Computer Architecture

Introduction

What will we Learn?

- Computer Architecture
  - The science and art of designing the hardware/software interface and designing, selecting, and interconnecting hardware components to create a computing system that meets functionality requirements, performance, energy consumption, cost, and other specific goals.

- Computer Architecture =

  Instruction Set Architecture + Computer Organization

What is “Computer Architecture” ?

- Computer Architecture =

  Instruction Set Architecture + Computer Organization
- Instruction Set Architecture (ISA)
  - WHAT the computer does (logical view)
- Computer Organization
  - HOW the ISA is implemented (physical view)
- We will study both in this course

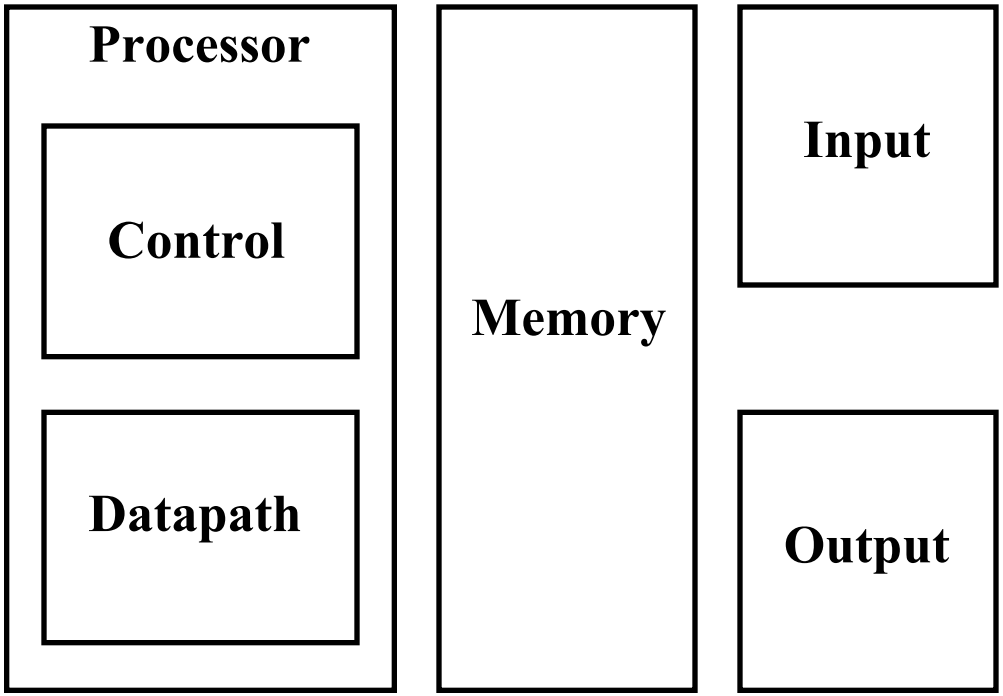
Instruction Set Architecture

- Instruction set architecture is the attributes of a computing system as seen by the assembly language programmer or compiler.
  - Instruction Set (what operations can be performed?)
  - Instruction Format (how are instructions specified?)
  - Data storage (where is data located?)
  - Addressing Modes (how is data accessed?)
  - Exceptional Conditions (what happens if something goes wrong?)

Machine Organization

- Machine organization is the view of the computer that is seen by the logic designer. This includes
  - Capabilities & performance characteristics of functional units (e.g., registers, ALU, shifters, etc.).
  - Ways in which these components are interconnected
  - How information flows between components
  - Logic and means by which such information flow is controlled
  - Coordination of functional units

Components of a Computer



Components of a Computer

- Datapath - performs arithmetic and logic operations
  - e.g., adders, multipliers, shifters
- Memory - holds data and instructions
  - e.g., cache, main memory, disk
- Input - sends data to the computer
  - e.g., keyboard, mouse
- Output - gets data from the computer
  - e.g., screen, sound card
- Control - gives directions to the other components
  - e.g., bus controller, memory interface unit

Information in a computer -- Instructions

- Instructions specify commands to:
  - Transfer information within a computer
    - (e.g., from memory to ALU)
  - Transfer of information between the computer and I/O devices
    - (e.g., from keyboard to computer, or computer to printer)
  - Perform arithmetic and logic operations
    - (e.g., Add two numbers, Perform a logical AND).

Information in a computer -- Instructions

- A sequence of instructions to perform a task is called a program, which is stored in the memory.

- Processor fetches instructions from the memory and performs the operations stated in those instructions.

- What do the instructions operate upon?

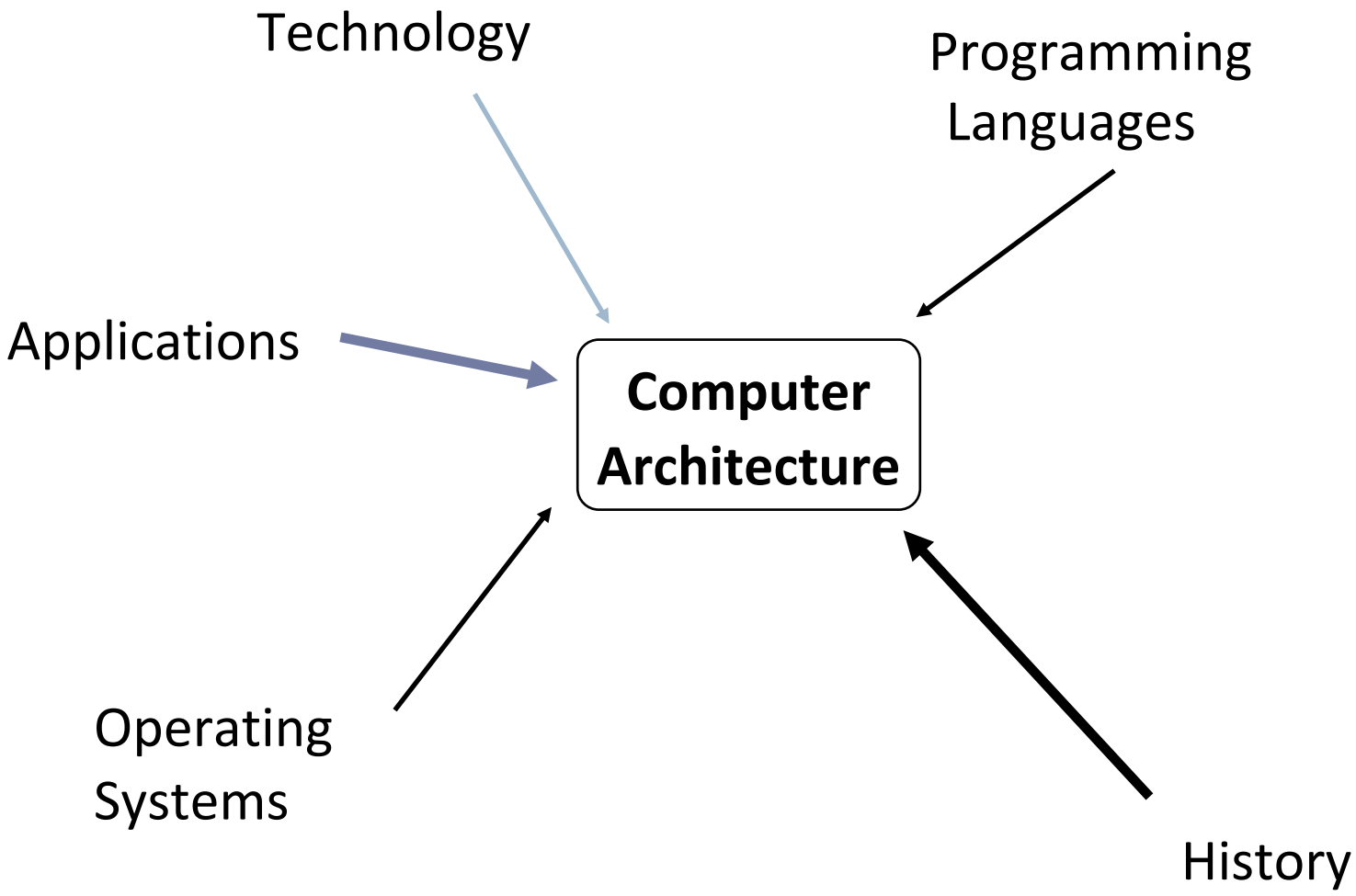
Information in a computer -- Data

- Data are the “operands” upon which instructions operate.
- Data could be:
  - Numbers,
  - Encoded characters.
- Data, in a broad sense means any digital information.
- Computers use data that is encoded as a string of binary digits called bits.

Classes of Computers

- Desktop / Notebook Computers
  - Low-end systems, high performance workstations.
  - Subject to cost/performance tradeoff
- Server Computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded Computers
  - Hidden as components of systems
  - Minimize memory and power. Often not programmable

Forces on Computer Architecture

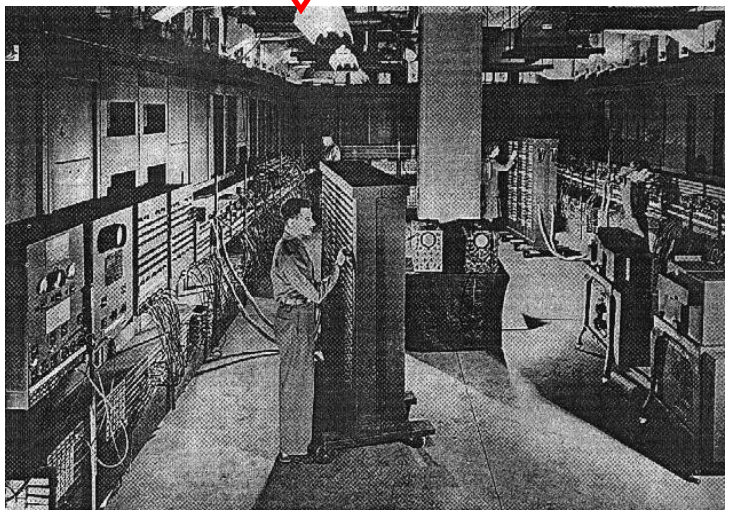


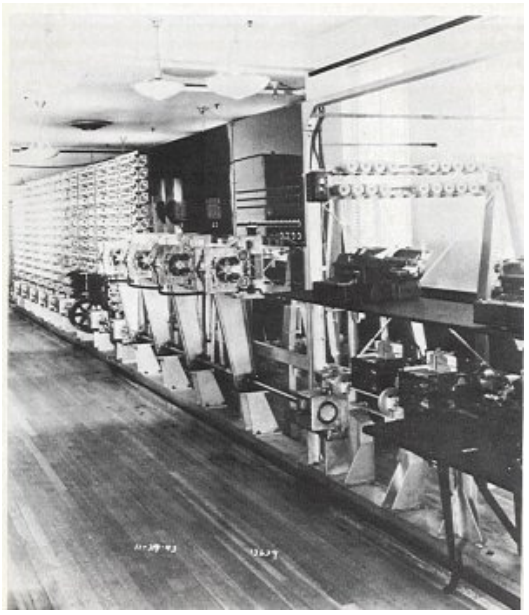
History

"Big Iron" Computers:

Used vacuum tubes, electric relays and bulk magnetic storage devices. No microprocessors. No memory.

Example: ENIAC (1945), IBM Mark 1 (1944)





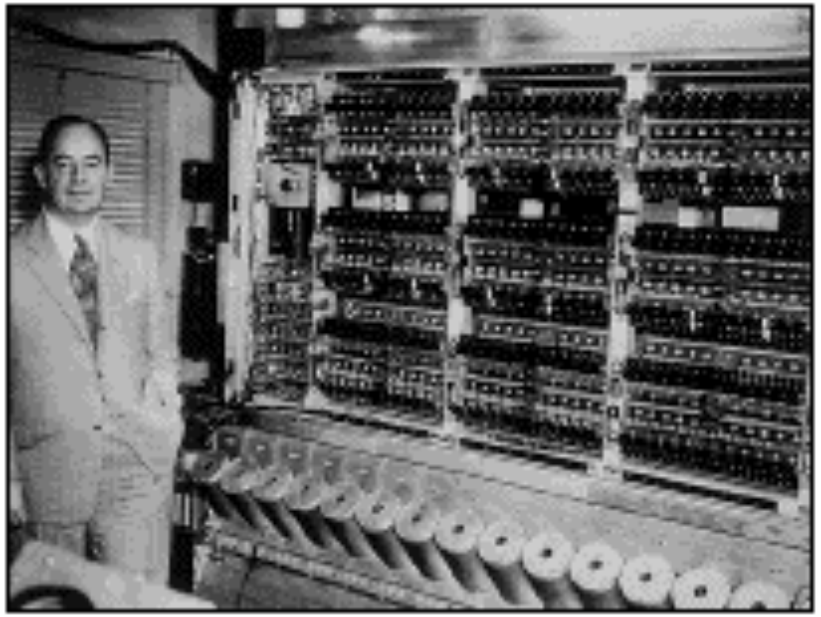
History

Von Newmann:

Invented EDSAC (1949).

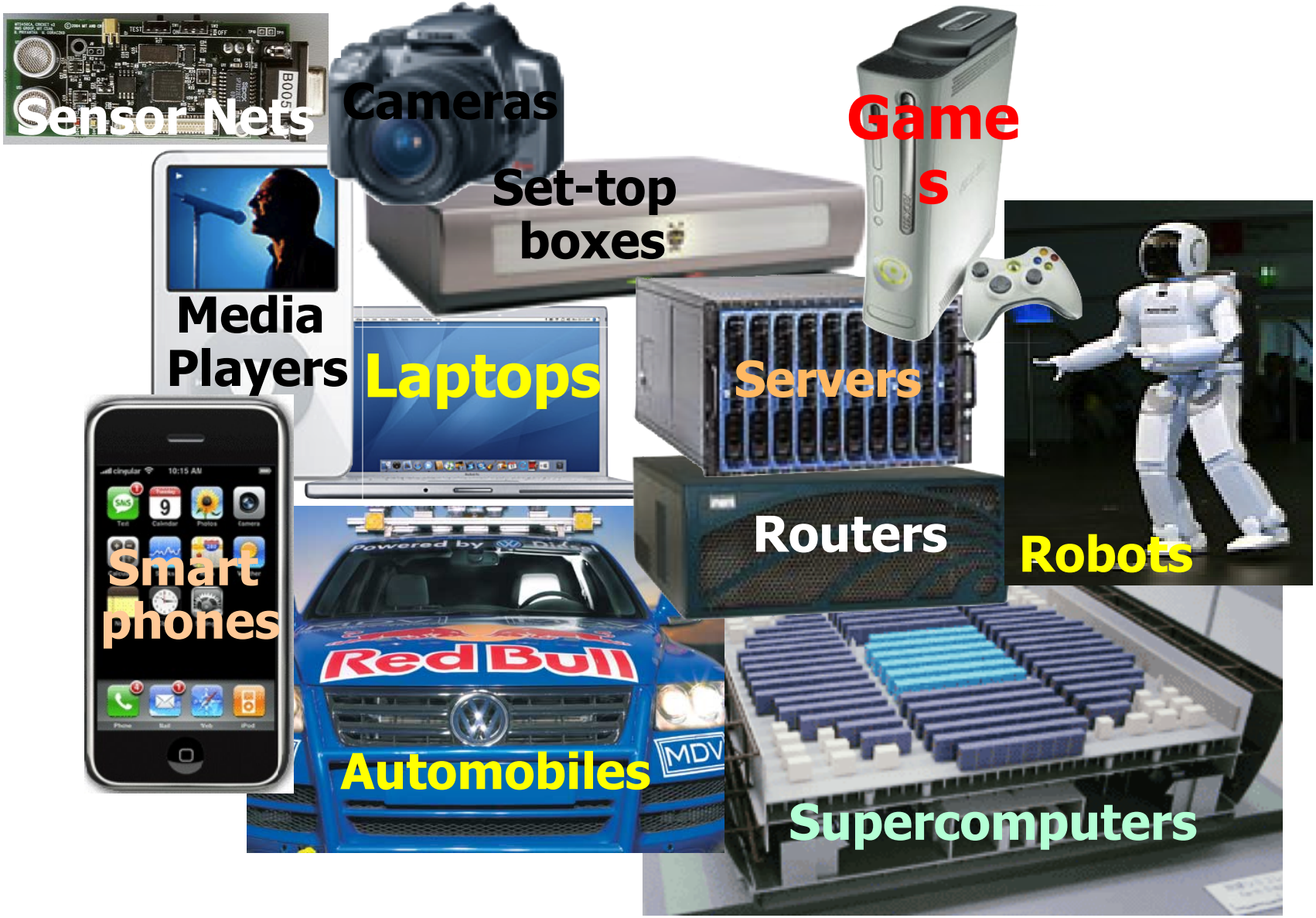
First Stored Program Computer.

Uses Memory.



Importance: We are still using the same basic design.

Computing Devices Now



Why Study Computer Architecture

- As computer scientists, software engineers, and sophisticated users, understanding how computers work is essential
  - To understand the components that make up a modern computing system
  - To understand how they impact a system's performance, efficiency, and usefulness
  - To be able to harness, modify, and extend them to solve problems effectively

Acknowledgements

- These slides contain material developed and copyright by:
  - Krste Asanovic (UCB) (Course# 152, Spring 2012)
  - James Hoe (CMU) (Course# 18-447, Spring 2011)
  - Li-Shiuan Peh (MIT) (Course# 6.823, Fall 2012)
  - Sudhakar Yalamanchili (GATECH) (Course# ECE3056, Fall 2012)
  - Amirali Baniasadi (UVIC) (Course# CENG450, Summer 2012)

The End