Introduction to Computer Architecture
EECS 645

David Andrews

Rm 527 JBHT

Dandrews@uark.edu

CSCE University of Arkansas

Course Introduction

Course Programmatics

- Grading
- Assignments
- Exams

What Is Computer Architecture

- Is It Science or Art ?

Programmatics

- Grading:
  - 2 Midterm Exams 300 pts
  - Final Exam 100 pts
  - Homework 100 pts
- Homework is important !
  - Where you mistakes and learn
  - Academic Honesty: Do Your Own Work
    - 1st Violation Will Result in 0 for all parties
    - 2nd Violation Will Result in "F" for Class, possible additional actions by Department, College, Univeristy

What is Computer Architecture ?

- **Computer Architecture:** An abstract computational model and it's realization.
  - Has evolved:
    - 1st *CPU Design*: Instruction Set Architecture (ISA) was abstract computational model (ACT)
    - 2nd *Computer Organization:* CPU, Cache,Memory, Bus, Programming Language + Operating System:= ACT
    - *3rd Parallel System Design:* Parallel Boards, interconnects. PL + OS + Middleware
  - More Than "Hardware Design". Abstractions of physical structures, abstractions of abstractions give modern ACT

What is a Computational Model

- Browne Defined New Computational Model

  Computer Vol. 17, pp. 83-87, July 1984

  - *Primitive Units* of computations or basic actions of the computer (the data types and operations defined by instruction set)
  - *Control mechanism* that selects primitive units into which the problem has been partitioned
  - *Data mechanism (*how data in memory is accessed); definition of address spaces available to the computation
  - Modes and patterns of *communication* among computers working in parallel, so they can exchange information
  - *Synchronization* mechanisms to ensure information arrives at correct time

A Hierarchy of Computational Models

- Tanenbaum Defined A Hierarchy of Abstractions
  - The *algorithm*, or high level plan of attack
  - The *high-level language* in which the user writes a program for the computer
  - The *operating system*, a program that allocates the resources of the system, using its own abstract picture of the system
  - The *physical machine* architecture, represented by it's instruction set

Computer Architecture

- Computer Architecture is:
  1. Defining and Creating Abstract Computational Model (What):
     - Examples include Data Flow, Quantum Computing, SIMD, MIMD, Stack
     - Programmers See:
       - ISA
       - Operating System
       - Middleware
  2. Computer Organization (Where):
     - High Level Organization of Abstract Machine Model
       - Memory, CPU, Bus, I/O Subsystems
  3. Hardware Realization (How):
     - Internal realization of the blocks defined in organizaiotn
     - ALU's, Logic, Registers

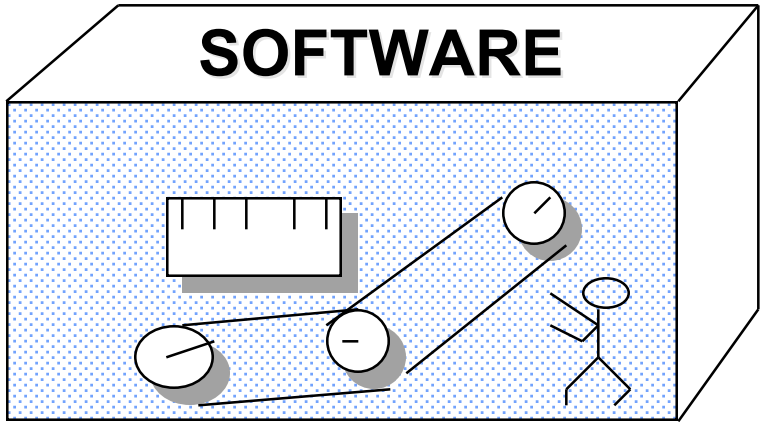
Computer Architecture

- The Computer Architect Must Understand all levels of Computational Models In Order to Do A Good Job……
  - What is it that the machine I am to design is trying to do ?
  - Is This a Good Abstract Model ?
  - How Do I support Efficient Interactions between Abstract Levels ?
    - Fast Communications, Disk Access, Scheduling, Arithmetic Operations
  - What Technology Do I Have Available ?
  - How Much Money Can I spend ?
    - We Live in the Real World !

From Book: Computer Architecture Is …

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, Blaaw, and Brooks, 1964



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:
  Computer Architecture Course = Design of CPU, memory system, I/O system, Multiprocessors
- 2000 - ?

  Computer Architecture Course= Systems (Desktop, Embedded, Signal Processing, Application Specific)….

Some Observations….

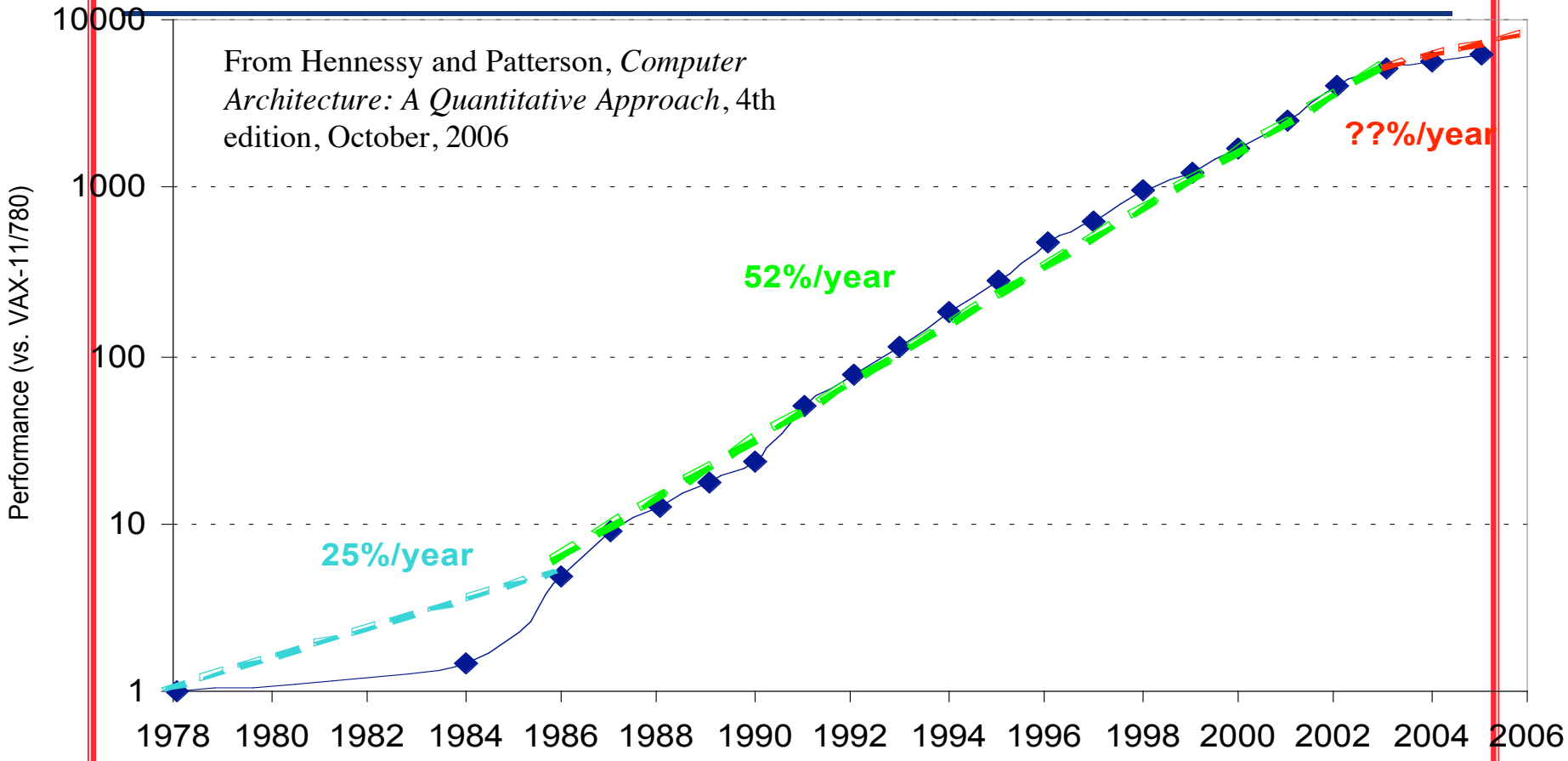
- Accepted Basic Von Neumann 1940's Stored Program Model
  - Model developed for scientific applications
  - Evolved into Desktop PC Platform
  - Extended To Parallel Computational Model
    - Flynn's Classification
      - SISD Single Instruction, Single Data (Basic Model)
      - MIMD Multiple Instruction, Multiple Data (Hypercubes etc)
      - SIMD Single Instruction, Multiple Data (Vector, Arrays)
      - MISD Multiple Instrucdtion, Single Data (Systolic ?)

Crossroads: Conventional Wisdom in Comp. Arch

- Old Conventional Wisdom: Power is free, Transistors expensive
- New Conventional Wisdom: "Power wall" Power expensive, Xtors free (Can put more on chip than can afford to turn on)
- Old CW: Sufficiently increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, VLIW, …)
- New CW: "ILP wall" law of diminishing returns on more HW for ILP
- Old CW: Multiplies are slow, Memory access is fast
- New CW: "Memory wall" Memory slow, multiplies fast (200 clock cycles to DRAM memory, 4 clocks for multiply)
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall
  - Uniprocessor performance now 2X / 5(?) yrs

⇒ Sea change in chip design: multiple "cores"
(2X processors per chip / ~ 2 years)
- More simpler processors are more power efficient

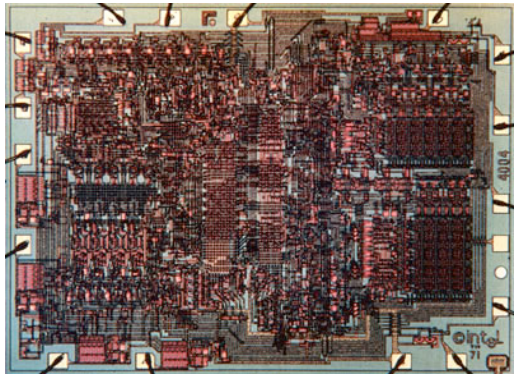
Crossroads: Uniprocessor Performance

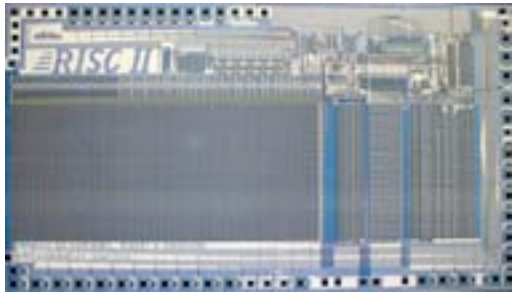


- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**
- **RISC + x86: ??%/year 2002 to present**

Sea Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm^2 chip
- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm^2 chip
- 125 mm^2 chip, 0.065 micron CMOS = 2312 RISC II+FPU+Icache+Dcache
  - RISC II shrinks to ~ 0.02 mm^2 at 65 nm
  - Caches via DRAM or 1 transistor SRAM (www.t-ram.com) ?
  - Proximity Communication via capacitive coupling at > 1 TB/s ? (Ivan Sutherland @ Sun / Berkeley)





- **Processor is the new transistor?**

Déjà vu all over again?

- Multiprocessors imminent in 1970s, '80s, '90s, …
- "… today's processors … are nearing an impasse as technologies approach the speed of light.."

  David Mitchell, *The Transputer: The Time Is Now* (1989)
- Transputer was premature
  ⇒ Custom multiprocessors strove to lead uniprocessors
  ⇒ Procrastination rewarded: 2X seq. perf. / 1.5 years
- "We are dedicating all of our future product development to multicore designs. … This is a sea change in computing"

  Paul Otellini, President, Intel (2004)
- Difference is all microprocessor companies switch to multiprocessors (AMD, Intel, IBM, Sun; all new Apples 2 CPUs)
  ⇒ Procrastination penalized: 2X sequential perf. / 5 yrs
  ⇒ Biggest programming challenge: 1 to 2 CPUs

Problems with Sea Change

- Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, … not ready to supply Thread Level Parallelism or Data Level Parallelism for 1000 CPUs / chip,
- Architectures not ready for 1000 CPUs / chip
  - Unlike Instruction Level Parallelism, cannot be solved by just by computer architects and compiler writers alone, but also cannot be solved *without* participation of computer architects
- 4th Edition of textbook Computer Architecture: A Quantitative Approach explores shift from Instruction Level Parallelism to Thread Level Parallelism / Data Level Parallelism

Reading Assignment

Read Background Material in Chapter 1.

Ch 1.4 - 1.6 Trends
Ch 1.8 - 1.9 Measurement and Analysis