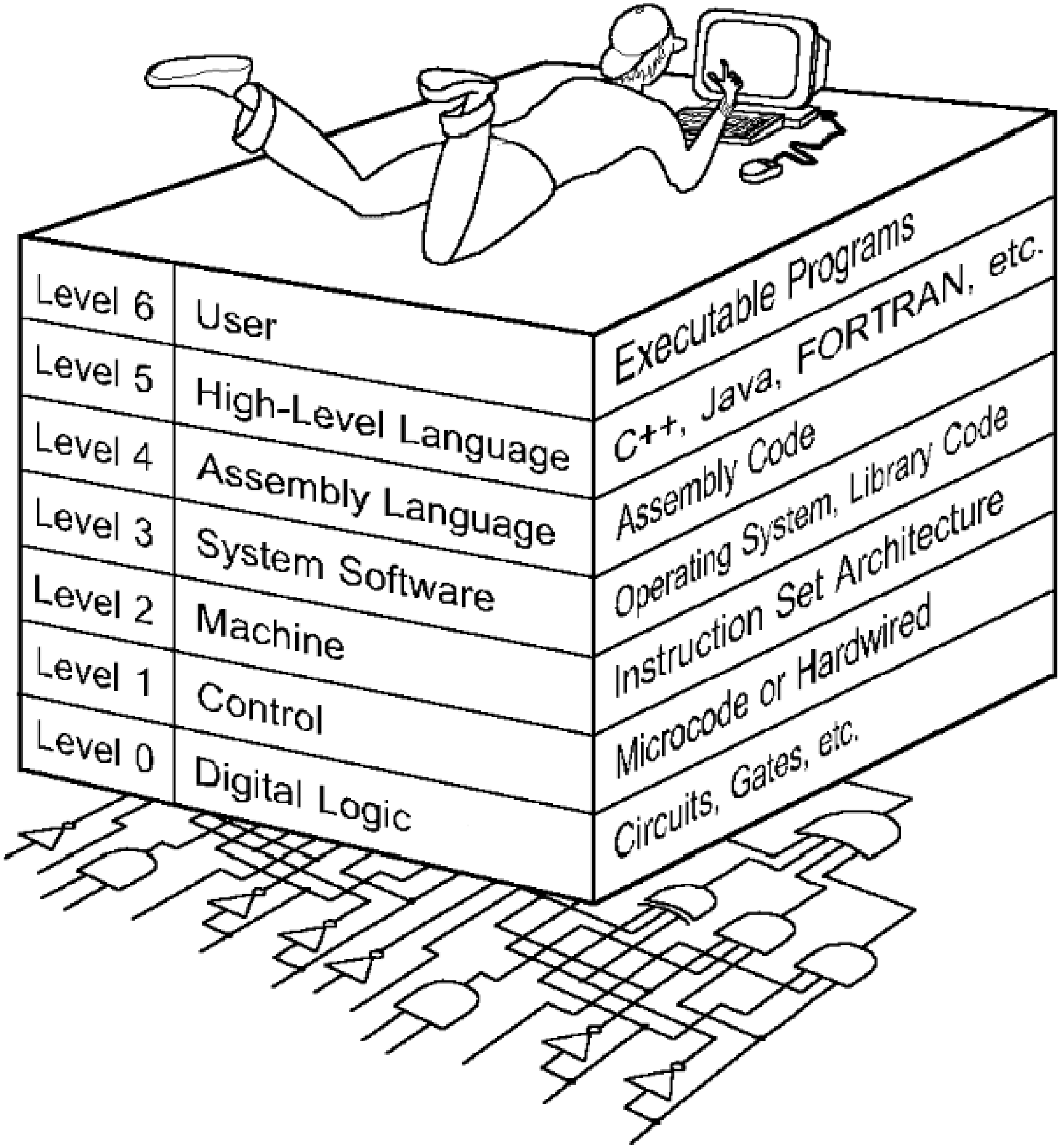
COMPUTER ORGANISATION

Computer Level Hierarchy



Program Execution

Translation: The entire high level program is translated into an equivalent machine language program. Then the machine language program is executed.

Interpretation: Another program reads the high level program instructions one-by-one and executes a equivalent series of machine language instructions.

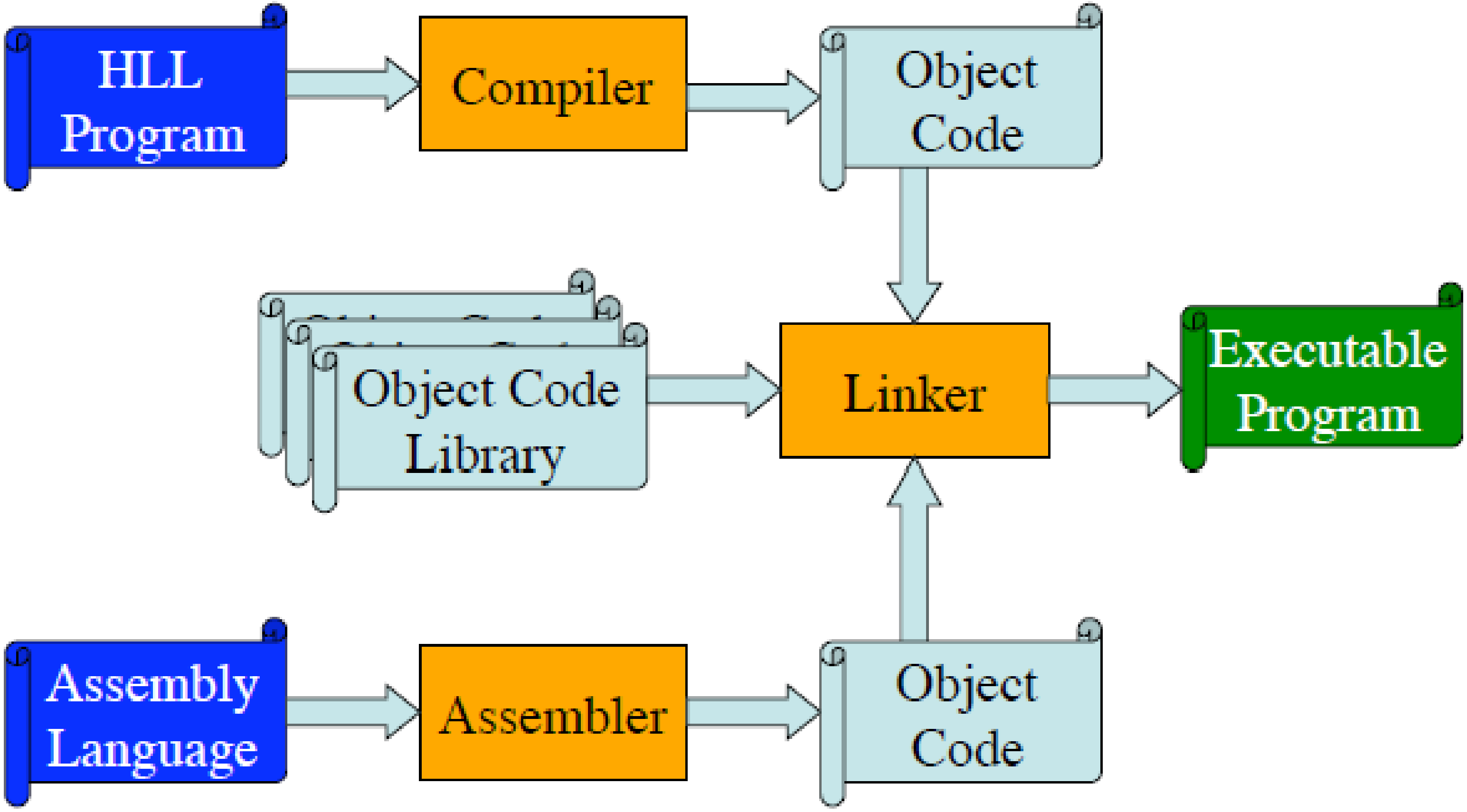
Program translation uses a collection of tools to perform the translation:

- **Compiler:** Translates high level language programs into a lower level language often called object code.

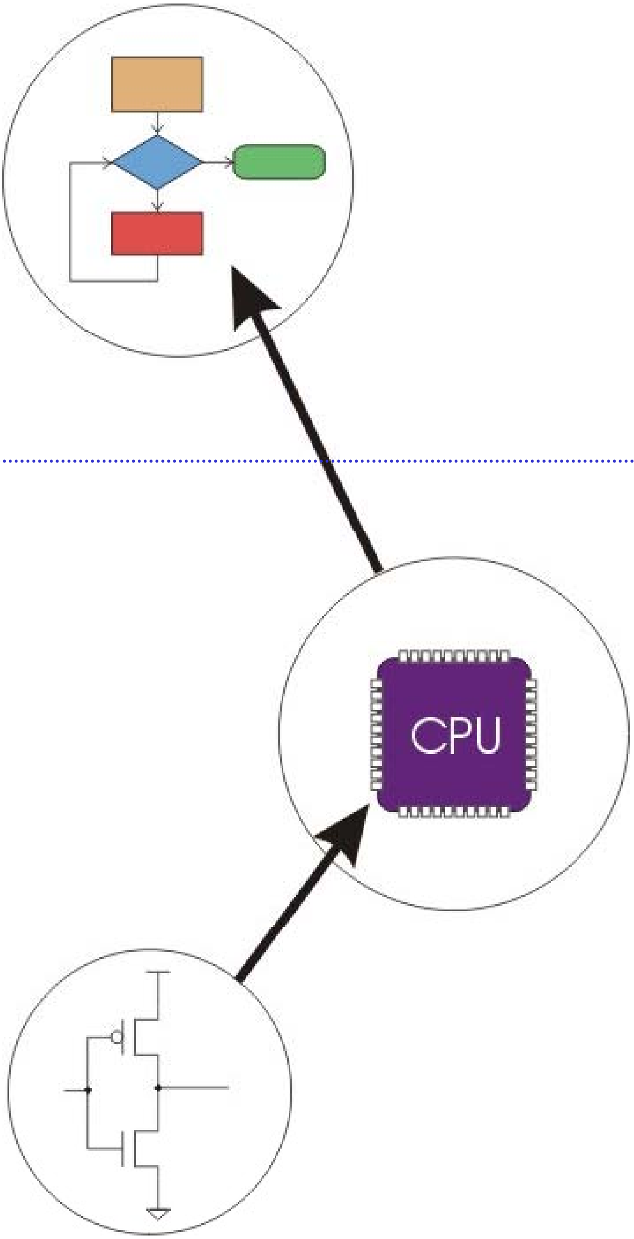
- **Assembler:** Translates assembly language instructions into object code.

- **Linker:** Combines collections of object code into a single executable machine language program.

Program Translation



Computer System: Layers of Abstraction



Application Program

Algorithms

Language

Software

Hardware

Instruction Set Architecture
(and I/O Interfaces)

Microarchitecture

Circuits

Devices

From Theory to Practice

In theory, computer can *compute* anything that's possible to compute

- **given enough *memory* and *time***

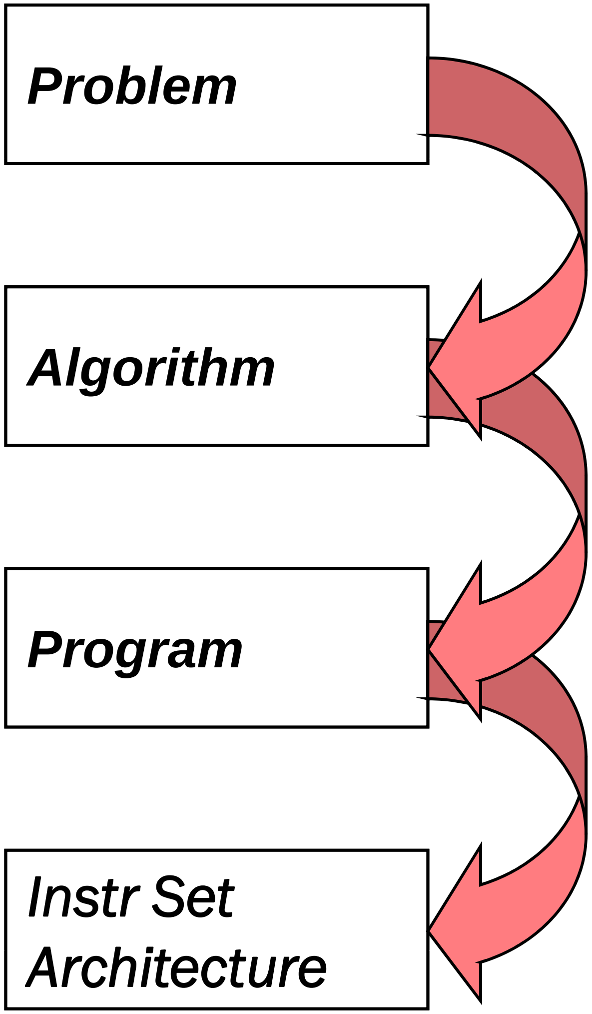
In practice, *solving problems* involves computing under constraints.

- **time**
  - **weather forecast, next frame of animation, ...**
- **cost**
  - **cell phone, automotive engine controller, ...**
- **power**
  - **cell phone, handheld video game, ...**

Transformations Between Layers

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.



Problem

Software Design:
choose algorithms and data structures

Algorithm

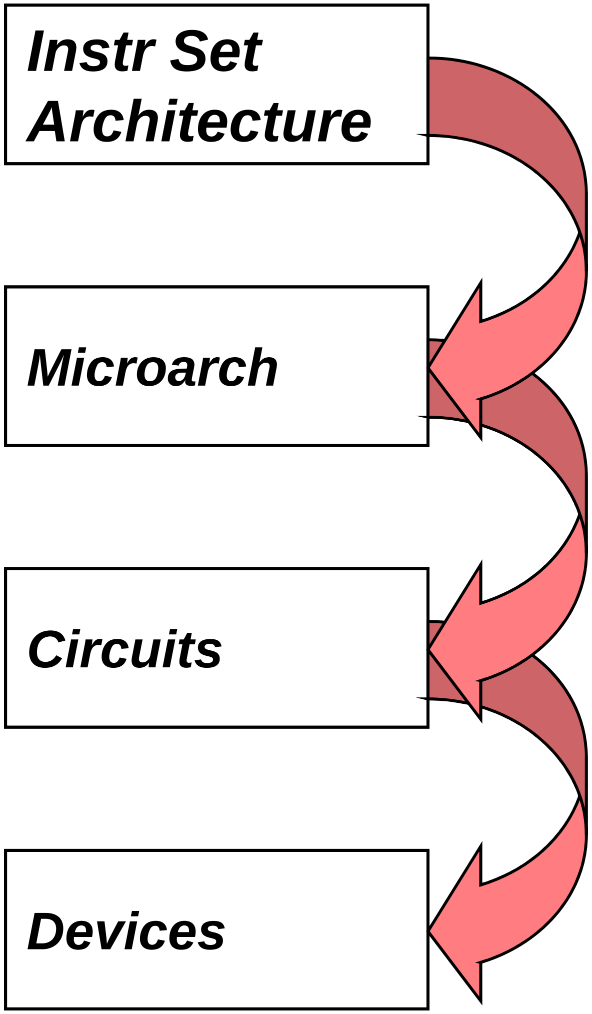
Programming:
use language to express design

Program

Compiling/Interpreting:
convert language to
machine instructions

Instr Set Architecture

Deeper and Deeper…



Instr Set Architecture

Microarch

Circuits

Devices

Processor Design:
choose structures to implement ISA

Logic/Circuit Design:
gates and low-level circuits to implement components

Process Engineering & Fabrication:
develop and manufacture lowest-level components

Descriptions of Each Level

Problem Statement

- stated using "natural language"
- may be ambiguous, imprecise

Algorithm

- step-by-step procedure, guaranteed to finish
- definiteness, effective computability, finiteness

Program

- express the algorithm using a computer language
- high-level language, low-level language

Instruction Set Architecture (ISA)

- specifies the set of instructions the computer can perform
- data types, addressing mode

Descriptions of Each Level (cont.)

Microarchitecture

- detailed organization of a processor implementation
- different implementations of a single ISA

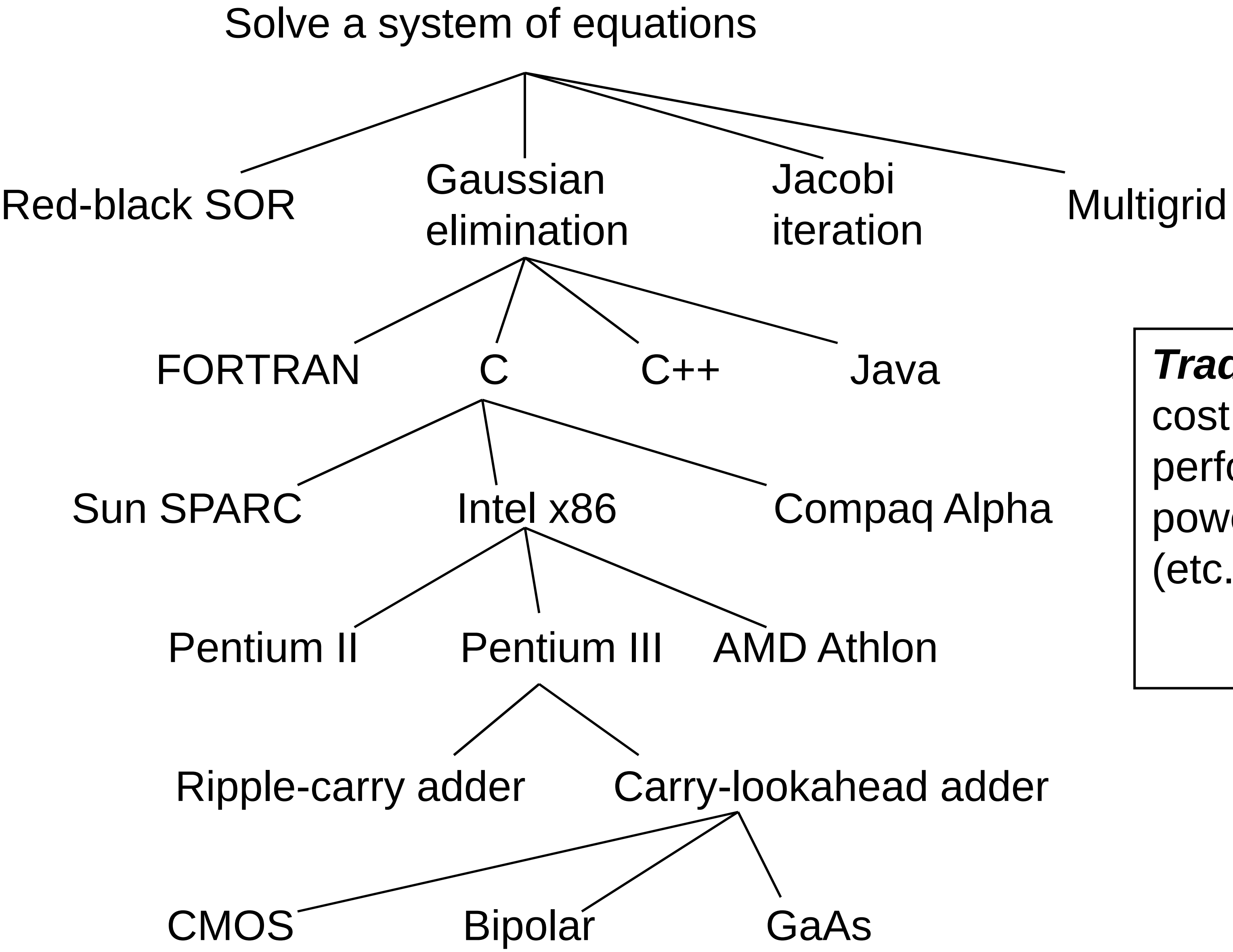
Logic Circuits

- combine basic operations to realize microarchitecture
- many different ways to implement a single function (e.g., addition)

Devices

- properties of materials, manufacturability

Many Choices at Each Level



Solve a system of equations

Red-black SOR

Gaussian elimination

Jacobi iteration

Multigrid

FORTRAN

C

C++

Java

Sun SPARC

Intel x86

Compaq Alpha

Pentium II

Pentium III

AMD Athlon

Ripple-carry adder

Carry-lookahead adder

CMOS

Bipolar

GaAs

Tradeoffs:
cost
performance
power
(etc.)

What’s Next

Bits and Bytes

- How do we represent information using electrical signals?

Digital Logic

- How do we build circuits to process information?

Processor and Instruction Set

- How do we build a processor out of logic elements?
- What operations (instructions) will we implement?

Assembly Language Programming

- How do we use processor instructions to implement algorithms?
- How do we write modular, reusable code? (subroutines)

I/O, Traps, and Interrupts

- How does processor communicate with outside world?

Structure and Function of a COMPUTER SYSTEM:

A computer is a complex system; For analysis, understanding and design - Identify the *hierarchical nature* of most complex system.

A hierarchical system is a set of interrelated subsystems, each in turn, hierarchical in structure; until at the lowest level we have elementary subsystems.

The hierarchical nature of complex systems is essential to both their design and their description. The designer need only deal with a particular level of the system at a time.

At each level, the system consists of a set of *components and their interrelationships*.

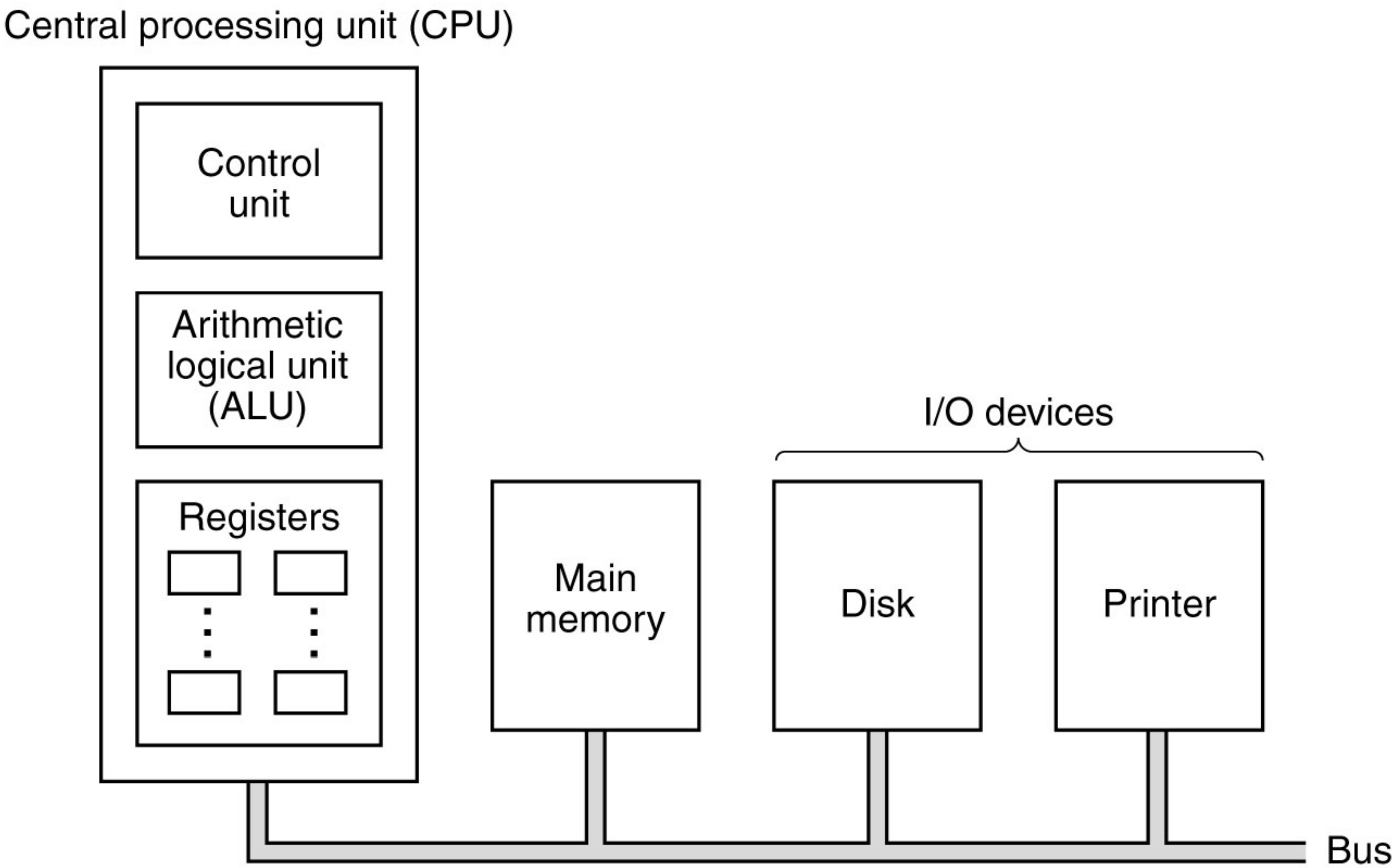
The behavior at each level depends only on a simplified, abstracted characterization of the system at the next lower level.

At each level, the designer is concerned with structure and function:

Structure: The way in which the components are interrelated.

Function: The operation of each individual component as part of the structure.

Central Processing Unit (CPU) based CO



The organization of a simple computer with one CPU and two I/O devices

There are four main **functions** of a computer:

- Data processing
- Data storage
- Data movement
- Control

MAIN **STRUCTURAL** BLOCKS / PARTS:

Central Processing Unit (CPU): Controls the operation of the computer and performs its data processing functions. Often simply referred to as processor.

Main Memory: Stores data.

I / O: Moves data between the computer and its external environment.

System Interconnection: e.g. BUS for communication among CPU, main memory, and I / O.

The major structural components of a CPU are:

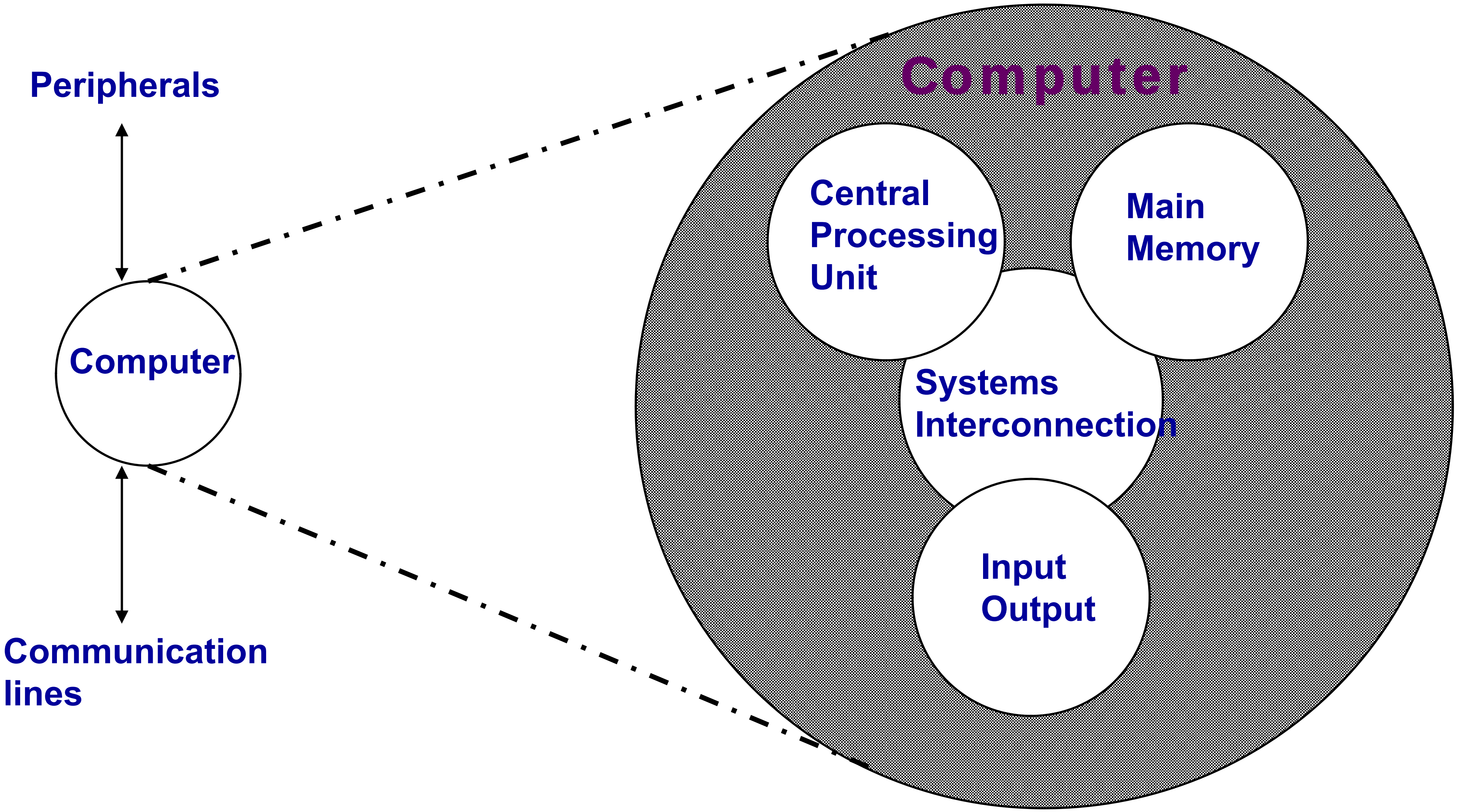
Control Unit (CU): Controls the operation of the CPU and hence the computer.

Arithmetic and Logic Unit (ALU): Performs computer’s data processing functions.

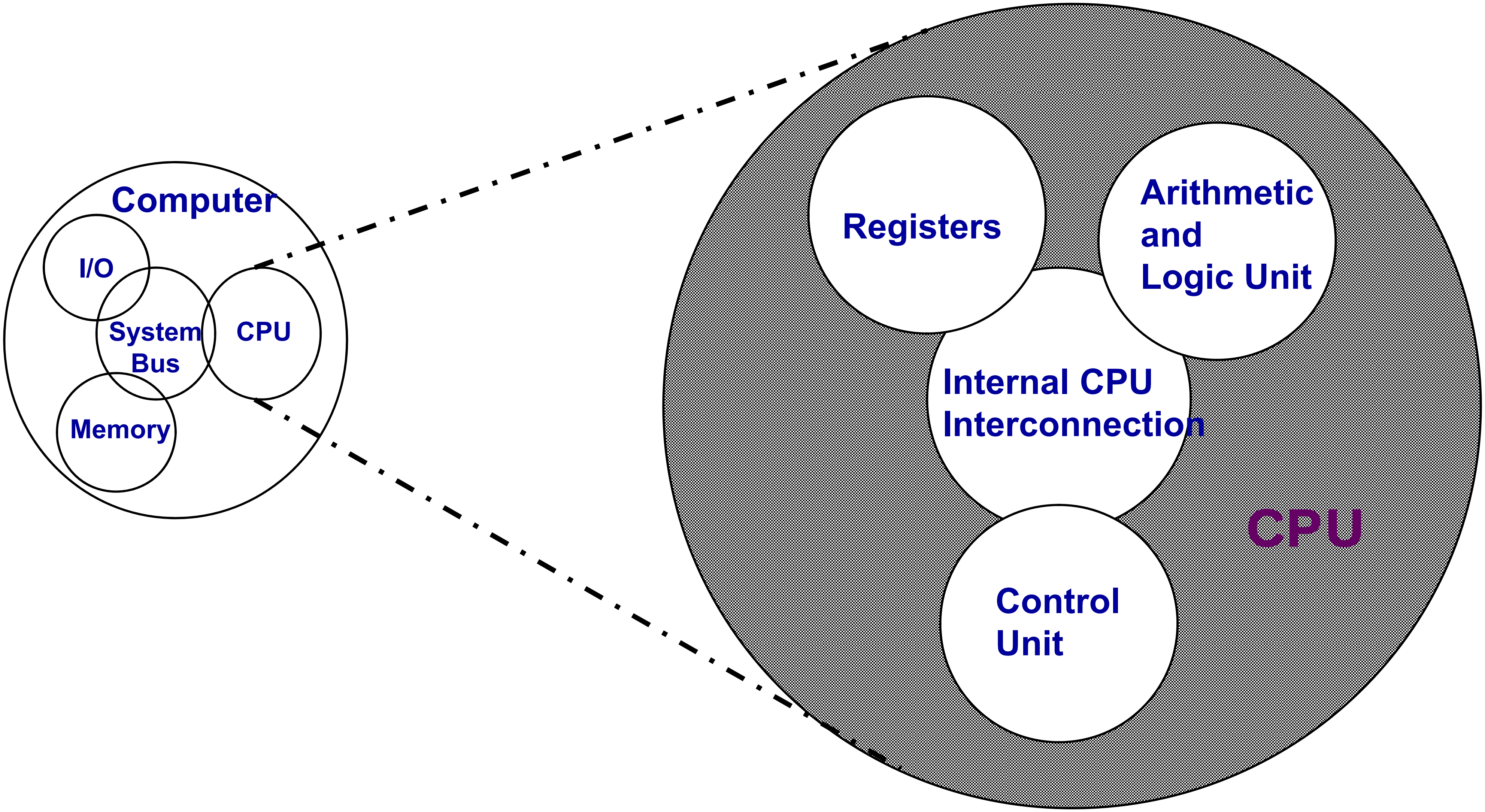
Register: Provides storage internal to the CPU.

CPU Interconnection: communication among the control unit, ALU, and register.

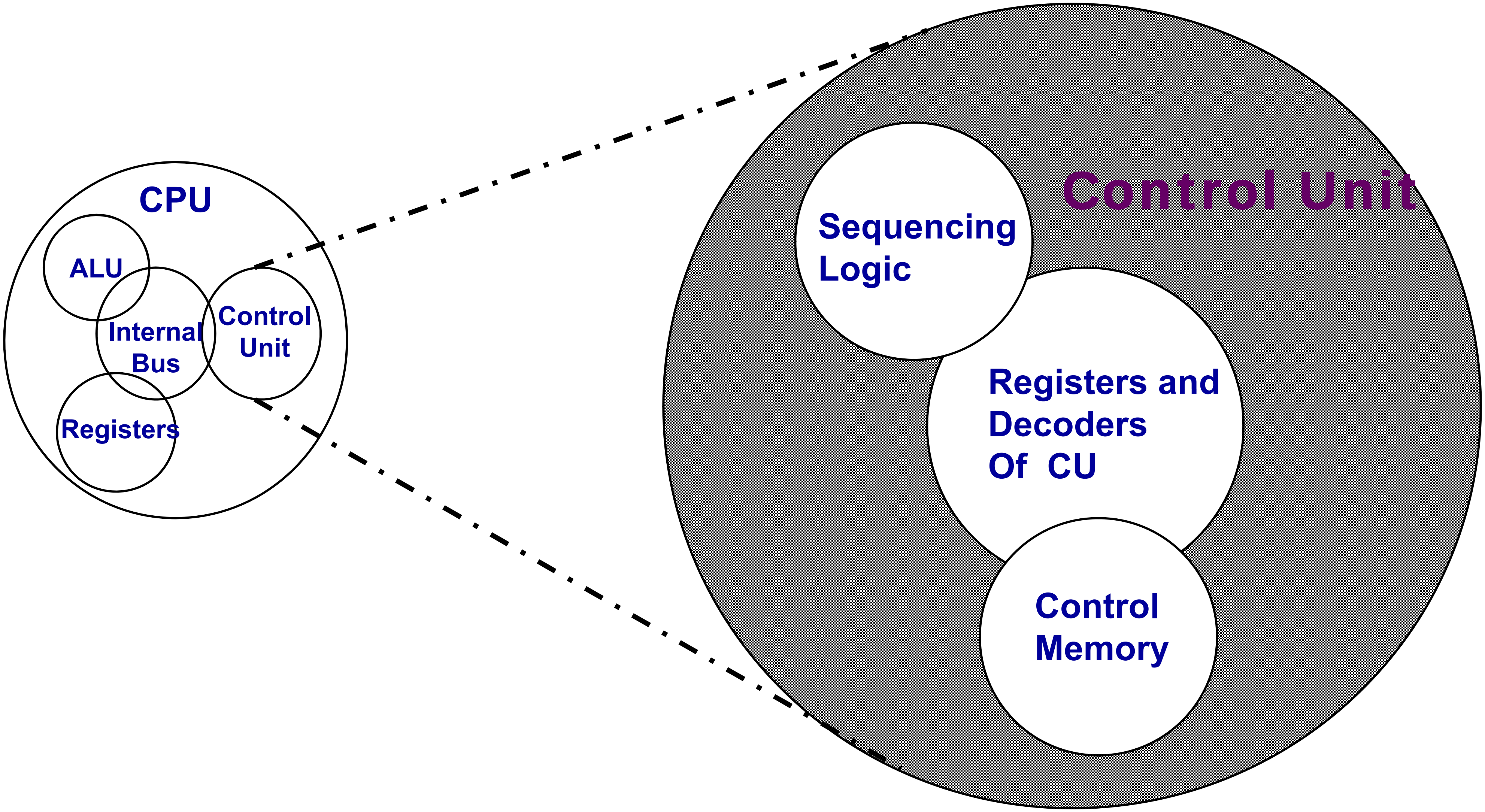
Structure - Top Level



Structure - The CPU



Structure - The Control Unit



- The First 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 Computer (ENIAC) by John Mauchly and J. Presper Eckertat the University of Pennsylvania, 1946
  - The IBM 650 first mass-produced computer. (1955). It was phased out in 1969.

- The Second 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.
- The Third Generation: Integrated Circuit Computers (1965 - 1980)
    - IBM 360
    - DEC PDP-8 and PDP-11
    - Cray-1 supercomputer
- 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).

The von Neumann Model

- The invention of stored program computers has 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.

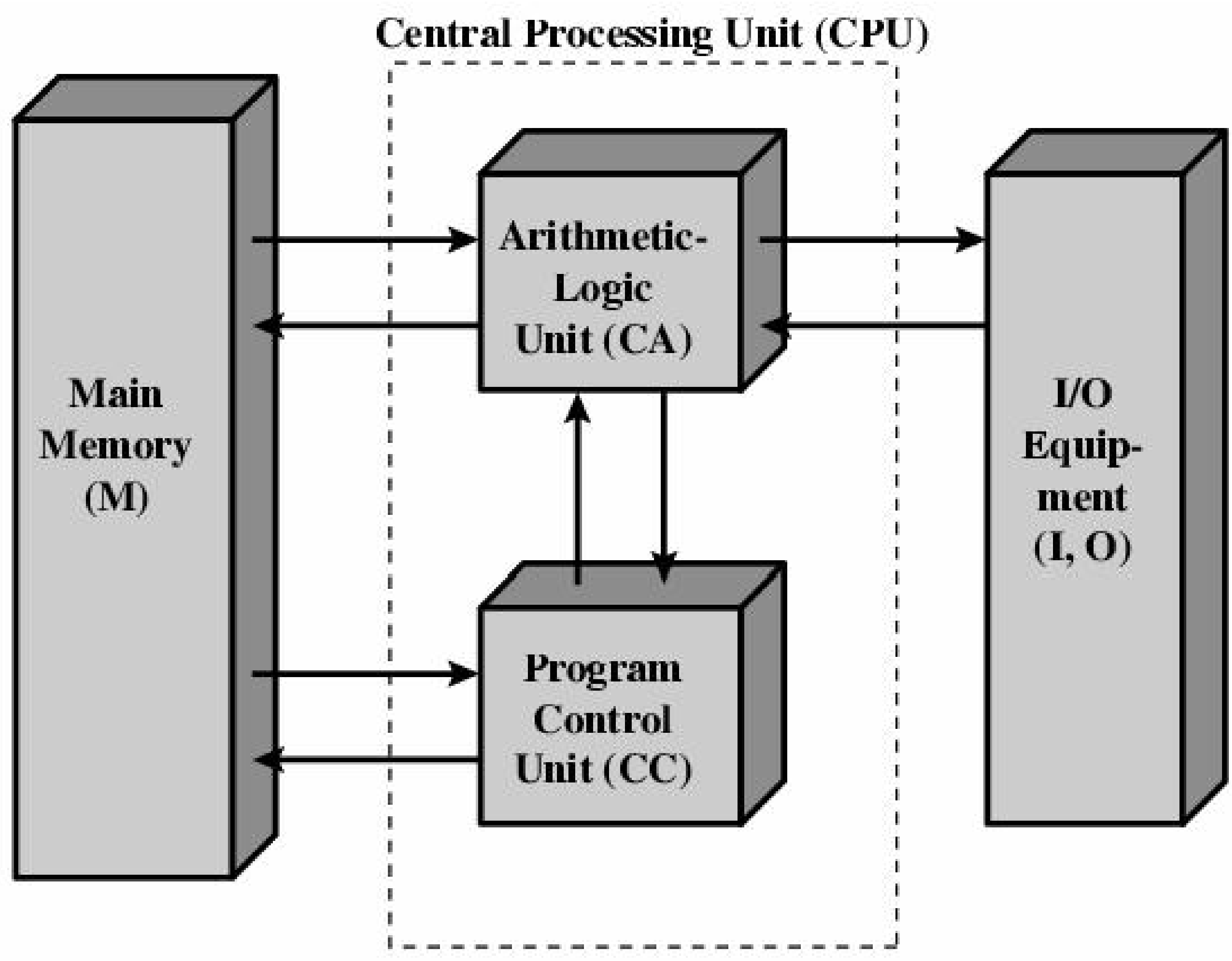
The von Neumann Model

- **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*.



IAS (Princeton) computer model by Von Neumann's group.

IAS computer consists of:

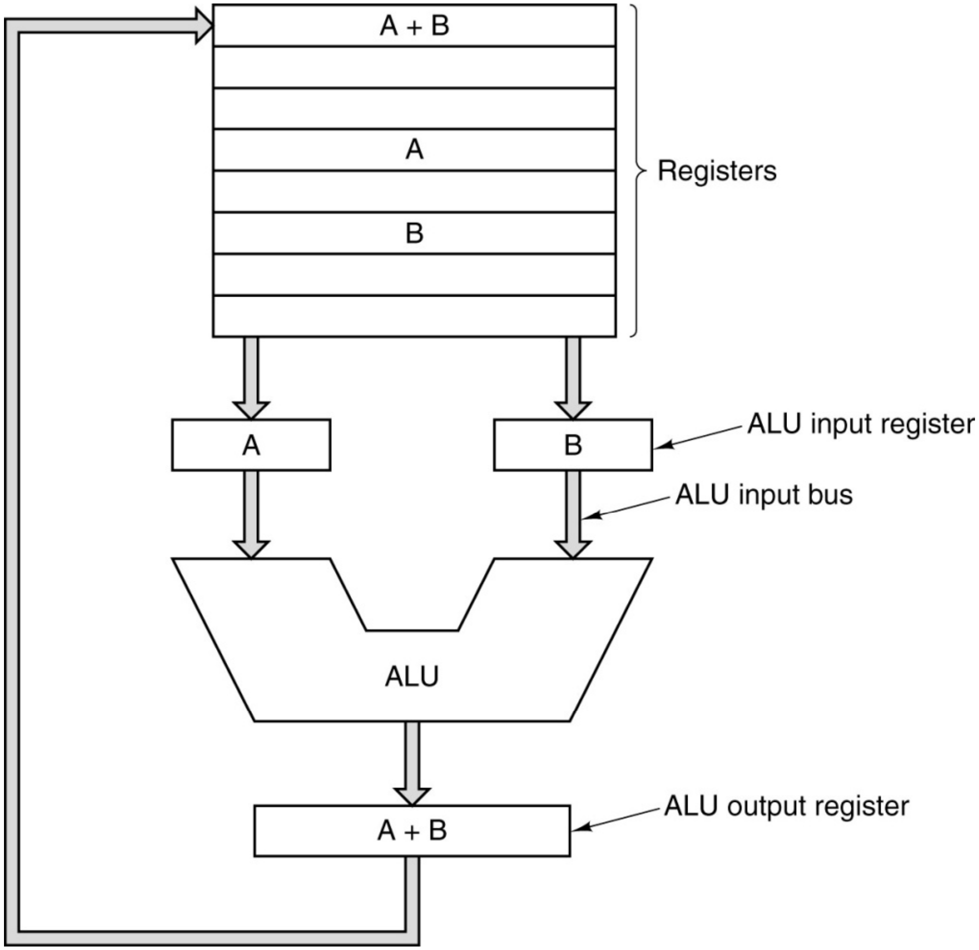
- A main memory, which stores both data and instructions.

- An arithmetic-logical unit (ALU) capable of operating on binary data.

- A control unit, which interprets the instructions in memory and causes them to be executed.

- Input and output (I/O) equipment operated by the control unit.

CPU Organization

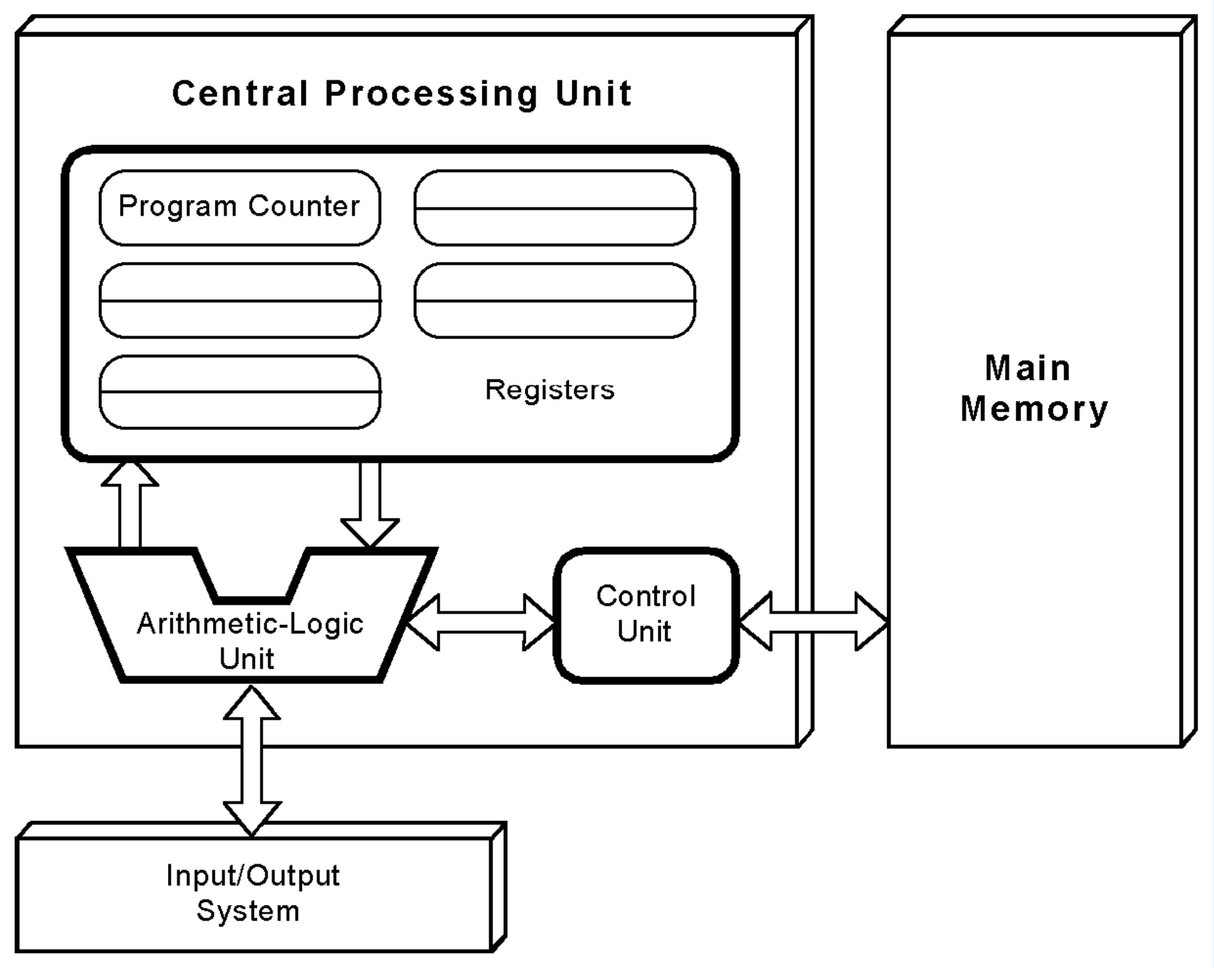


The data path of a typical Von Neumann machine.

The von Neumann Model

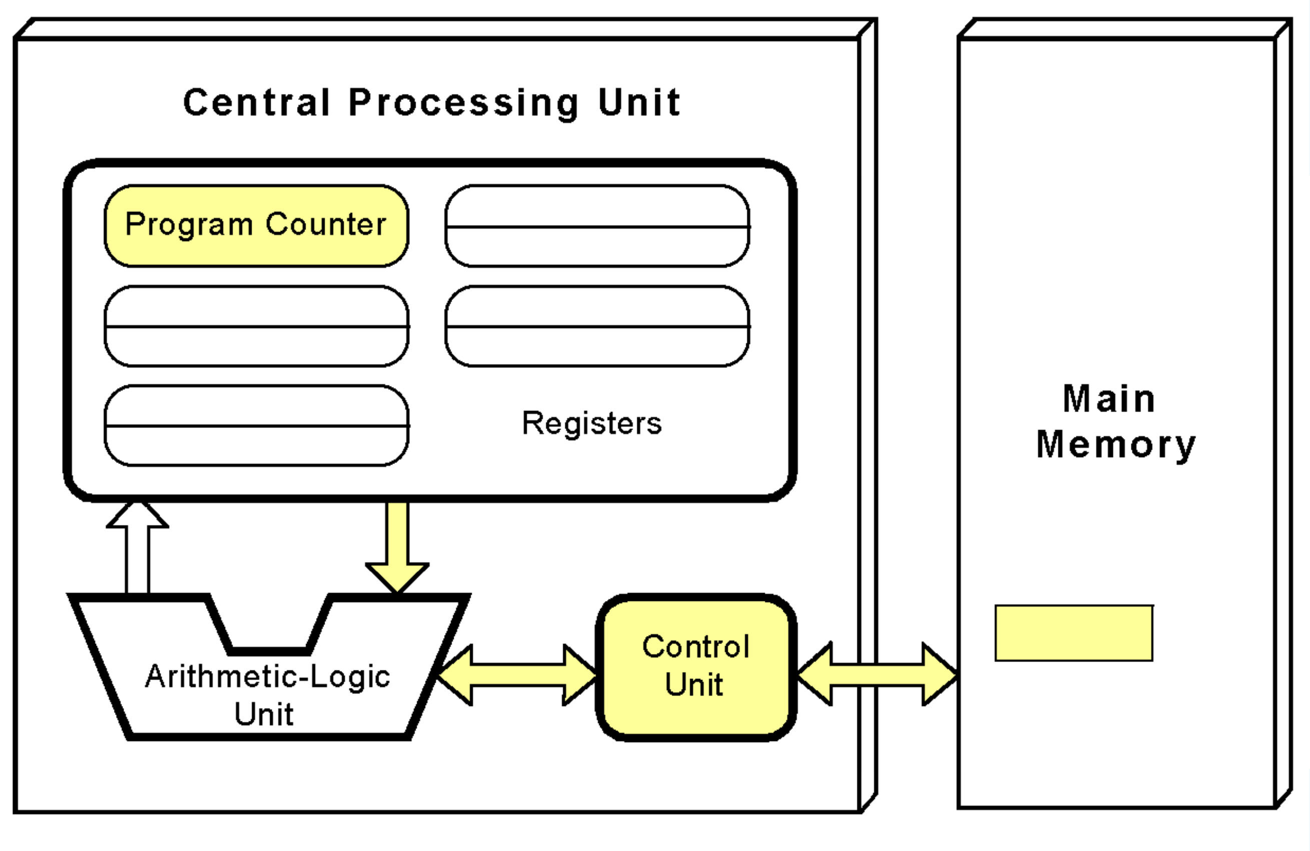
- **This is a general depiction of a von Neumann system:**

- **These computers employ a fetch-decode-execute cycle to run programs as follows . . .**



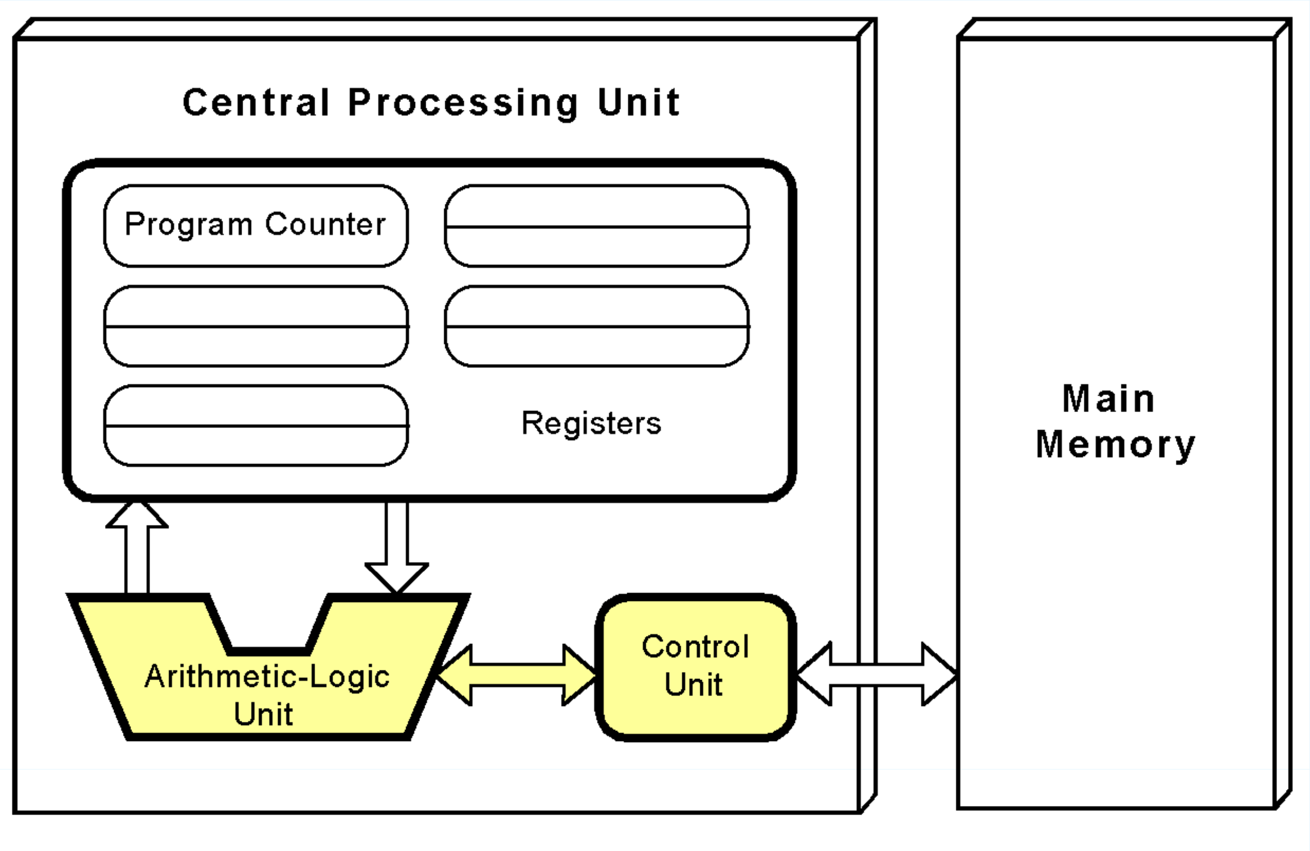
The von Neumann Model

- **The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.**



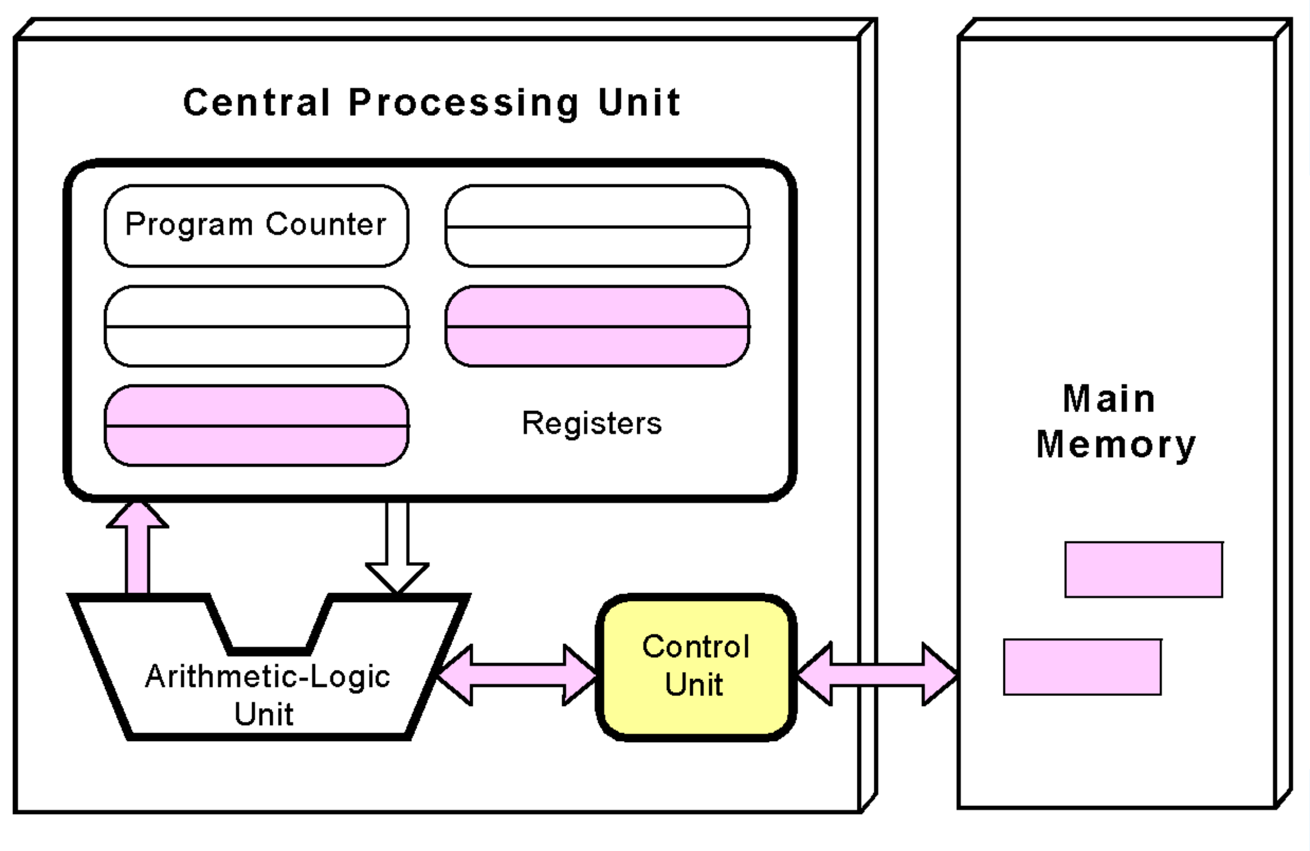
The von Neumann Model

- **The instruction is decoded into a language that the ALU can understand.**



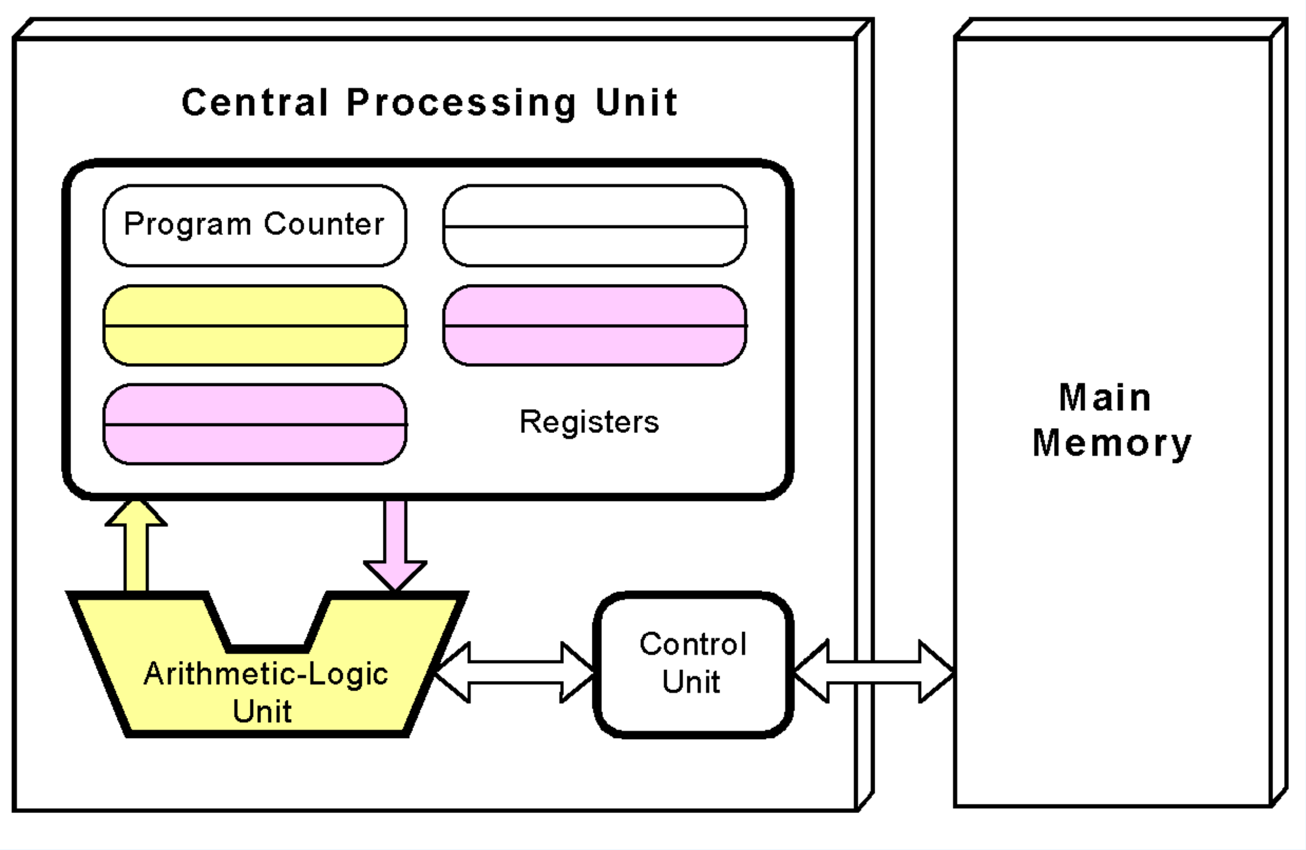
The von Neumann Model

- **Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.**



The von Neumann Model

- **The ALU executes the instruction and places results in registers or memory.**



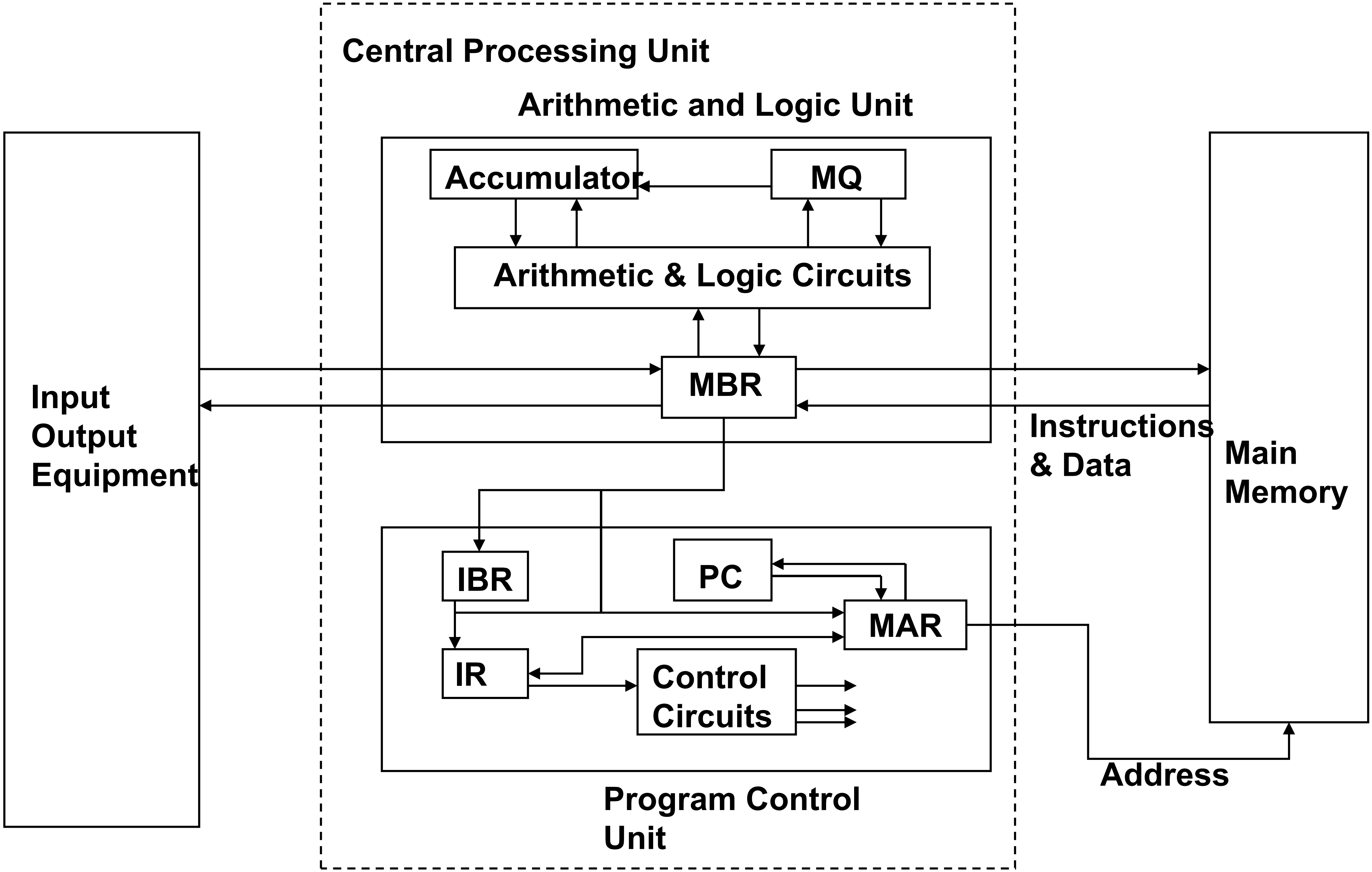
IAS – Von Neumann (1952+)

- **1024 x 40 bit words (= 5KB memory)**
  - **Binary number (2's complement)**
  - **2 x 20 bit instructions**
- **Set of registers (storage in CPU)**
  - **Memory Buffer Register**
  - **Memory Address Register**
  - **Instruction Register**
  - **Instruction Buffer Register**
  - **Program Counter**
  - **Accumulator**
  - **Multiplier Quotient**

Addition time was 62 microseconds and the multiplication time was 713 microseconds.

It was an asynchronous machine.

Structure of IAS



MQ - Multiplier/Quotient

Non-von Neumann Models

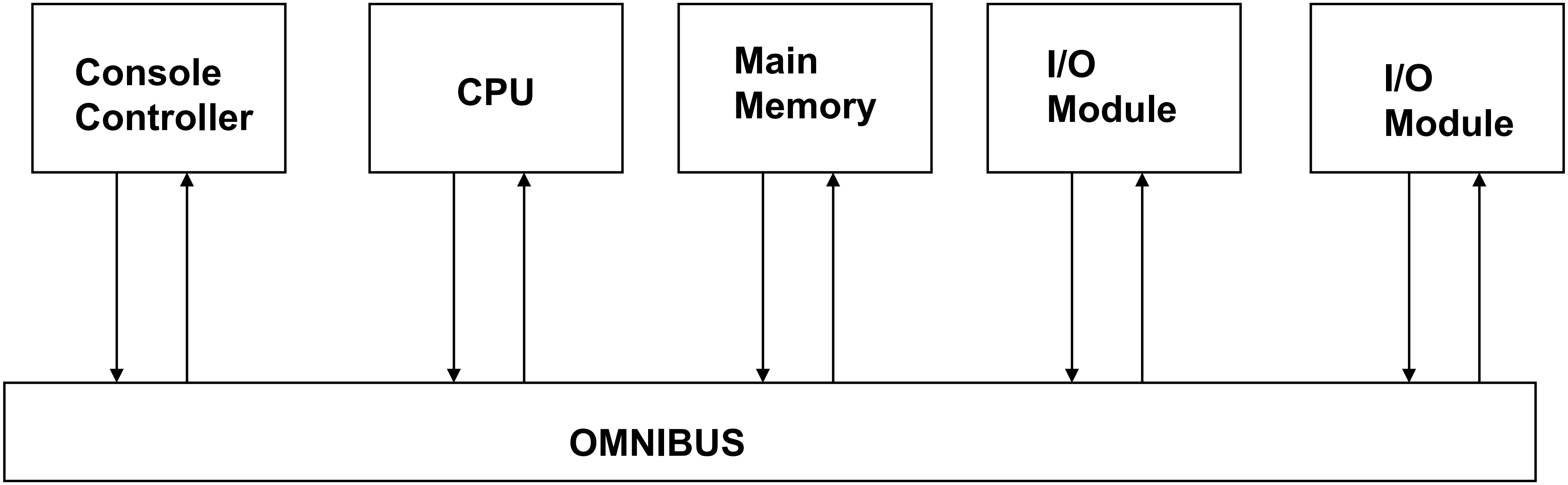
- Conventional stored-program computers have undergone many incremental improvements over the years.

- These improvements include adding specialized buses, floating-point units, and cache memories, to name only a few.

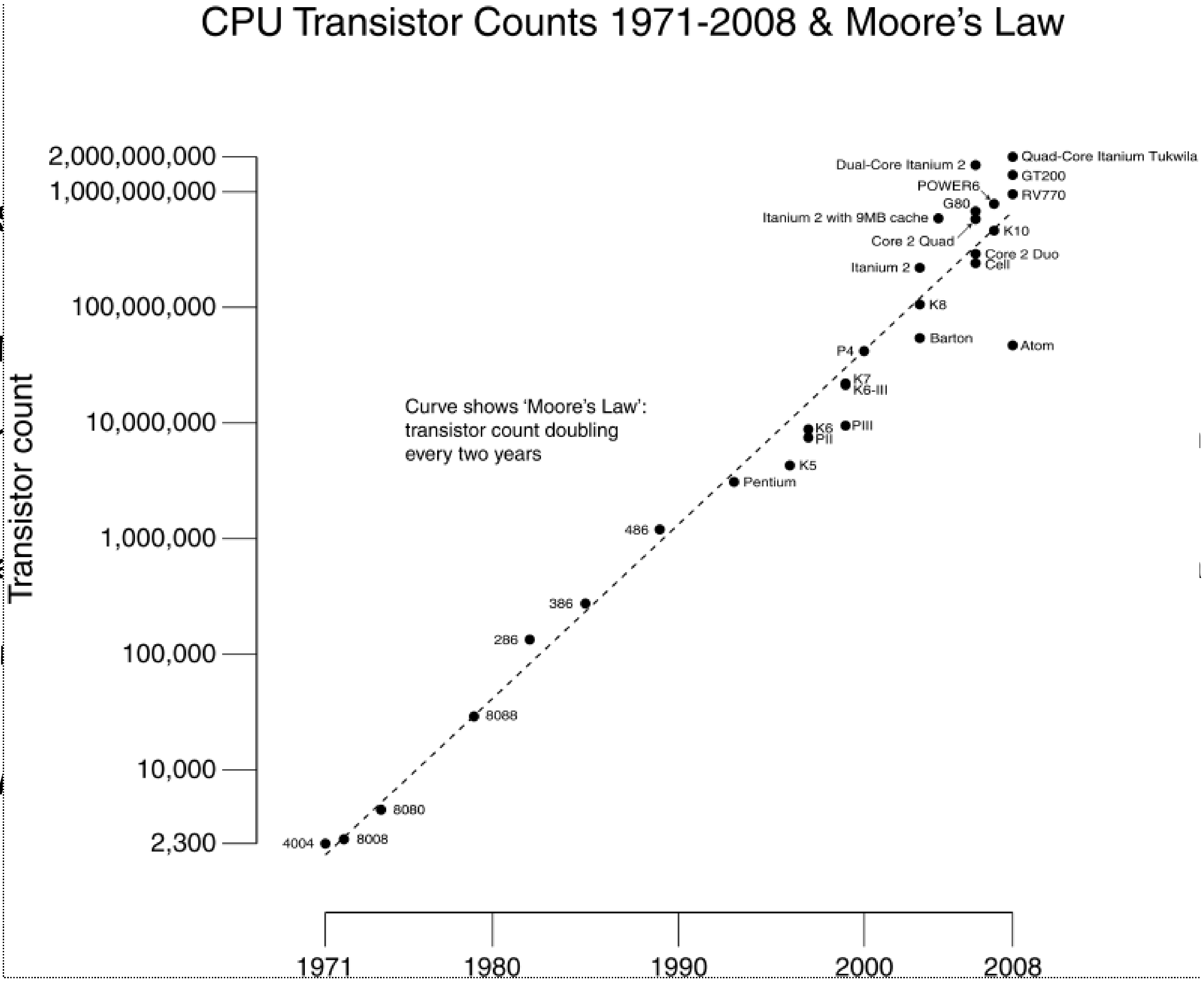
- But enormous improvements in computational power require departure from the classic von Neumann architecture.

- Adding processors is one approach.

DEC - PDP-8 Bus Structure



The Omnibus - a backplane of undedicated slots;


CPU Transistor Counts 1971-2008 & Moore's Law

S

I

Lar **ip**

Very la **chip**

Ult **ip**

M

Architecture vs. Organization

Often used interchangeably – in book titles and as keywords.

Thin line of difference – should be clear as we progress through the course material.

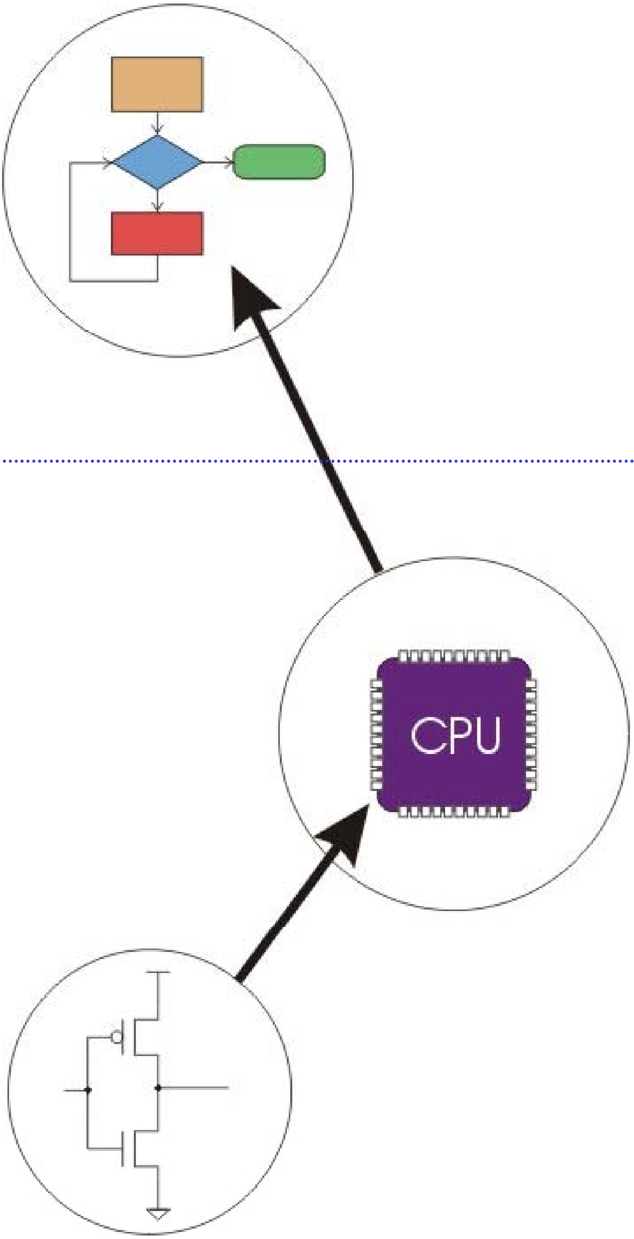
An **instruction set** is a list of all the instructions, that a processor can execute.

Typical Categories of Instructions:

- **Arithmetic** - *add, subtract*
- **Logic** - *and, or and not*
- **Data** - *move, input, output, load* and *store*
- **Control flow** - *goto, if ... goto, call* and *return.*

An **instruction set, or instruction set architecture (ISA),** is the part of the computer architecture related to programming, including the native **data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O**; also includes a specification of the set of **opcodes (machine language)** - the native commands for a particular processor.

Computer System: Layers of Abstraction



Application Program

Algorithms

Language

Software

Hardware

Instruction Set Architecture (and I/O Interfaces)

Microarchitecture

Circuits

Devices

Computer Architecture

Logical aspects of system implementation as seen by the programmer; such as, instruction sets (ISA) and formats, opcode, data types, addressing modes and I/O.

Instruction set architecture (ISA) is different from "microarchitecture", which consist of various processor design techniques used to implement the instruction set.

Computers with different microarchitectures can share a common instruction set.

For example, the Intel Pentium and the AMD Athlon implement nearly identical versions of the x86 instruction set, but have radically different internal designs.

Computer architecture is the conceptual design and fundamental operational structure of a computer system. It is a **functional description** of requirements and design implementations for the various parts of a computer.

It is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.

It deals with the **architectural attributes** like physical address memory, CPU and how they should be designed and **made to coordinate with each other** keeping the goals in mind.

Analogy: “building the design and architecture of house” – architecture may take more time due to planning and then organization is building house by bricks or by latest technology keeping the basic layout and architecture of house in mind.

Computer architecture comes before computer organization.

Computer organization (CO) is how operational attributes are linked together and contribute to realise the architectural specifications.

CO encompasses all physical aspects of computer systems

e.g. Circuit design, control signals, memory types.

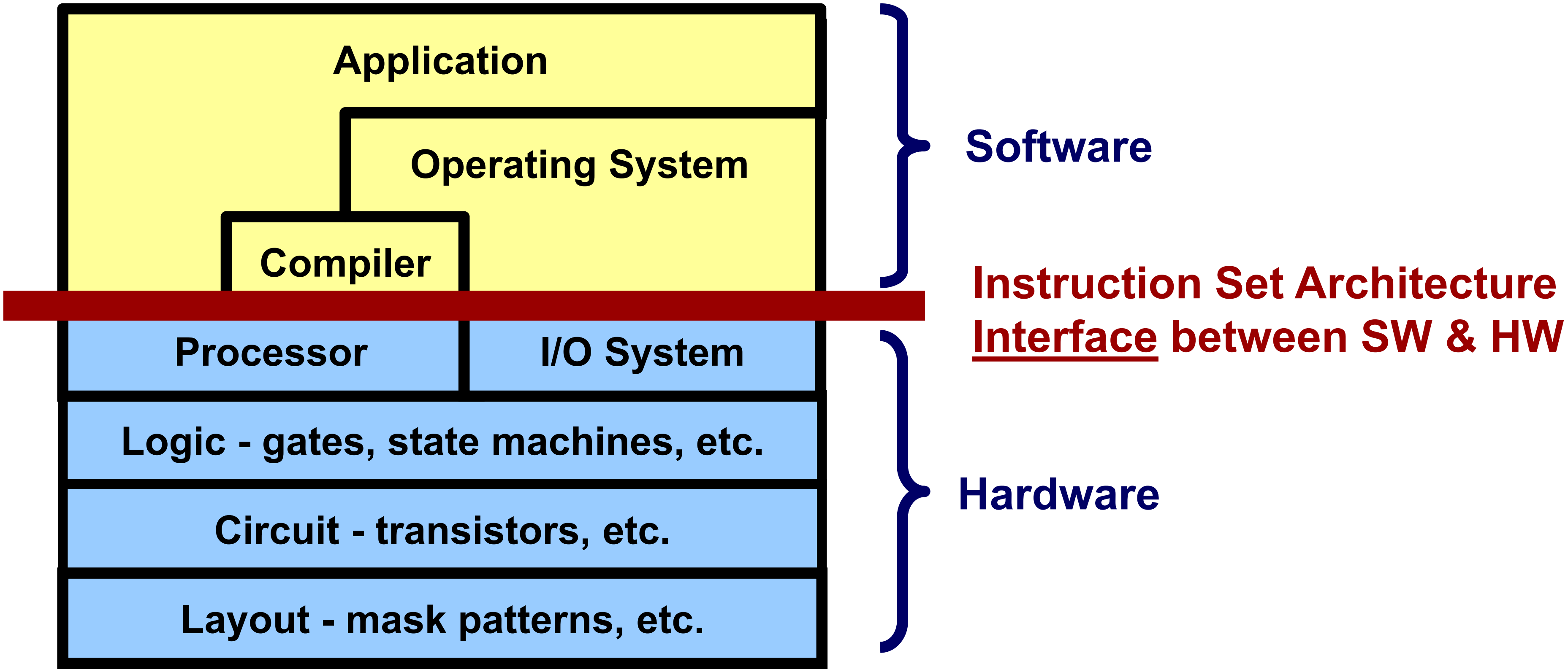
Microarchitecture, also known as **Computer organization** is a lower level, more concrete and detailed, description of the system that involves how the **constituent parts of the system are interconnected and how they interoperate** in order to implement the ISA.

The size of a computer's cache, for example, is an organizational issue that generally has nothing to do with the ISA.

Another example: it is an architectural design issue whether a computer will have a **multiply** instruction. It is an organizational issue whether that instruction will be implemented by a special **multiply unit** or by a mechanism that makes repeated use of the **add unit** of the system.

Instruction Set Architecture (ISA) - The Hardware-Software Interface

- The **most important** abstraction of computer design



Application

Operating System

Compiler

Processor

I/O System

Logic - gates, state machines, etc.

Circuit - transistors, etc.

Layout - mask patterns, etc.

Software

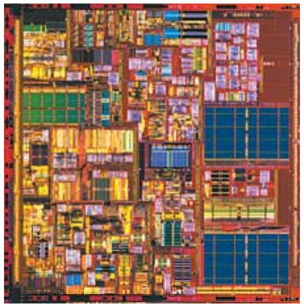
Instruction Set Architecture

Interface between SW & HW

Hardware

Important Building Blocks

- **Microprocessor**

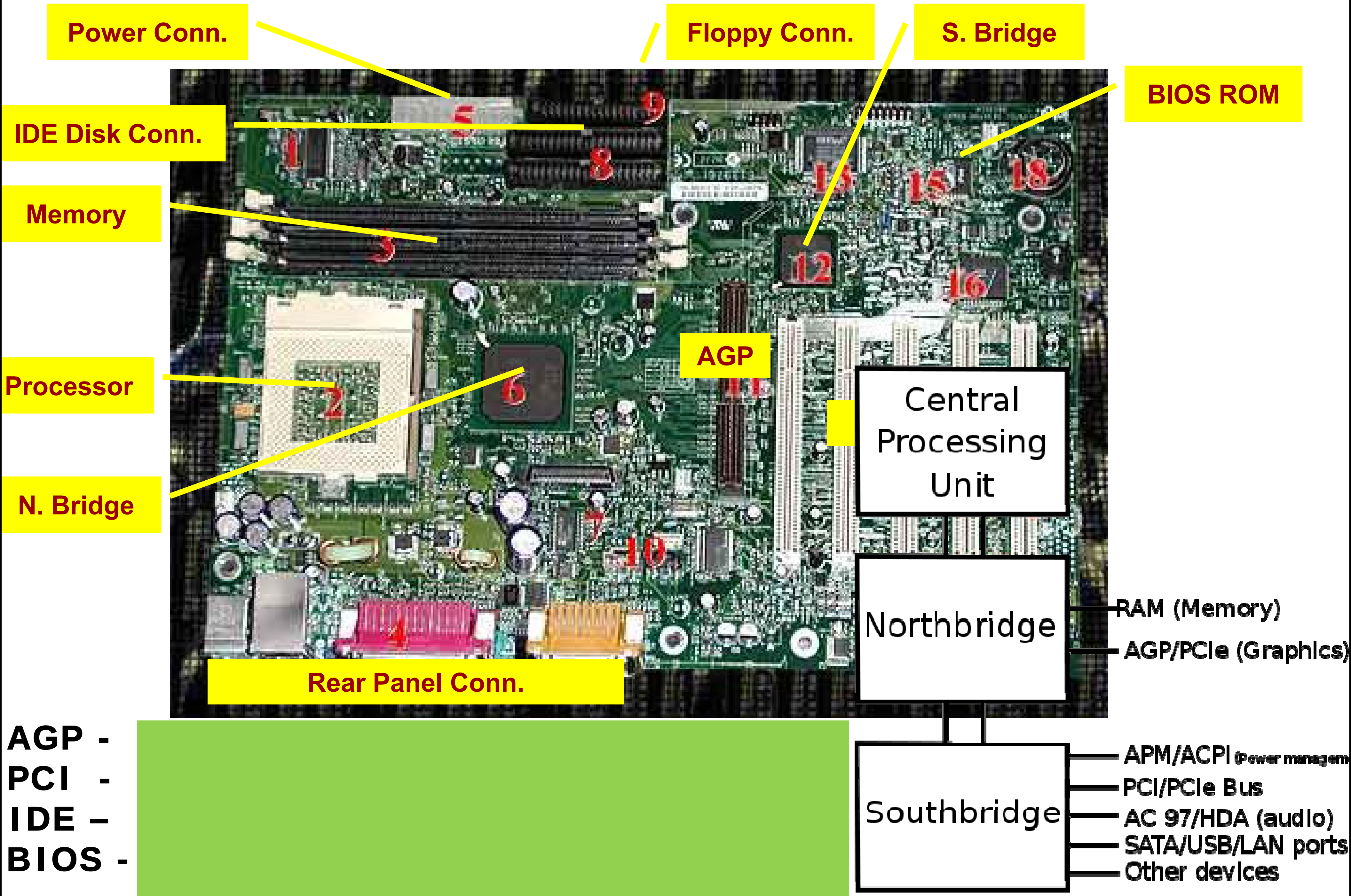


- **Memory**

- **Mass Storage (Disk)**

- **Network Interface**

Typical Motherboard (Pentium III)



Central Processing Unit

Northbridge

RAM (Memory)

AGP/PCIe (Graphics)

Southbridge

APM/ACPI (Power managem

PCI/PCIe Bus

AC 97/HDA (audio)

SATA/USB/LAN ports

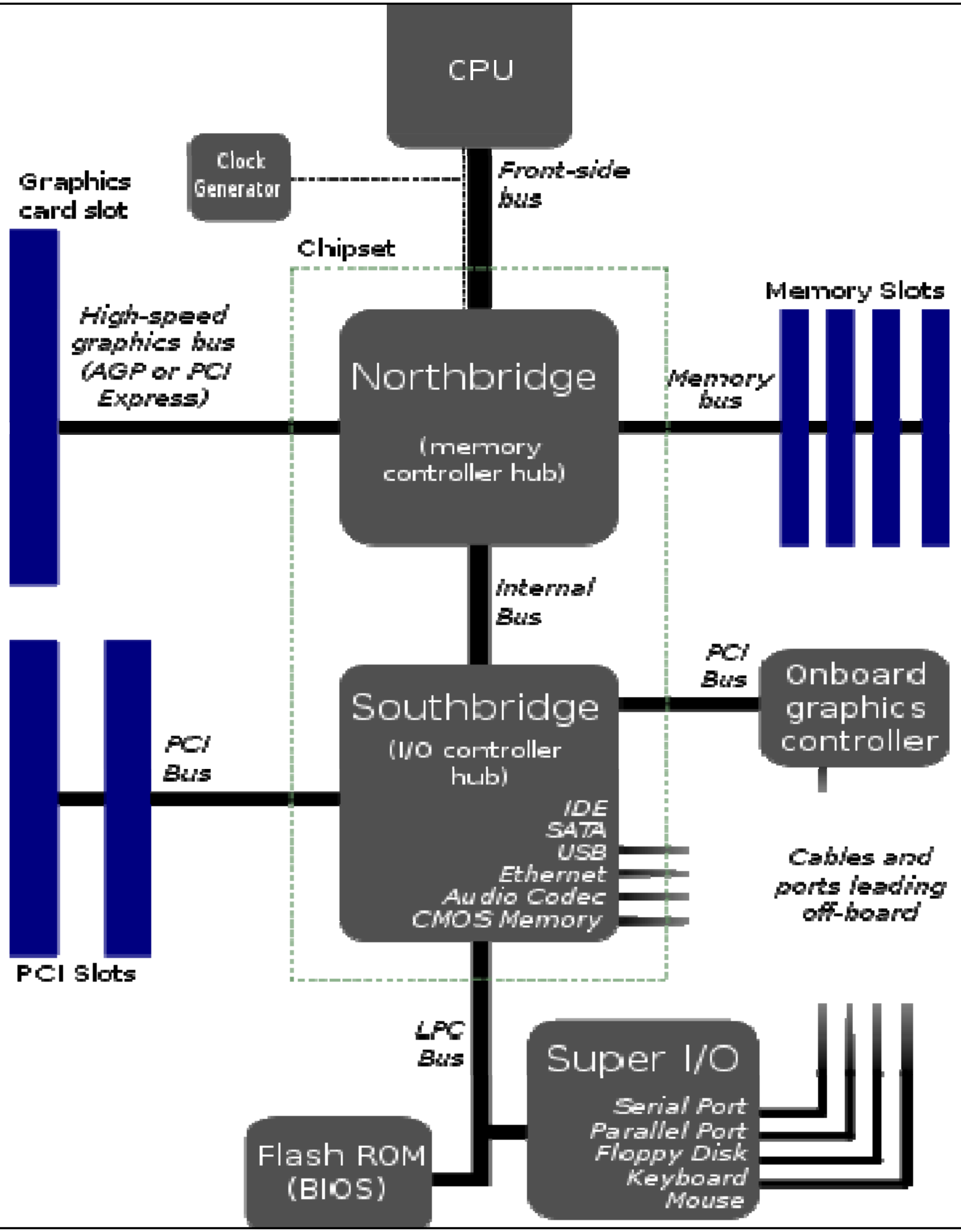
Other devices

AGP -

PCI -

IDE –

BIOS -


CPU
Clock Generator
Front-side bus
Graphics card slot
Chipset
Memory Slots
High-speed graphics bus (AGP or PCI Express)
Northbridge
(memory controller hub)
Memory bus
Internal Bus
PCI Bus
Onboard graphics controller
PCI Bus
Southbridge
(I/O controller hub)
IDE
SATA
USB
Ethernet
Audio Codec
CMOS Memory
Cables and ports leading off-board
PCI Slots
LPC Bus
Super I/O
Serial Port
Parallel Port
Floppy Disk
Keyboard
Mouse
Flash ROM (BIOS)

Why design issues matter:

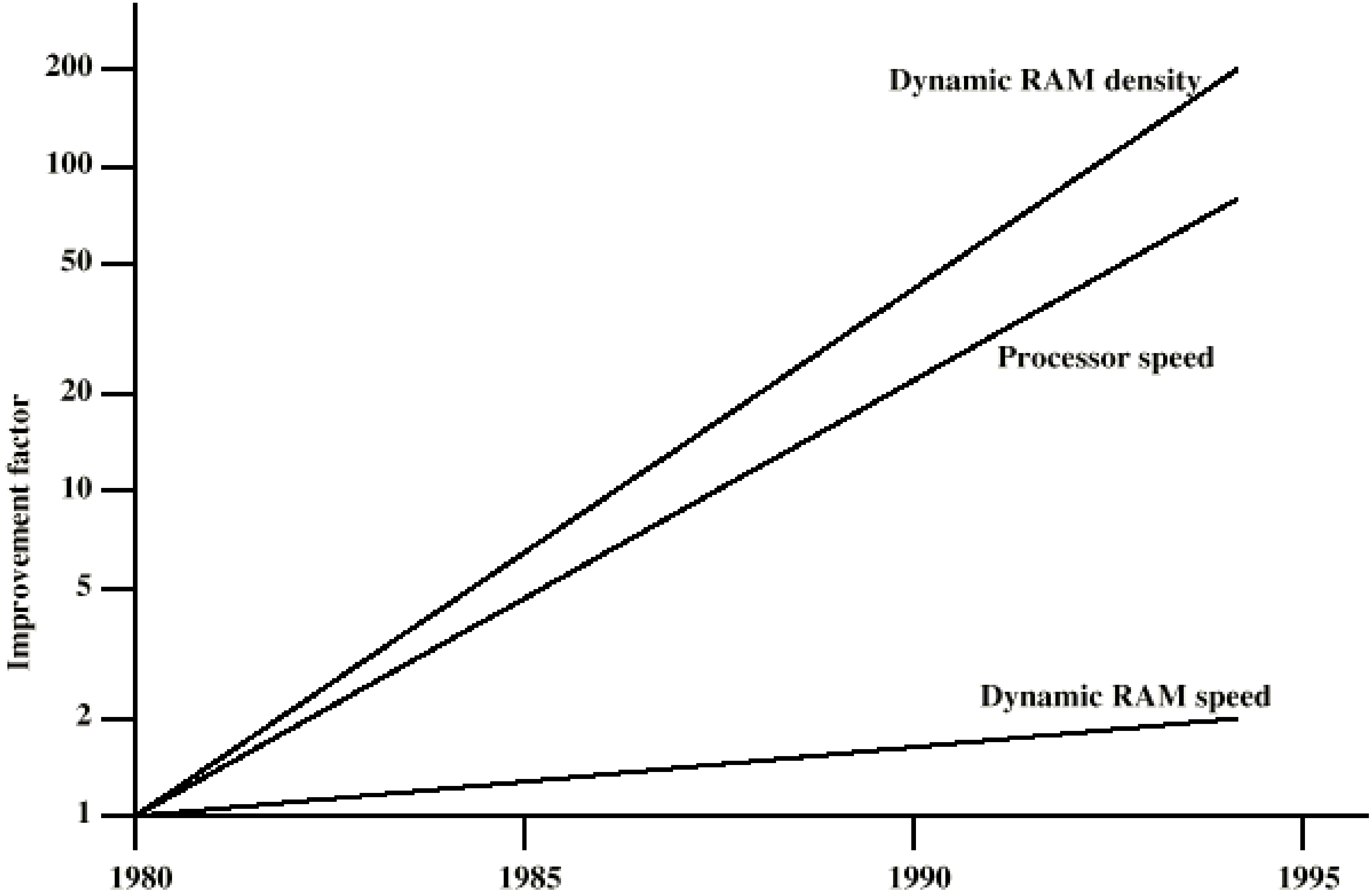
- Cannot assume infinite speed and memory.
- Speed mismatch between memory and processor
- handle bugs and errors (bad pointers, overflow etc.)
- multiple processors, processes, threads
- shared memory
- disk access
- better performance with reduced power
-

Enhancing Performance (speed)

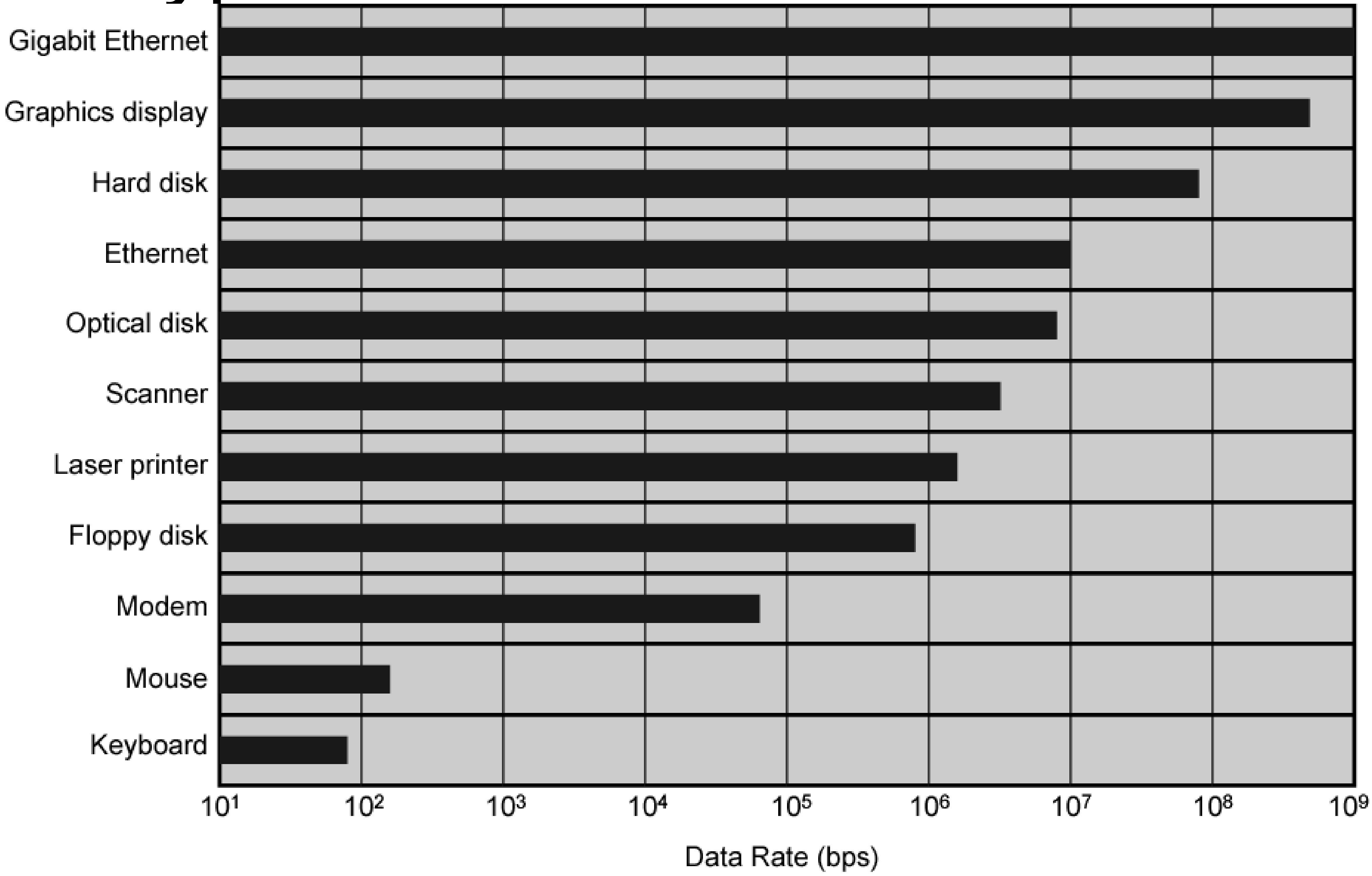
- Pipelining
- On board cache
- On board L1 & L2 cache

- Branch prediction
- Data flow analysis (in compilers)
- Speculative execution

DRAM and Processor Characteristics



Typical I/O Device Data Rates



Performance Analysis

A basic performance equation:

$$T = \frac{N * S}{R}$$

T – processor time required to execute a program (not total time used);

N - Actual number of machine instructions (including that due to loops);

S – Average No. of clock cycles/instruction;

R – Cycle/sec

Earlier measures –

MIPS (Millions of Instructions per sec.)

MFLOPS – Million floating point operations per sec.

CPI – Cycles per Instruction;

IPC – Instructions per cycle = 1/CPI;

Speedup = (Earlier execution time) / (Current execution time);

The Unix “time <a.out>” command gives:

“User CPU” time; “system (kernel) CPU” time and the “elapsed” real-time.

e.g. A: 0.327u 0.010s 0:01.15; %-age elapsed time in CPU:

$$\frac{0.327+0.01}{75}=0.45\%$$

e.g. B: 90.7u 12.9s 0:12.39; %-age elapsed time in CPU:

$$\frac{90.7+12.9}{159}=65\%$$

A better situation, for exploitation of CPU time.

CPU execution time for a program =

$$\frac{\text{CPU clock cycles}}{\text{Clock rate}}=\text{CPU clock cycles}*\text{clock cycle time}$$

CPU execution time "for a program" =

$$\frac{\text{CPU clock cycles}}{\text{Clock rate}} = \text{CPU clock cycles} * \text{clock cycle time}$$

CPU clock cycles = No. of instructions * Avg. clock cycles/instruction

$$\text{CPU clock cycles} = \sum_{i=1}^{n} N_i * \text{CPI}_\text{i}$$

CPI = cycles/instruction; N – No. of instructions;

- **CPU execution time for program**

= <u>Instruction Count</u> x <u>CPI</u> x <u>Clock Cycle Time</u>

A better measure: $\frac{Exec\_Time(A)}{Exec\_Time(B)}$ $\quad Exec\_time = \frac{1}{n}\sum_{i=1}^{n} Time_i$

$$\frac{seconds}{program} = \frac{Instructions}{program} * \frac{clock\ cycle}{Instruction} * \frac{\text{seconds}}{clock\ cycle}$$

Performance - SPECS

- CPU
- Graphics/Workstations
- MPI/OMP

  (Orthogonal Multi-Processor)
- Java Client/Server
- Mail Servers
- Network File System
- Power
- SIP

(Session Initiation Protocol)
- SOA

(Service Oriented Architecture)
- Virtualization
- Web Servers

SPEC MPI2007 focuses on performance of compute intensive applications using the Message-Passing Interface (MPI), which means these benchmarks emphasize the performance of:

- computer processor (CPU),
- number of computer processors,
- MPI Library,
- communication interconnect,
- memory architecture,
- compilers, and
- shared file system.

Not for **Graphics, O/S and I/O.**

MPI2007 is SPEC's benchmark suite for evaluating MPI-parallel, floating point, compute intensive performance across a wide range of cluster and SMP (Symmetric multi-processing) hardware.

CFP2006 is used for measuring and comparing compute-intensive floating point performance.

SPEC rating (ratio) = TR / TC;

TR = Running time of the Reference Computer;

TC = Running time of the Computer under test;

$$SPEC = \left(\prod_{i=1}^{n} SPEC_i \right)^{1/n}$$

Higher the SPEC score, better the performance.

n – No. of programs in the SPEC Suite.

Benchmark	**Language**	**Application Area**
104.milc	C	Quantum Chromodynamics
107.leslie3d	Fortran	Computational Fluid Dynamics CFD
113.GemsFDTD	Fortran	Computational Electromagnetics
115.fds4	C/Fortran	CFD: Fire dynamics simulator
121.pop2	C/Fortran	Climate Modeling
122.tachyon	C	Graphics: Ray Tracing
126.lammps	C++	Molecular Dynamics
127.wrf2	C/Fortran	Weather Forecasting
128.GAPgeofem	C/Fortran	Heat Transfer using FEM
129.tera_tf	Fortran	3D Eulerian Hydrodynamics
130.socorro	C/Fortran	Molecular Dynamics
132.zeusmp2	C/Fortran	Computational Astrophysics
137.lu	Fortran	Implicit CFD

From first to fifth/sixth generation systems, the following factors were also taken into consideration, to improve the performance.

- Reduced Power dissipation

- Reduced Space Area

- More increase in Speed and Registers (GPRs) for operation

- More memory size

- Use of Cache

- Set of cores on CPU

- pipelining and special MMX hardware.

-

Increase in CPU performance, may come from three factors:

- **Increase in clock rate**
- **Improvement in processor design to lower CPI**
- **Compiler enhancements for lower average CPI**
- **Better memory organization**
-

Key terminologies:

- Microcontroller
- CPU design
- Hardware description language
- Von-Neumann architecture
- Multi-core (computing)
- Datapath
- Dataflow architecture
- Stream processing
- Instruction-level parallelism (ILP)
- Vector processor
- Control Path
- ALU, FPU, GPU etc.
- Pipelining
- Cache
- Superscalar
- Out-of-order execution
- Register renaming
- multi-threading
- RISC, CISC
- Addressing Modes
- Instruction set

- **SIMD, MIMD**
- **Flynn’s taxonomy**
- **MMX instructions**
-