Computer Architecture
(CSC-3501)
Lecture 1
(15 Jan 2008)

Seung-Jong Park (Jay)
http://www.csc.lsu.edu/~sjpark

Handouts

- Class information
  - http://www.csc.lsu.edu/~sjpark/cs3501/overview.html
- Schedule (check online frequently)
  - http://www.csc.lsu.edu/~sjpark/cs3501/schedule.html

Goals of CSC-3501

- Principles on Computer Architecture
  - From hardware to software
- Topics
  - Computer history
  - Boolean algebra and digital logic
  - Components of computer
    - CPU, BUS, Clock, I/O, Memory
  - Instruction set architecture
  - Memory & Cache
  - I/O & Storage
  - CISC vs RISC
  - Performance measurement
  - Networks

Administrative Information

- Seung-Jong Park
  - Room 289, Coates Hall
  - Email : sjpark at csc.lsu.edu
  - Office hours: Tue, Thr: 3:00pm-4:30pm
- Class website
  - http://www.csc.lsu.edu/~sjpark
- TA
  - TBA
  - TA hour(2 hours per week officially, additional hours need appointment with TA) and place will be announced soon

Policies

- Lecture slides will be available on website before the day of class
- Lecture slides are NOT comprehensive – students are expected to attend classes to "fill in" information
- Miss classes at your own risk. Professor and TAs will NOT be responsible for any information you might not have because of a missed class
- Zero tolerance policy toward cheating

Textbook

- Textbook
  - Computer Organization and Architecture
    - written by Linda Null and Julia Lobur
- Recommended reading texts
  - Structured Computer Organization
    - Written by Tanenbaum
- Additional reading list will be available on website

Grading

- Exams
  - 1 midterm (1 x 25% = 25%)
  - 1 final (1 x 30% = 30%)
- Homework
  - 10 Homework (10 x 3% = 30%)
- Quiz
  - 10 Pop quizzes (10 x 1% = 10%)
    - Each quiz has 3 – 5 questions to check whether students understand the content s of class
- Class points
  - Attendance & Interaction in class (1 x 5% = 5%)

Chapter 1

The Goal of Chapter 1

- Know the difference between computer organization and computer architecture.
- Appreciate the evolution of computers.
- Understand the computer as a layered system.
- Be able to explain the von Neumann architecture and the function of basic computer components.

Why do we learn this stuff?

- You want to call yourself a “computer scientist”
- You want to build software which people use (need performance)
- You need to make a purchasing decision or offer “expert” advice

- Both Hardware and Software affect performance:
  - Algorithm determines number of source-level statements
  - Language/Compiler/Architecture determine machine instructions
  - Processor/Memory determine how fast instructions are executed

☺ **AT LEAST, Know how to order components of a computer when you order it (Go to Dell.com)**

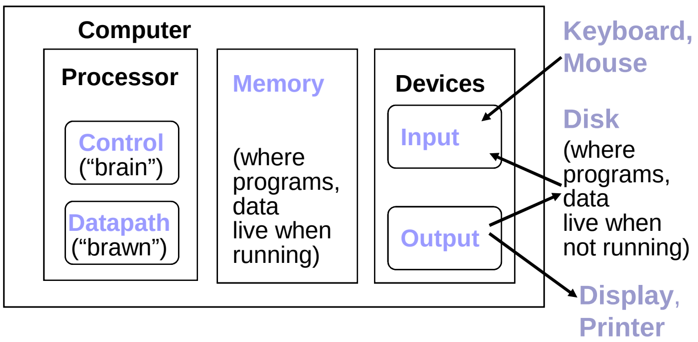
Overview

- Computer organization
  - Encompasses all ***physical*** aspects of computer systems.
  - E.g., circuit design, control signals, memory types.
  - *How does a computer work?*
- Computer architecture
  - ***Logical*** aspects of system implementation as seen by the programmer.
  - E.g., instruction sets, instruction formats, data types, addressing modes.
  - *How do I design a computer?*

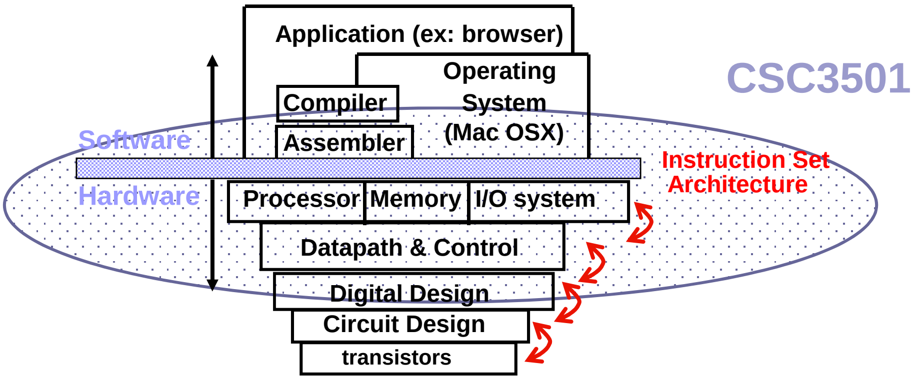
- BUT, there is no clear distinction between matters related to computer organization and matters relevant to computer architecture.

Computer Organization

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



Computer Architecture



* Coordination of many
  levels (layers) of abstraction

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:
  - IA-32, PowerPC, MIPS, SPARC, ARM, and others
- Execute binary executable codes at different machines

How do computers work?

- Need to understand abstractions such as:
  - Applications software
  - Systems software
  - Assembly Language
  - Machine Language
  - Architectural Issues: i.e., Caches, Virtual Memory, Pipelining
  - Sequential logic, finite state machines
  - Combinational logic, arithmetic circuits
  - Boolean logic, 1s and 0s
  - Transistors used to build logic gates (CMOS)
  - Semiconductors/Silicon used to build transistors
  - Properties of atoms, electrons, and quantum dynamics

- So much to learn!

Different Levels of Representation

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g.,MIPS)

Assembler

Machine Language Program (binary)

Machine Interpretation

Hardware Architecture Description (Logic, Logisim, etc.)

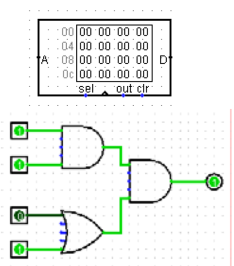
Architecture Implementation

Logic Circuit Description (Logisim, etc.)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

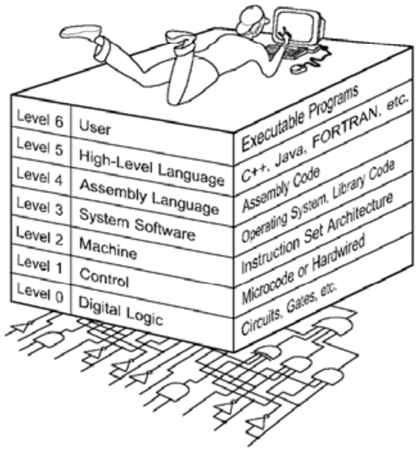
```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



Computer Level Hierarchy

- **Each virtual machine layer is an abstraction of the level below it.**
- **The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required.**
- **Computer circuits ultimately carry out the work.**



- Level 6: The User Level
  - Program execution and user interface level.
  - The level with which we are most familiar.
- Level 5: High-Level Language Level
  - The level with which we interact when we write programs in languages such as C, Pascal, Lisp, and Java.
- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.
- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Assembly language instructions often pass through Level 3 without modification.
- Level 2: Machine Level
  - Also known as the Instruction Set Architecture (ISA) Level.
  - Consists of instructions that are particular to the architecture of the machine.
  - Programs written in machine language need no compilers, interpreters, or assemblers.

- Level 1: Control Level
  - A *control unit* decodes and executes instructions and moves data through the system.
  - Control units can be *microprogrammed* or *hardwired*.
  - A microprogram is a program written in a low-level language that is implemented by the hardware.
  - Hardwired control units consist of hardware that directly executes machine instructions.
- Level 0: Digital Logic Level
  - This level is where we find digital circuits (the chips).
  - Digital circuits consist of gates and wires.
  - These components implement the mathematical logic of all other levels.

Historical Development

- To fully appreciate the computers of today, it is helpful to understand how things got the way they are.
- The evolution of computing machinery has taken place over several centuries.
- In modern times computer evolution is usually classified into **four generations** according to the salient technology of the era.

Generation Zero

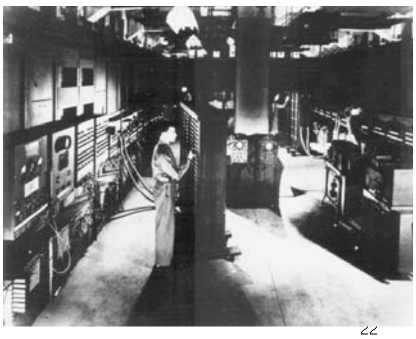
- Mechanical Calculating Machines (1642 - 1945)
  - Calculating Clock - Wilhelm Schickard (1592 - 1635).

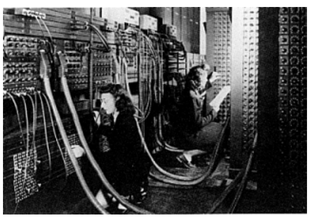
  - Pascaline - Blaise Pascal (1623 - 1662).
  - Difference Engine - Charles Babbage (1791 - 1871), also designed but never built the Analytical Engine.

  - Punched card tabulating machines - Herman (1860 - 1929).

1st Generation

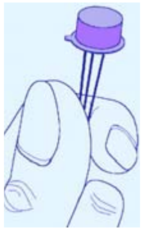
- Vacuum Tube Computers (1945 - 1953)
  - Atanasoff Berry Computer (1937 - 1938) solved systems of linear equations.
  - John Atanasoff and Clifford Berry of Iowa State University
  - Electronic Numerical Integrator and Computer (ENIAC)
    - The first *general-purpose* computer.
    - John Mauchly and J. Presper Eckert
    - University of Pennsylvania, 1946





2nd Generation

- Transistorized Computers (1954 - 1965)
  - IBM 7094 (scientific) and 1401 (business)
  - Digital Equipment Corporation (DEC) PDP-1
  - Univac 1100
  - Control Data Corporation 1604.
  - . . . and many others.

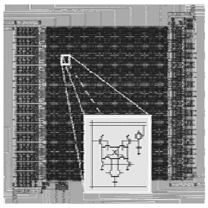


3rd Generation

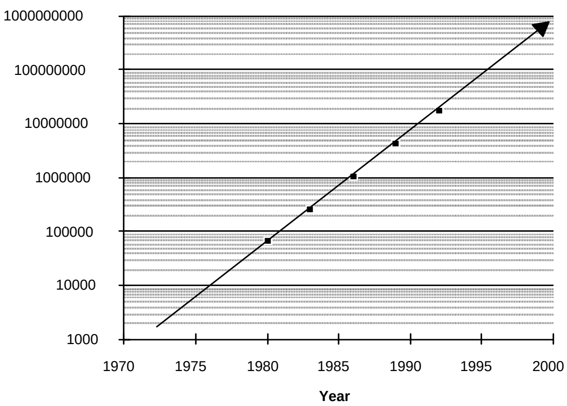
- Integrated Circuit Computers (1965 - 1980)
  - IBM 360
  - DEC PDP-8 and PDP-11
  - Cray-1 supercomputer
  - . . . and many others.
- By this time, IBM had gained overwhelming dominance in the industry.
  - Computer manufacturers of this era were characterized as IBM and the BUNCH (Burroughs, Unisys, NCR, Control Data, and Honeywell).

4th Generation

- VLSI Computers (1980 - ????)
  - Very large scale integrated circuits (VLSI) have more than 10,000 components per chip.
  - Enabled the creation of microprocessors.
  - The first was the 4-bit Intel 4004.
  - Later versions, such as the 8080, 8086, and 8088 spawned the idea of "personal computing."



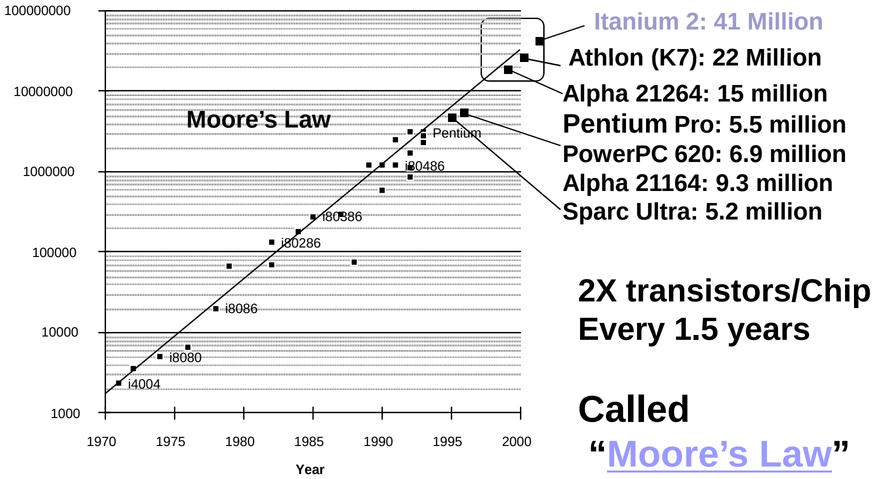
Technology Trends: Memory Capacity (Single-Chip DRAM)size



year	size (Mbit)
1980	0.0625
1983	0.25
1986	1
1989	4
1992	16
1996	64
1998	128
2000	256
2002	512

- **Now 1.4X/yr, or 2X every 2 years.**
- **8000X since 1980!**

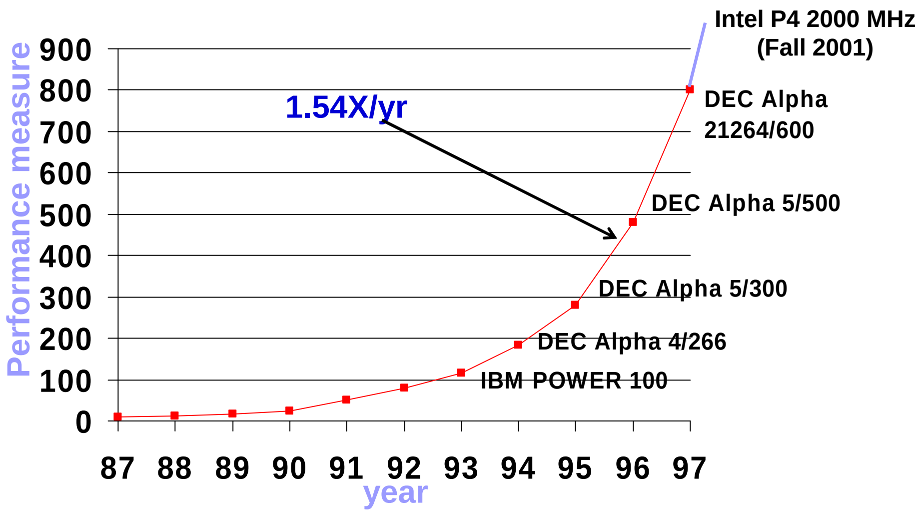
Technology Trends: Microprocessor Complexity



2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Technology Trends: Processor Performance



Computer Technology - Dramatic Change!

- Memory
  - DRAM capacity: 2x / 2 years (since '96);
    64x size improvement in last decade.
- Processor
  - Speed 2x / 1.5 years (since '85);
    100X performance in last decade.
- Disk
  - Capacity: 2x / 1 year (since '97)
    250X size in last decade.

Computer Technology - Dramatic Change!

- State-of-the-art PC when you graduate:
  (at least…)
  - Processor clock speed: 5000 MegaHertz (5.0 GigaHertz)
  - Memory capacity: 8000 MegaBytes (8.0 GigaBytes)
  - Disk capacity: 2000 GigaBytes (2.0 TeraBytes)
  - New units! Mega => Giga, Giga => Tera

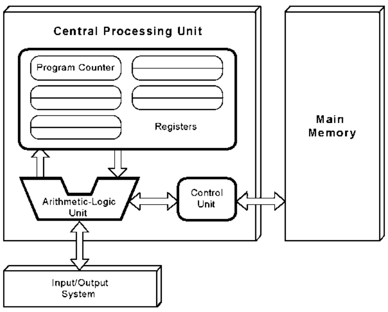
    (Tera => Peta, Peta => Exa, Exa => Zetta
    Zetta => Yotta = 10^{24})

The Von Neumann Model

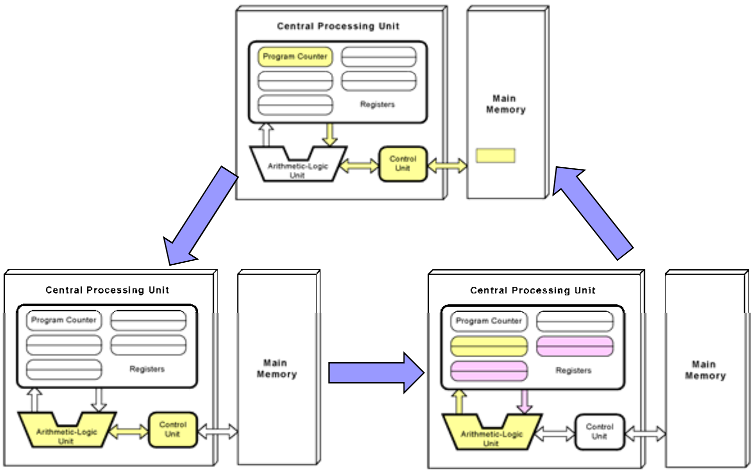
- Non-Von Neumann Model
  - On the ENIAC, all programming was done at the digital logic level.
  - Programming the computer involved moving plugs and wires.
  - A different hardware configuration was needed to solve every unique problem type.
  - **Configuring the ENIAC to solve a "simple" problem required many days labor by skilled technicians. => So painful jobs**
- Von Neumann Model
  - Inventors of the ENIAC, John Mauchley and J. Presper Eckert, conceived of a computer that could store instructions in memory.
  - The invention of this idea has since been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
  - Stored-program computers have become known as von Neumann Architecture systems.

The Von Neumann Model

- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.
    - This single path is known as the *von Neumann bottleneck*.
- **These computers employ a fetch-decode-execute cycle to run programs as follows .**



Cycles of Von Neumann Model



Alternatives of Non-Von Neumann

- In the late 1960s, high-performance computer systems were equipped with dual processors to increase computational throughput.
- In the 1970s supercomputer systems were introduced with 32 processors.
- Supercomputers with 1,000 processors were built in the 1980s.
- In 1999, IBM announced its Blue Gene system containing over 1 million processors.
- Parallel processing is only one method of providing increased computational power.
- More radical systems have reinvented the fundamental concepts of computation.
- These advanced systems include genetic computers, quantum computers, and dataflow systems.
- At this point, it is unclear whether any of these systems will provide the basis for the next generation of computers.