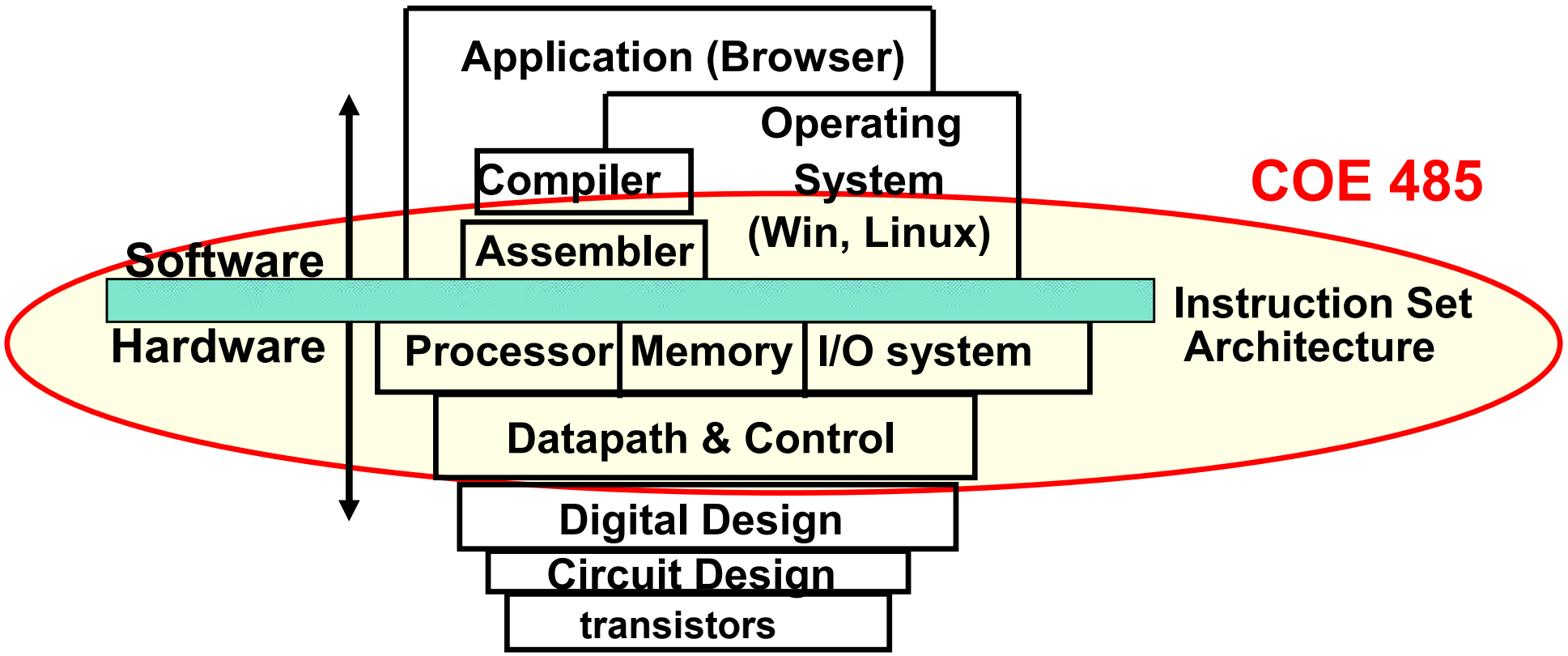
Introduction

Introduction

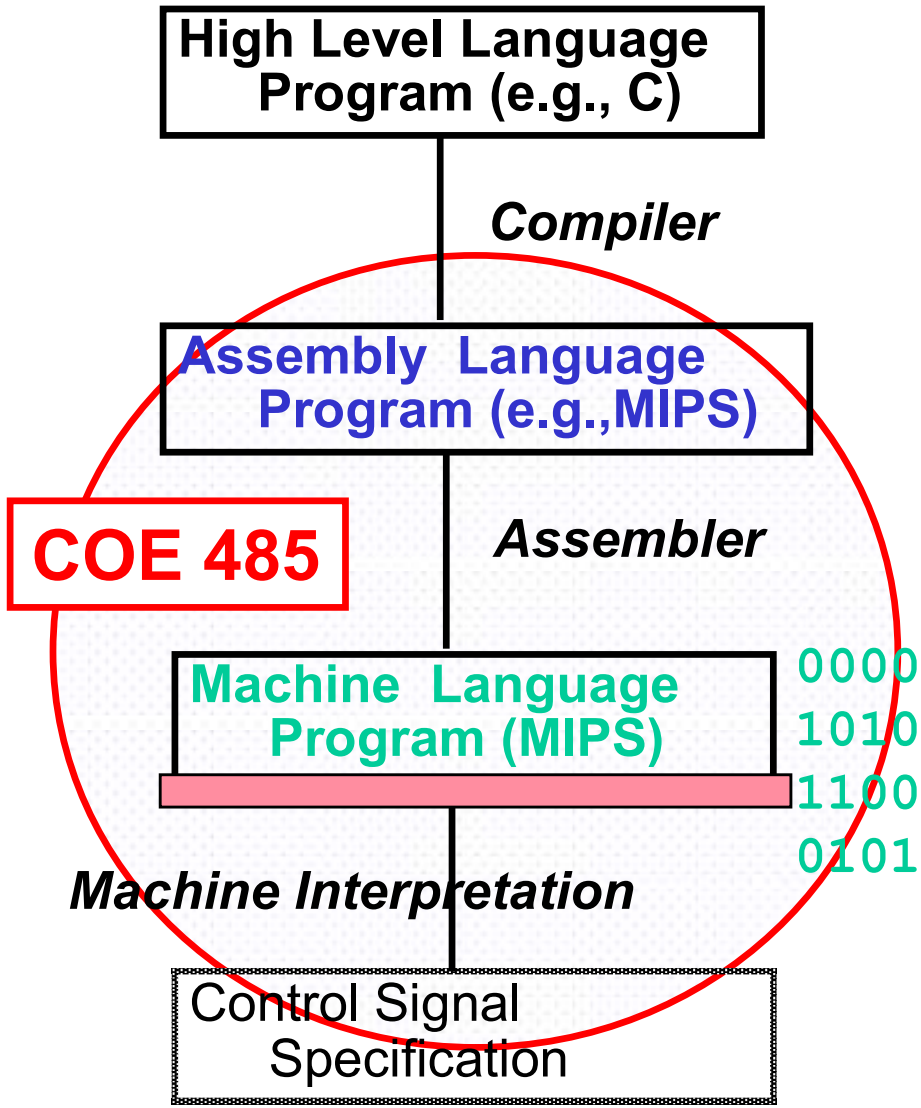
- **Rapidly changing field:**
  - vacuum tube -> transistor -> IC -> VLSI
  - doubling every 1.5 years (Moore's law):
    - *Memory capacity*
    - *Processor speed (Due to advances in technology and organization)*
- **Things you'll be learning:**
  - how computers work, a basic foundation
  - how to analyze their performance
  - how to improve performance
  - issues affecting modern processors (caches, pipelines)
- **Why learn this stuff?**
  - you want to call yourself a "computer engineer"
  - you want to build a computing system that performs
  - you need to make a purchasing decision or offer advice

Computing System



- Coordination of many *levels of abstraction*

Levels of Representation

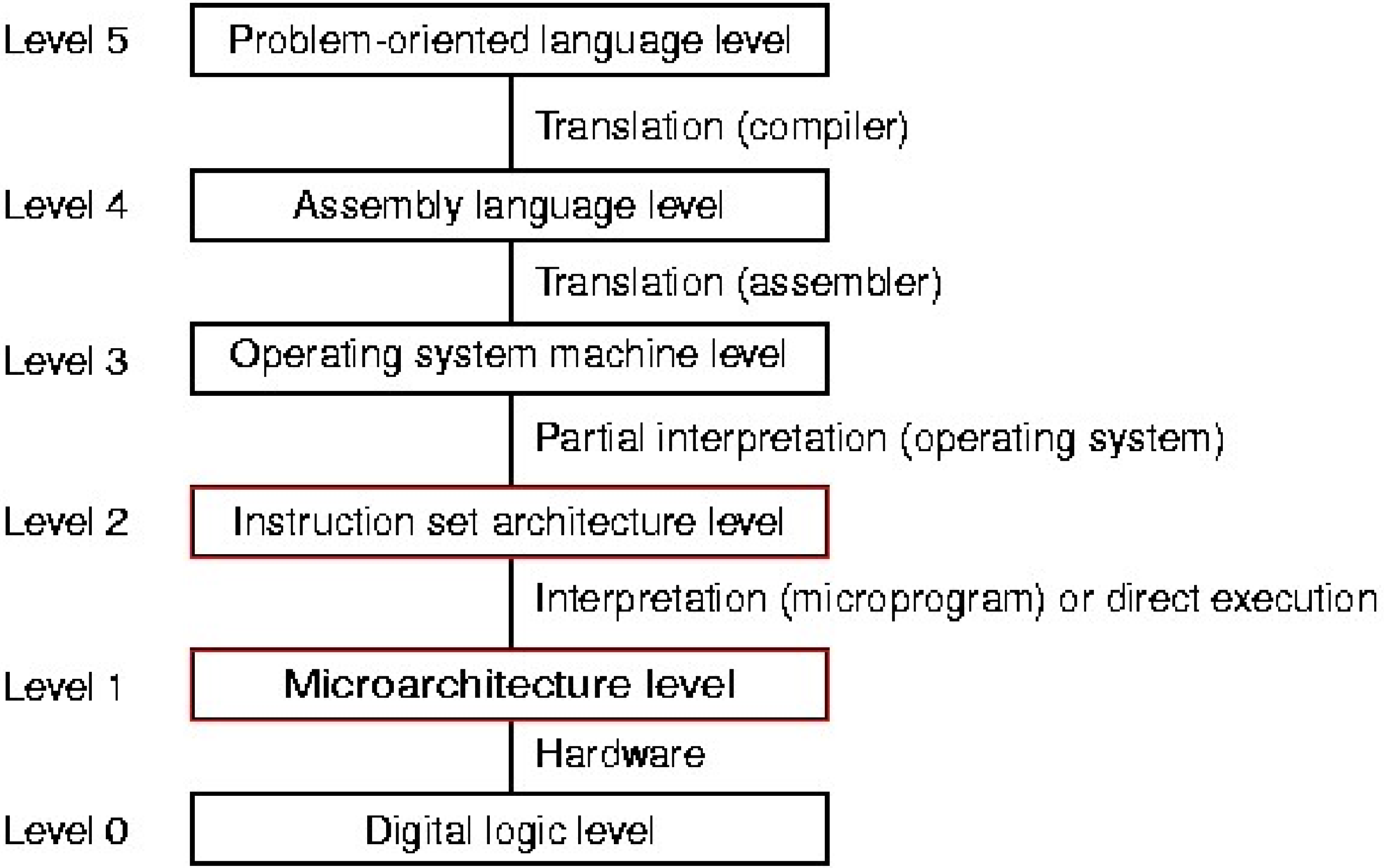



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

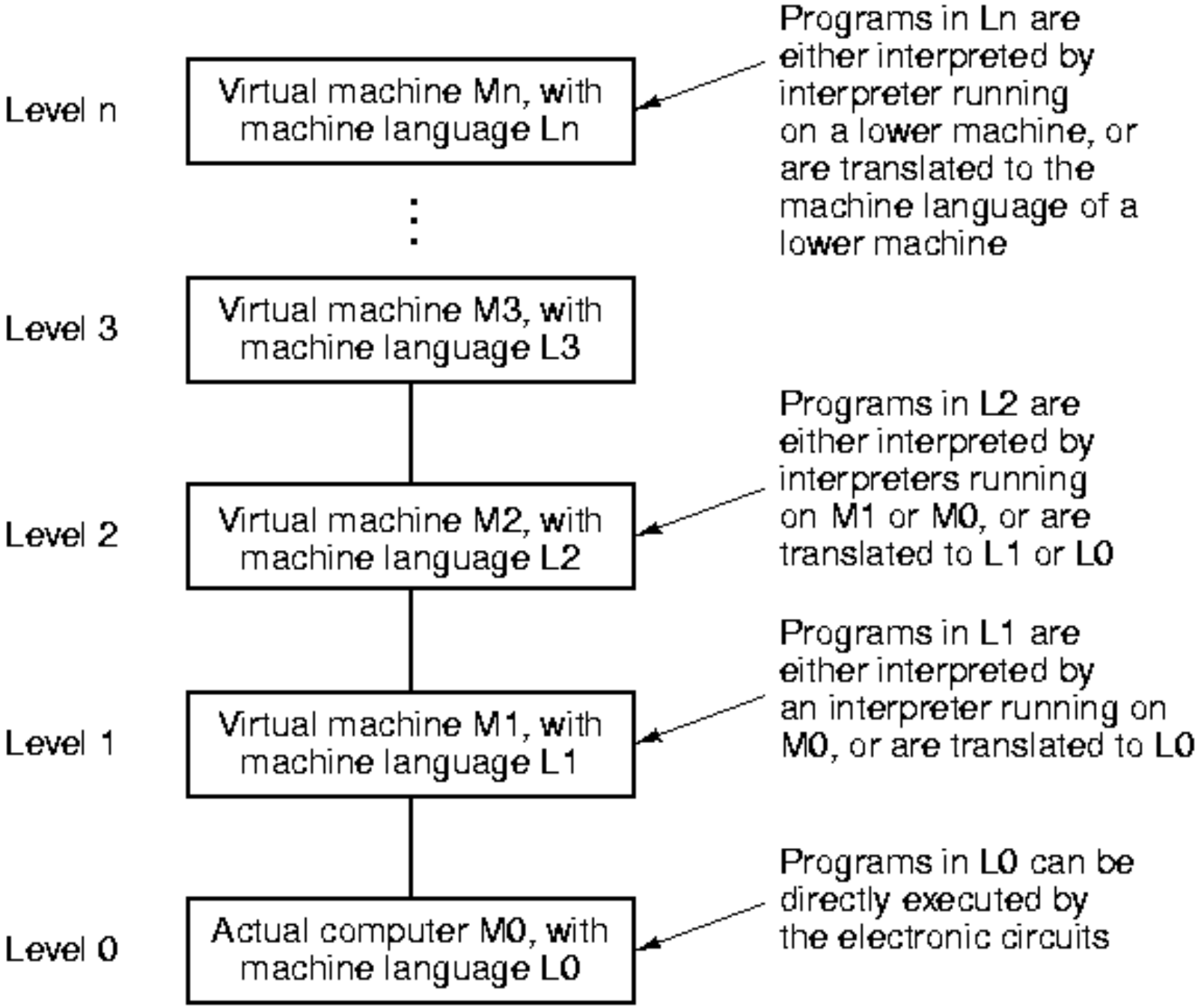
```
lw $t0,   0($2)
lw $t1,   4($2)
sw$t1,   0($2)
sw$t0,   4($2)
```

```
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
```

A Six-Level Computer



Big Idea: Multilevel Machine



Evolution of Multilevel Machines

1. Bare hardware
2. Microprogramming
3. Operating system
4. Compilers
5. **Hardware / software interface**
   - Simple ISA
   - CISC
   - RISC
   - FISC

RISC Design Principles

- CISC vs. RISC
- Instructions directly executed by hardware
- Maximize instruction issue rate (ILP)
- Simple instructions (easy to decode)
- Access to memory only via load/store
- Plenty of registers
- Pipelining

Computer Architecture

- **Computer architecture** refers to
  - attributes of a system visible to a programmer
  - attributes that have a direct impact on the logical execution of a program
- **Instruction set architecture (ISA)** is a term often used interchangeably with computer architecture.
  - The ISA defines instruction formats, instruction opcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution
- Examples of architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory

Computer Organization (1)

- **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications
- Organizational attributes include those hardware details transparent to the programmer
  - control signals;
  - interfaces between the computer and peripherals;
  - memory technology used
- For 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.

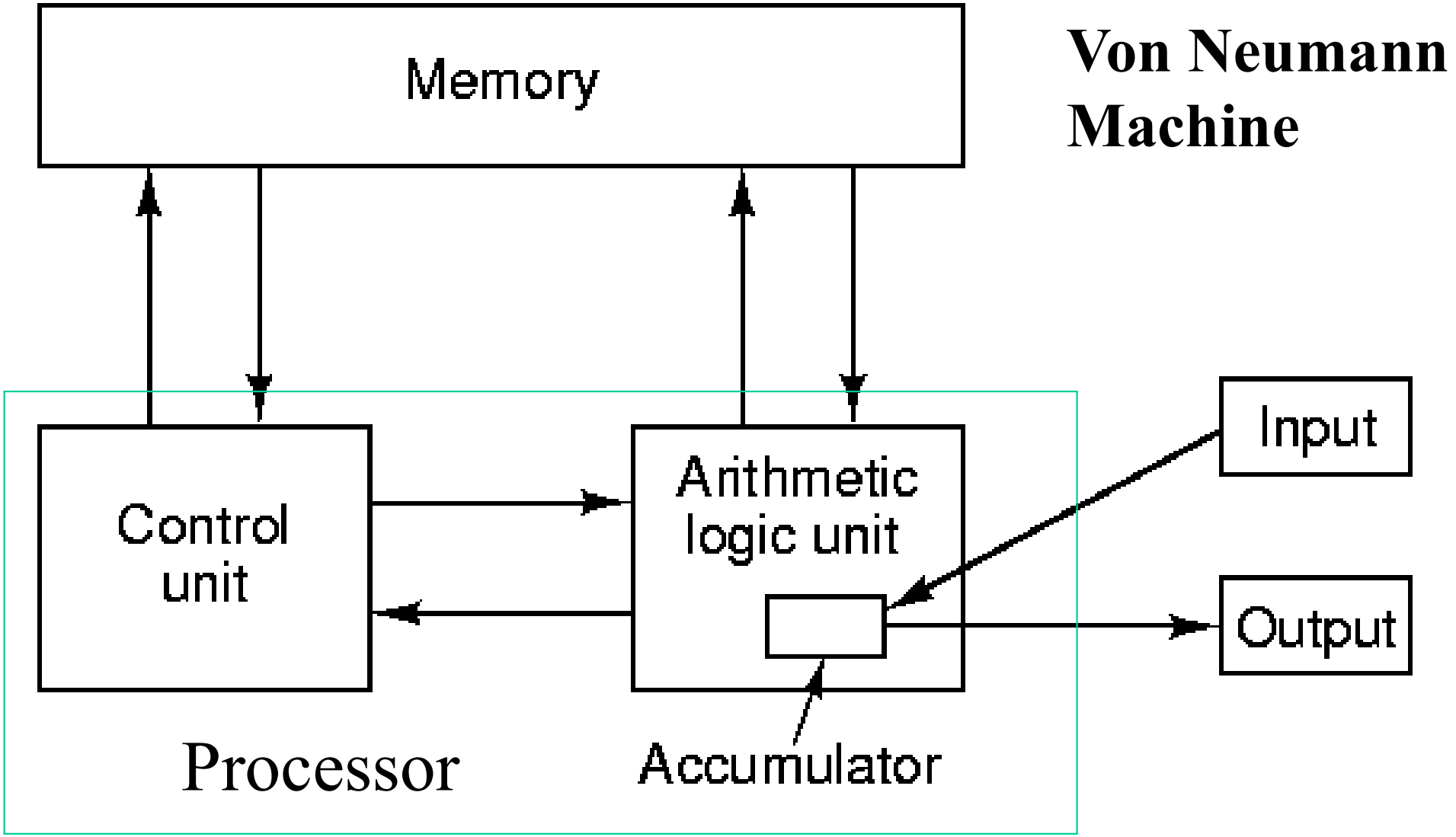
Computer Organization (2)

- The organizational decision may be based on
  - the anticipated frequency of use of the multiply instruction,
  - the relative speed of the two approaches, and
  - the cost and physical size of a special multiply unit

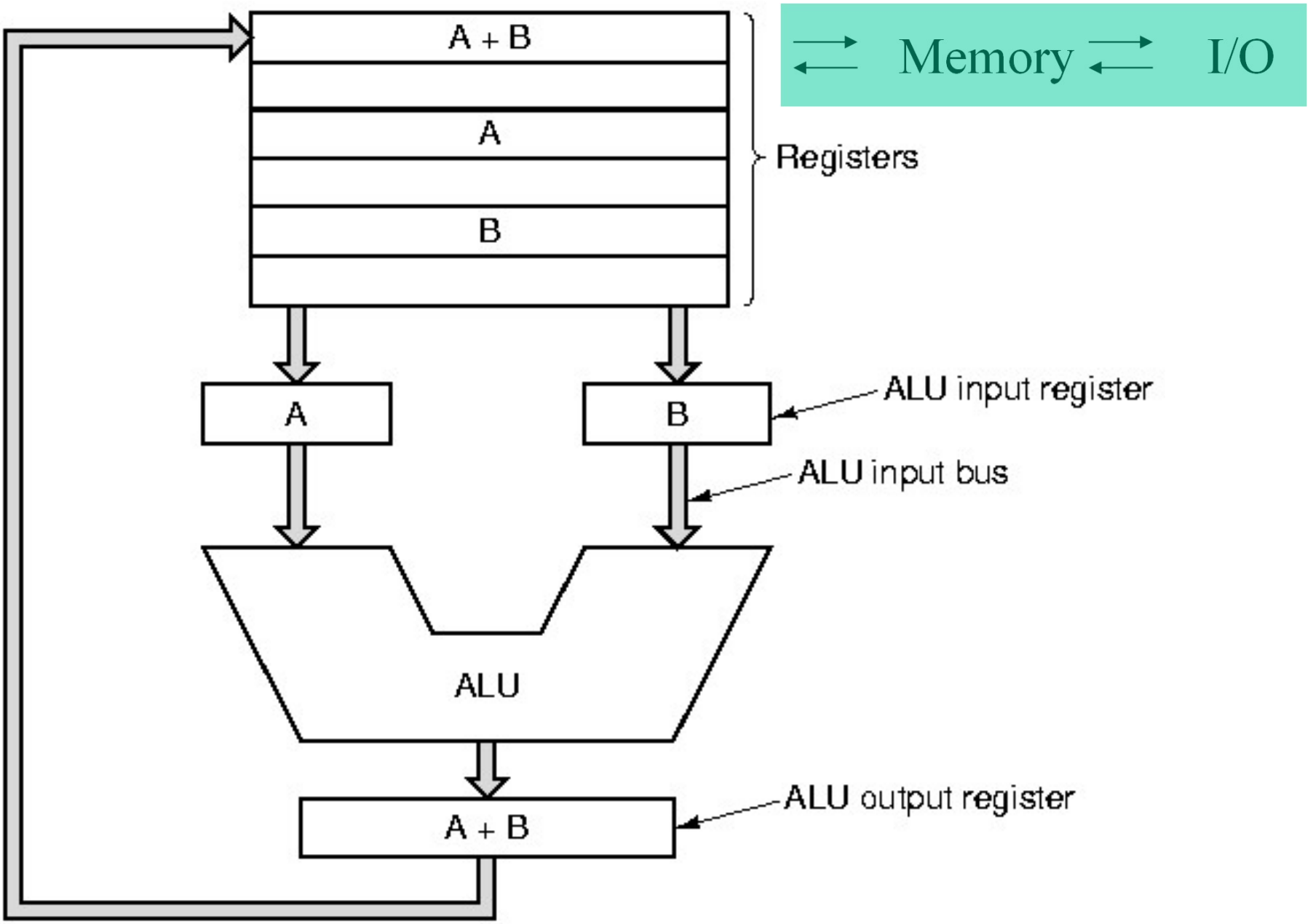
Architecture vs. Organization

- One family of computer models with the same architecture may have differences in organization
- Gives rise to different models in the family having different price and performance characteristics
- Furthermore, a particular architecture may span many years
  - encompassing a number of different computer models, with organization changing with changing technology
- A prominent example of both these phenomena is the IBM System/370 architecture, which was first introduced in 1970

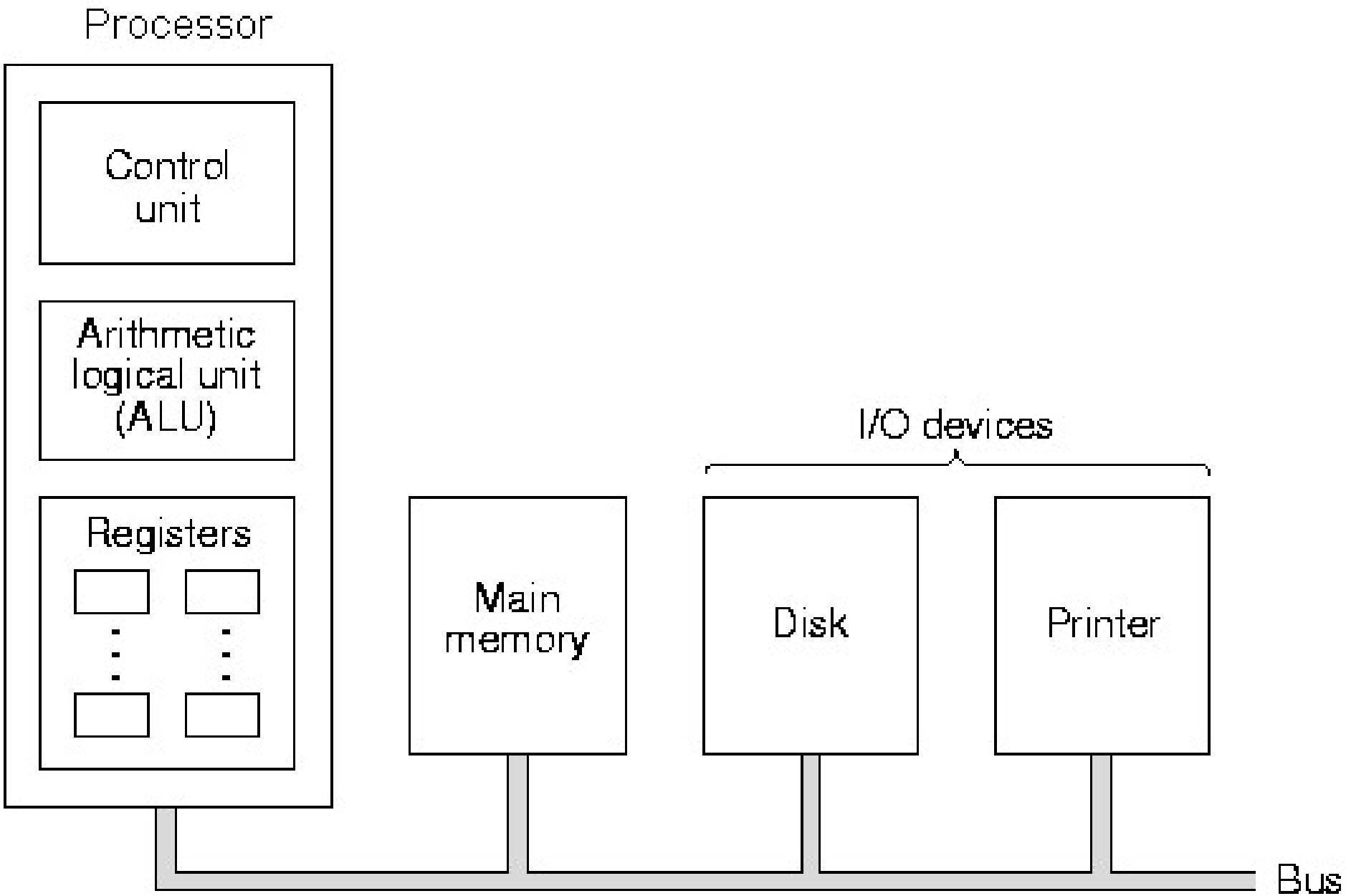
Computer Organization



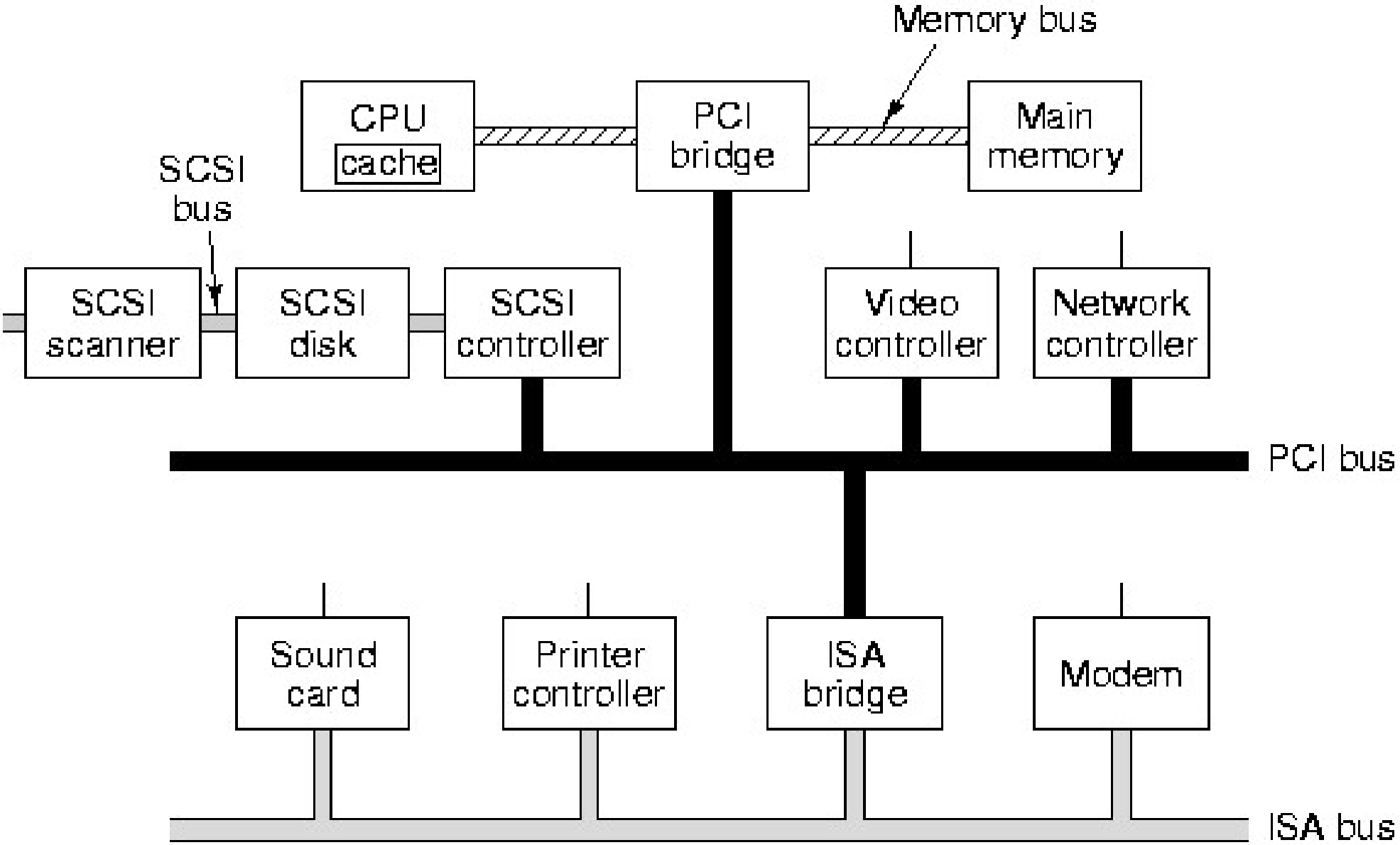
Datapath


A + B
A
B
Registers
Memory
I/O
A
B
ALU input register
ALU input bus
ALU
A + B
ALU output register

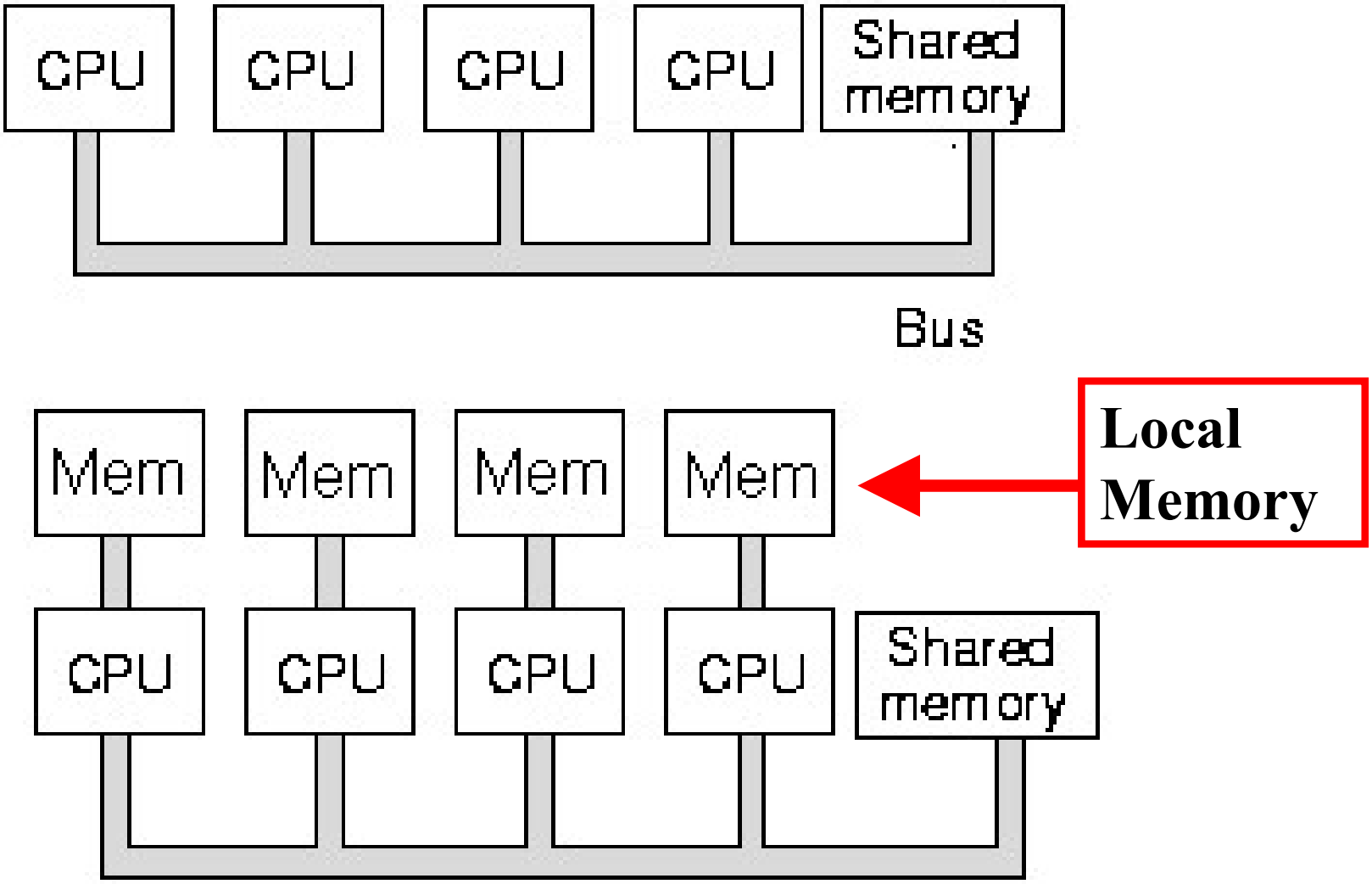
Bus-Based Computer



Anatomy of a Modern PC



Multiprocessors



Conclusion

- Principle of *abstraction*, used to build systems as layers
- *Pliable Data*: a program determines what it is
- *Stored program* concept: instructions are just data
- Principle of *Locality*, exploited via memory hierarchy
- Greater performance by exploiting *parallelism* (pipeline)
- *Compilation v. interpretation* to move downward through layers of system
- Principles of *Performance Measurement*

Course Administration

- Instructor:
  - Prof. K. O. Boateng (koboat2000@gmail.com, **boat.coe@knust.edu.gh** ; 414 Ceasar Building)
- TAs: Jummai & Onesimus
- Texts:
  - ***Computer Organization and Architecture***, **Tenth Edition**, William Stallings, 2016
  - ***Computer Organization and Design: The Hardware Software Interface***, **Fifth Edition**, Patterson and Hennessy, 2014
- Also recommended:
  - ***Computer Architecture: A Quantitative Approach***, **Sixth Edition**, Hennessy and Patterson, 2019

Course Overview

- Arithmetic and how to build an ALU (P 3)
- von Neumann model
- Instruction set architecture (W12.1-12.2; W13.1-13.3)
- CPU organization (W14.1-14.3; W20.1-20.3; W21.1-21.3)
- Performance issues (P 1)
- A specific instruction set architecture (P 2)
-

Course Overview

- Constructing a processor to execute selected instructions (P 4)
- Pipelining to improve performance (P 4)
- Caches, main, and virtual memory, I/O (W4.1-4.3; W5.1-5.4; W6.1-6,5; W7.1-7.6)
- Parallel processing (W17.1, 17.2, 17.4)
- Future Computing Technologies

Course Evaluation

- Grade breakdown
  - Midsem Exam 20%
  - Final Exam 70%
  - Quizzes (5) 5%
  - Home Assignments (5) 10%

TOTAL 100%