# Introduction to Simple Computer Part I

Hui Chen

Computer & Information Science

CUNY Brooklyn College

Fall 2023

# 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.

# CPU

- Fetches, decodes, and executes program instructions.
- Principal parts: 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.*

# CPU Storage Units: Registers

- 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.

# ALU

- The arithmetic-logic unit (ALU) carries out logical and arithmetic operations as directed by the control unit.

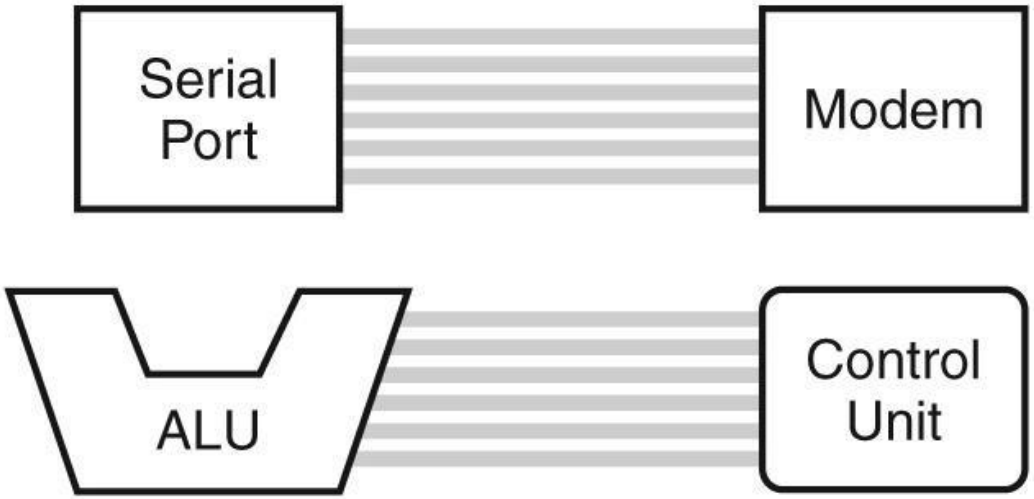
# Control Unit and Registers

- The control unit determines which actions to carry out according to the values in a *program counter register* and a *status register*.

# The Bus

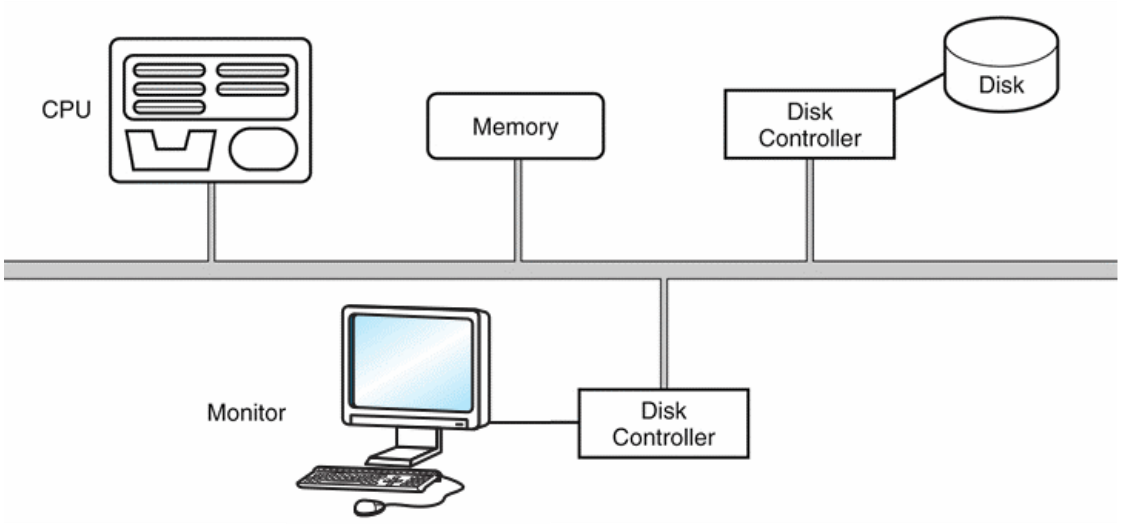
- The CPU shares data with other system components by way of a data bus.
  - A bus is a set of wires that simultaneously convey a single bit along each line.
- Two types of buses are commonly found in computer systems:
  - *point-to-point*, and
  - *multipoint* buses.

# Point-to-Point Bus



# Multipoint Bus

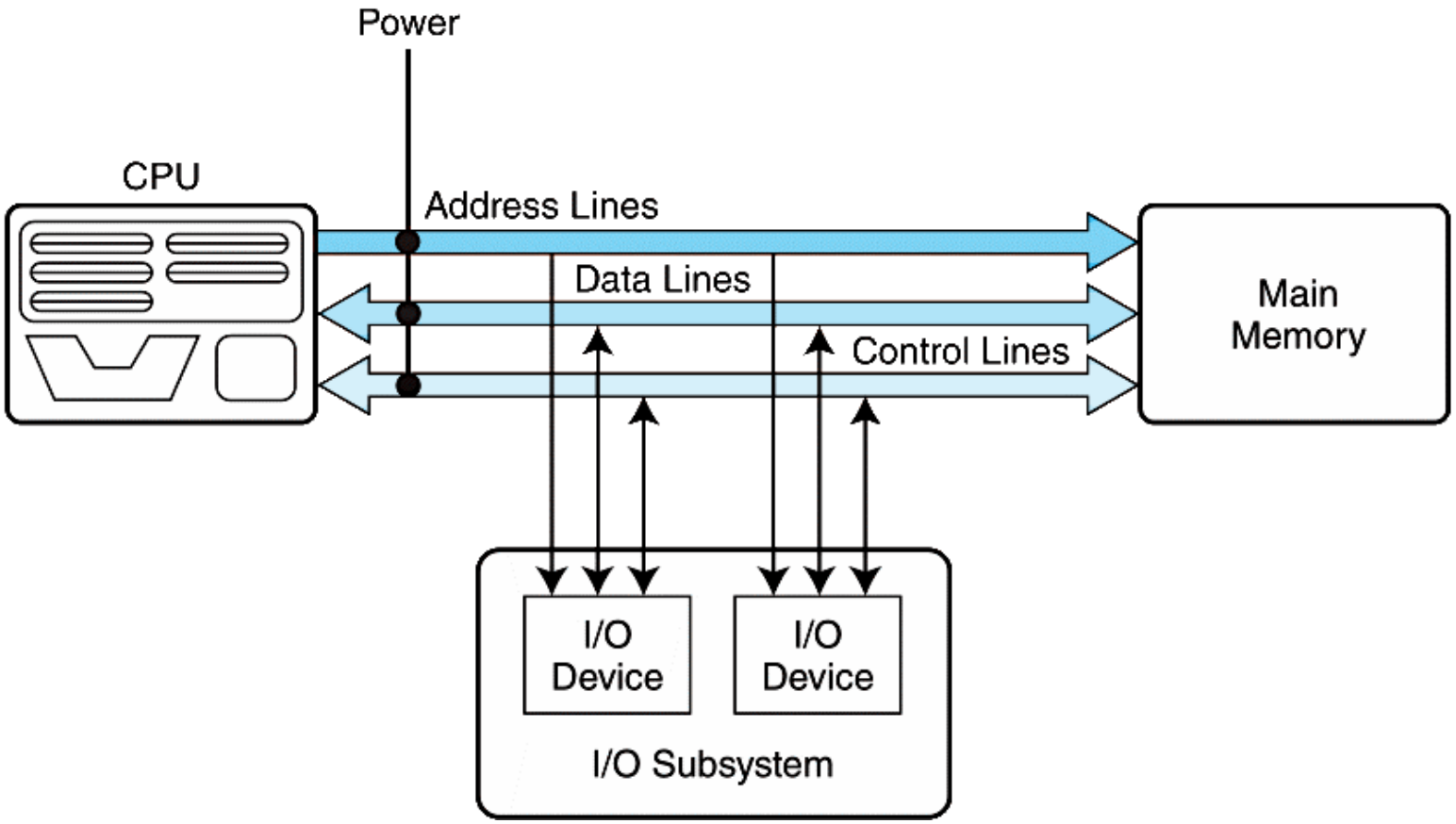
- Because a multipoint bus is a shared resource, access to it is controlled through protocols, which are built into the hardware.



# BUS Lines

- Buses consist of data lines, control lines, and address lines.
  - 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.

# Example Bus Configuration



# Bus Configuration

- Master-slave configuration
  - 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.

# Clocks

- 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.

# Clock Speed vs. Clock Performance

- 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 later.

# 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 later if time permits

# Memory

- 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 4M $\times$ 16 RAM chip gives us 4 megabytes of 16-bit memory locations.

# Memory Addressing

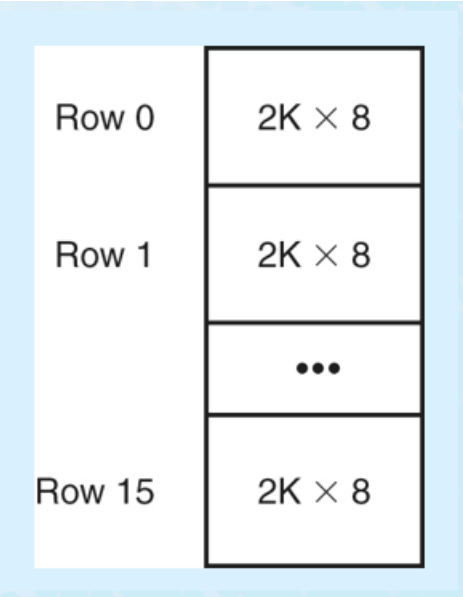
- How does the computer access a memory location corresponds to a particular address?
- We observe that 4M can be expressed as $2^2 \times 2^{20} = 2^{22}$ words.
- The memory locations for this memory are numbered 0 through $2^{22} – 1$.
- Thus, the memory bus of this system requires at least 22 address 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.

# Memory Organization

- 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.

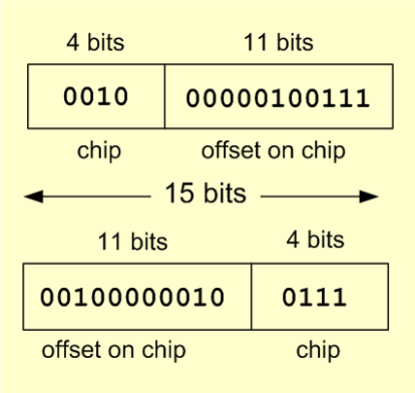
# Example Setup.

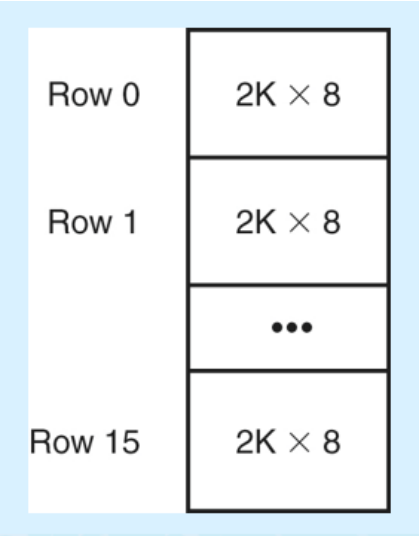
- Suppose we have a memory consisting of 16 2K x 8 bit chips.
  - Memory is 32K = $2^5 \times 2^{10} = 2^{15}$
  - 15 bits are needed for each address.
  - We need 4 bits to select the chip, and 11 bits for the offset into the chip that selects the byte.



# Example:

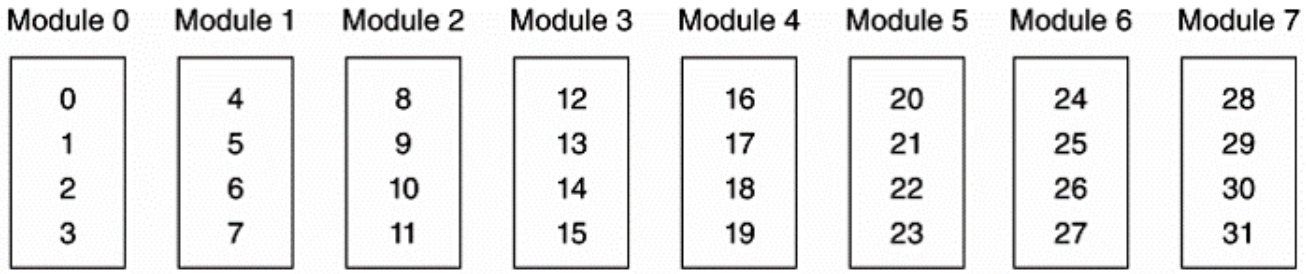
- In high-order interleaving the high-order 4 bits select the chip.
- In low-order interleaving the low-order 4 bits select the chip.



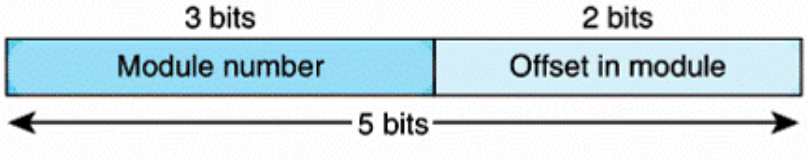



# Addressing Modules

a)




b)



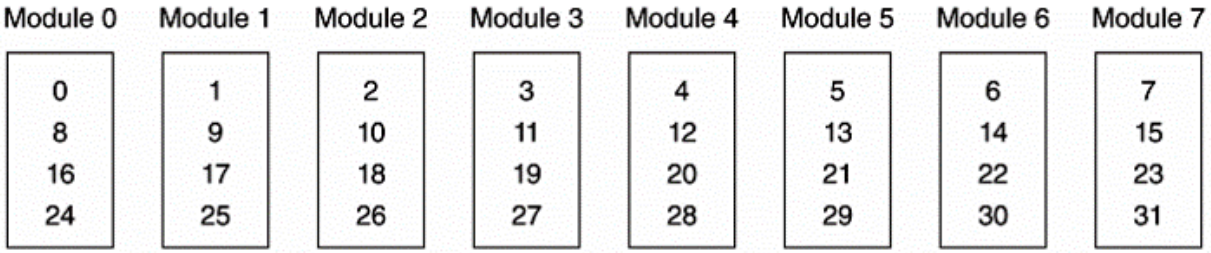

c)

| 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 |

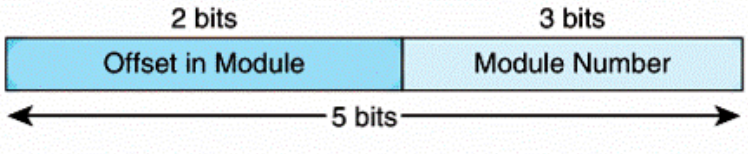
a) High-Order Memory Interleaving b) Address Structure c) First Two Modules

# Addressing Modules

a)




b)



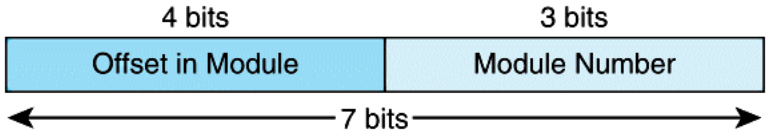

c)

| Module | Decimal Word Address | Binary Address | Address Split per Given Structure | Offset in Module | Module Number |
|---|---|---|---|---|---|
| Module 0 | 0 | 00000 | 00 000 | 0 | 0 |
| | 8 | 01000 | 01 000 | 1 | 0 |
| | 16 | 10000 | 10 000 | 2 | 0 |
| | 24 | 11000 | 11 000 | 3 | 0 |
| Module 1 | 1 | 00001 | 00 001 | 0 | 1 |
| | 9 | 01001 | 01 001 | 1 | 1 |
| | 17 | 10001 | 10 001 | 2 | 1 |
| | 25 | 11001 | 11 001 | 3 | 1 |

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

# Example

- 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 ($128 = 2^7$).



# 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.

# Summary

- The major components of a computer system are its control unit, registers, memory, ALU, and data path.
- A built-in clock keeps everything synchronized.

# Questions?

- CPU
- Bus
- Clock
- I/O System
- Memory