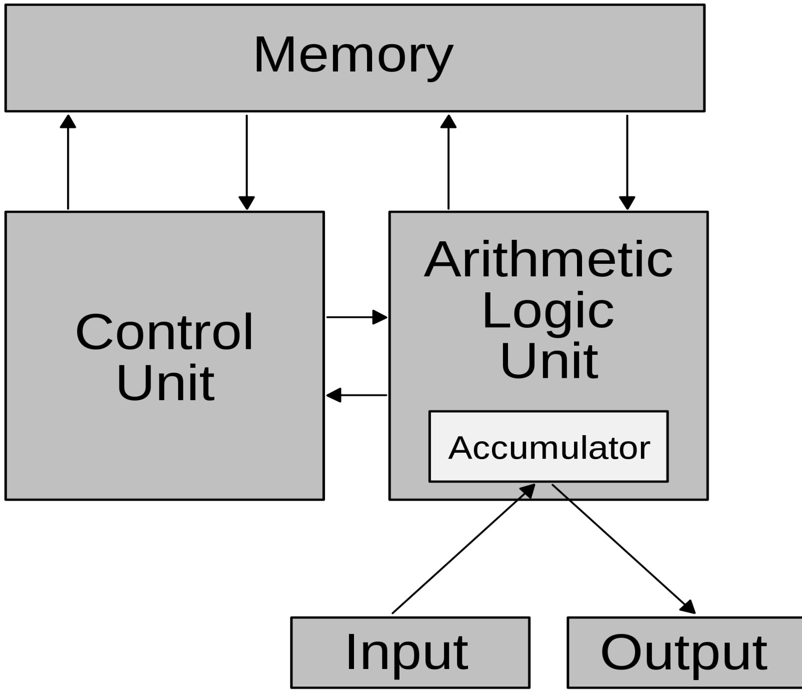
CSL7070: Computer Architecture
Lecture 02, 13th January 2022

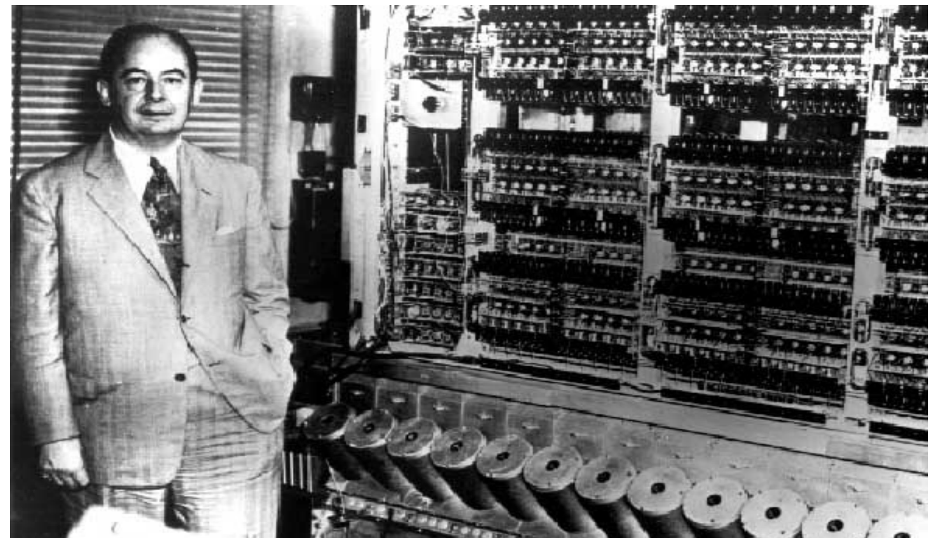
Dip Sankar Banerjee

|| त्वं ज्ञानमयो विज्ञानमयोऽसि ||

Indian Institute of Technology, Jodhpur
January-April 2022

Von-Neumann Architecture



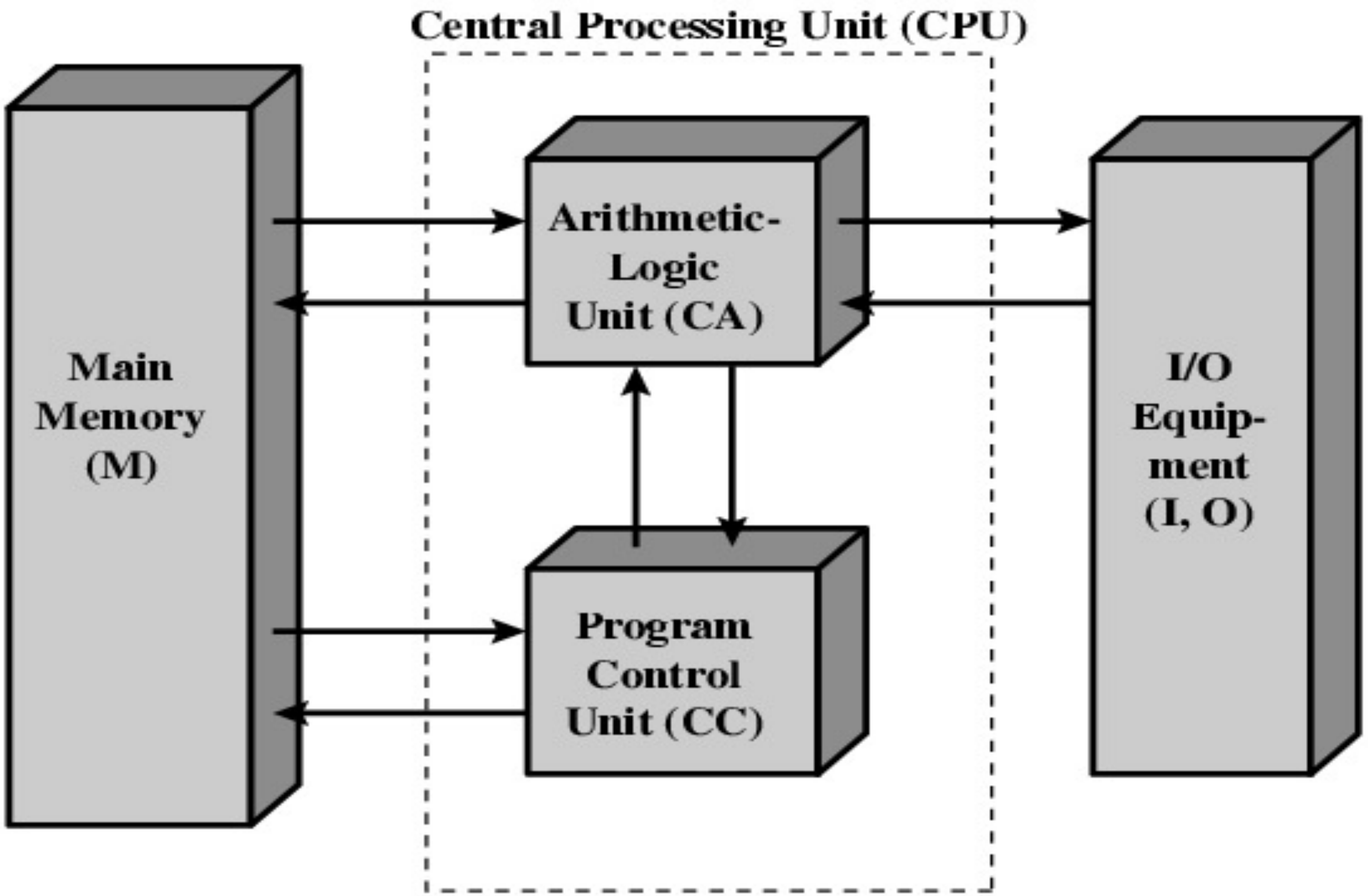


Source: http://www.gap-system.org/~history/ PictDisplay/Von_Neumann.html

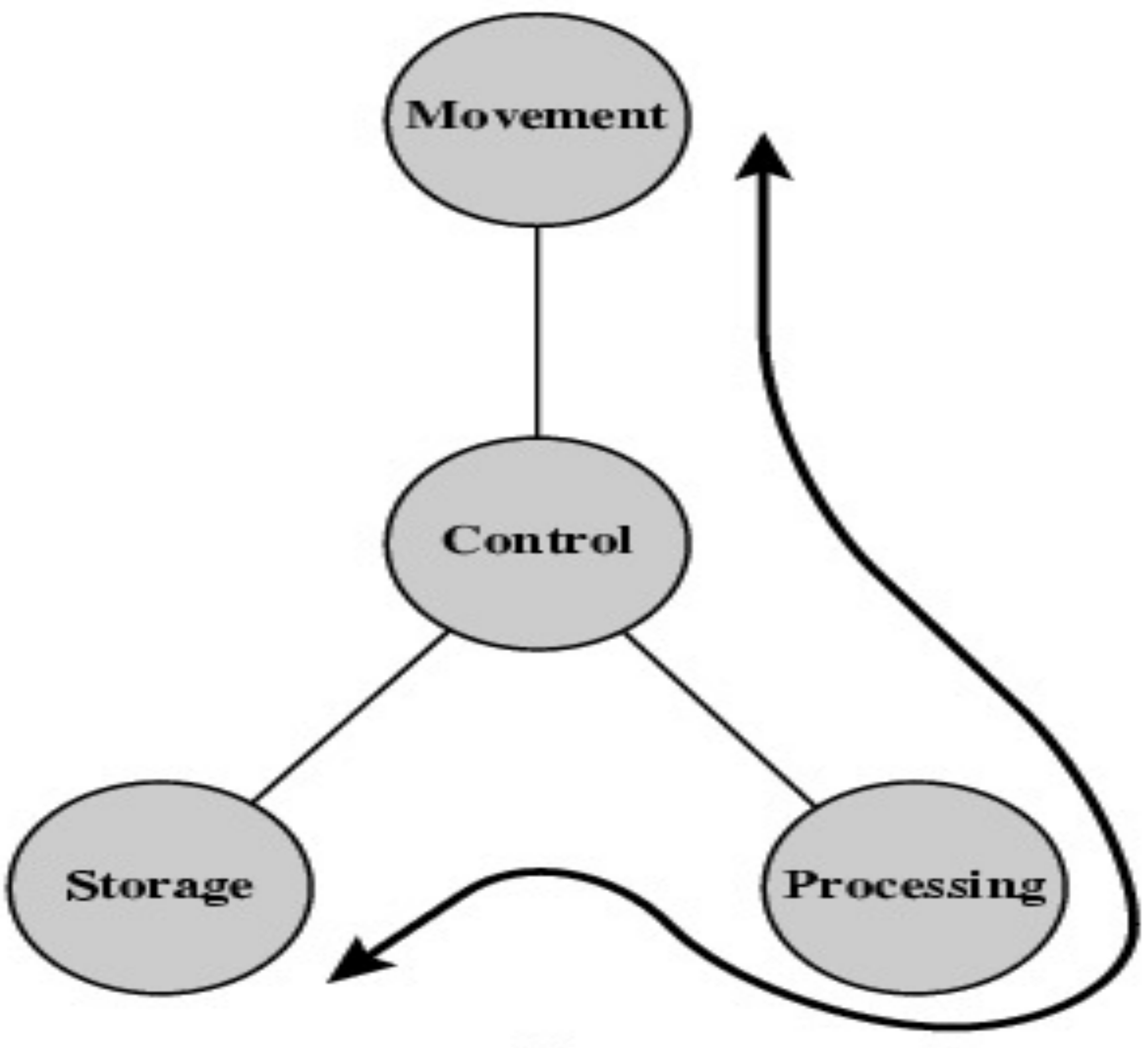
Von-Neumann/Turing Architecture

- Stored Program concept
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
  - IAS
  - Completed 1952

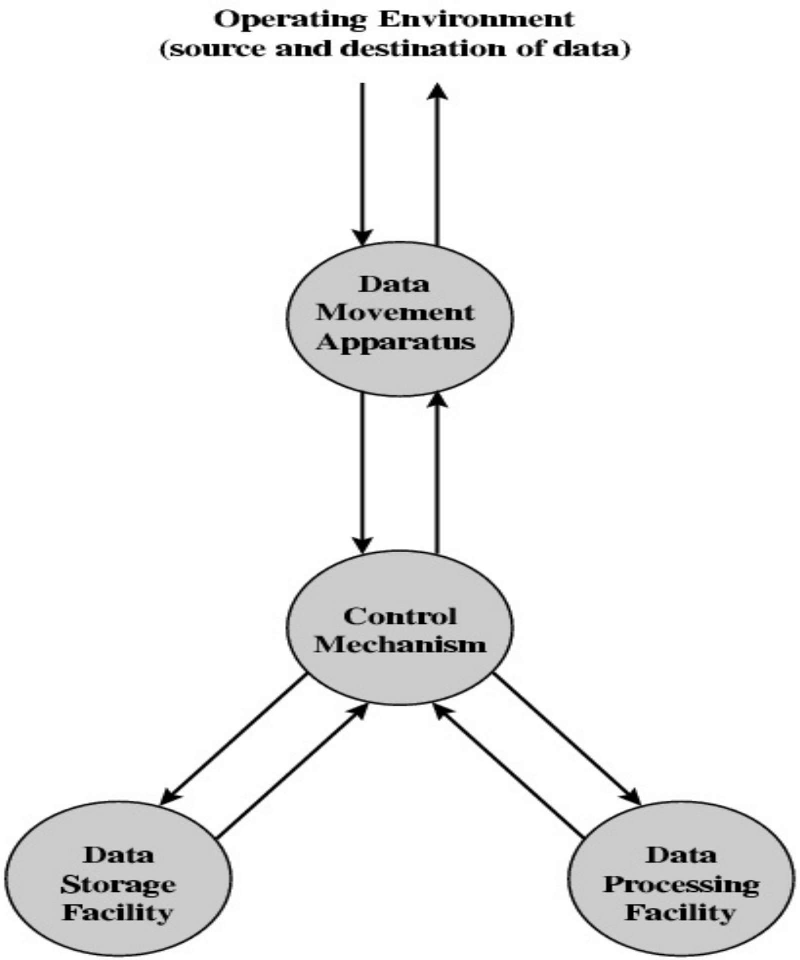
Von-Neumann Architecture (IAS)



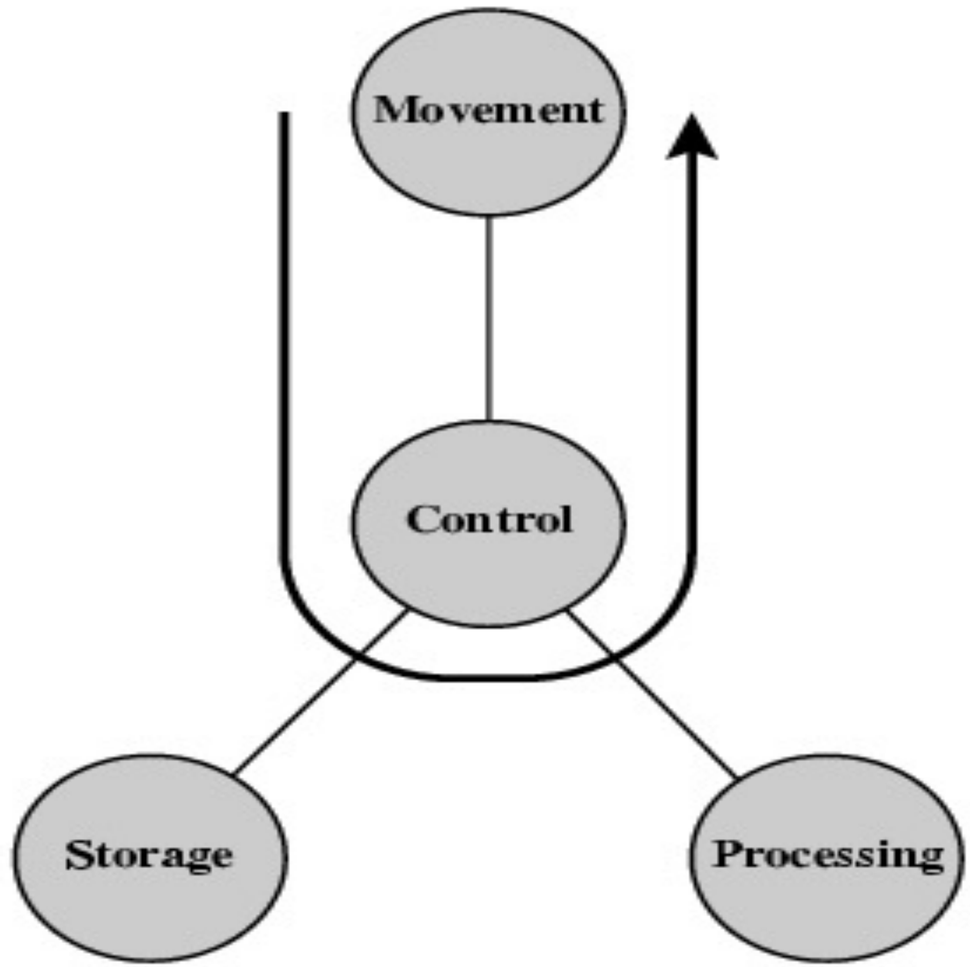
Processing from Storage to I/O



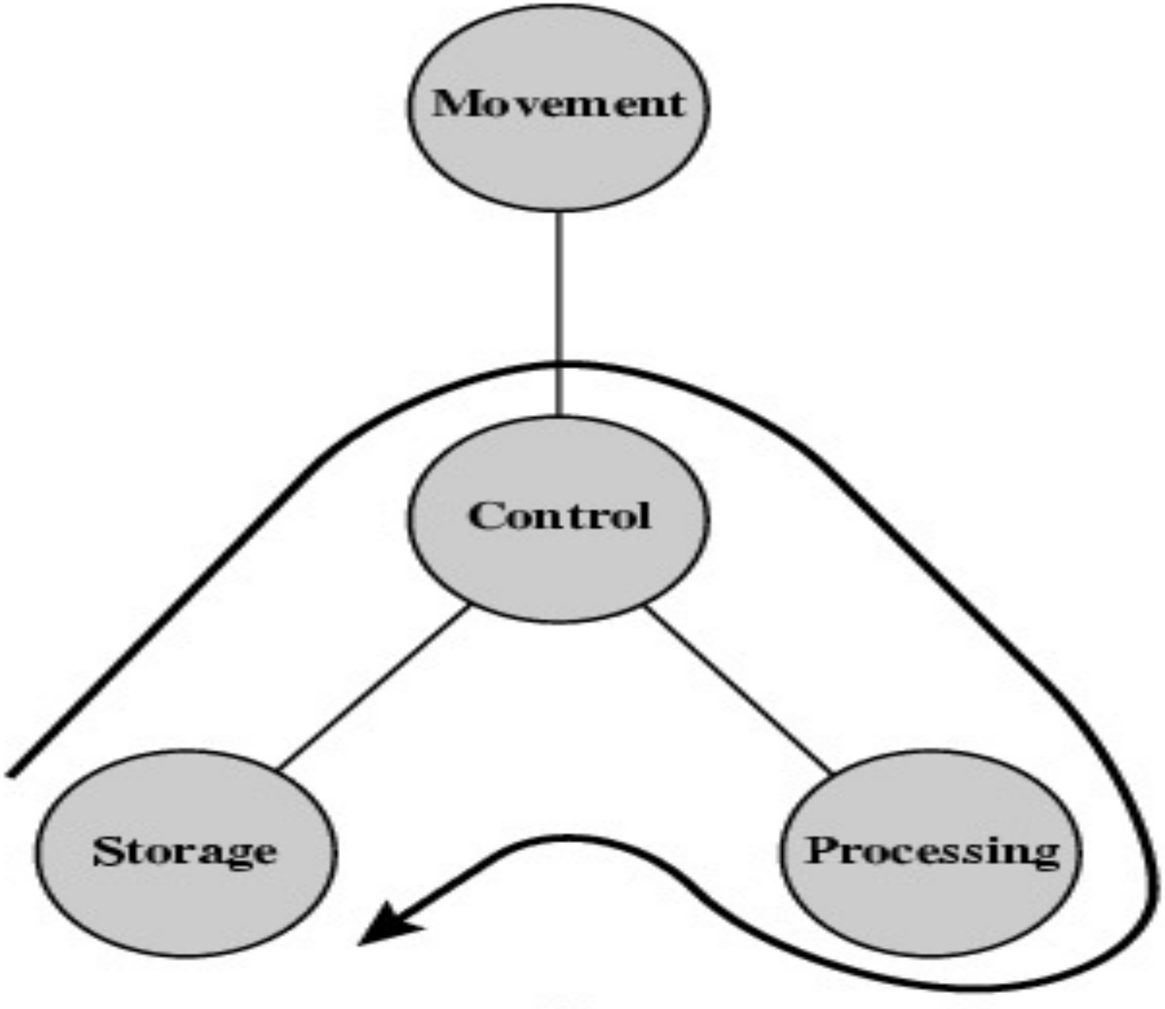
Functional View



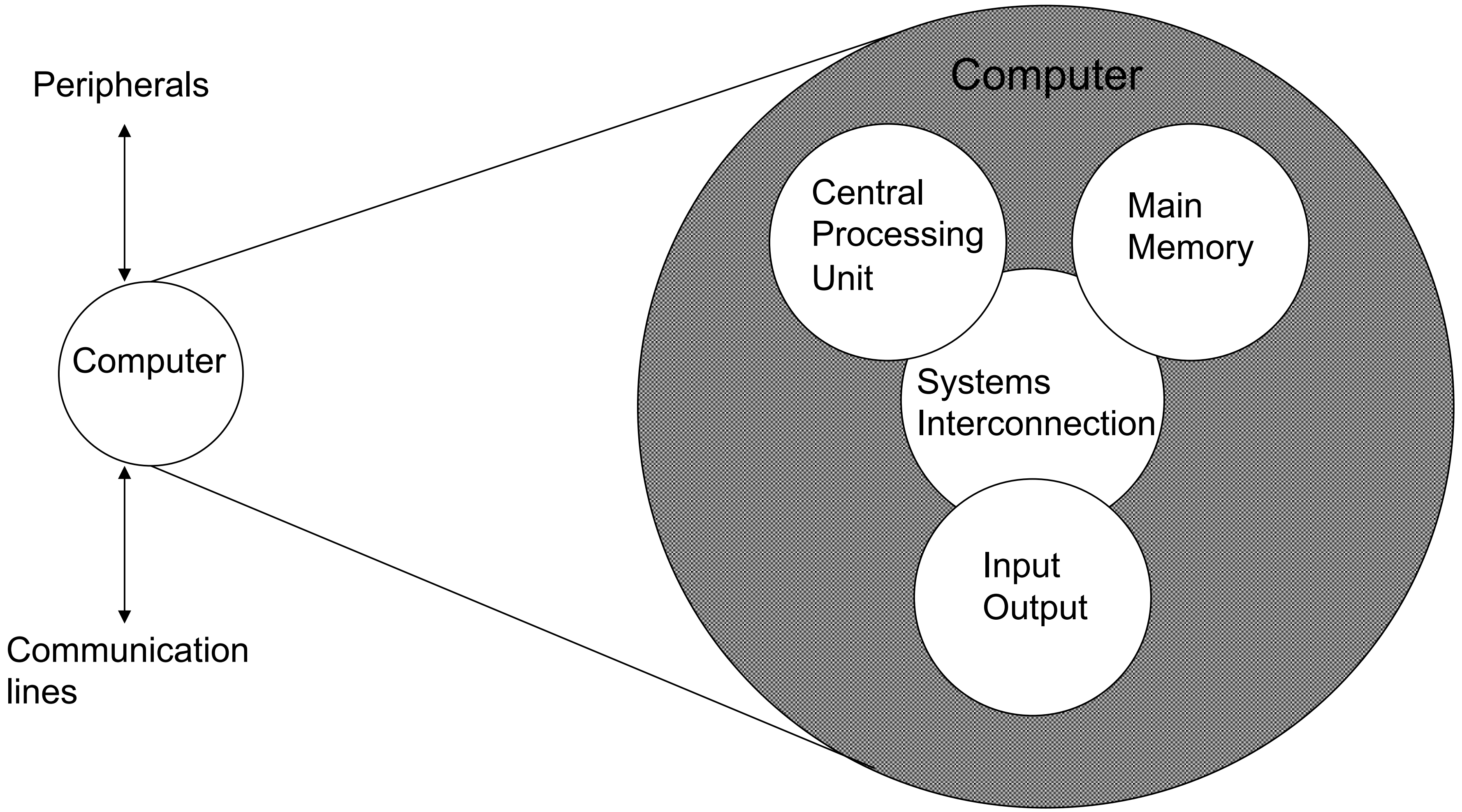
Data Movement



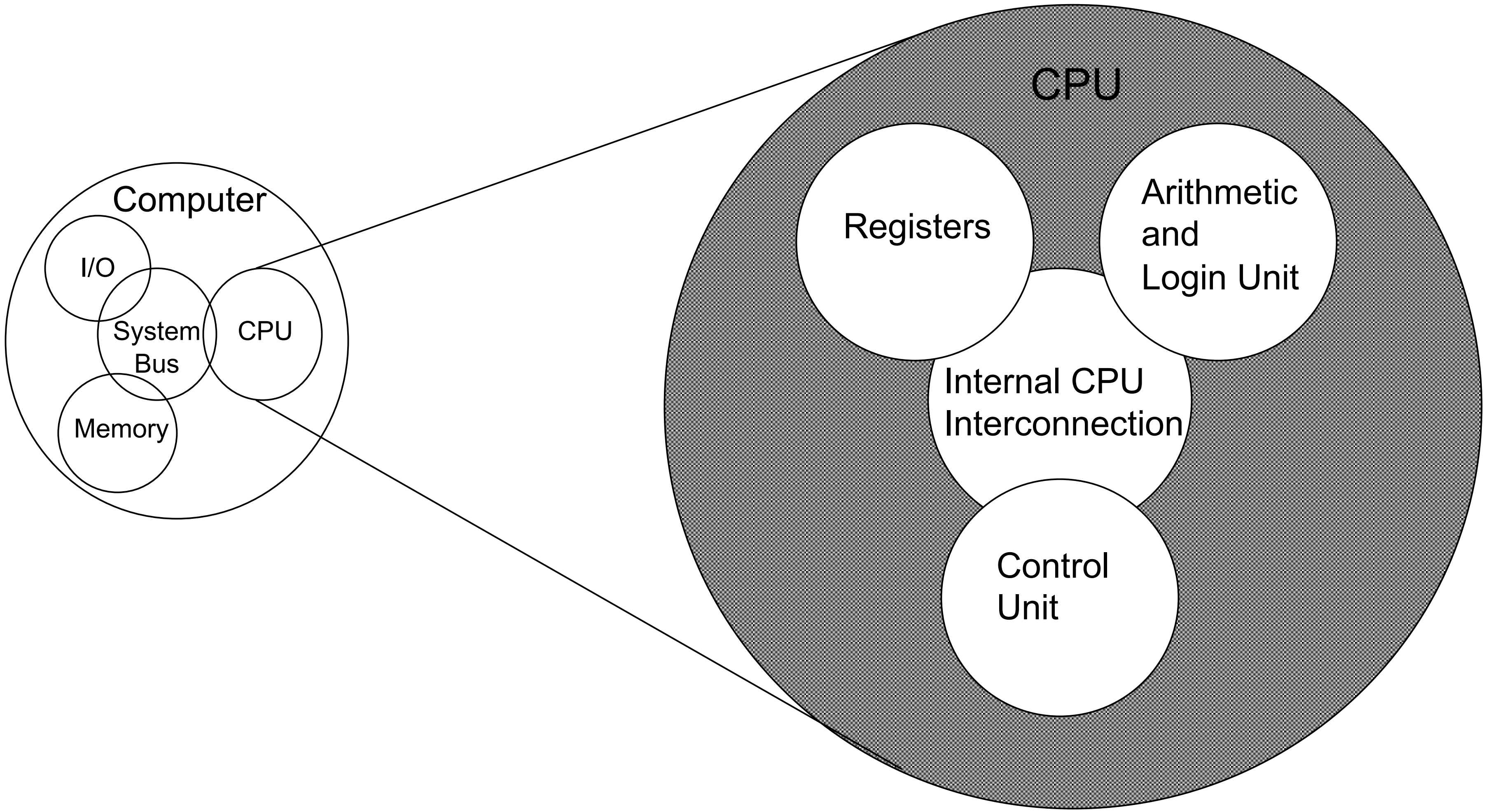
Processing To and From Storage



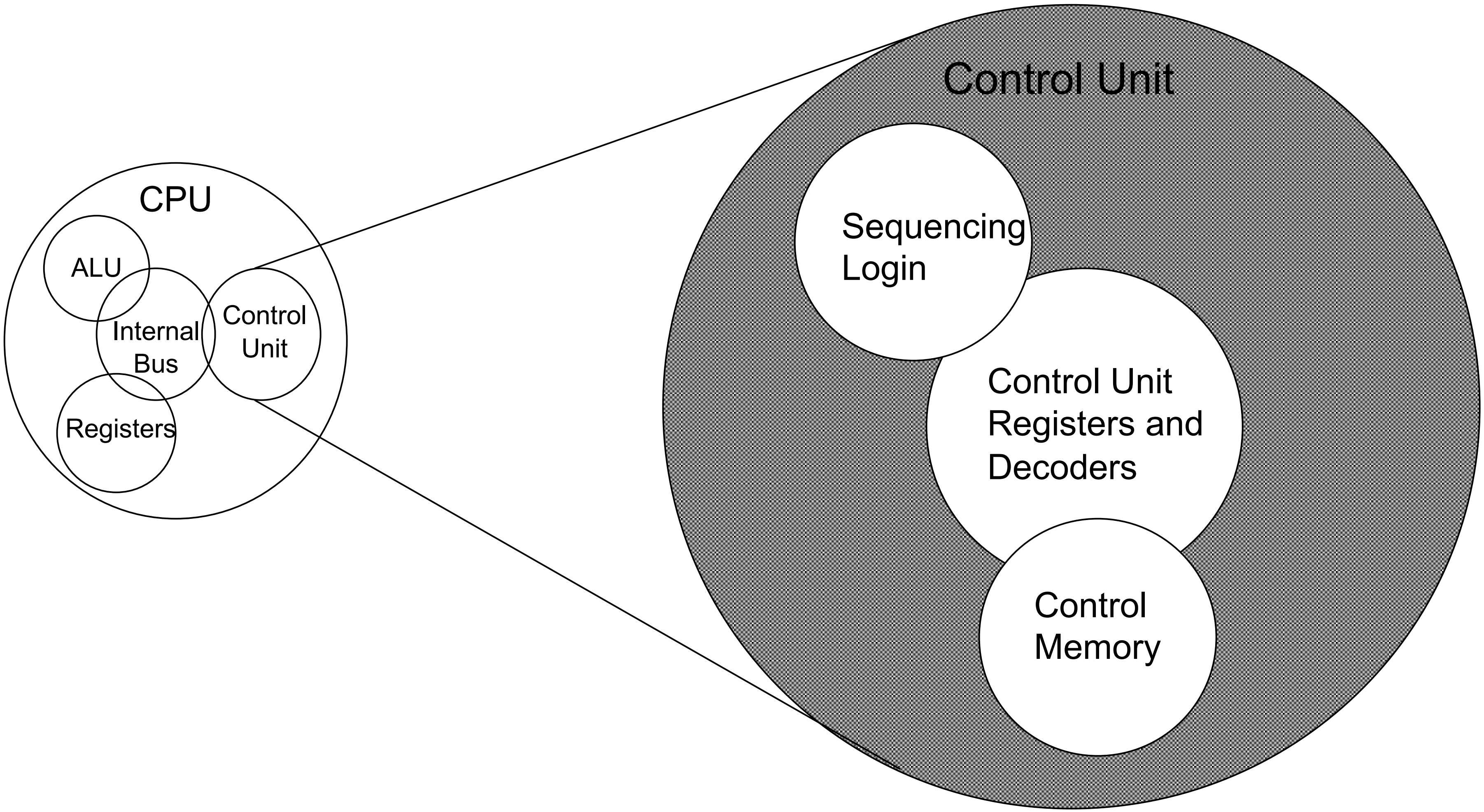
Top Level Structure



Structure of CPU



Structure of Control Unit



Technologies building the Computer

- Earliest technology was vacuum tube.
- Transistors changed everything.
  - It is an on/off switch controlled by electricity.
- Integrated circuit (IC) – combined dozen to hundreds of transistors into a single chip.
- Very Large-Scale integrated circuit (VLSI) – the number of transistors increased from hundreds to millions.

Vacuum Tube

Source: Wikepedia

Transistors

- Replaced vacuum tubes
- Smaller
- Cheaper
- Less heat dissipation
- Solid State device
- Made from Silicon (Sand)
- Invented 1947 at Bell Labs
- William Shockley et al.

Generations of Computers

- Vacuum tube - 1946-1957
- Transistor - 1958-1964
- Small scale integration - 1965 on
  - Up to 100 devices on a chip
- Medium scale integration - to 1971
  - 100-3,000 devices on a chip
- Large scale integration - 1971-1977
  - 3,000 - 100,000 devices on a chip
- Very large scale integration - 1978 -1991
  - 100,000 - 100,000,000 devices on a chip
- Ultra large scale integration – 1991 -
  - Over 100,000,000 devices on a chip

History of Computers

- There are five generations of computers.
- In each generation there was fundamental changes in computers in terms of its efficiency, reliability, size, cost, and power consumption.
- First generation of computers (1940-1956):
  - Vacuum tube was the main component for circuitry and magnetic drum was used for memory.
  - Used punch cards/tapes, machine language coding.
  - Expensive bulky, unreliable, power consuming.

History of Computers

- Second generation (1956-1963)
  - –Transistors replaced vacuum tubes
  - –Efficient, Smaller, faster, cheaper, and more reliable than vacuum tubes
  - –Relied on punched cards for input and printouts for output.
  - –Moved to assembly language coding. Some early HLL such as FORTRAN and COBOL got introduced.

History of Computers

- Third generation (1964-1971):
  - Technology shifted to integrated circuits where transistors were miniaturized and placed on a single chip.
  - Speed and efficiency drastically changed
  - Keyboard and monitor got introduced as Input and output devices
  - Early operating system emerged.

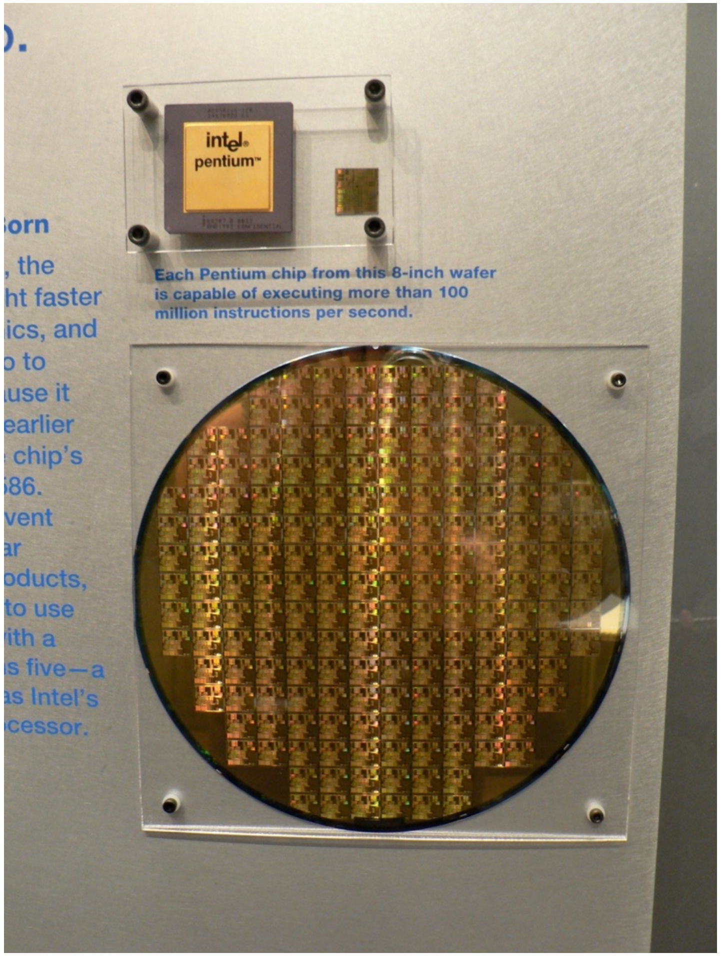
History of Computers

- Fourth generation (1971 to present)
  - –Microprocessor arrived with LSI and VLSI ICs.
  - –Made computers much smaller in size and cost with enhanced performance.
  - –Home computers became affordable.
  - –Embedded computing became possible
  - –Graphical interface and mouse came into picture.

History of Computers

- Fifth generation of computers (present and beyond)
  - Based on artificial intelligence concepts
  - Voice input and output
  - Natural language input and output.
  - Parallel computing
  - Quantum computing (nanotechnology).
  - DNA computing.

Wafers

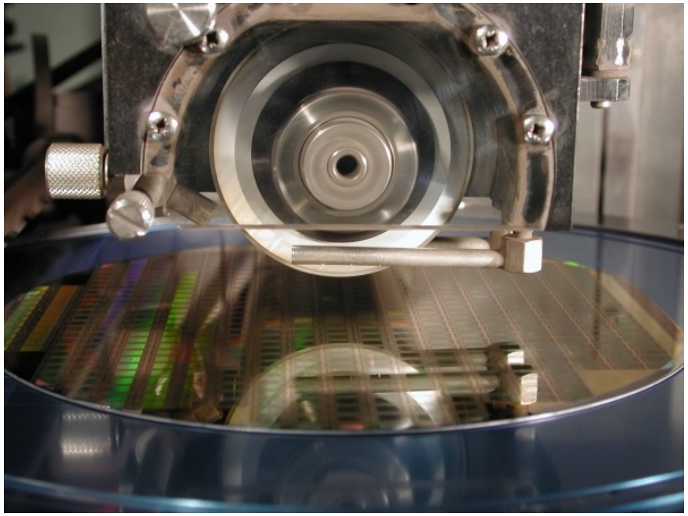


Source: http://commons.wikimedia.org/wiki/File:Wafer_with_Pentium_chips_.jpg

Source: http://www.txchnologist.com/2011/10-factories-that-changed-the-world/intels-fab-1-through-42

Dicing

- Dicing: Each of the individual chips on the wafer are diced.



Source: http://www.aspentechnologies.com/manufacturing/capabilities_technologies/wafer_dicing_inspection.html

- Previously shown wafer has 196 Pentium dies.
- Since a wafer costs about the same no matter what is on it, fewer dies mean higher costs.

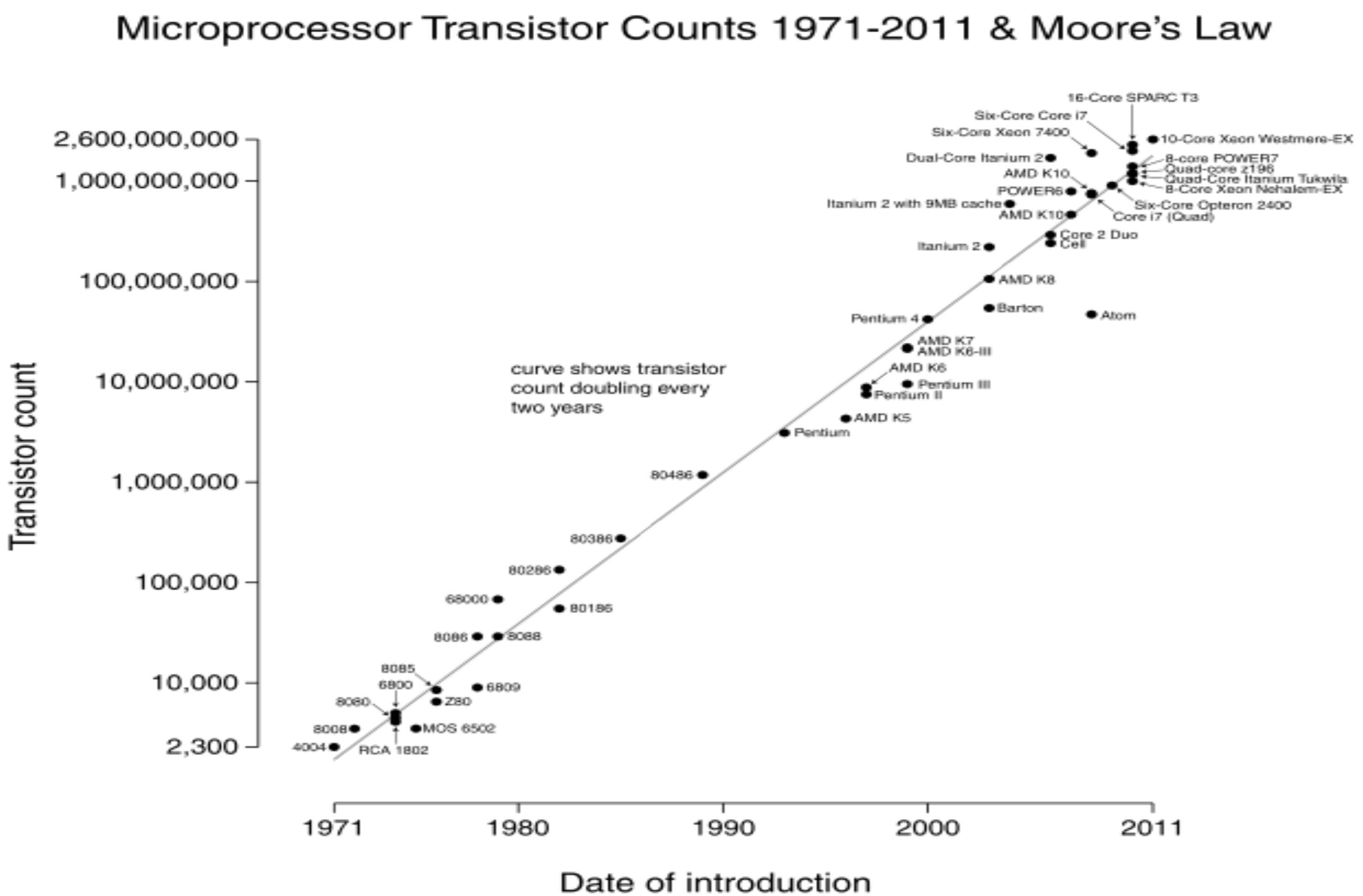
Moore’s Law

- “The number of transistors than can be placed inexpensively on an integrated circuit doubles approximately every 18 months” – Moore’s law

- Intel co-founder Gordon E. Moore, suggested this trend in his 1965 paper.

- It’s a information revolution.

Moore’s Law


Microprocessor Transistor Counts 1971-2011 & Moore’s Law
Transistor count
Date of introduction
curve shows transistor count doubling every two years
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
1971
1980
1990
2000
2011
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
Quad-Core Itanium Tukwila
8-Core Xeon Nehalem-EX
AMD K10
POWER6
Six-Core Opteron 2400
Itanium 2 with 9MB cache
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802

ENIAC

- Electronic Numerical Integrator And Computer
- Eckert and Mauchly
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
- Used until 1955

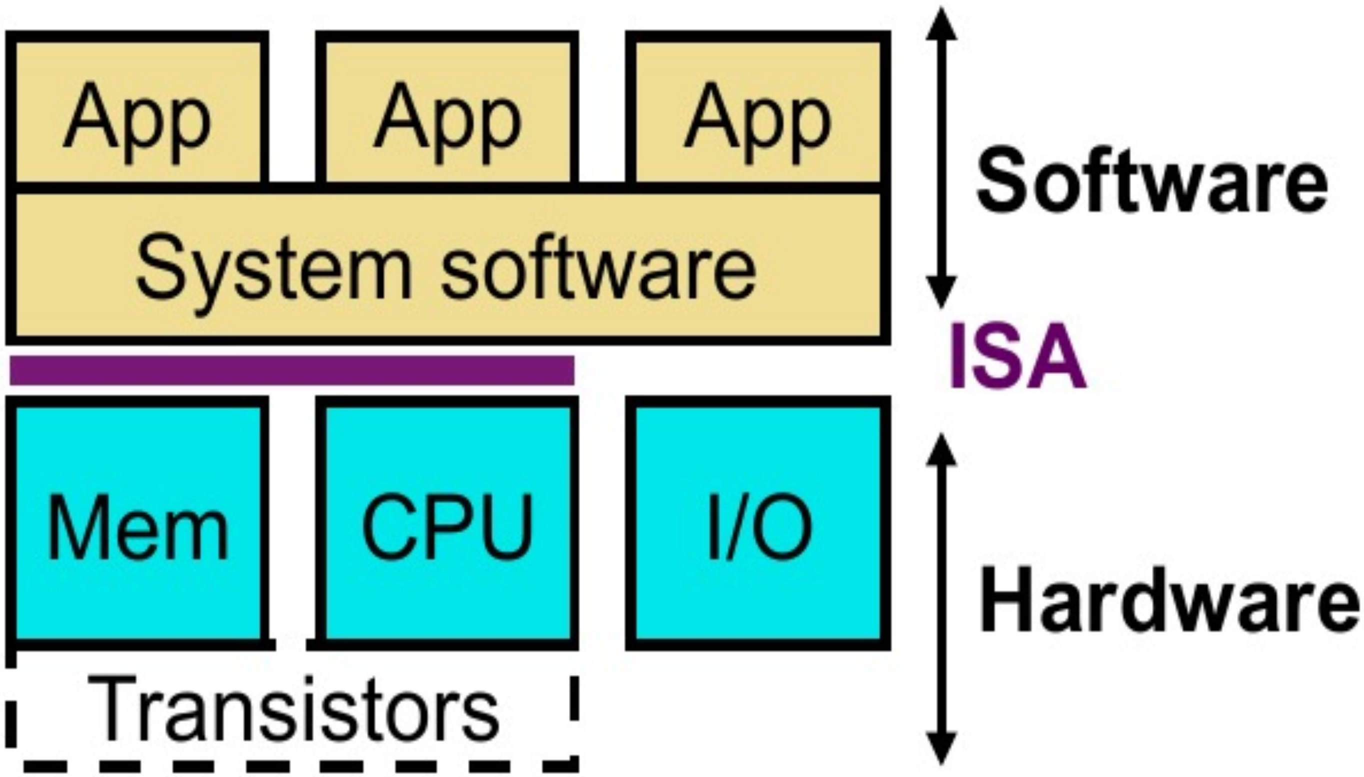
ENIAC

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

Commercial Computers

- 1947 - Eckert-Mauchly Computer Corporation
- UNIVAC I (Universal Automatic Computer)
- US Bureau of Census 1950 calculations
- Became part of Sperry-Rand Corporation
- Late 1950s - UNIVAC II
  - Faster
  - More memory

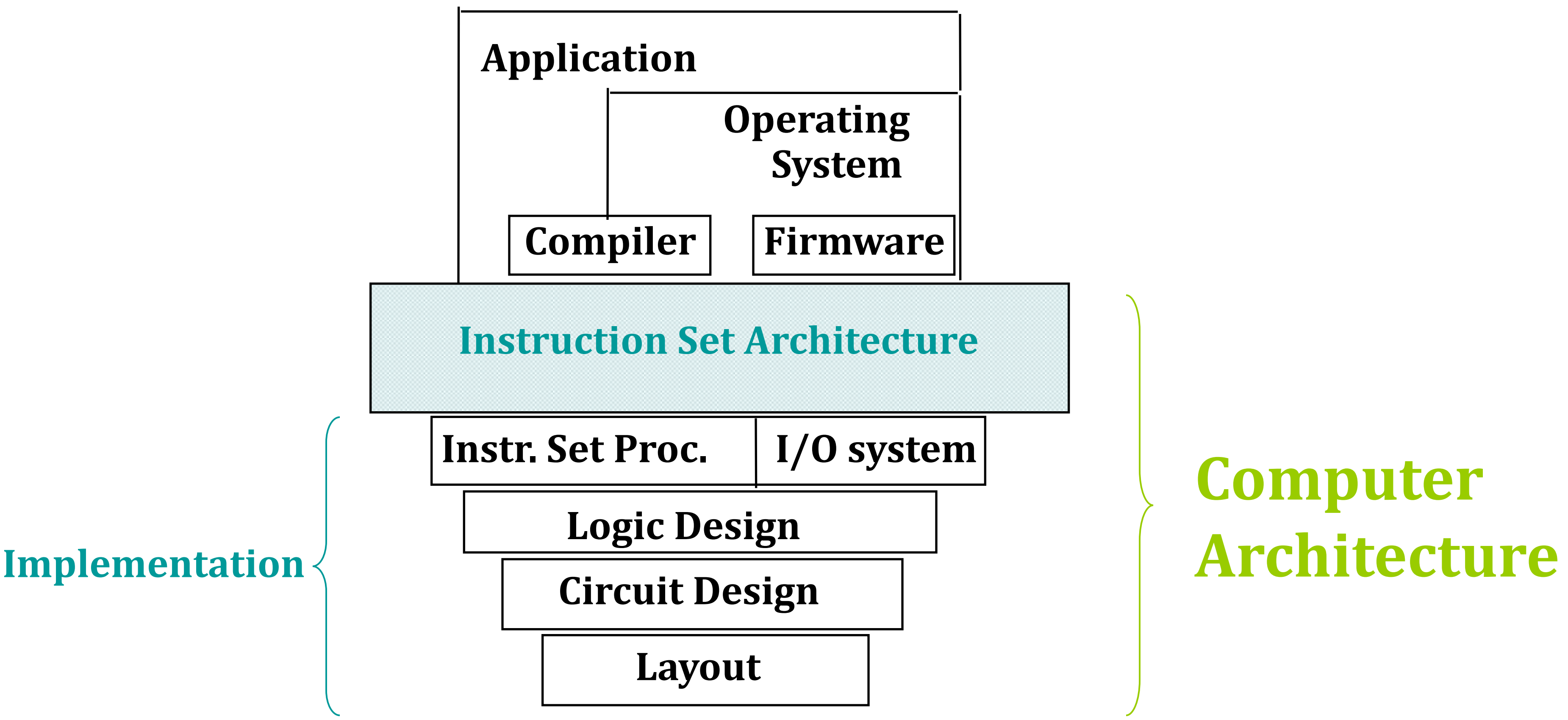
Abstraction, Layering and Computers



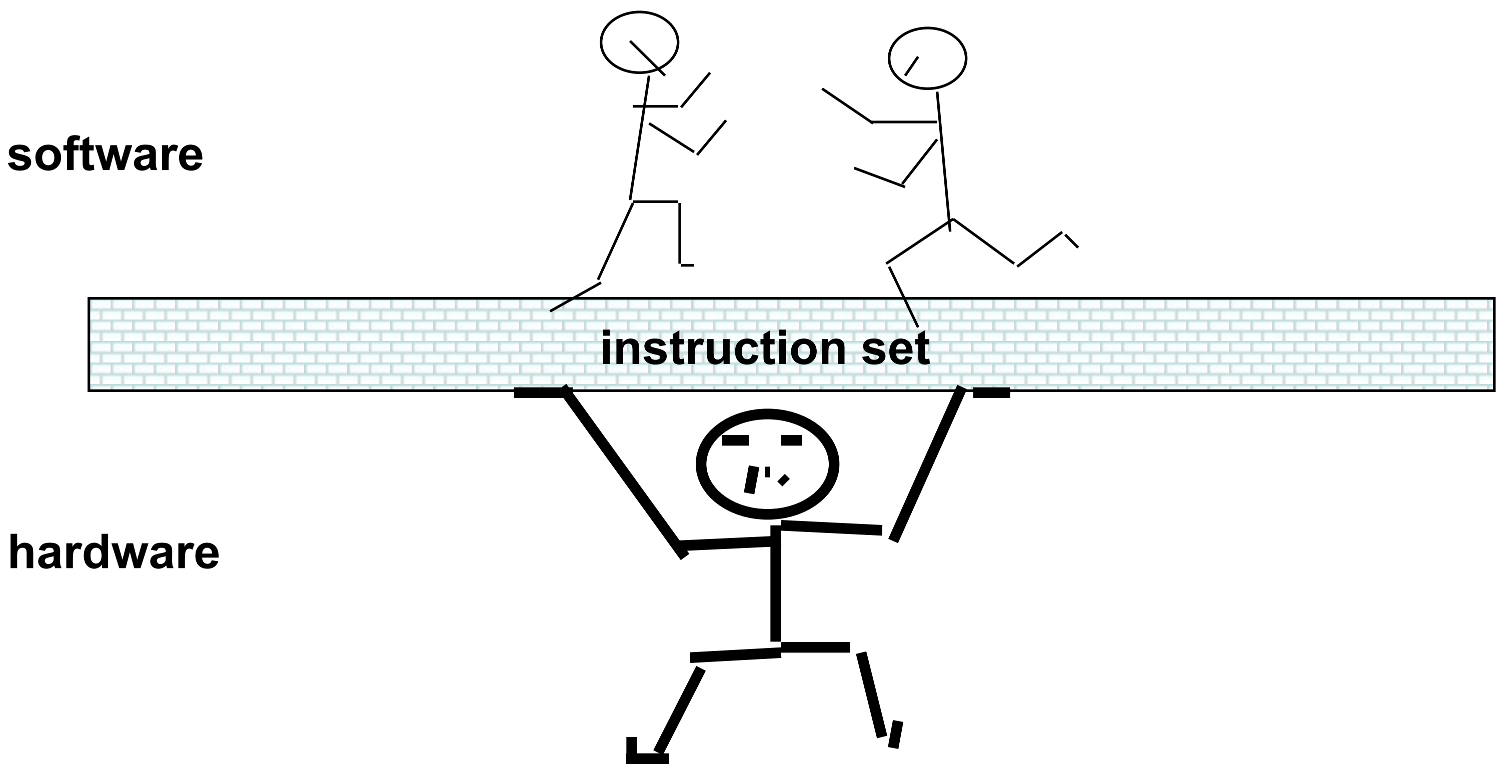
Abstraction, Layering and Computers

- Computers are complex, built in layers
  - Several software layers: assembler, compiler, OS, applications
  - Instruction set architecture (ISA)
  - Several hardware layers: transistors, gates, CPU/Memory/IO
- 99% of users don't know hardware layers implementation
- 90% of users don't know implementation of any layer
  - That's okay, world still works just fine
  - But sometimes it is helpful to understand what's "under the hood"

Computer Architecture



The Instruction Set: a Critical Interface



Instruction Set Architecture

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

- Common instruction set architectures:
  - IA-32, PowerPC, MIPS, SPARC, ARM, and others

Instruction Set Architecture

- ISA, or simply architecture – the abstract interface between the hardware and the lowest level software that encompasses all the information necessary to write a machine language program, including instructions, registers, memory access, I/O, …

- ISA Includes
  - Organization of storage
  - Data types
  - Encoding and representing instructions
  - Instruction Set (or opcodes)
  - Modes of addressing data items/instructions
  - Program visible exception handling
- Specifies requirements for binary compatibility across implementations (ABI)

Case Study: MIPS ISA

- Instruction Categories
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
  - Memory Management
  - Special

R0 - R31

PC

HI

LO

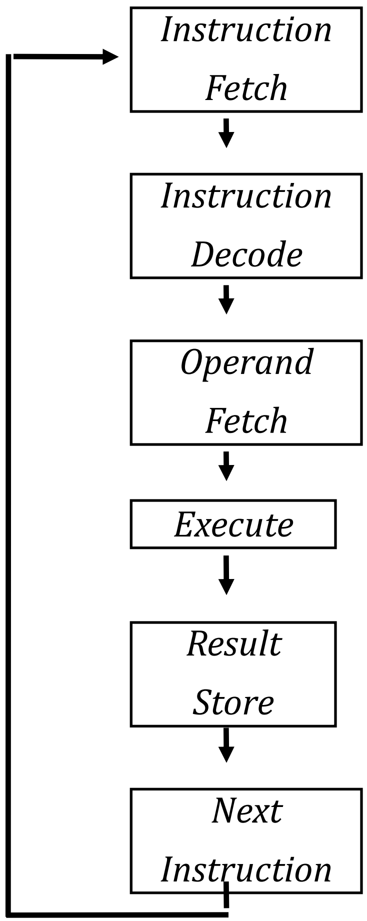
3 Instruction Formats, 32 bits wide

OP	rs	rt	rd	sa	funct

OP	rs	rt	immediate

OP	jump target

Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Levels of Representation

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

Control Signal Specification

temp = v[k];

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

v[k+1] = temp;

```
lw   $15,   0($2)
lw   $16,   4($2)
sw   $16,   0($2)
sw   $15,   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
```

ALUOP[0:3] <= InstReg[9:11] & MASK
[i.e.high/low on control lines]

Advantage of HLLs

- Higher-level languages (HLLs)
  - Allow the programmer to think in a more natural language and for their intended use (Fortran for scientific computation, Cobol for business programming, Lisp for symbol manipulation, Java for web programming, …)
  - Improve programmer productivity – more understandable code that is easier to debug and validate
  - Improve program maintainability
  - Allow programs to be independent of the computer on which they are developed (compilers and assemblers can translate high-level language programs to the binary instructions of any machine)
  - Emergence of optimizing compilers that produce very efficient assembly code optimized for the target machine
- Compilers convert source code to object code
- Libraries simplify common tasks