Introduction

- Rapidly changing field:
  - vacuum tube -> transistor -> IC -> VLSI (see section 1.4)
  - memory capacity and processor speed is doubling every 1.5 years:
- Things you'll be learning:
  - how computers work, a basic foundation
  - how to analyze their performance (or how not to!)
  - issues affecting design of modern processors (caches, pipelines)
- Why learn this stuff?
  - You want to design state-of-art system
  - you want to call yourself a "computer scientist or engineer"
  - you want to build software people use (need performance)
  - you need to make a purchasing decision or offer "expert" advice

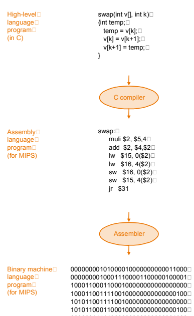
What is a computer?

- Components:
  - input (mouse, keyboard)
  - output (display, printer)
  - memory (disk drives, DRAM, SRAM, CD)
  - network
- Our primary focus:
  - understanding performance
  - the processor (datapath and control)
  - implemented using millions of transistors
  - impossible to understand by looking at each transistor
  - we need an abstraction

Abstraction

- Delving into the depths reveals more information
- An abstraction omits unneeded detail, helps us cope with complexity

What are some of the details that appear in these familiar abstractions?



What is Computer Architecture?

- A programmer's view of machine
- What does it include?

- *What is Computer Organization, Structure, and Function?*

Instruction Set Architecture

- A very important abstraction
  - interface between hardware and low-level software
  - standardizes instructions, machine language bit patterns, etc.
  - advantage: *different implementations of the same architecture*
  - disadvantage: *sometimes prevents using new innovations*

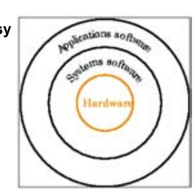
  True or False: Binary compatibility is extraordinarily important?

- Modern instruction set architectures:
  - 80x86/Pentium/K6, PowerPC, DEC Alpha, MIPS, SPARC, HP

- Historical Perspective

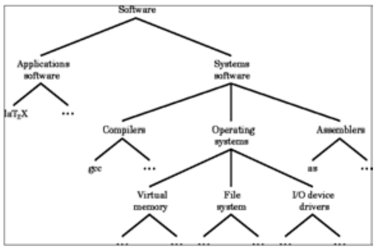
A View of Hardware/Software Hierarchy

- Hardware and software are layered
- Some functions can be moved back and forth
- System software is a collection of programs
  - OS, compilers are some examples
  - It makes job of individuals user easy
- Application software programs
  - Used by many users



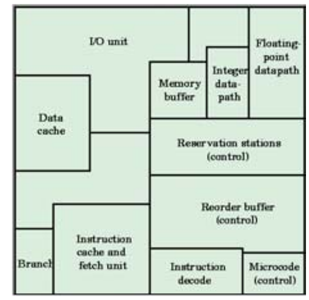
View of Software

- Software means different things to different people

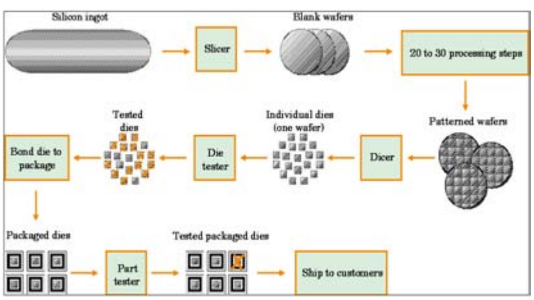


Internal Structure of a Processor Chip

- Major Components
  - Instruction cache
  - Instruction Fetch
  - Instruction Decode
  - Control/Microcode
  - Register File
  - Data path
  - Data Cache
  - I/O Unit
  - Memory Buffer
  - Advanced Units



Chip Manufacturing Process



Where we are headed

- Performance issues (Chapter 2) *vocabulary and motivation*
- A specific instruction set architecture (Chapter 3)
- Other instruction set example (From Outside)
- Arithmetic and how to build an ALU (Chapter 4)
- Constructing a processor to execute our instructions (Chapter 5)
- Pipelining to improve performance (Chapter 6)
- Memory: caches and virtual memory (Chapter 7)
- I/O (Chapter 8)

Key to a good grade: attending classes, reading the book!

Historical Perspective

- 1642 Pascal: Mechanical Computer
- 1671: Gottfried Leibniz ADD/SUB/MUL/DIV
- 1801: Automatic Control of Weaving Process
- 1827 The Difference Engine by Charles Babbage
- 1936: Zuse Z1: electromechanical computers
- 1941: Zuse Z2
- 1943: Zuse Z3
- 1944: Aiken: Ark 1 at Harvard
- 1942-45: ABC at Iowa State (Atanasoff-Berry Computer)
- 1946: ENIAC: Eckert and Mauchley: Vacuum Tube
- 1945 EDVAC by von-Neumann machine, father of modern computing

Difference Engine

- Based on computing differences, a finite n-th order polynomial can be differentiated n times, which can be represented by a difference
- For example
- $y = \text{Sin}(x) = x - x^3/3!$
- To compute value of sin(x) at x0, x1, x2, x3, x4, x5, x6…. such that difference in two consecutive values is small, we can calculate y0, y1, y2, and y3 by hand and call them $\Delta^0 y0$, $\Delta^0 y1$, $\Delta^0 y2$, and $\Delta^0 y3$
- Then first order difference is $\Delta^1 y0 = y1-y0$; $\Delta^1 y1 = y2-y1$; $\Delta^1 y2 = y3-y2$;
- Second order difference is $\Delta^2 y0 = \Delta^1 y1 - \Delta^1 y0 = y2-2y1+y0$; and so on
- Third order difference is $\Delta^3 y0 = \Delta^2 y1 - \Delta^2 y0 = y3-3y2+3y1-y0$
- $\Delta^4 y0 = 0$
- Using this we can recursively compute $\Delta^3 y1$, $\Delta^2 y1$, and $\Delta^1 y1$, and $\Delta^0 y1$
- And so on….