Chapter 5: Computer Systems Organization

Invitation to Computer Science,

C++ Version, Third Edition

Objectives

In this chapter, you will learn about:

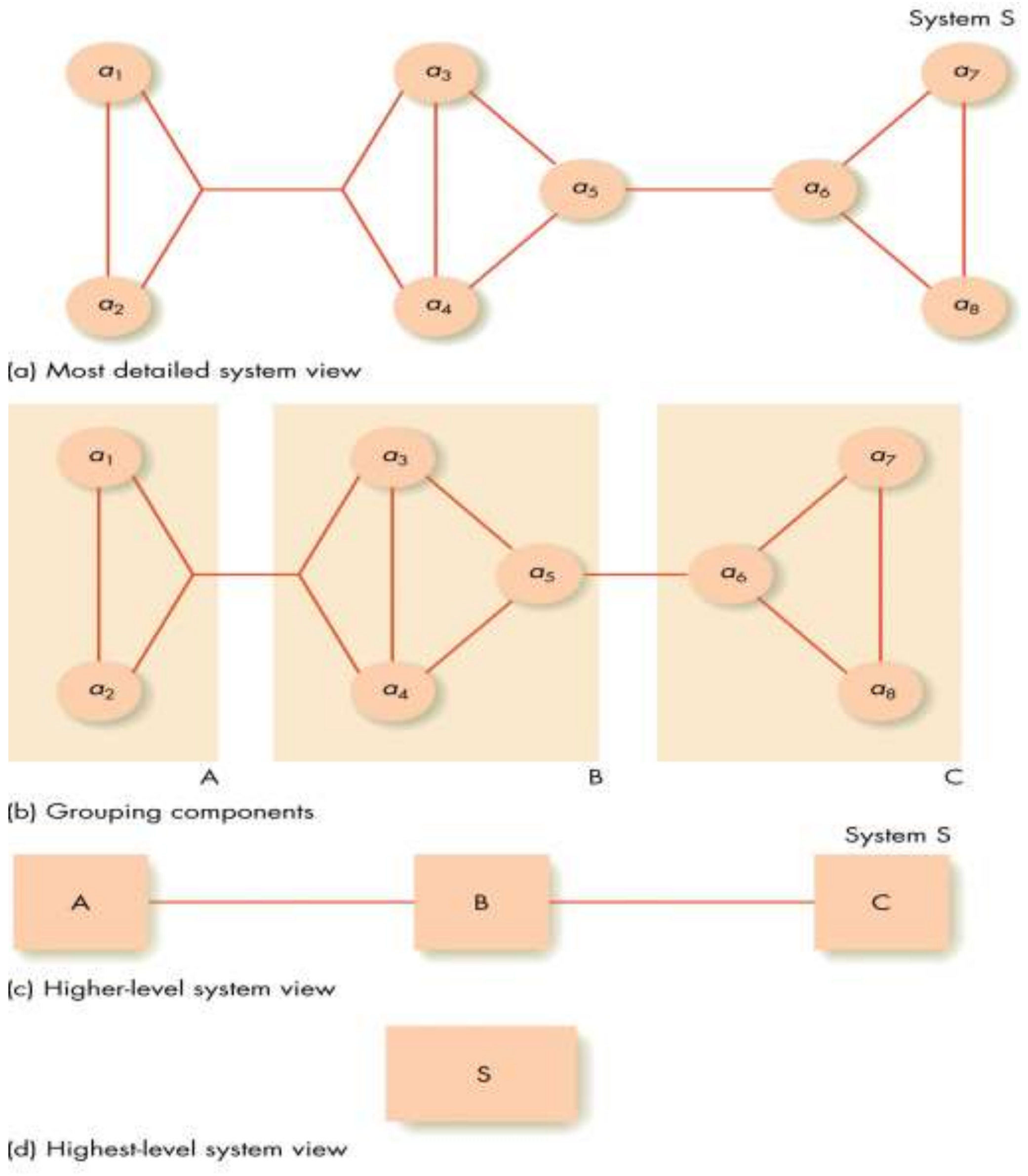
- The components of a computer system
- Putting all the pieces together – the Von Neumann architecture
- The future: non-Von Neumann architectures

Introduction

- Computer organization examines the computer as a collection of interacting “functional units”
- Functional units may be built out of the circuits already studied
- Higher level of abstraction assists in understanding by reducing complexity

Figure 5.1

The Concept of Abstraction


System S
a_1
a_2
a_3
a_4
a_5
a_6
a_7
a_8
(a) Most detailed system view
A
B
C
(b) Grouping components
System S
A
B
C
(c) Higher-level system view
S
(d) Highest-level system view

The Components of a Computer System

- Von Neumann architecture has four functional units:
  - Memory
  - Input/Output
  - Arithmetic/Logic unit
  - Control unit
- Sequential execution of instructions
- Stored program concept

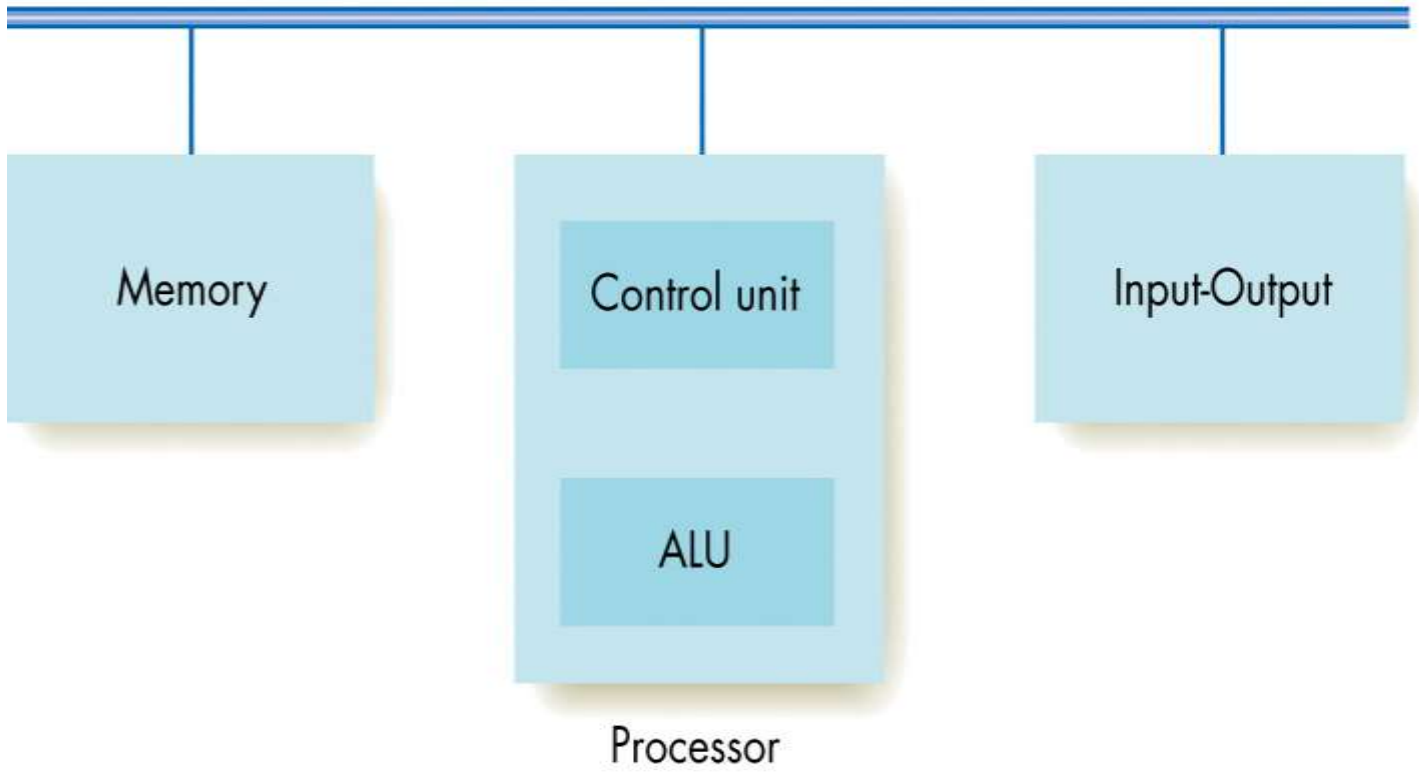


Figure 5.2

Components of the Von Neumann Architecture

Memory and Cache

- Information stored and fetched from memory subsystem
- Random Access Memory maps addresses to memory locations
- Cache memory keeps values currently in use in faster memory to speed access times

Memory and Cache (continued)

- RAM (Random Access Memory)
  - Memory made of addressable “cells”
  - Current standard cell size is 8 bits
  - All memory cells accessed in equal time
  - Memory address
    - Unsigned binary number N long
    - Address space is then 2^N cells

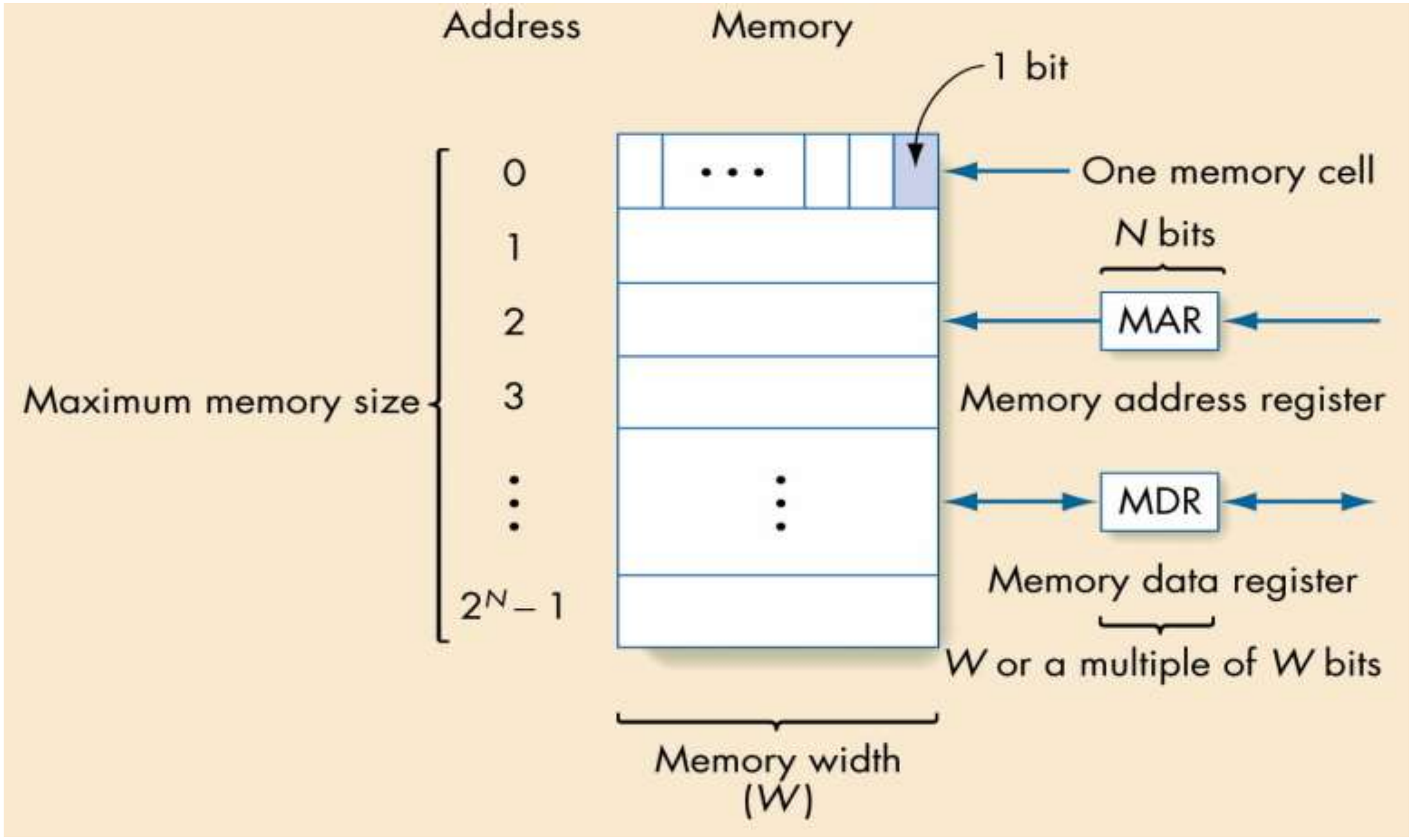


Figure 5.3

Structure of Random Access Memory

Memory and Cache (continued)

- Parts of the memory subsystem
  - Fetch/store controller
    - Fetch: retrieve a value from memory
    - Store: store a value into memory
  - Memory address register (MAR)
  - Memory data register (MDR)
  - Memory cells, with decoder(s) to select individual cells

Memory and Cache (continued)

- Fetch operation
  - The address of the desired memory cell is moved into the MAR
  - Fetch/store controller signals a “fetch,” accessing the memory cell
  - The value at the MAR’s location flows into the MDR

Memory and Cache (continued)

- Store operation
  - The address of the cell where the value should go is placed in the MAR
  - The new value is placed in the MDR
  - Fetch/store controller signals a “store,” copying the MDR’s value into the desired cell

Memory and Cache (continued)

- Memory register
  - Very fast memory location
  - Given a name, not an address
  - Serves some special purpose
  - Modern computers have dozens or hundreds of registers

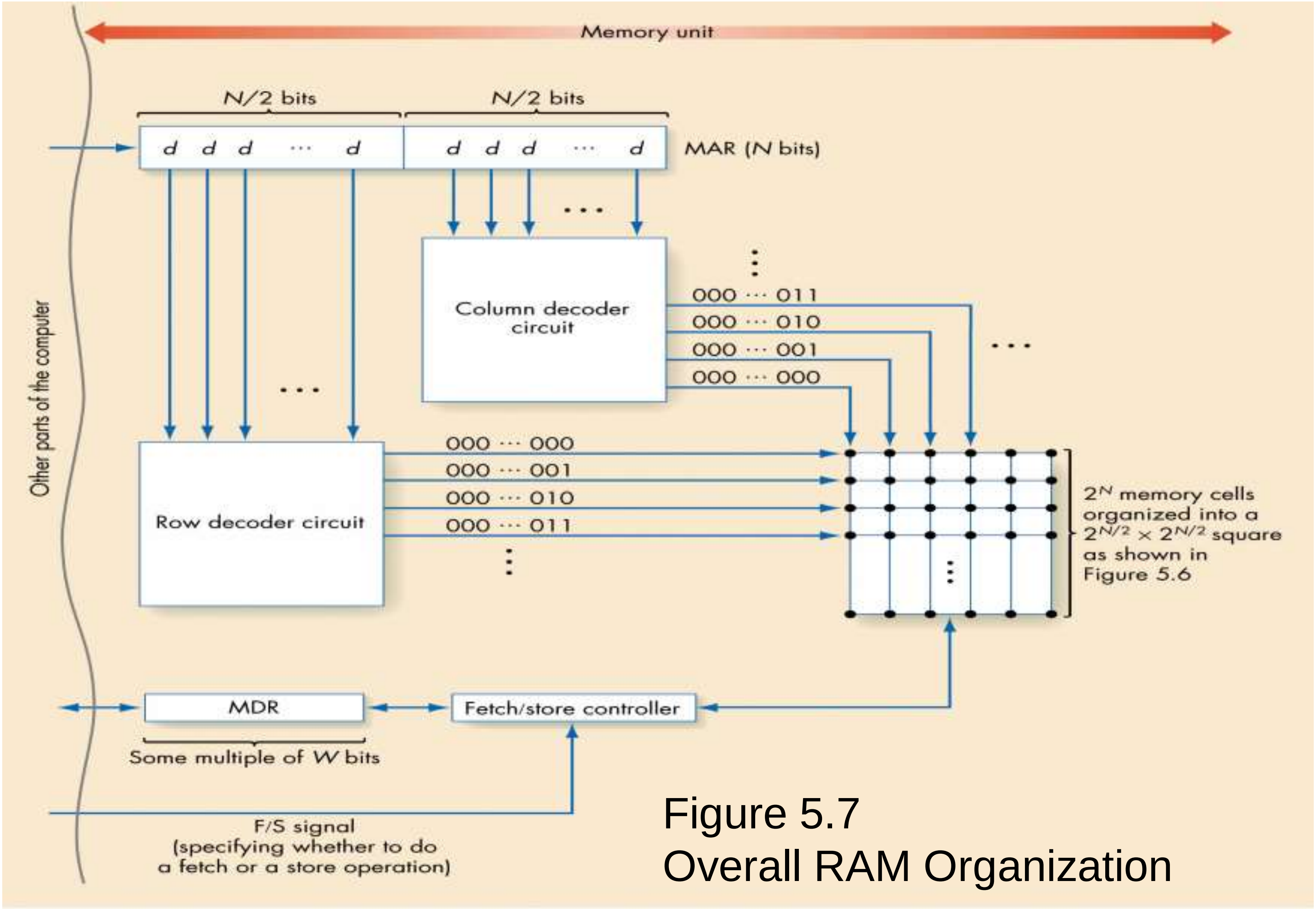


Figure 5.7
Overall RAM Organization

Cache Memory

- Memory access is much slower than processing time
- Faster memory is too expensive to use for all memory cells
- Locality principle
    - Once a value is used, it is likely to be used again
- Small size, fast memory just for values currently in use speeds computing time

Input/Output and Mass Storage

- Communication with outside world and external data storage
  - Human interfaces: monitor, keyboard, mouse
  - Archival storage: not dependent on constant power
- External devices vary tremendously from each other

Input/Output and Mass Storage (continued)

- Volatile storage
  - Information disappears when the power is turned off
  - Example: RAM
- Nonvolatile storage
  - Information does not disappear when the power is turned off
  - Example: mass storage devices such as disks and tapes

Input/Output and Mass Storage (continued)

- Mass storage devices
  - Direct access storage device
    - Hard drive, CD-ROM, DVD, etc.
    - Uses its own addressing scheme to access data
  - Sequential access storage device
    - Tape drive, etc.
    - Stores data sequentially
    - Used for backup storage these days

Input/Output and Mass Storage (continued)

- Direct access storage devices
  - Data stored on a spinning disk
  - Disk divided into concentric rings (sectors)
  - Read/write head moves from one ring to another while disk spins
  - Access time depends on:
    - Time to move head to correct sector
    - Time for sector to spin to data location

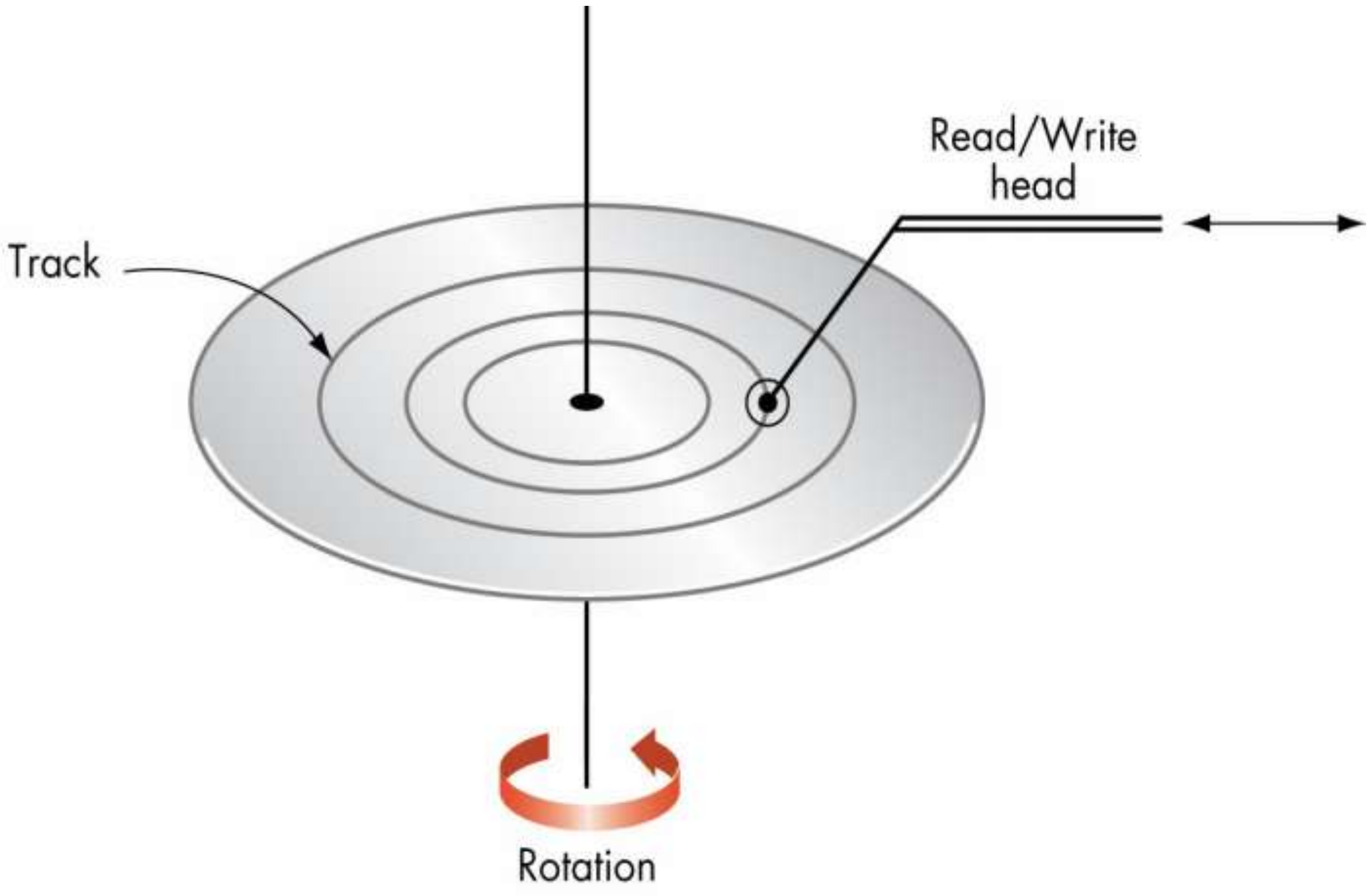


Figure 5.8

Overall Organization of a Typical Disk

Input/Output and Mass Storage (continued)

- I/O controller
  - Intermediary between central processor and I/O devices
  - Processor sends request and data, then goes on with its work
  - I/O controller interrupts processor when request is complete

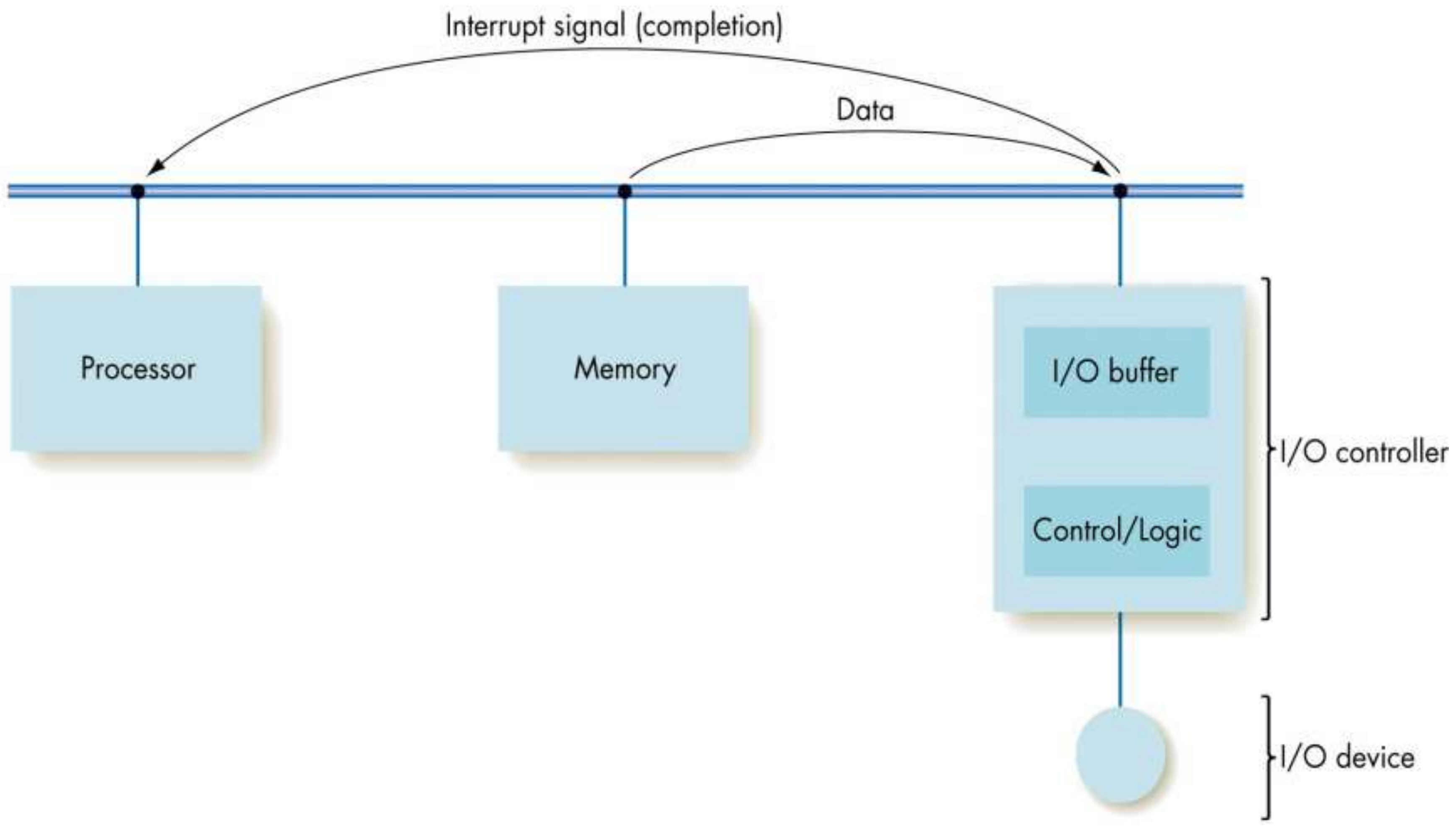


Figure 5.9

Organization of an I/O Controller

The Arithmetic/Logic Unit

- Actual computations are performed
- Primitive operation circuits
  - Arithmetic (ADD, etc.)
  - Comparison (CE, etc.)
  - Logic (AND, etc.)
- Data inputs and results stored in registers
- Multiplexor selects desired output

The Arithmetic/Logic Unit (continued)

- ALU process
  - Values for operations copied into ALU’s input register locations
  - All circuits compute results for those inputs
  - Multiplexor selects the one desired result from all values
  - Result value copied to desired result register

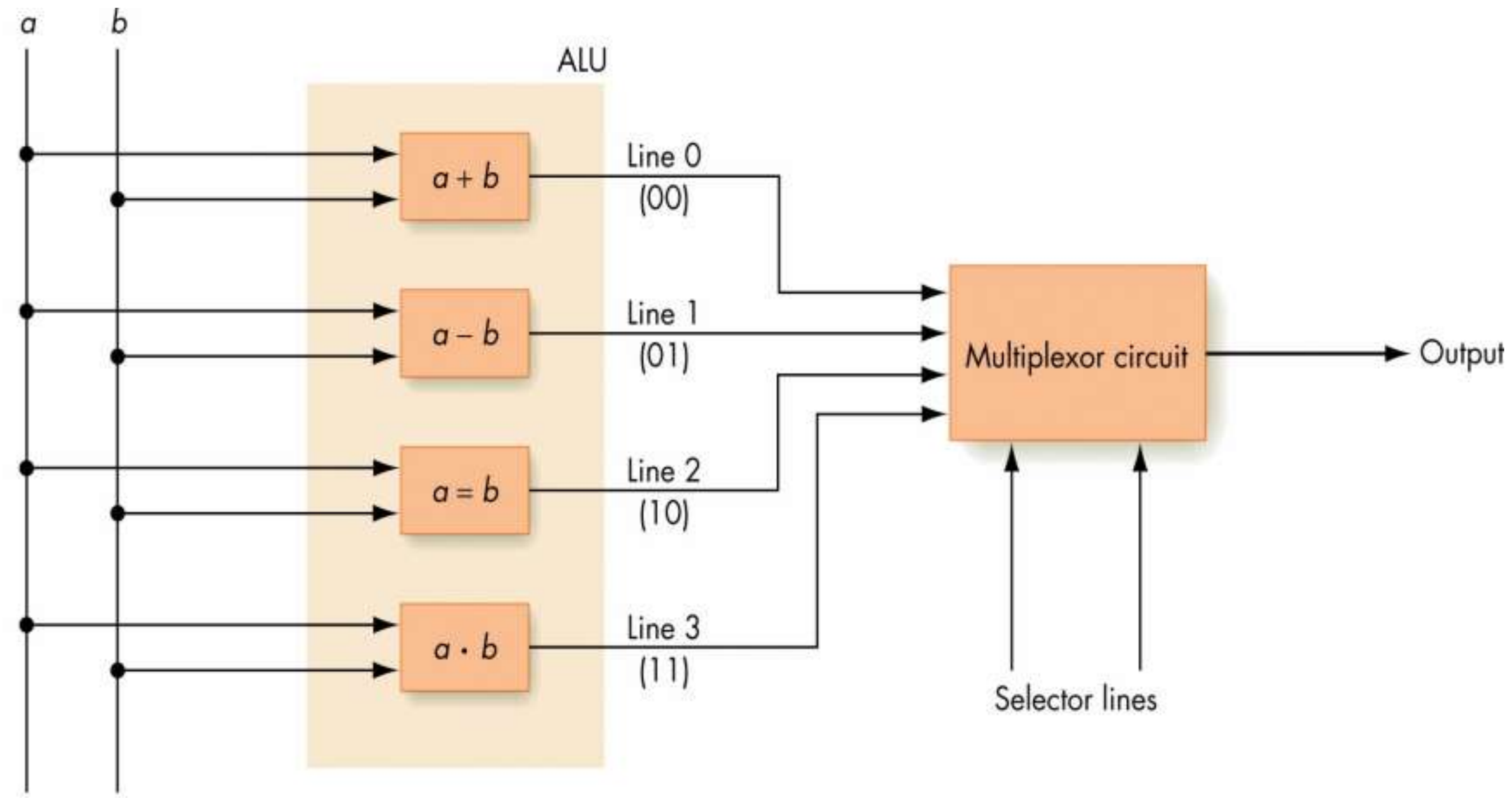


Figure 5.12

Using a Multiplexor Circuit to Select the Proper ALU Result

The Control Unit

- Manages stored program execution
- Task
  - Fetch from memory the next instruction to be executed
  - Decode it: determine what is to be done
  - Execute it: issue appropriate command to ALU, memory, and I/O controllers

Machine Language Instructions

- Can be decoded and executed by control unit
- Parts of instructions
  - Operation code (op code)
    - Unique unsigned-integer code assigned to each machine language operation
  - Address field(s)
    - Memory addresses of the values on which operation will work

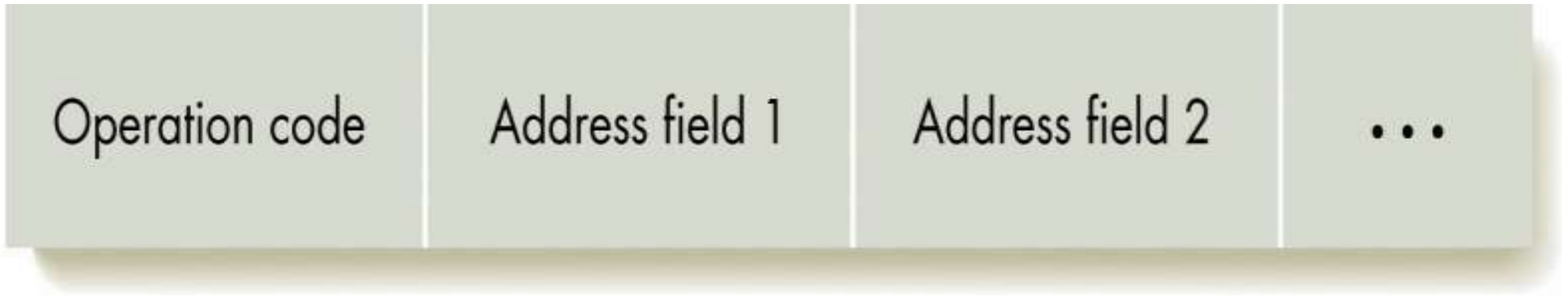


Figure 5.14

Typical Machine Language Instruction Format

Machine Language Instructions (continued)

- Operations of machine language
  - Data transfer
    - Move values to and from memory and registers
  - Arithmetic/logic
    - Perform ALU operations that produce numeric values

Machine Language Instructions (continued)

- Operations of machine language (continued)
  - Compares
    - Set bits of compare register to hold result
  - Branches
    - Jump to a new memory address to continue processing

Control Unit Registers And Circuits

- Parts of control unit
  - Links to other subsystems
  - Instruction decoder circuit
  - Two special registers:
    - Program Counter (PC)
      - Stores the memory address of the next instruction to be executed
    - Instruction Register (IR)
      - Stores the code for the current instruction

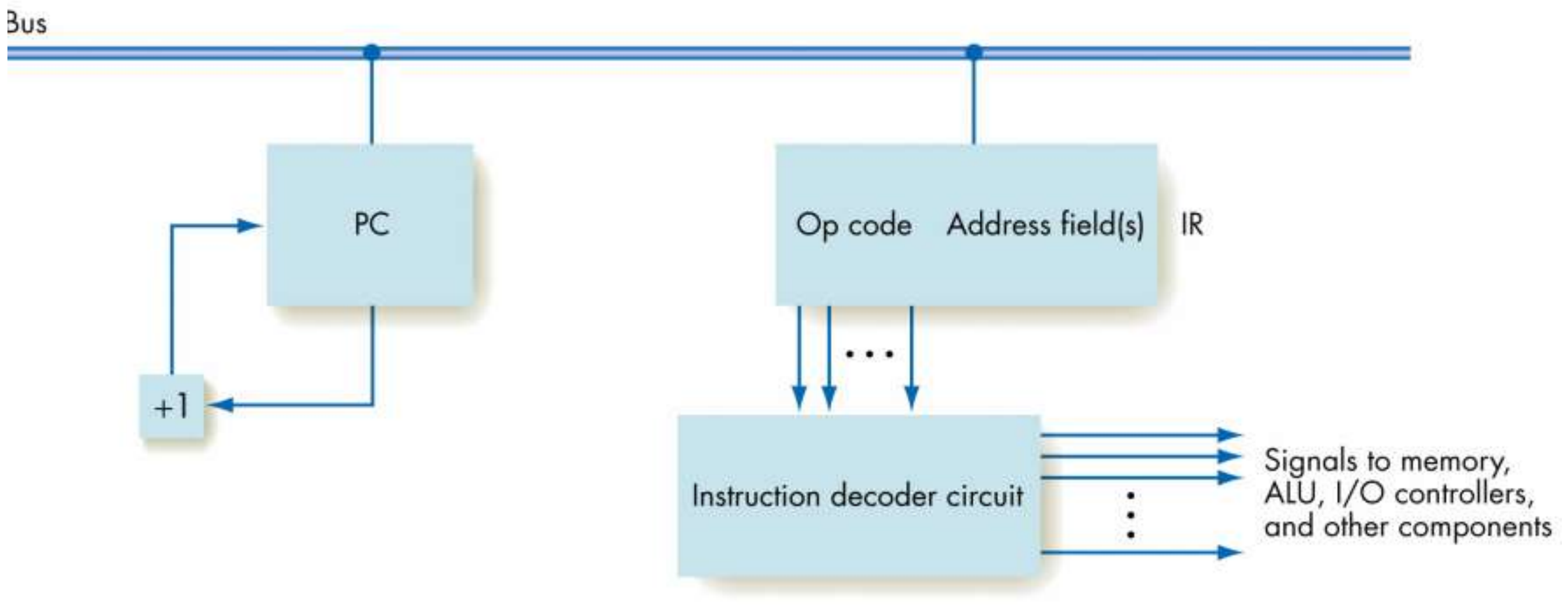

Bus
PC
+1
Op code
Address field(s)
IR
Instruction decoder circuit
Signals to memory, ALU, I/O controllers, and other components

Figure 5.16

Organization of the Control Unit Registers and Circuits

Putting All the Pieces Together—the Von Neumann Architecture

- Subsystems connected by a bus
  - Bus: wires that permit data transfer among them
- At this level, ignore the details of circuits that perform these tasks: Abstraction!
- Computer repeats fetch-decode-execute cycle indefinitely

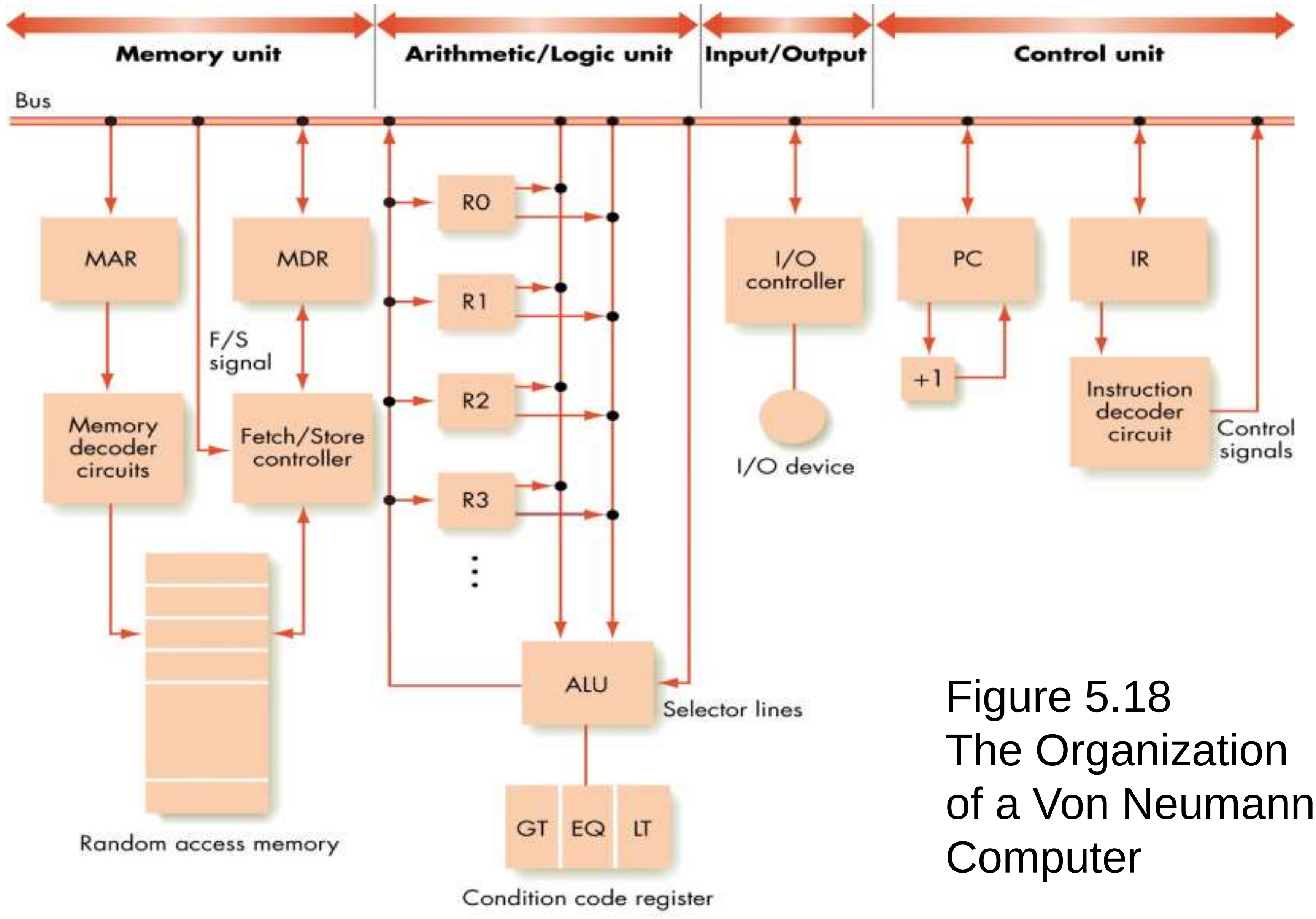


Figure 5.18
The Organization of a Von Neumann Computer

The Future: Non-Von Neumann Architectures

- Physical limitations on speed of Von Neumann computers
- Non-Von Neumann architectures explored to bypass these limitations
- Parallel computing architectures can provide improvements: multiple operations occur at the same time

The Future: Non-Von Neumann Architectures (continued)

- SIMD architecture
  - Single instruction/Multiple data
  - Multiple processors running in parallel
  - All processors execute same operation at one time
  - Each processor operates on its own data
  - Suitable for "vector" operations

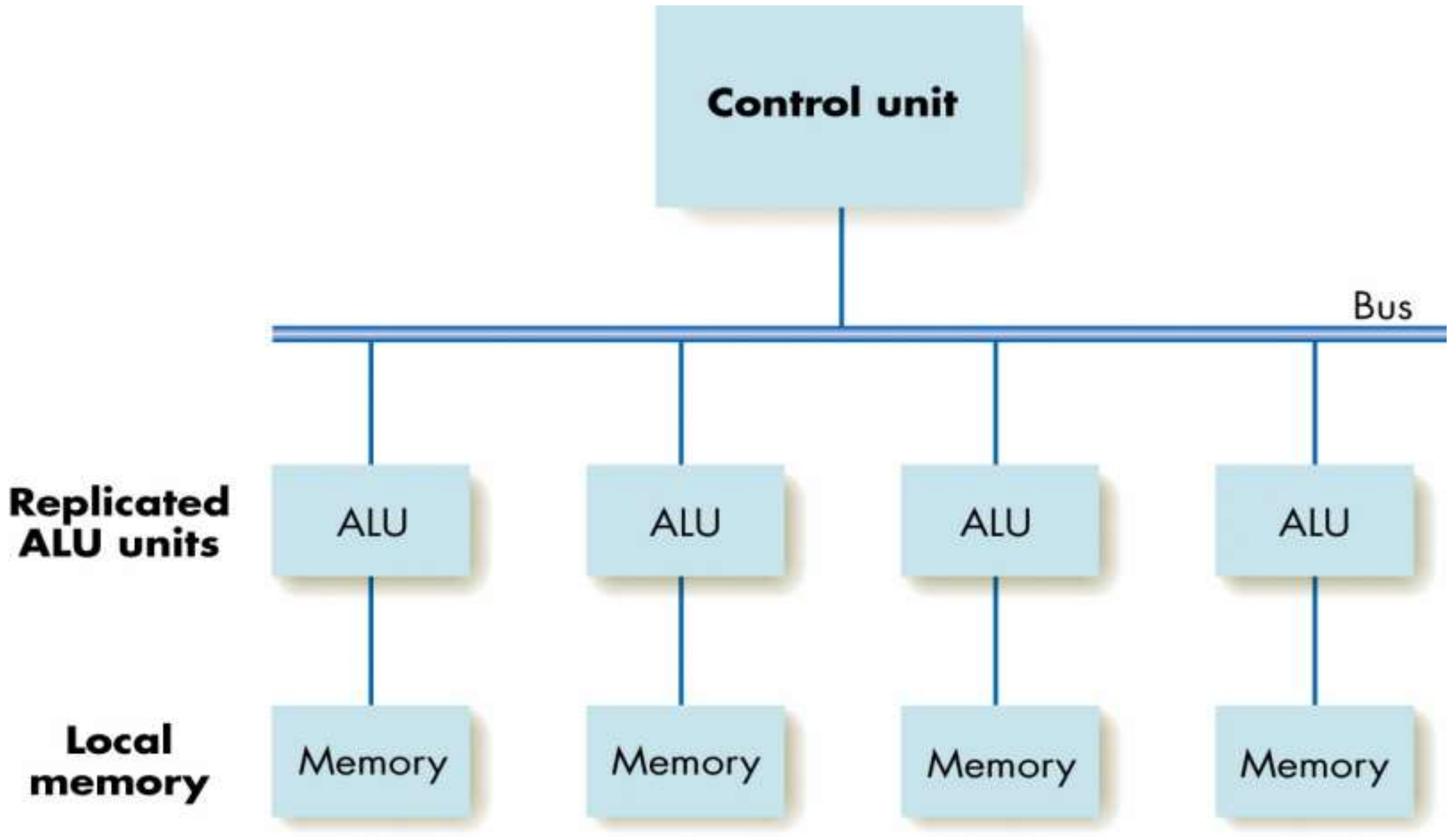


Figure 5.21

A SIMD Parallel Processing System

The Future: Non-Von Neumann Architectures (continued)

- MIMD architecture
  - Multiple instruction/Multiple data
  - Multiple processors running in parallel
  - Each processor performs its own operations on its own data
  - Processors communicate with each other

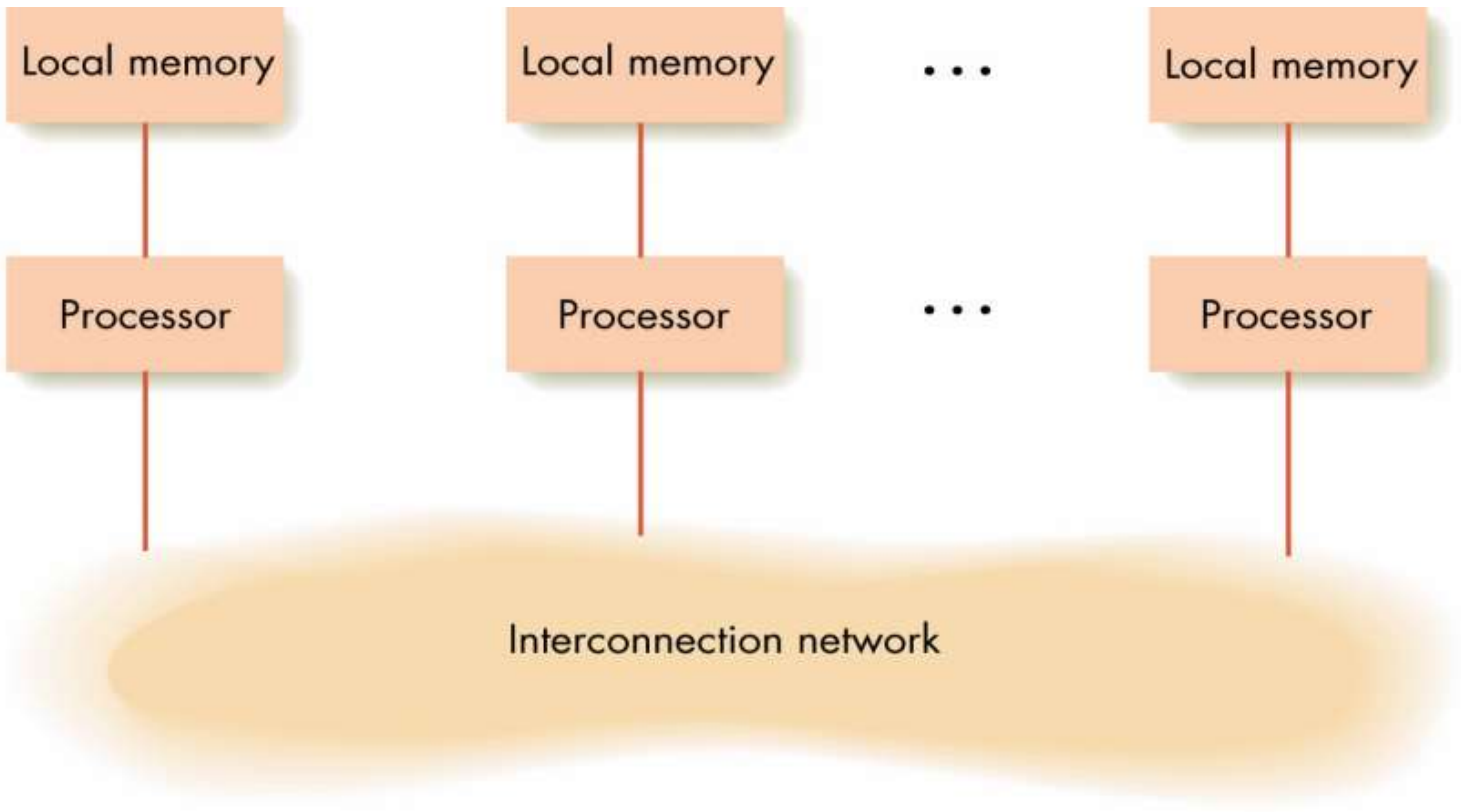


Figure 5.22

Model of MIMD Parallel Processing

Summary of Level 2

- Focus on how to design and build computer systems
- Chapter 4
  - Binary codes
  - Transistors
  - Gates
  - Circuits

Summary of Level 2 (continued)

- Chapter 5
  - Von Neumann architecture
  - Shortcomings of the sequential model of computing
  - Parallel computers

Summary

- Computer organization examines different subsystems of a computer: memory, input/output, arithmetic/logic unit, and control unit
- Machine language gives codes for each primitive instruction the computer can perform, and its arguments
- Von Neumann machine: sequential execution of stored program
- Parallel computers improve speed by doing multiple tasks at one time