00000 000000 00000 11110
11111000010 0000000000 00 01001 01010
11111000010 0000010000 00 01001 01011
11111000000 0000000000 00 01001 01011
11111000000 0000010000 00 01001 01010
11010110000 00000 000000 00000 11110

CPU

Memory

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
Computer Organization

Five classic components

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