Computer Architecture

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

Computer Architecture

Lecture 0 (05 March 2013)

CADSL

Architecture

Latin *architectura, from the Greek ἀρχιτέκτων – arkhitekton, from ἀρχι- "chief" and τέκτων "builder, carpenter, mason") is both the process and product of planning, designing and construction. Architectural works, in the material form of buildings, are often perceived as cultural symbols and as works of art. Historical civilizations are often identified with their surviving architectural achievements.*

Computer Architecture

In computer science and engineering, **computer architecture** is the art that specifies the relations and parts of a computer system. In the architecture of buildings, this art is normally visual, but computer architecture is logical, defining systems to serve particular purposes. In both instances (building and computer), a complete design has many details, and some details are implied by common practice.

Computer Architecture

- Instruction Set Architecture (IBM 360)
  - *… the attributes of a [computing] system as seen by the programmer. I.e. the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation. -- Amdahl, Blaauw, & Brooks, 1964*
- Machine Organization (microarchitecture)
  - ALUS, Buses, Caches, Memories, etc.
- Machine Implementation (realization)
  - Gates, cells, transistors, wires

Running Program on Processor

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) (CPI) (cycle time)

Architecture --> Implementation --> Realization

Compiler Designer **Processor Designer** **Chip Designer**

Problem Solving: Main Steps

1. Problem definition
2. Algorithm design/ Algorithm specification
3. Algorithm analysis
4. Implementation
5. Testing
6. [Maintenance]

This Course in Context

- Prerequisites
  - Digital Design – gates, logic, memory, organization
  - Programming Languages – high-level language down to machine language interface

- This course – puts it all together
  - Implement the logic that provides ISA interface
  - Must do datapath and control, but no magic
  - Manage tremendous complexity with abstraction

- Follow-on courses explore trade-offs
  - Advanced computer Architecture, Multi-core Architectures

Why Take CA?

- To become a computer designer

- To learn what is *under the hood* of a computer
  - Innate curiosity
  - To better understand when things break
  - To write better code/applications
  - To write better system software (O/S, compiler, etc.)

- Because it is intellectually fascinating!
  - What is the most complex man-made device?

About This Course

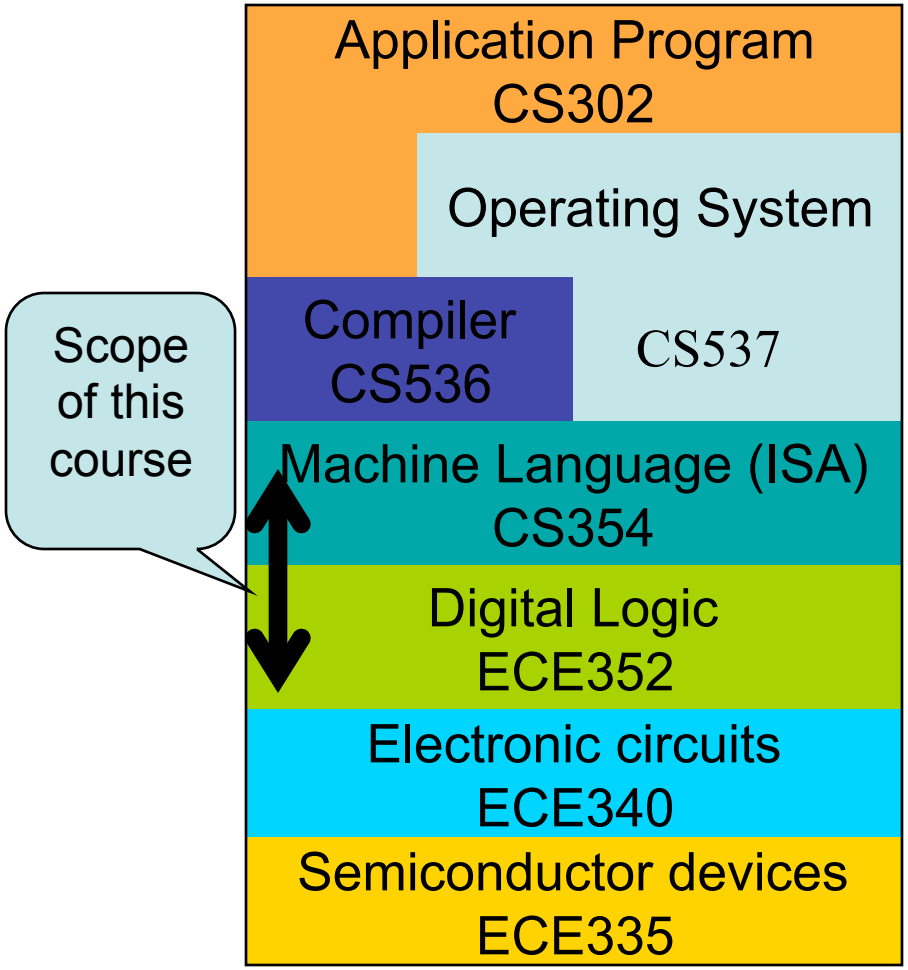
- Course Textbook
  - D.A. Patterson and J.L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 4th edition, Elsevier/Morgan Kauffman.
  - 3rd edition OK if 4th edition not available.
- Homework
  - Homework assignments, unequally weighted
  - Some group, some individual

About This Course

- Project
  - Implement processor for IIT-Mandi13 ISA
  - Priority: working nonpipelined version
  - Extra credit: pipelined version
  - Groups of 3 students, no individual projects
    - Form teams early
  - Must demo and submit written report

Abstraction and Complexity

- Abstraction helps us manage complexity
- Complex interfaces
  - Specify what to do
  - Hide details of how
- Goal: remove magic



Computer Architecture

- Exercise in engineering tradeoff analysis
  - Find the fastest/cheapest/power-efficient/etc. solution
  - Optimization problem with 100s of variables
- All the variables are changing
  - At non-uniform rates
  - With inflection points
  - Only one guarantee: Today's right answer will be wrong tomorrow
- Two high-level effects:
  - Technology push
  - Application Pull

Technology Push

- What do these two intervals have in common?
  - 1776-1999 (224 years)
  - 2000-2001 (2 years)
- Answer: Equal progress in processor speed!
- The power of exponential growth!
- Driven by Moore's Law
  - Device per chips doubles every 18-24 months
- Computer architects work to turn the additional resources into speed/power savings/functionality!

Thank You