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

CP-226: Computer Architecture

Lecture 2 (28 Jan 2013)

Computer Architecture's Changing Definition

- 1950s to 1960s:
  Computer Architecture Course = Computer Arithmetic
- 1970s to mid 1980s:
  Computer Architecture Course = Instruction Set Design, especially ISA appropriate for compilers
- 1990s onwards:
  Computer Architecture Course = Design of CPU (Processor Microarchitecture), memory system, I/O system, Multiprocessors

This Course in Context

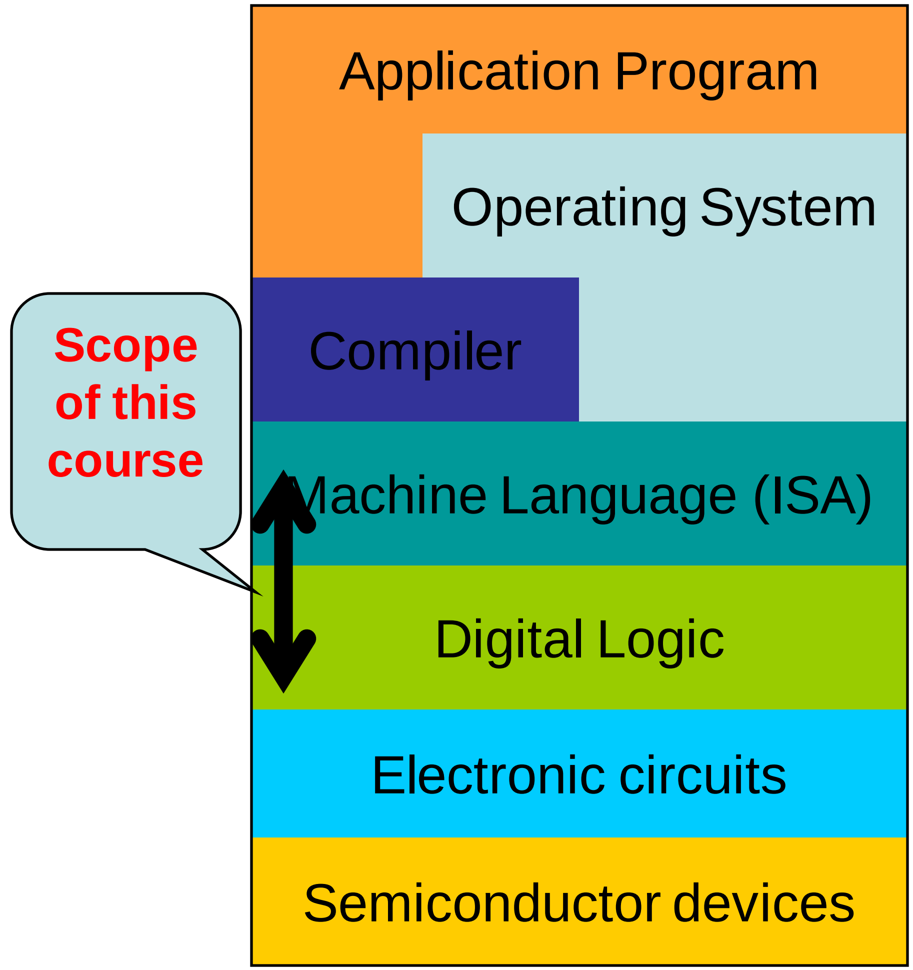
- Prerequisites
  - Digital Design – gates, logic, memory, organization
  - Programming Languages – high-level language down to machine language interface or instruction set architecture (ISA)
- This course – puts it all together
  - Implement the logic that provides ISA interface
  - Must do datapath and control, but no magic
  - Manage tremendous complexity with abstraction
- Follow-on courses explore trade-offs
  - 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?

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!

Some History

Date	Event	Comments
1939	First digital computer	John Atanasoff (UW PhD ' 30)
1947	1st transistor	Bell Labs
1958	1st IC	Jack Kilby (MSEE ' 50) @TI Winner of 2000 Nobel prize
1971	1st microprocessor	Intel
1974	Intel 4004	2300 transistors
1978	Intel 8086	29K transistors
1989	Intel 80486	1.M transistors, pipelined
1995	Intel Pentium Pro	5.5M transistors
2005	Intel Montecito	1B transistors

Performance Growth

Unmatched by any other industry !

[John Crawford, Intel]

- Doubling every 18 months (1982-1996): 800x
  - Cars travel at 44,000 mph and get 16,000 mpg
  - Air travel: LA to NY in 22 seconds (MACH 800)
  - Wheat yield: 80,000 bushels per acre

- Doubling every 24 months (1971-1996): 9,000x
  - Cars travel at 600,000 mph, get 150,000 mpg
  - Air travel: LA to NY in 2 seconds (MACH 9,000)
  - Wheat yield: 900,000 bushels per acre

Technology Push

- Technology advances at varying rates
  - E.g. DRAM capacity increases at 60%/year
  - But DRAM speed only improves 10%/year
  - Creates gap with processor frequency!
- Inflection points
  - Crossover causes rapid change
  - E.g. enough devices for multicore processor (2001)
- Current issues causing an "inflection point"
  - Power consumption
  - Reliability
  - Variability

Application Pull

- Corollary to Moore's Law:
  Cost halves every two years

  In a decade you can buy a computer for less than its sales tax today. –Jim Gray

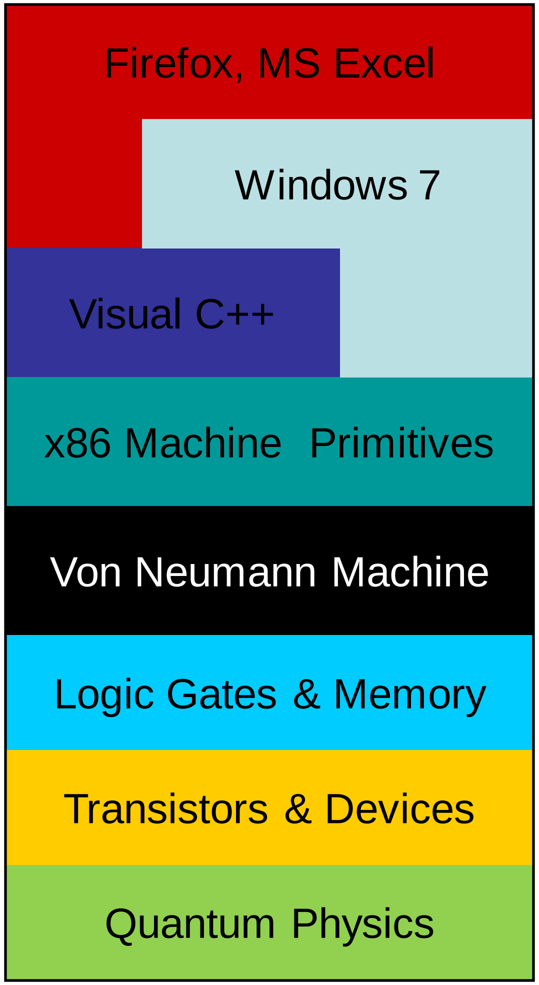
- Computers cost-effective for
  - National security – weapons design
  - Enterprise computing – banking
  - Departmental computing – computer-aided design
  - Personal computer – spreadsheets, email, web
  - Pervasive computing – prescription drug labels

Application Pull

- What about the future?
- Must dream up applications that are not cost-effective today
  - Virtual reality
  - Telepresence
  - Mobile applications
  - Sensing, analyzing, actuating in real-world environments
- This is your job

What’s the Big Deal?

- Tower of abstraction
- Complex interfaces implemented by layers below
- Abstraction hides detail
- Hundreds of engineers build one product
- Complexity unmanageable otherwise



Bottom Line

- Designers must know BOTH software and hardware
- Both contribute to layers of abstraction
- IC costs and performance
- Compilers and Operating Systems

About This Course

- Course Textbook
  - D.A. Patterson and J.L. Hennessy, *Computer Architecture and Design: The Hardware/Software Interface*, 4th edition, Elsevier/Morgan Kauffman.
  - 3rd edition OK if 4th edition not available.
- Homework
  - Couple of homework assignments, unequally weighted
- Tests
  - Periodic tests will be conducted (some are scheduled and some surprise)

About This Course

- Project
  - Implement processor for MNIT-CS13 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

Thank You