Introduction

Computer Architecture

Riad Bourbia

Computer Sciences department

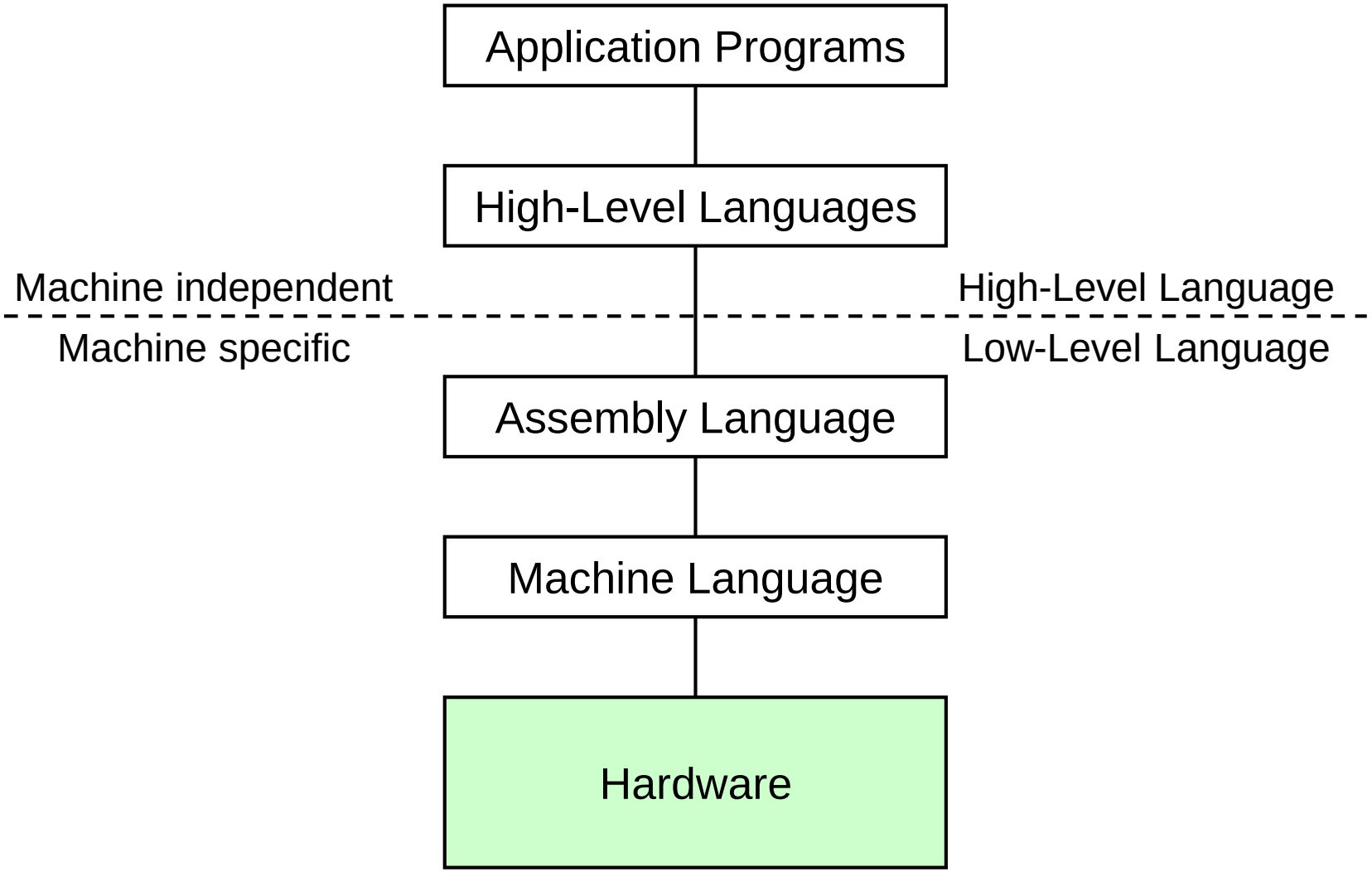
Guelma University

[Adapted from slides of Dr. A. El-maleh]

Next . . .

- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Technology Improvements
- Programmer's View of a Computer System

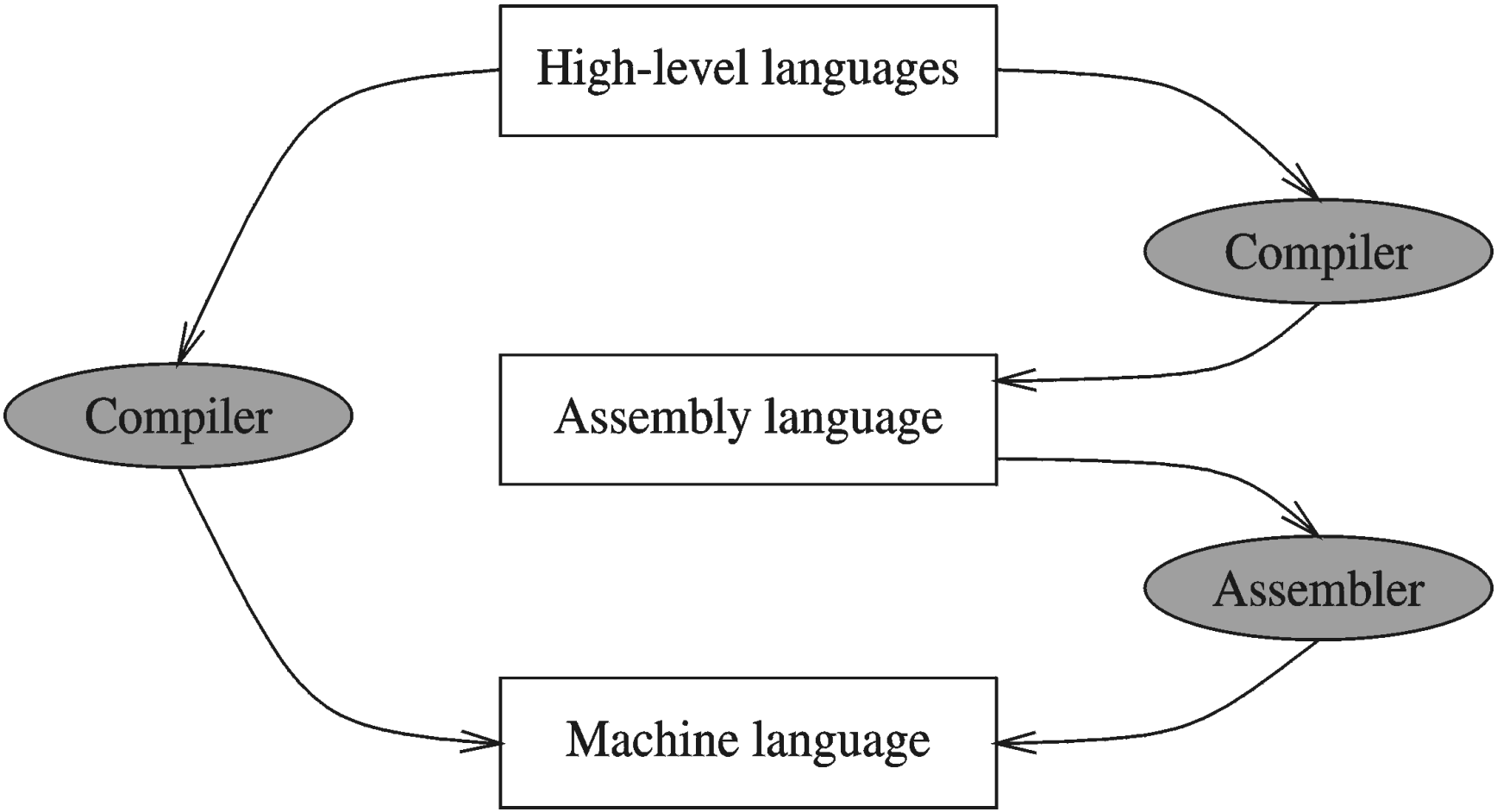
A Hierarchy of Languages



Assembly and Machine Language

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler

Compiler and Assembler



Instructions and Machine Language

- Each command of a program is called an instruction (it instructs the computer what to do).
- Computers only deal with binary data, hence the instructions must be in binary format (0s and 1s) .
- The set of all instructions (in binary form) makes up the computer's machine language. This is also referred to as the instruction set.

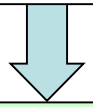
Instruction Fields

- Machine language instructions usually are made up of several fields. Each field specifies different information for the computer. The major two fields are:
- Opcode field which stands for operation code and it specifies the particular operation that is to be performed.
  - Each operation has its unique opcode.
- Operands fields which specify where to get the source and destination operands for the operation specified by the opcode.
  - The source/destination of operands can be a constant, the memory or one of the general-purpose registers.

Translating Languages

Program (C Language):

```
swap(int v[], int k) {
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Compiler

MIPS Assembly Language:

```
sll $2,$5, 2
add $2,$4,$2
lw  $15,0($2)
lw  $16,4($2)
sw  $16,0($2)
sw  $15,4($2)
jr  $31
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

A statement in a high-level language is translated typically into several machine-level instructions

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

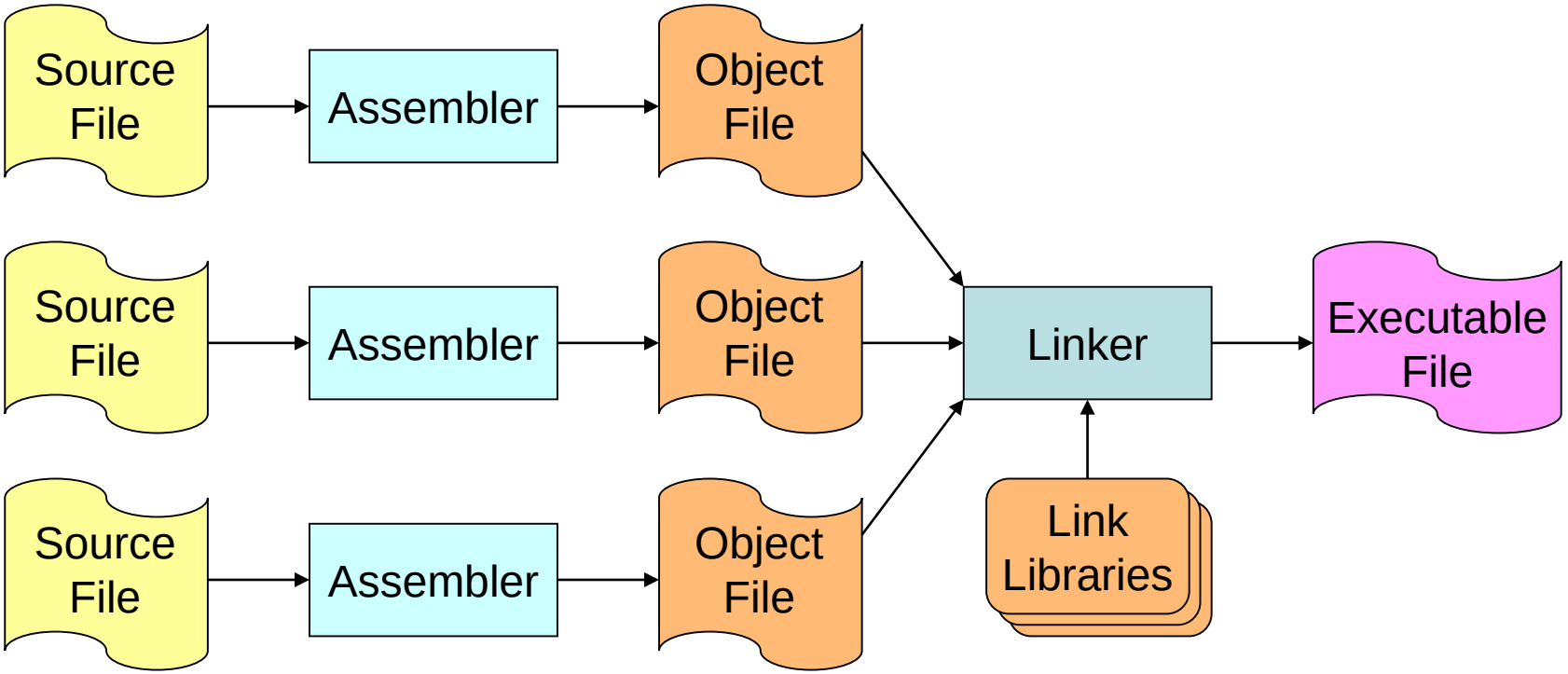
Why Learn Assembly Language?

- Many reasons:
  - Accessibility to system hardware
  - Space and time efficiency
- Accessibility to system hardware
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- Space and Time efficiency
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code

Assembly Language Programming Tools

- Editor
  - Allows you to create and edit assembly language source files
- Assembler
  - Converts assembly language programs into object files
  - Object files contain the machine instructions
- Linker
  - Combines object files created by the assembler with link libraries
  - Produces a single executable program
- Debugger
  - Allows you to trace the execution of a program
  - Allows you to view machine instructions, memory, and registers

Assemble and Link Process

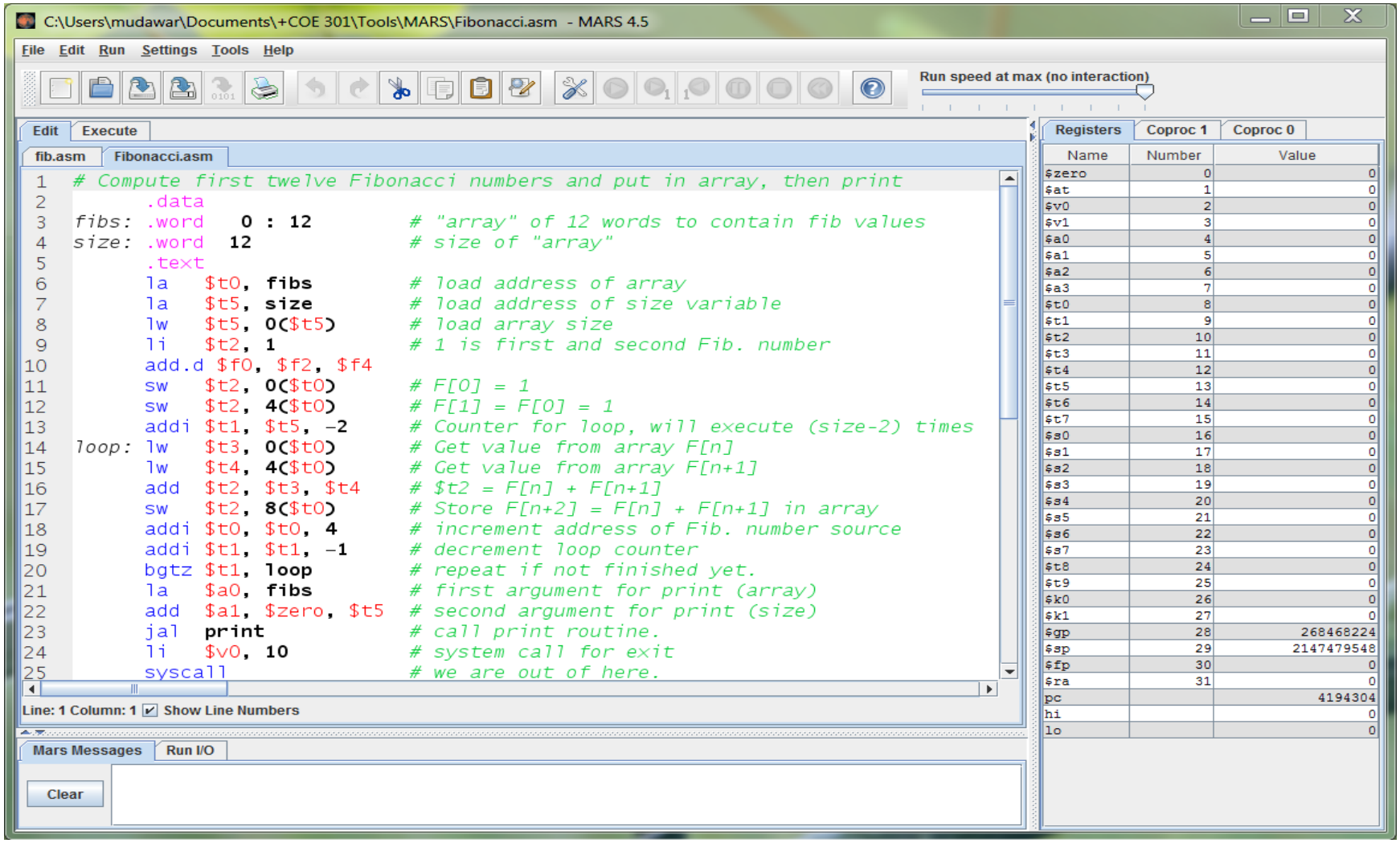


A project may consist of multiple source files

Assembler translates each source file separately into an object file

Linker links all object files together with link libraries

MARS Assembler and Simulator Tool



MARS Assembler and Simulator Tool

- Simulates the execution of a MIPS program
  - No direct execution on the underlying Intel processor
- Editor with color-coded assembly syntax
- Assembler

  Converts MIPS assembly language programs into object files
- Debugger
  - Allows you to trace the execution of a program and set breakpoints
  - Allows you to view machine instructions, edit registers and memory

Next . . .

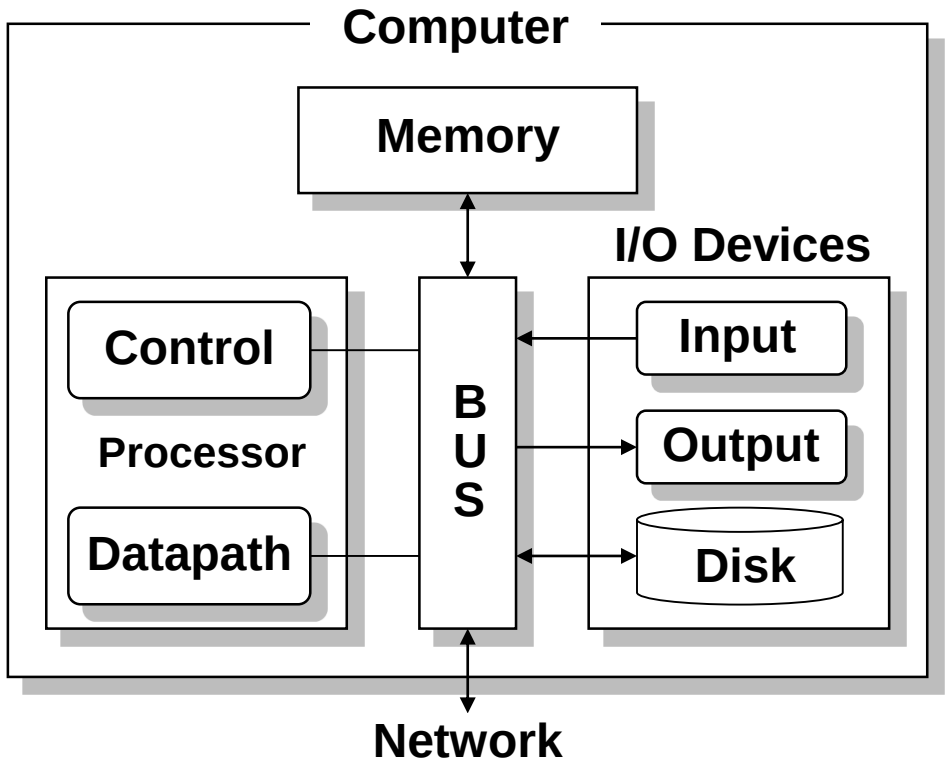
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Technology Improvements
- Programmer's View of a Computer System

Von Neumann Architecture

John von Neumann (1903–1957) was a Hungarian-American mathematician and physicist.

Components of a Computer System

- Processor
  - Datapath
  - Control
- Memory & Storage
  - Main Memory
  - Disk Storage
- Input devices
- Output devices

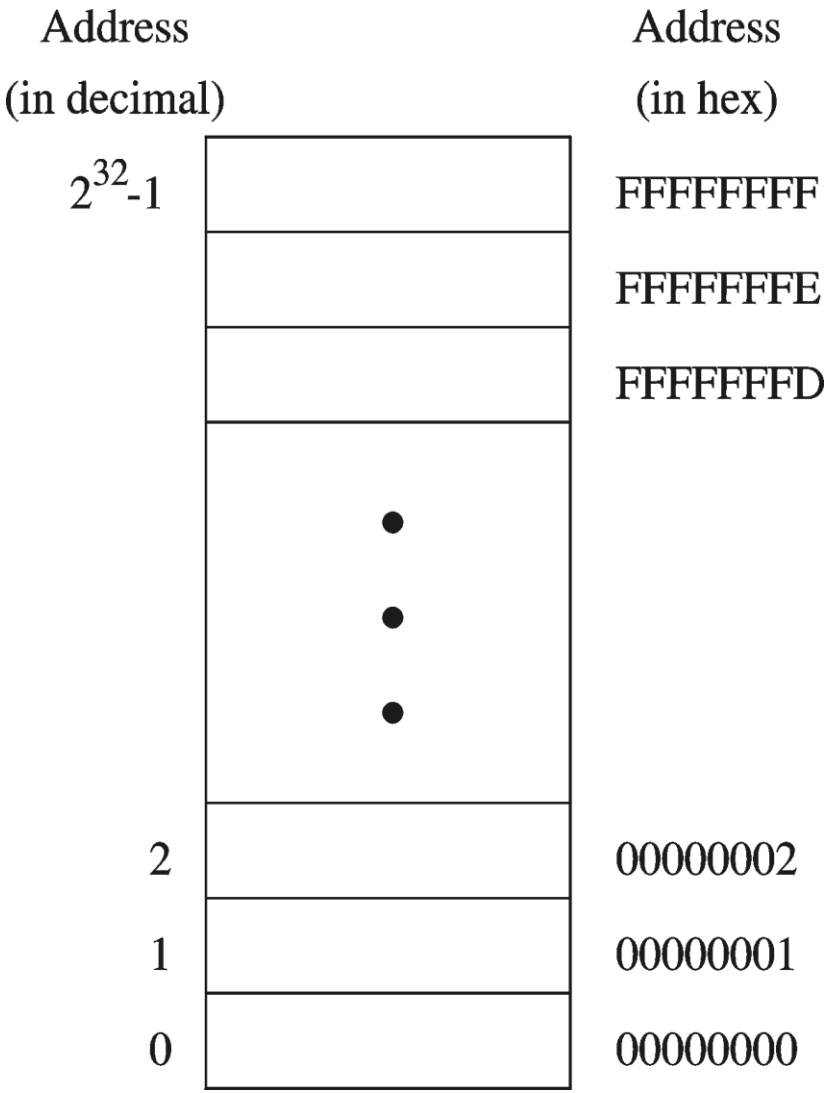


- Bus: Interconnects processor to memory and I/O
- Network: newly added component for communication

Memory

- Ordered sequence of bytes
  - The sequence number is called the memory address
- Byte addressable memory
  - Each byte has a unique address
  - Supported by almost all processors
- Physical address space
  - Determined by the address bus width
  - Pentium has a 32-bit address bus
    - Physical address space = **4GB = 2^{32} bytes**
  - Itanium with a 64-bit address bus can support
    - Up to **2^{64} bytes** of physical address space

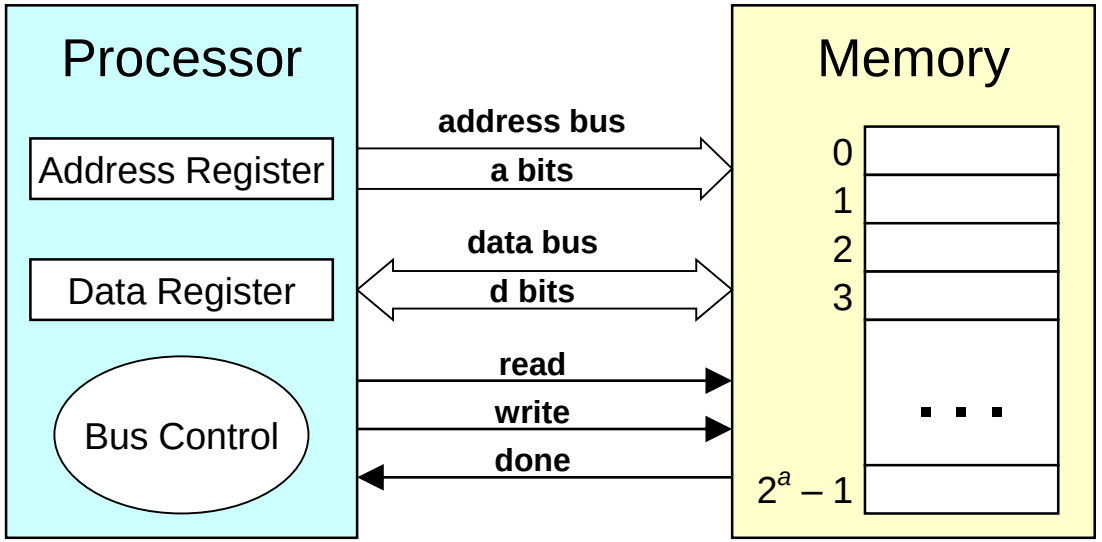
Address Space



Address Space is the set of memory locations (bytes) that can be addressed

Address, Data, and Control Bus

- Address Bus
  - Memory address is put on address bus
  - If memory address = *a* bits then 2^a locations are addressed
- Data Bus: bi-directional bus
  - Data can be transferred in both directions on the data bus
- Control Bus
  - Signals control transfer of data
  - Read request
  - Write request
  - Done transfer



Memory Devices

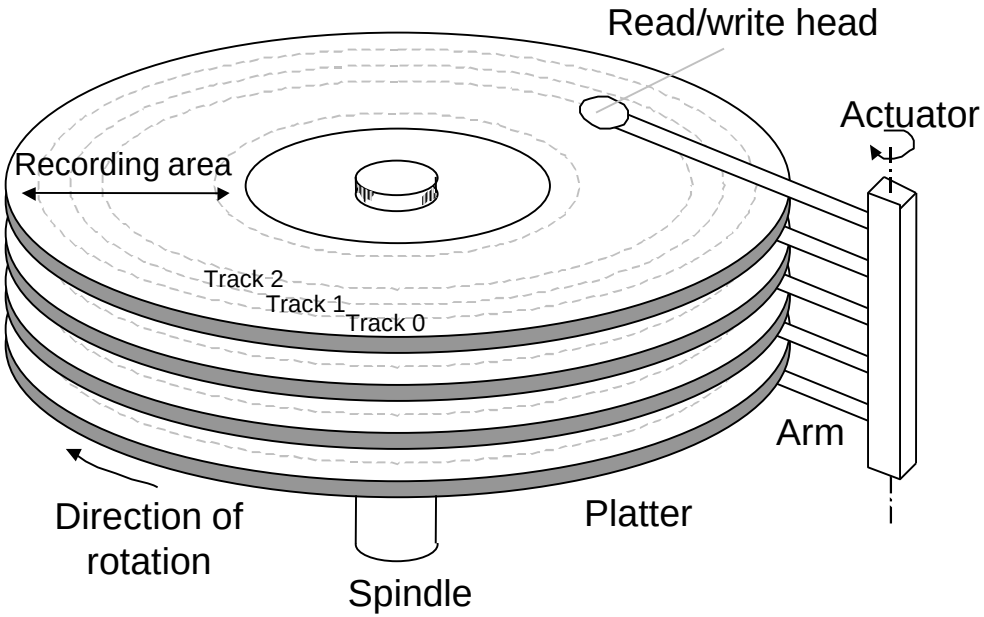
- Volatile Memory Devices
  - Data is lost when device is powered off
  - RAM = Random Access Memory
  - DRAM = Dynamic RAM
    - 1-Transistor cell + trench capacitor
    - Dense but slow, must be refreshed
    - Typical choice for main memory

  - SRAM: Static RAM
    - 6-Transistor cell, faster but less dense than DRAM
    - Typical choice for cache memory
- Non-Volatile Memory Devices
  - Stores information permanently
  - ROM = Read Only Memory
  - Used to store the information required to startup the computer
  - Many types: ROM, EPROM, EEPROM, and FLASH
  - FLASH memory can be erased electrically in blocks

Magnetic Disk Storage

A Magnetic disk consists of a collection of platters

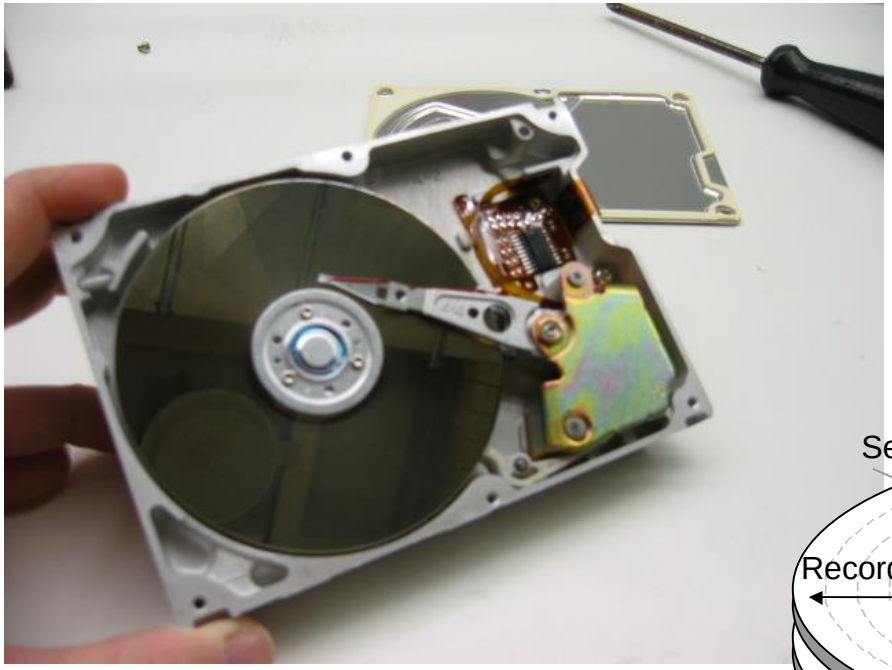
Provides a number of recording surfaces



Arm provides read/write heads for all surfaces

The disk heads are connected together and move in conjunction

Magnetic Disk Storage

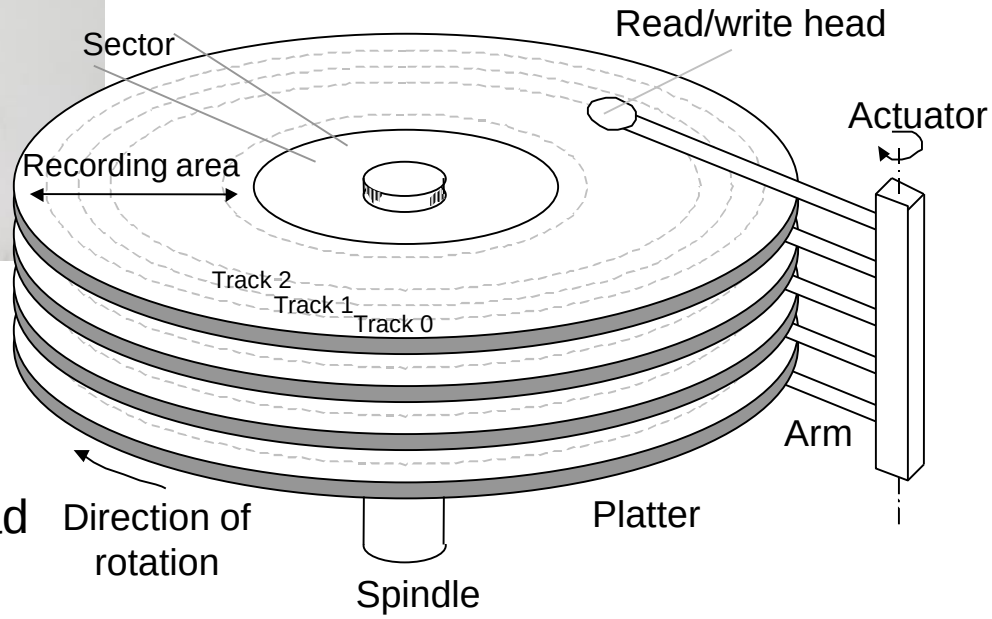


Disk Access Time =

Seek Time +

Rotation Latency +

Transfer Time



Seek Time: head movement to the desired track (milliseconds)

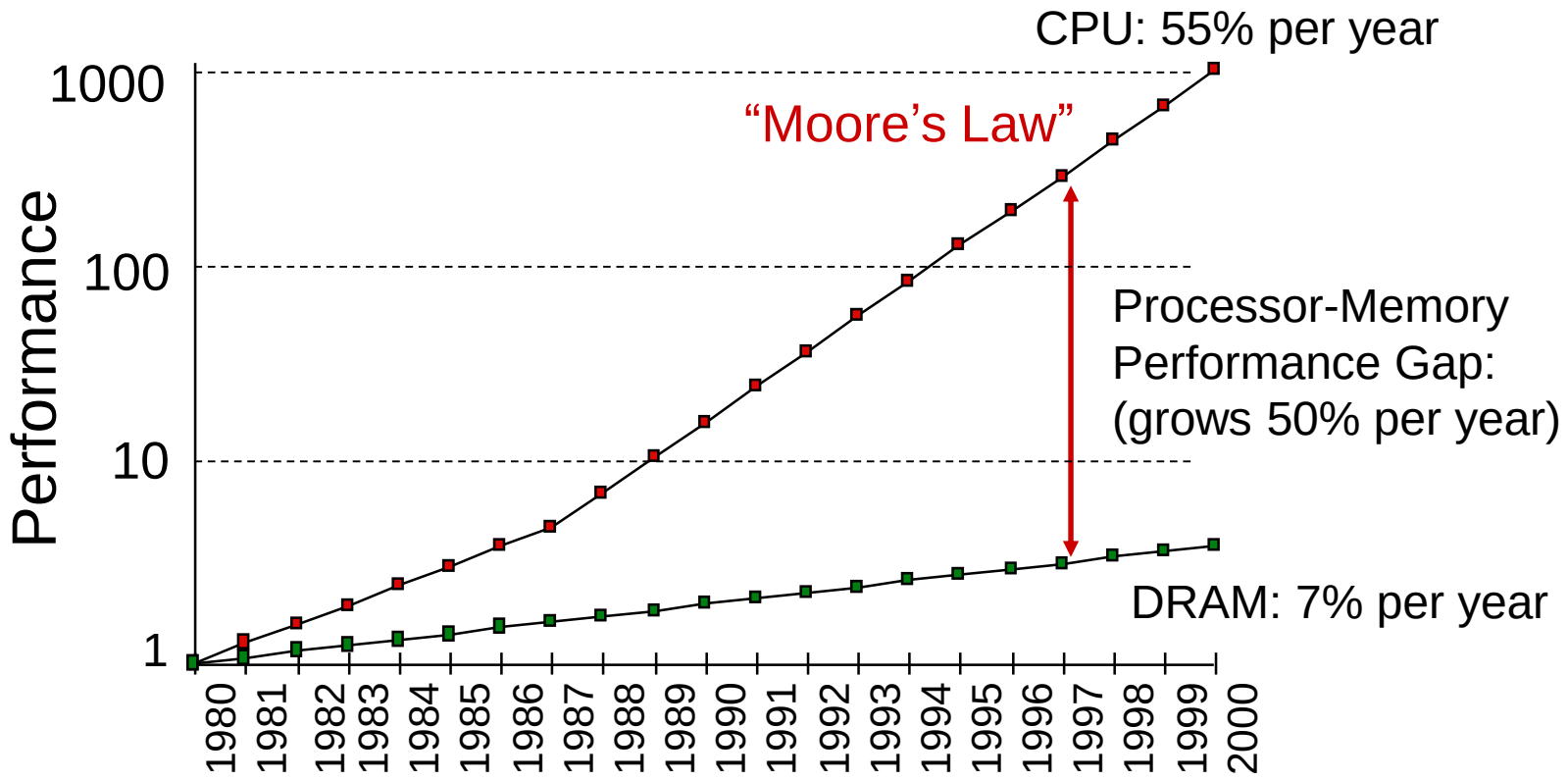
Rotation Latency: disk rotation until desired sector arrives under the head

Transfer Time: to transfer data

Example on Disk Access Time

- Given a magnetic disk with the following properties
  - Rotation speed = 7200 RPM (rotations per minute)
  - Average seek = 8 ms, Sector = 512 bytes, Track = 200 sectors
- Calculate
  - Time of one rotation (in milliseconds)
  - Average time to access a block of 32 consecutive sectors
- Answer
  - Rotations per second = 7200/60 = 120 RPS
  - Rotation time in milliseconds = 1000/120 = 8.33 ms
  - Average rotational latency = time of half rotation = 4.17 ms
  - Time to transfer 32 sectors = (32/200) * 8.33 = 1.33 ms
  - Average access time = 8 + 4.17 + 1.33 = 13.5 ms

Processor-Memory Performance Gap



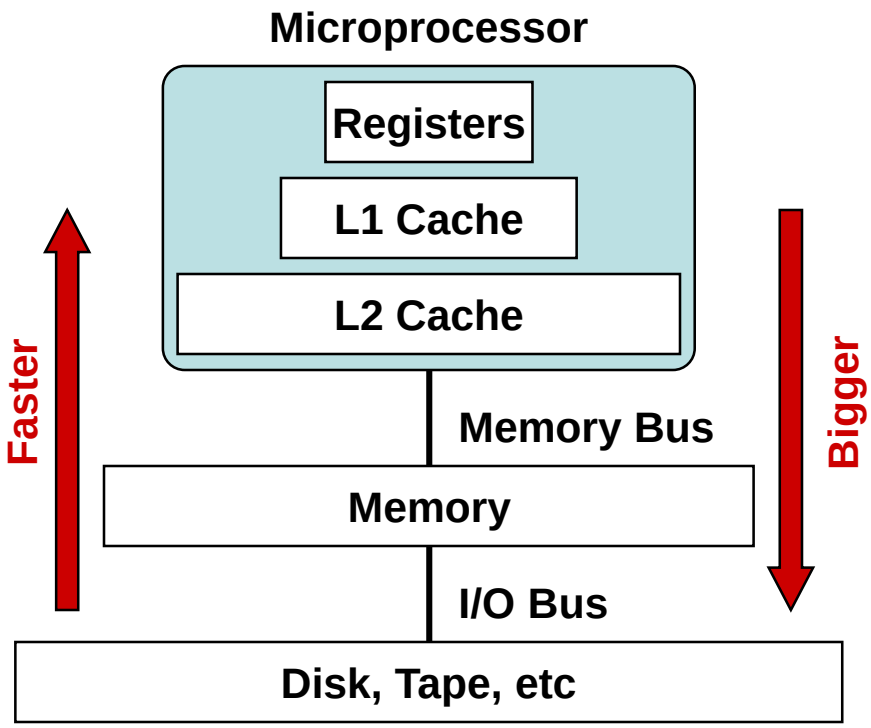
- 1980 – No cache in microprocessor
- 1995 – Two-level cache on microprocessor

The Need for a Memory Hierarchy

- Widening speed gap between CPU and main memory
  - Processor operation takes less than 1 ns
  - Main memory requires more than 50 ns to access
- Each instruction involves at least one memory access
  - One memory access to fetch the instruction
  - A second memory access for load and store instructions
- Memory bandwidth limits the instruction execution rate
- Cache memory can help bridge the CPU-memory gap
- Cache memory is small in size but fast

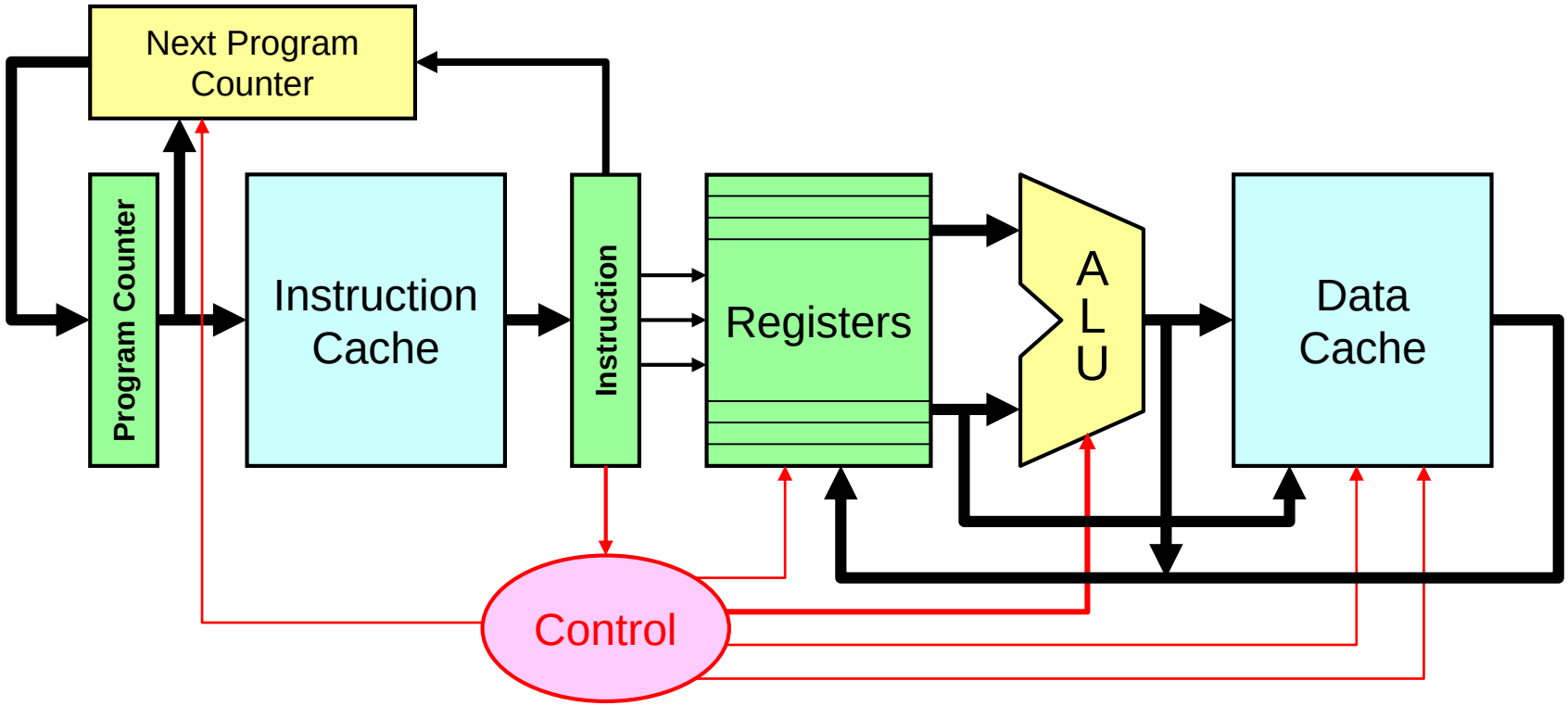
Typical Memory Hierarchy

- Registers are at the top of the hierarchy
  - Typical size < 1 KB
  - Access time < 0.5 ns
- Level 1 Cache (8 – 64 KB)
  - Access time: 0.5 – 1 ns
- L2 Cache (64 KB – 8 MB)
  - Access time: 2 – 10 ns
- Main Memory (1 – 64 GB)
  - Access time: 50 – 70 ns
- Disk Storage (> 200 GB)
  - Access time: milliseconds



Processor

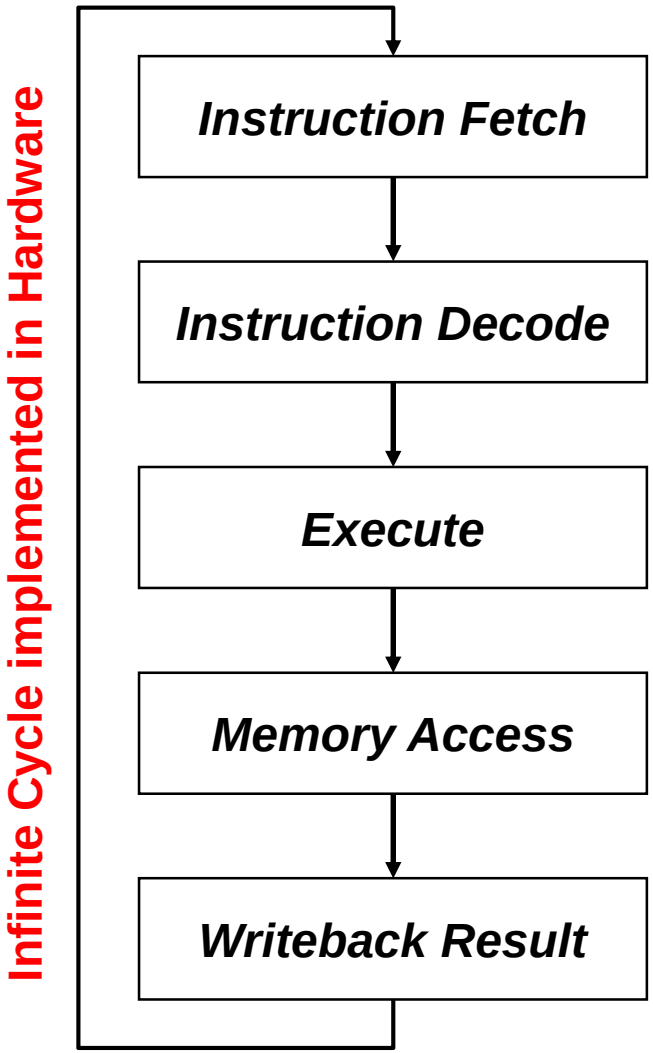
- Datapath: part of a processor that executes instructions
- Control: generates control signals for each instruction



Datapath Components

- Program Counter (PC)
  - Contains address of instruction to be fetched
  - Next Program Counter: computes address of next instruction
- Instruction Register (IR)
  - Stores the fetched instruction
- Instruction and Data Caches
  - Small and fast memory containing most recent instructions/data
- Register File
  - General-purpose registers used for intermediate computations
- ALU = Arithmetic and Logic Unit
  - Executes arithmetic and logic instructions
- Buses
  - Used to wire and interconnect the various components

Fetch - Execute Cycle



Fetch instruction
Compute address of next instruction

Generate control signals for instruction
Read operands from registers

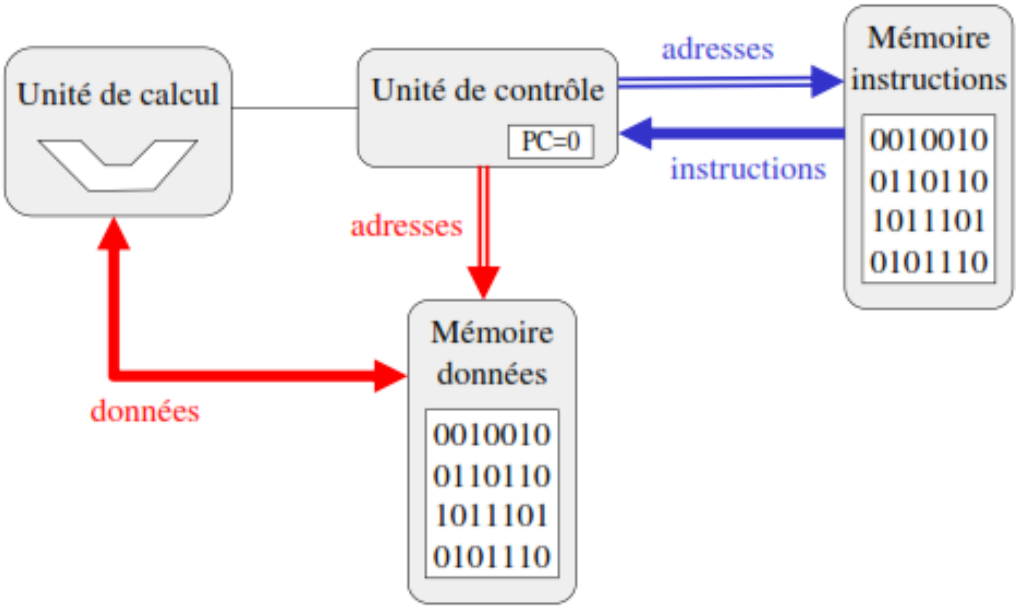
Compute result value

Read or write memory (load/store)

Writeback result in a register

Harvard Architecture

The Harvard architecture is a processor design that physically separates data memory from program memory. Access to each of the two memories is done through two distinct buses.



Difference between Von Neumann and Harvard Architecture

Comparison Table

It is a theoretical design based on the stored-program computer concept.	It is a modern computer architecture based on the Harvard Mark I relay-based computer model.
It uses same physical memory address for instructions and data.	It uses separate memory addresses for instructions and data.
Processor needs two clock cycles to execute an instruction.	Processor needs one cycle to complete an instruction.
Simpler control unit design and development of one is cheaper and faster.	Control unit for two buses is more complicated which adds to the development cost.
Data transfers and instruction fetches cannot be performed simultaneously.	Data transfers and instruction fetches can be performed at the same time.
Used in personal computers, laptops, and workstations.	Used in microcontrollers and signal processing.

Next . . .

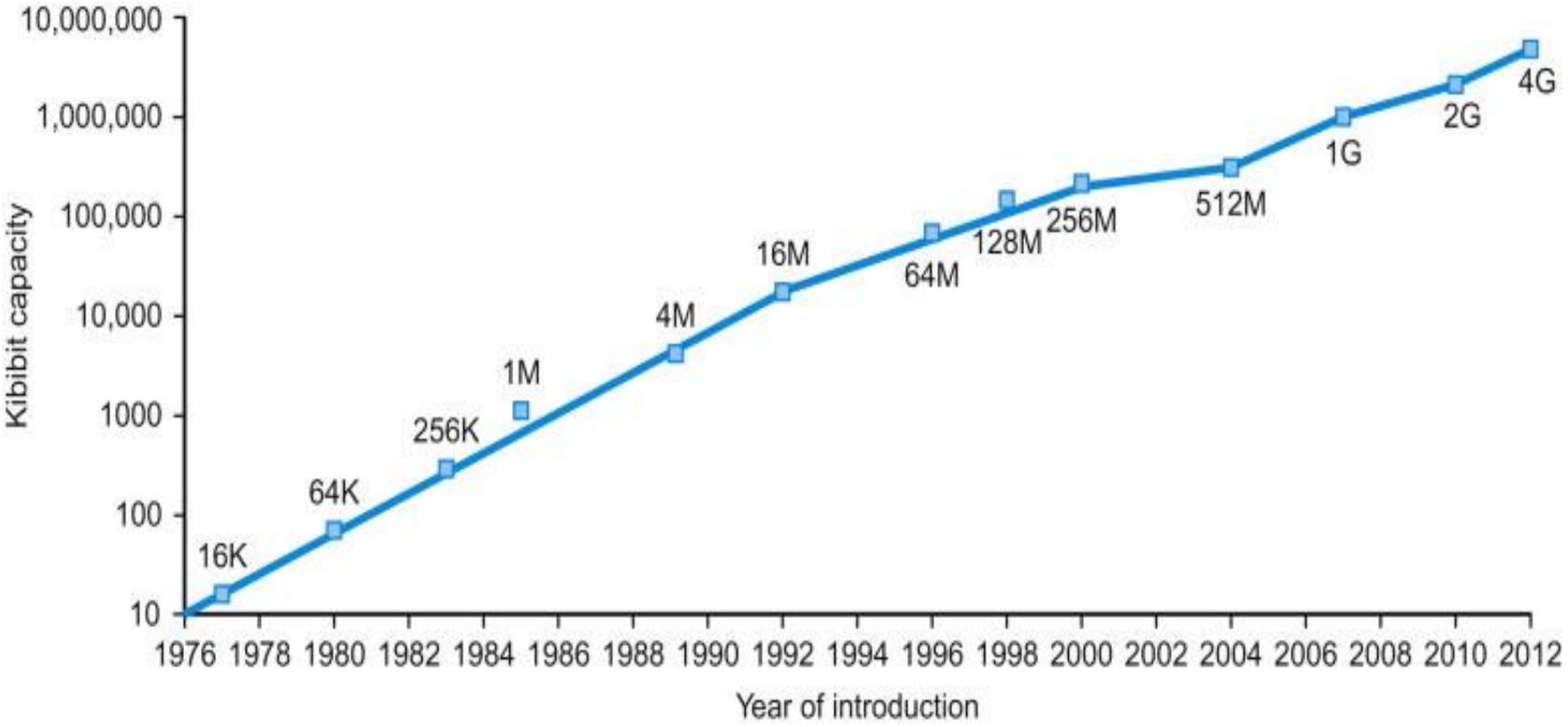
- Assembly-, Machine-, and High-Level Languages
- Components of a Computer System
- Technology Improvements
- Programmer's View of a Computer System

Technology Improvements

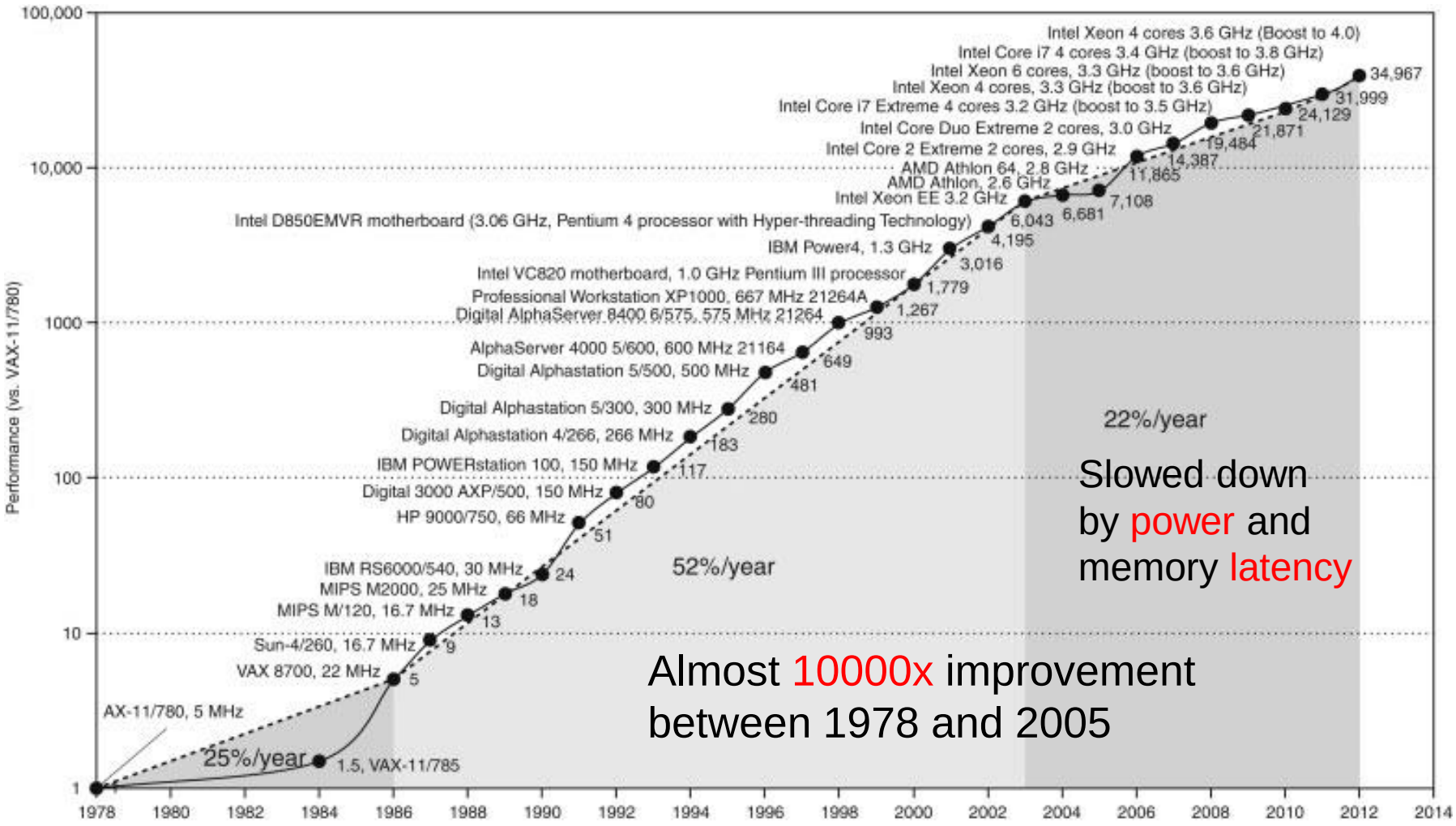
- Vacuum tube → transistor → → VLSI
- Processor
  - Transistor count: about 30% to 40% per year
- Memory
  - DRAM capacity: about 60% per year (4x every 3 yrs)
  - Cost per bit: decreases about 25% per year
- Disk
  - Capacity: about 60% per year
- Opportunities for new applications
- Better organizations and designs

Growth of Capacity per DRAM Chip

- DRAM capacity quadrupled almost every 3 years
  - 60% increase per year, for 20 years



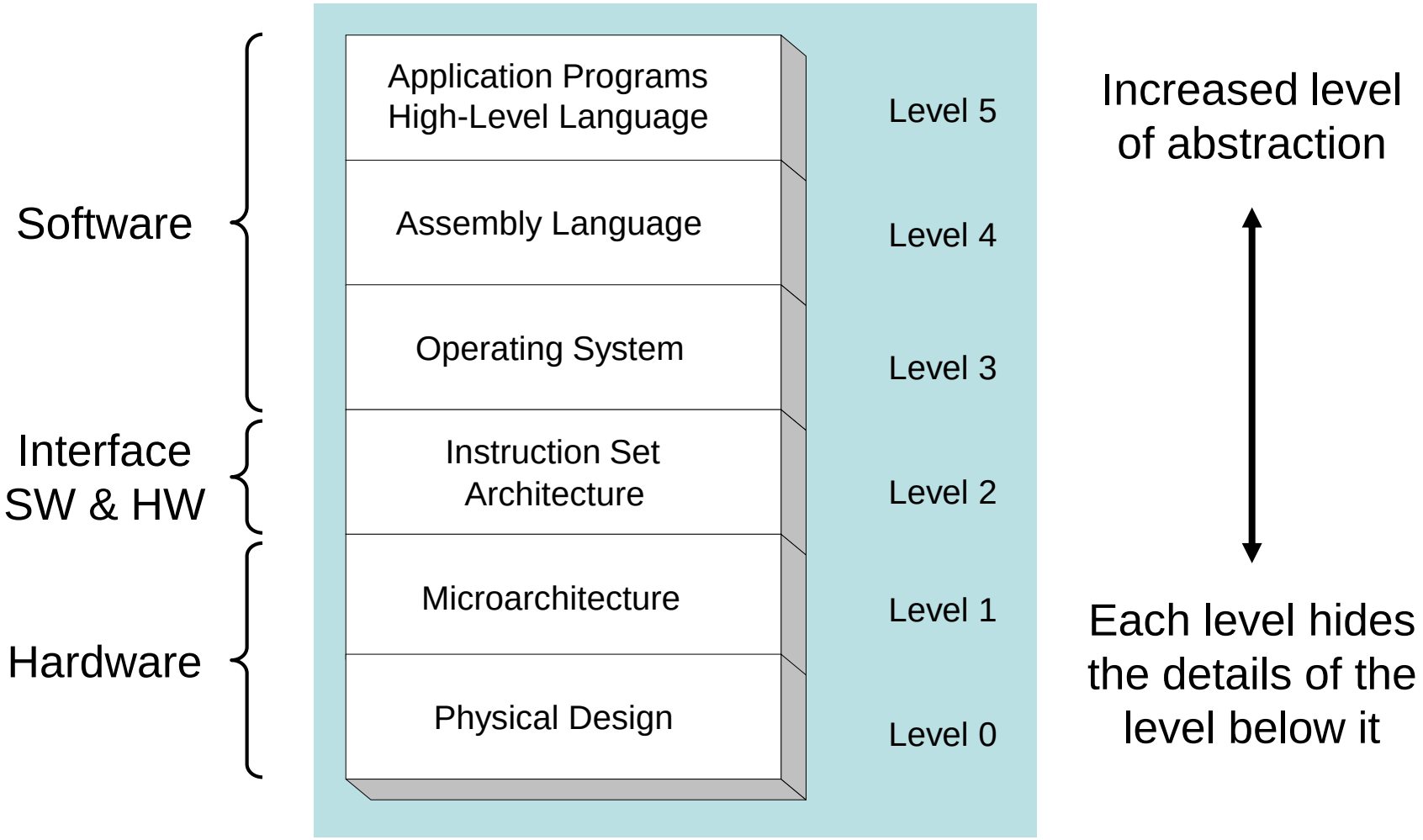
Processor Performance



Next . . .

- Assembly-, Machine-, and High-Level Languages
- Components of a Computer System
- Technology Improvements
- Programmer's View of a Computer System

Programmer's View of a Computer System



Programmer's View of a Computer System

- Application Programs (Level 5)
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- Assembly Language (Level 4)
  - Instruction mnemonics (symbols) are used
  - Have one-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)

Programmer's View of a Computer System

- Operating System (Level 3)
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)
- Instruction Set Architecture (Level 2)
  - Interface between software and hardware
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)

Programmer's View of a Computer System

- Microarchitecture (Level 1)
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic
- Physical Design (Level 0)
  - Implements the microarchitecture at the transistor-level
  - Physical layout of circuits on a chip