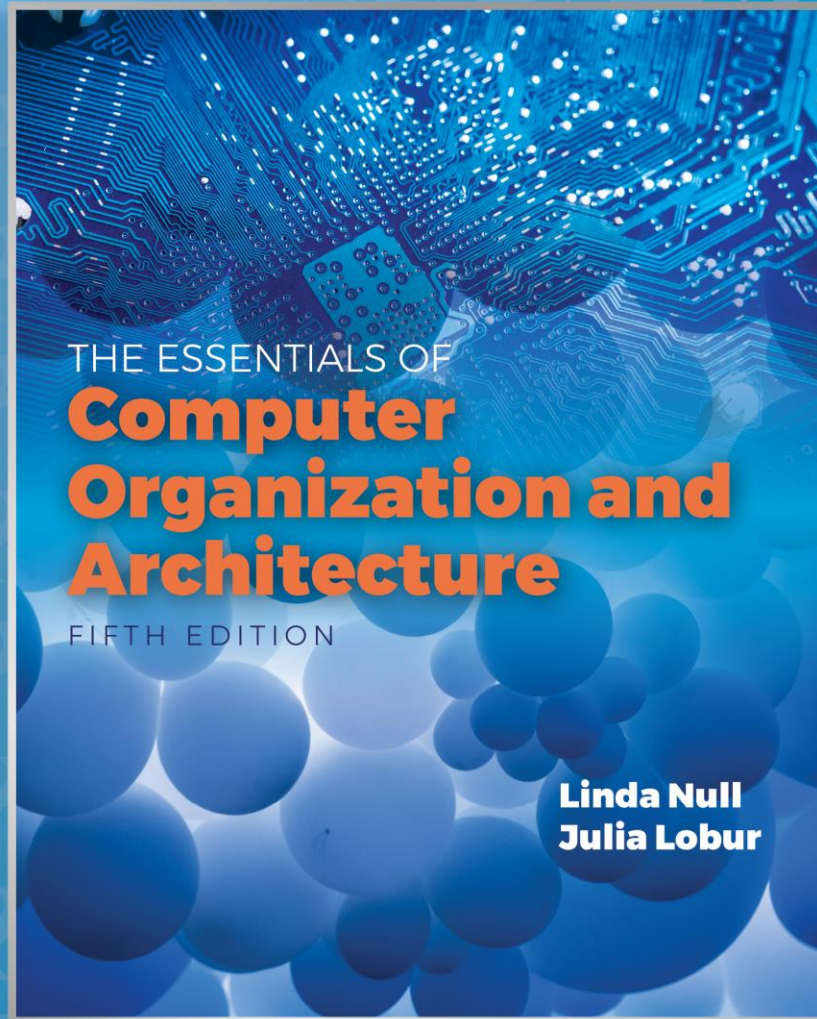


This is the first
lecture of
Chapter 4

Chapter 4

MARIE: An
Introduction
to a Simple Computer



Objectives

- Learn the components common to every modern computer system.
- Be able to explain how each component contributes to program execution.
- Understand a simple architecture invented to illuminate these basic concepts, and how it relates to some real architectures.
- Know how the program assembly process works.

4.1 Introduction

- Chapter 1 presented a general overview of computer systems.
- In Chapter 2, we discussed how data is stored and manipulated by various computer system components.
- Chapter 3 described the fundamental components of digital circuits.
- Having this background, we can now understand how computer components work, and how they fit together to create useful computer systems.

4.2 CPU Basics (1 of 2)

- The computer's CPU fetches, decodes, and executes program instructions.
- The two principal parts of the CPU are the *datapath* and the *control unit*.
 - The datapath consists of an arithmetic-logic unit and storage units (registers) that are interconnected by a data bus that is also connected to main memory.
 - Various CPU components perform sequenced operations according to signals provided by its control unit.

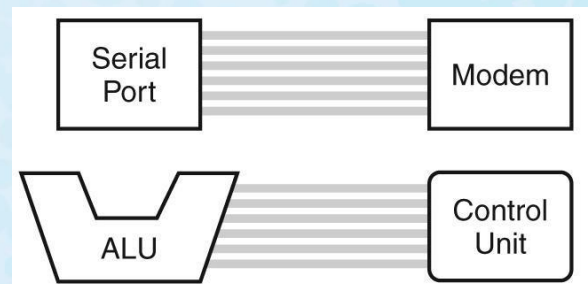
4.2 CPU Basics (2 of 2)

- Registers hold data that can be readily accessed by the CPU.
- They can be implemented using D flip-flops.
 - A 32-bit register requires 32 D flip-flops.
- The arithmetic-logic unit (ALU) carries out logical and arithmetic operations as directed by the control unit.
- The control unit determines which actions to carry out according to the values in a program counter register and a status register.

4.3 The Bus (1 of 5)

- The CPU shares data with other system components by way of a data bus.
 - A bus is a set of wires that simultaneously convey a single bit along each line.
- Two types of buses are commonly found in computer systems: *point-to-point*, and *multipoint* buses.

These are
point-to-point buses:

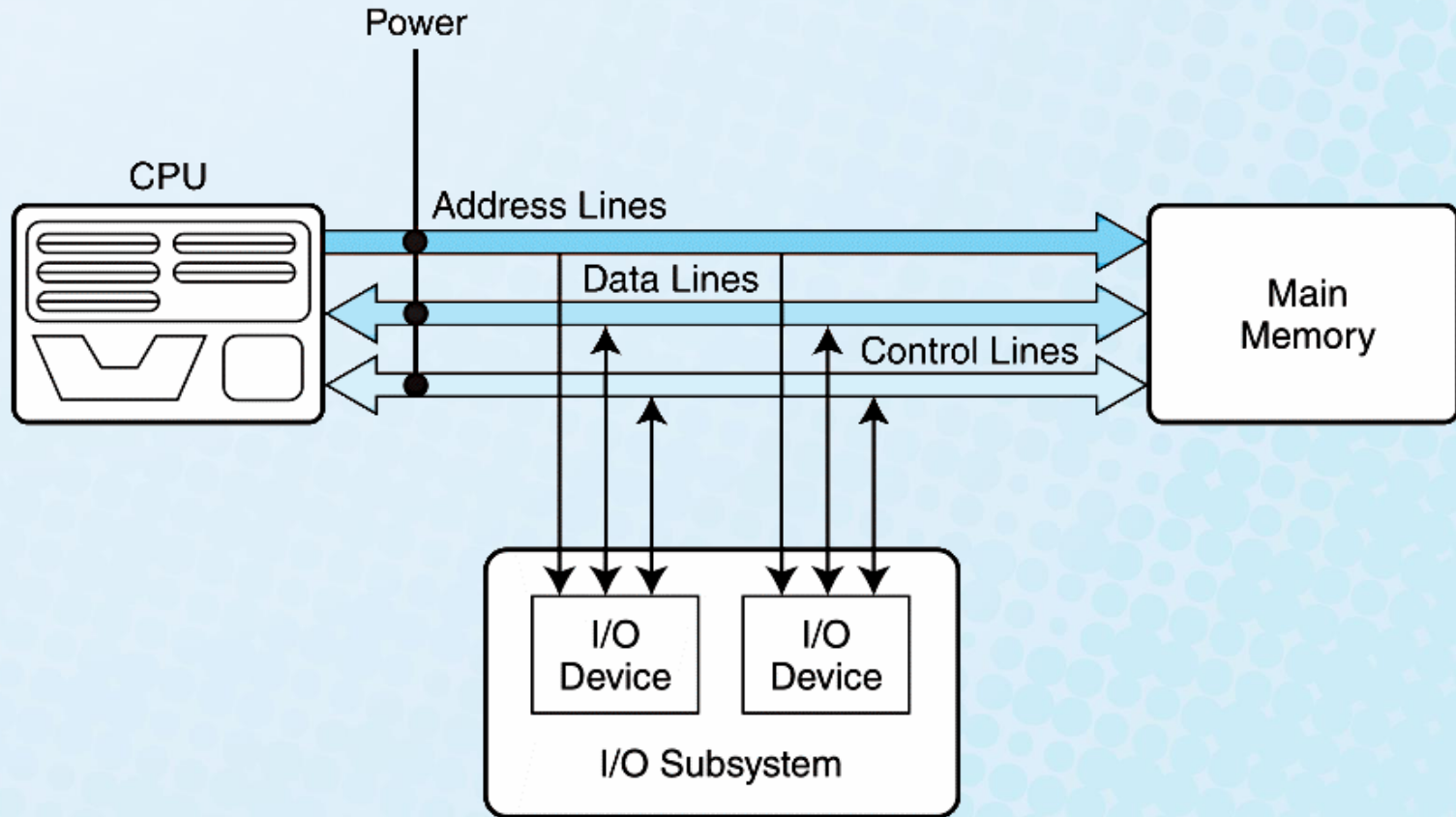


4.3 The Bus (2 of 5)

- Buses consist of data lines, control lines, and address lines.
- While the data lines convey bits from one device to another, control lines determine the direction of data flow, and when each device can access the bus.
- Address lines determine the location of the source or destination of the data.

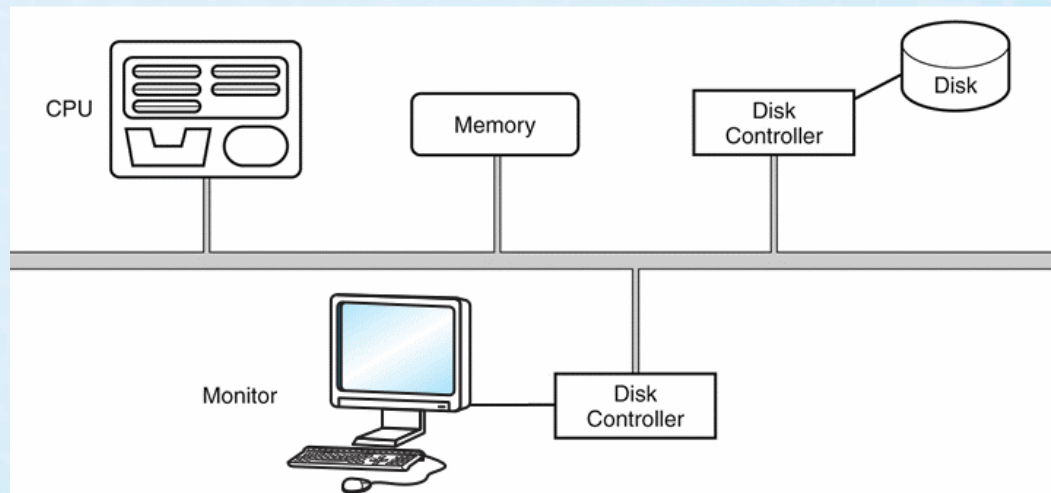
The next slide shows a model bus configuration.

4.3 The Bus (3 of 5)



4.3 The Bus (4 of 5)

- A multipoint bus is shown below.
- Because a multipoint bus is a shared resource, access to it is controlled through protocols, which are built into the hardware.



4.3 The Bus (5 of 5)

- In a master-slave configuration, where more than one device can be the bus master, concurrent bus master requests must be arbitrated.
- Four categories of bus arbitration are:
 - **Daisy chain:** Permissions are passed from the highest-priority device to the lowest.
 - **Centralized parallel:** Each device is directly connected to an arbitration circuit.
 - **Distributed using self-detection:** Devices decide which gets the bus among themselves.
 - **Distributed using collision-detection:** Any device can try to use the bus. If its data collides with the data of another device, it tries again.

4.4 Clocks (1 of 2)

- Every computer contains at least one clock that synchronizes the activities of its components.
- A fixed number of clock cycles are required to carry out each data movement or computational operation.
- The clock frequency, measured in megahertz or gigahertz, determines the speed with which all operations are carried out.
- Clock cycle time is the reciprocal of clock frequency.
 - An 800 MHz clock has a cycle time of 1.25 ns.

4.4 Clocks (2 of 2)

- Clock speed should not be confused with CPU performance.
- The CPU time required to run a program is given by the general performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- We see that we can improve CPU throughput when we reduce the number of instructions in a program, reduce the number of cycles per instruction, or reduce the number of nanoseconds per clock cycle.

We will return to this important equation in later chapters.

4.5 The Input/Output Subsystem

- A computer communicates with the outside world through its input/output (I/O) subsystem.
- I/O devices connect to the CPU through various interfaces.
- I/O can be memory-mapped—where the I/O device behaves like main memory from the CPU's point of view.
- Or I/O can be instruction-based, where the CPU has a specialized I/O instruction set.

We study I/O in detail in chapter 7.

4.6 Memory Organization (1 of 8)

- Computer memory consists of a linear array of addressable storage cells that are similar to registers.
- Memory can be byte-addressable, or word-addressable, where a word typically consists of two or more bytes.
- Memory is constructed of RAM chips, often referred to in terms of length \times width.
- If the memory word size of the machine is 16 bits, then a $4\text{M} \times 16$ RAM chip gives us 4 mega 16-bit memory locations.

4.6 Memory Organization (2 of 8)

- How does the computer access a memory location corresponds to a particular address?
- We observe that 4M can be expressed as $2^2 \times 2^{20} = 2^{22}$ words.
- The memory locations for this memory are numbered 0 through $2^{22} - 1$.
- Thus, the memory bus of this system requires at least 22 address lines.
 - The address lines “count” from 0 to $2^{22} - 1$ in binary. Each line is either “on” or “off” indicating the location of the desired memory element.

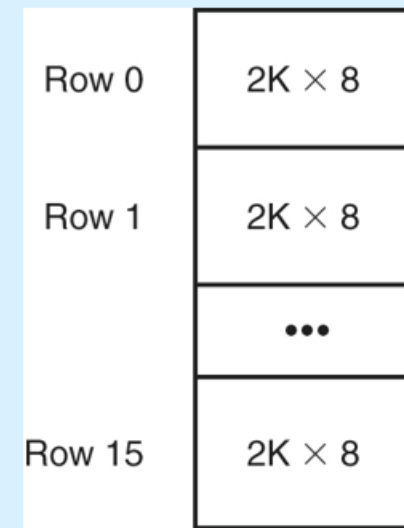
4.6 Memory Organization (3 of 8)

- Physical memory usually consists of more than one RAM chip.
- Access is more efficient when memory is organized into banks of chips with the addresses interleaved across the chips
- With low-order interleaving, the low order bits of the address specify which memory bank contains the address of interest.
- Accordingly, in high-order interleaving, the high order address bits specify the memory bank.

The next two slides illustrate these two ideas.

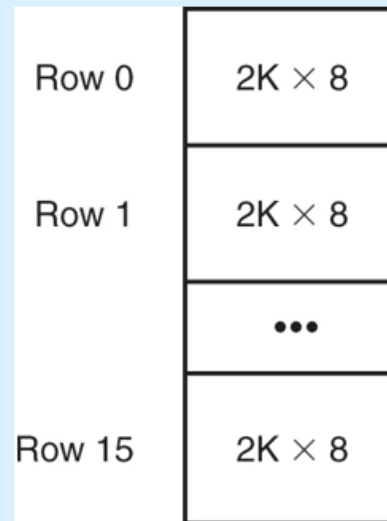
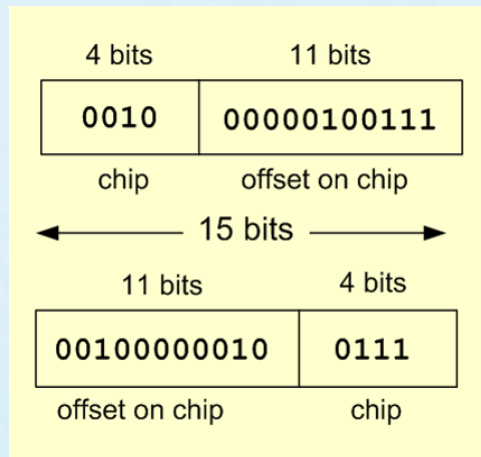
4.6 Memory Organization (4 of 8)

- Example: Suppose we have a memory consisting of 16 2K x 8 bit chips.
 - Memory is $32K = 2^5 \times 2^{10} = 2^{15}$
 - 15 bits are needed for each address.
 - We need 4 bits to select the chip, and 11 bits for the offset into the chip that selects the byte.



4.6 Memory Organization (5 of 8)

- In high-order interleaving the high-order 4 bits select the chip.
- In low-order interleaving the low-order 4 bits select the chip.



4.6 Memory Organization (6 of 8)

a)

Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7
0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

b)

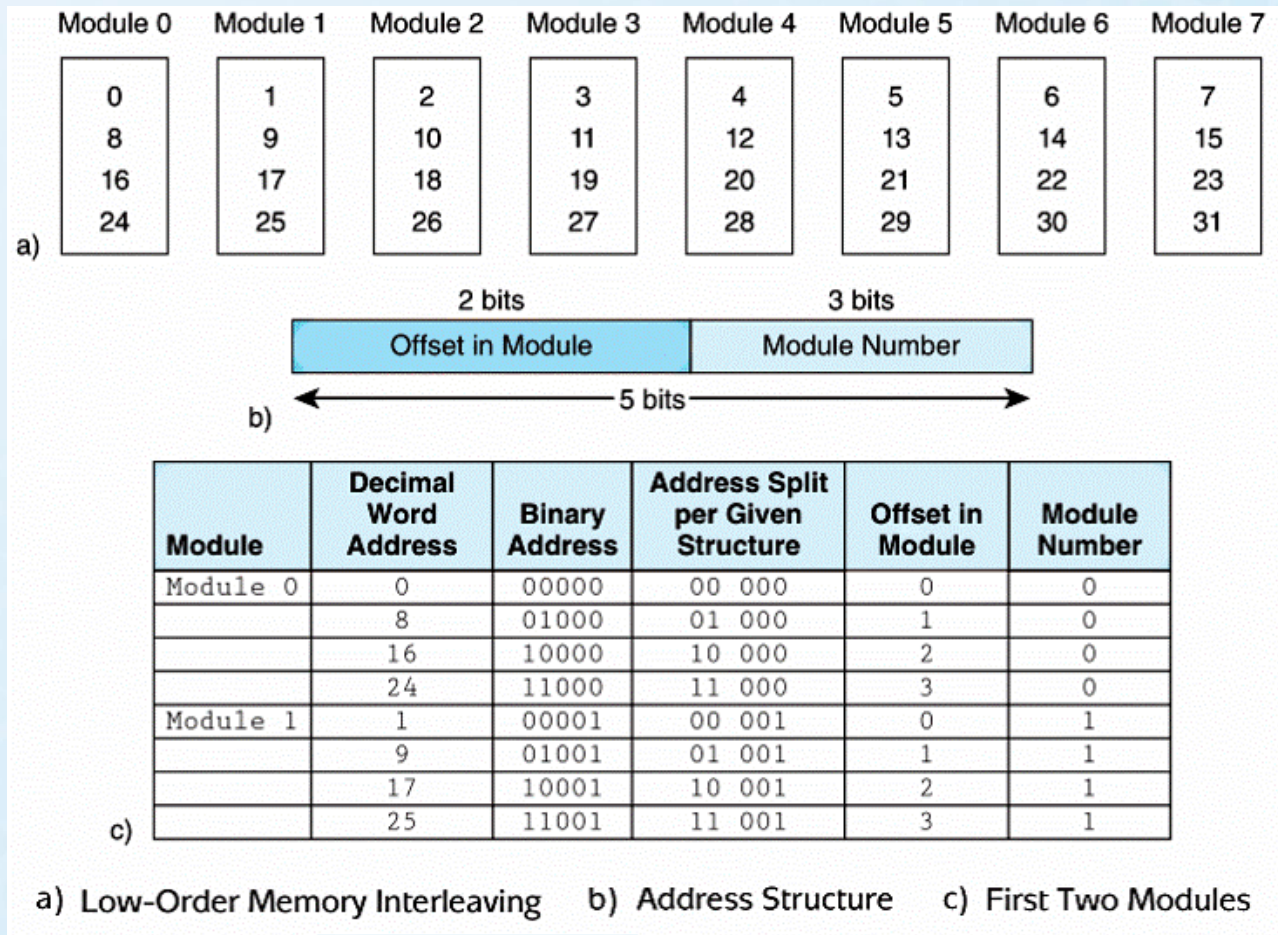
The diagram shows a horizontal bar representing a 5-bit address. The left portion is labeled '3 bits' and 'Module number'. The right portion is labeled '2 bits' and 'Offset in module'. A double-headed arrow below the bar indicates the total width is 5 bits.

c)

Module	Decimal Word Address	Binary Address	Address Split per Given Structure	Module Number	Offset in Module
Module 0	0	00000	000 00	0	0
	1	00001	000 01	0	1
	2	00010	000 10	0	2
	3	00011	000 11	0	3
Module 1	4	00100	001 00	1	0
	5	00101	001 01	1	1
	6	00110	001 10	1	2
	7	00111	001 11	1	3

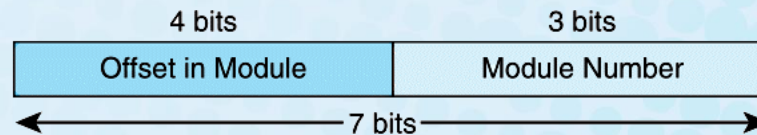
a) High-Order Memory Interleaving b) Address Structure c) First Two Modules

4.6 Memory Organization (7 of 8)



4.6 Memory Organization (8 of 8)

- EXAMPLE 4.1: Suppose we have a 128-word memory that is 8-way low-order interleaved
 - which means it uses 8 memory banks; $8 = 2^3$
- So we use the low-order 3 bits to identify the bank.
- Because we have 128 words, we need 7 bits for each address ($128 = 2^7$).



4.7 Interrupts

- The normal execution of a program is altered when an event of higher-priority occurs. The CPU is alerted to such an event through an interrupt.
- Interrupts can be triggered by I/O requests, arithmetic errors (such as division by zero), or when an invalid instruction is encountered.
- Each interrupt is associated with a procedure that directs the actions of the CPU when an interrupt occurs.
 - Nonmaskable interrupts are high-priority interrupts that cannot be ignored.