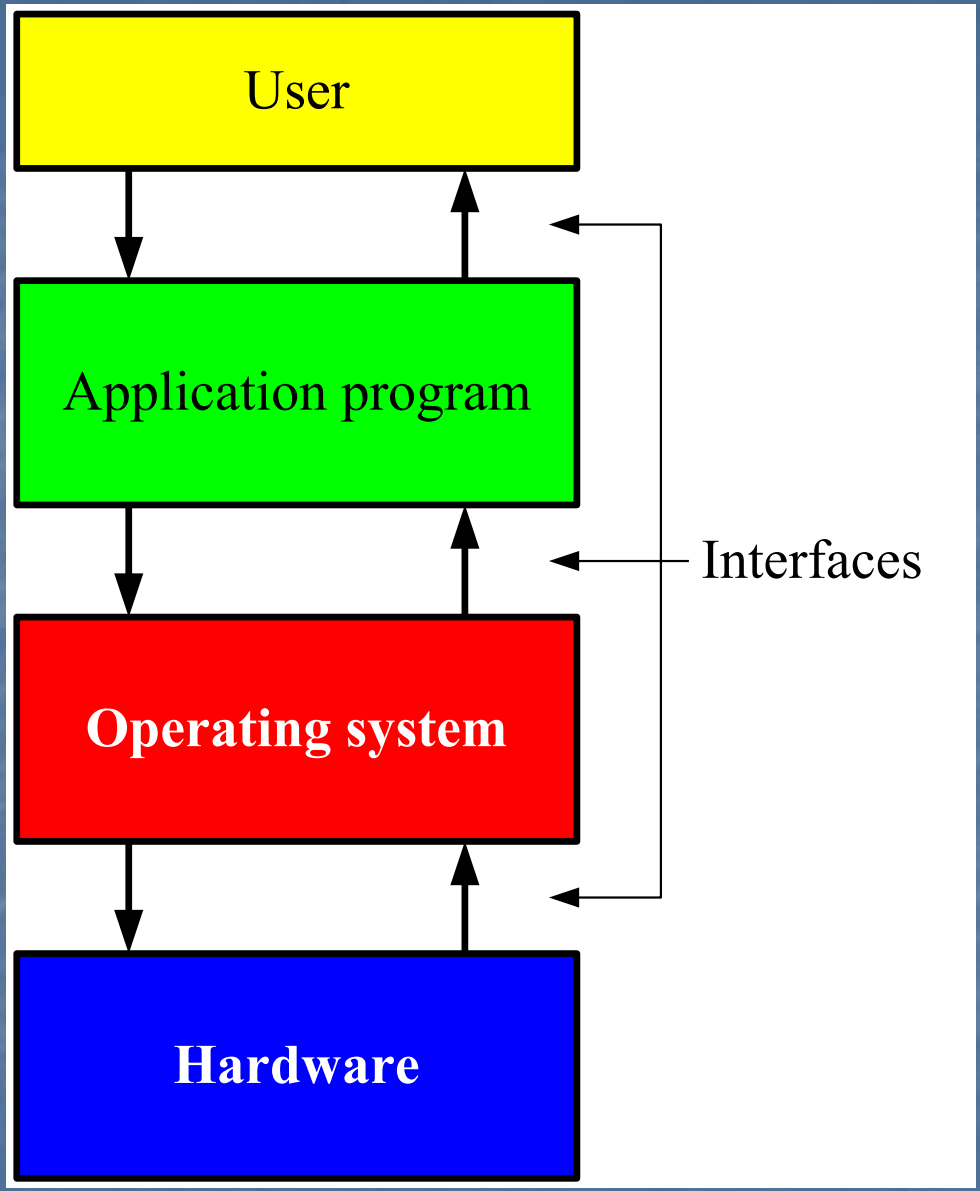
Operating Systems

Introduction to Operating Systems
and Computer Hardware

Introduction and Overview

The operating system is a set of system software routines that interface between an application program and the hardware.



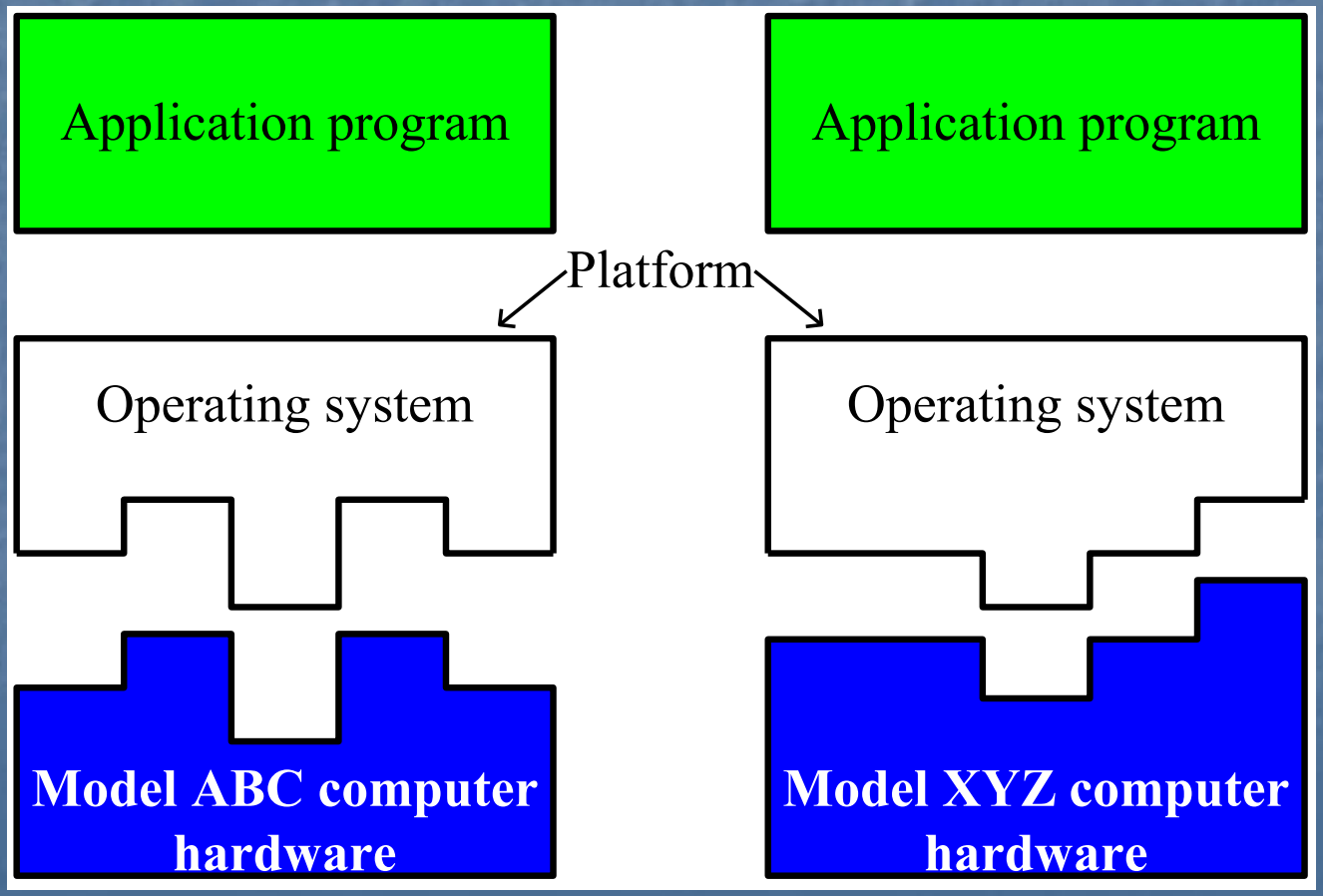
Interfaces

- Points of connection or linkage
  - User interfaces with application program
  - Application program interfaces with operating system
  - Operating system interfaces with hardware
- Application program accesses hardware
  - Through the operating system
  - Following the operating system's rules

Operating System Services

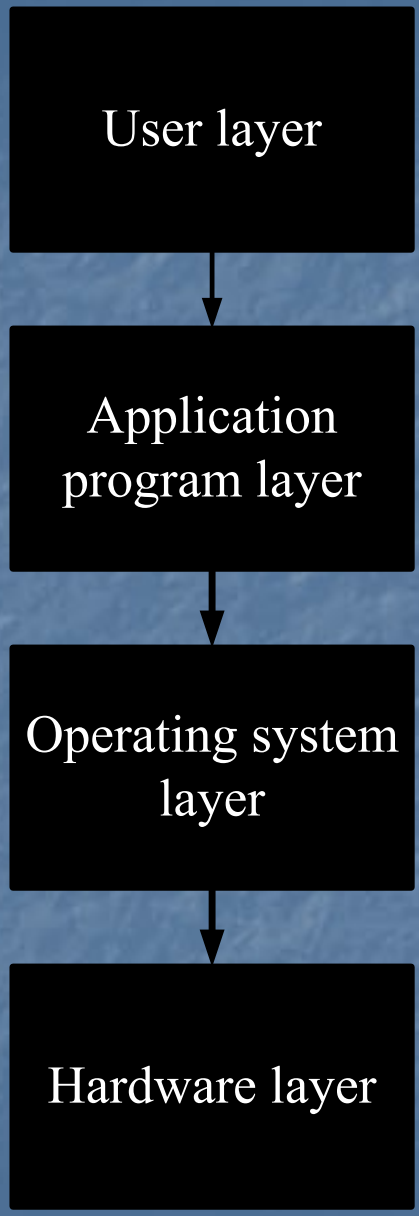
- Service
  - Software routine that runs to support another program:
    - Is small and simple
    - Performs single task
  - An operating system is a repository of common services
    - I/O support—open, close, read, and write
    - Program launch

The operating system routines that interface with the application program represent a consistent platform for running the program.



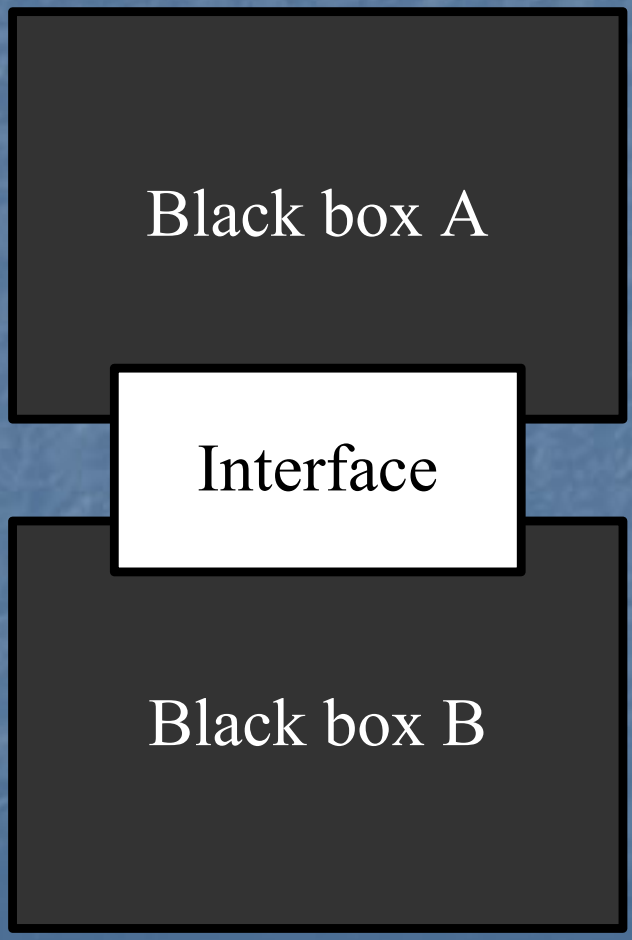
Layers of Abstraction

- Abstraction
  - A simplified view of an object that ignores internal details.
  - Each layer of abstraction is a black box.
- Black box
  - Inputs/outputs known
  - Contents hidden
  - Functionally independent

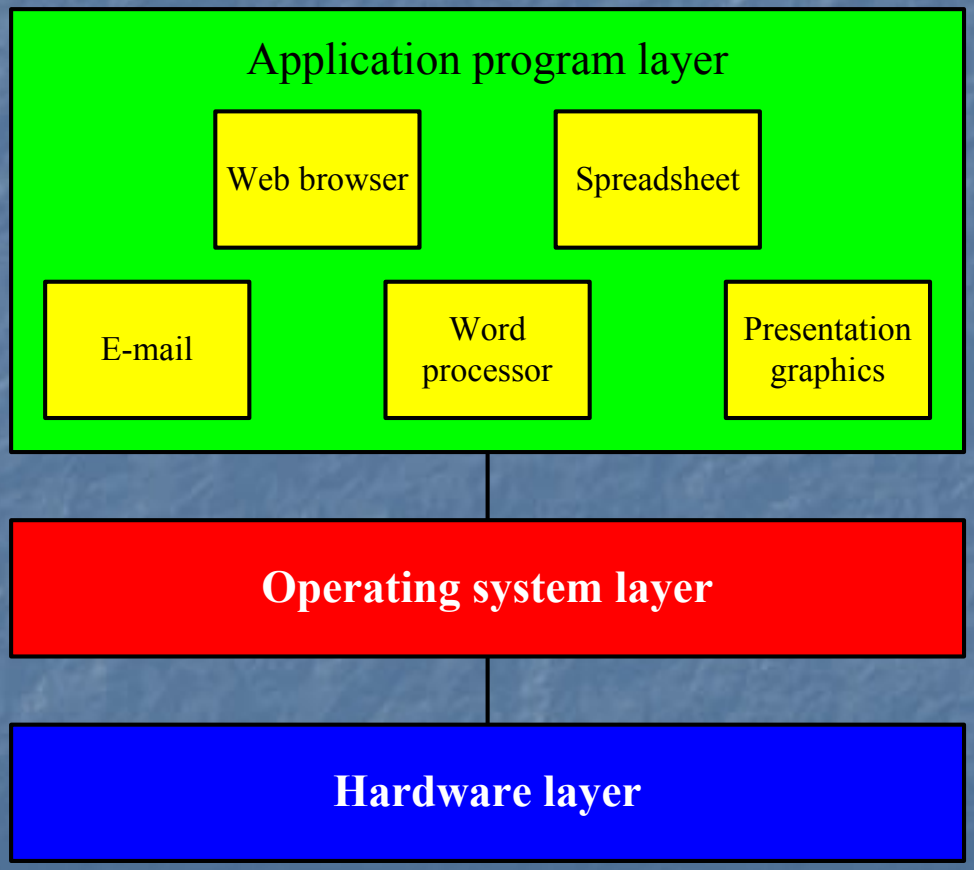


Communication Between Two Black Boxes

- Two black boxes communicate through a shared interface.
- To use a black box, all you must know are its interface rules.

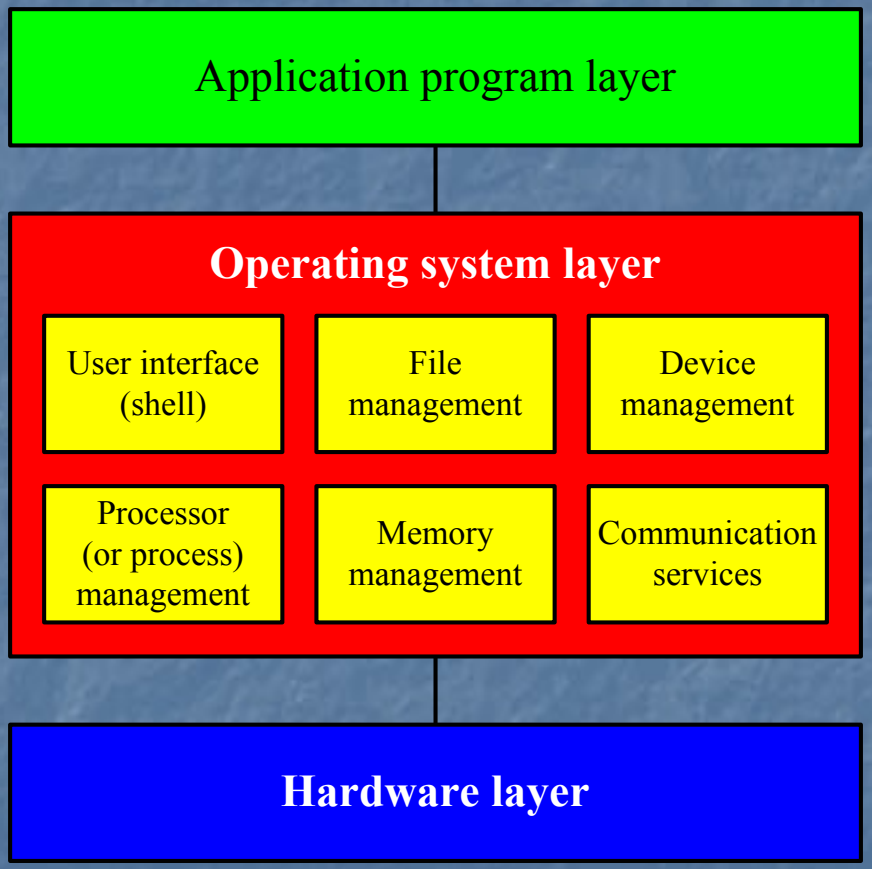


An operating system can support many application programs.



Primary Components of an Operating System

- Apparent to user
  - Shell
  - File system
  - Device management
- Transparent
  - Processor management
  - Memory management
  - Communication services

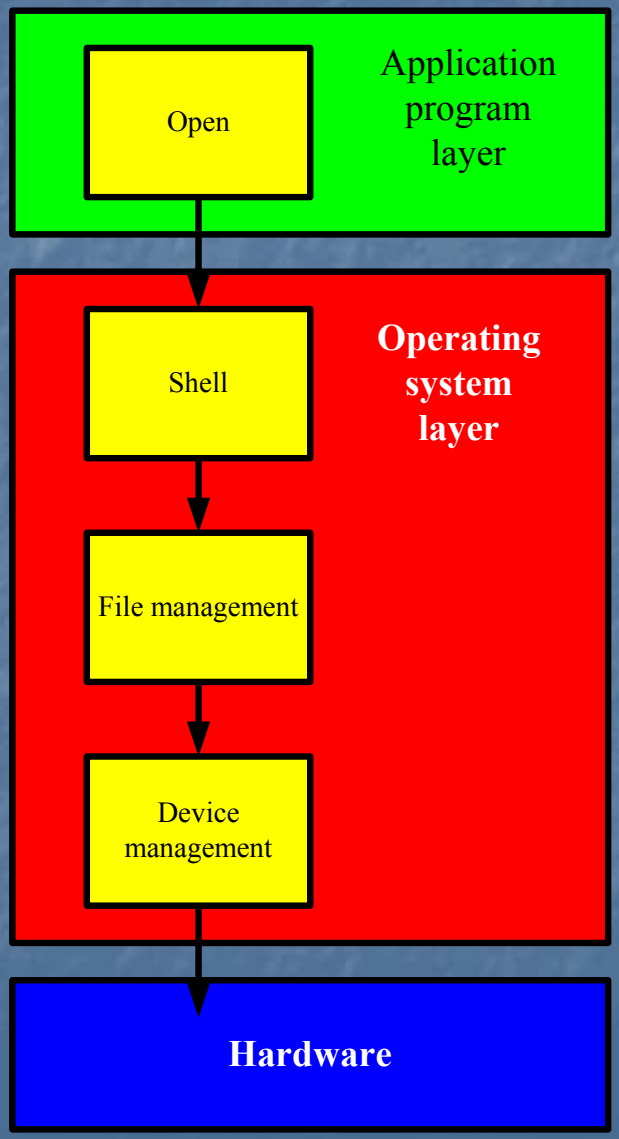


Resources that need to be managed in a computer system

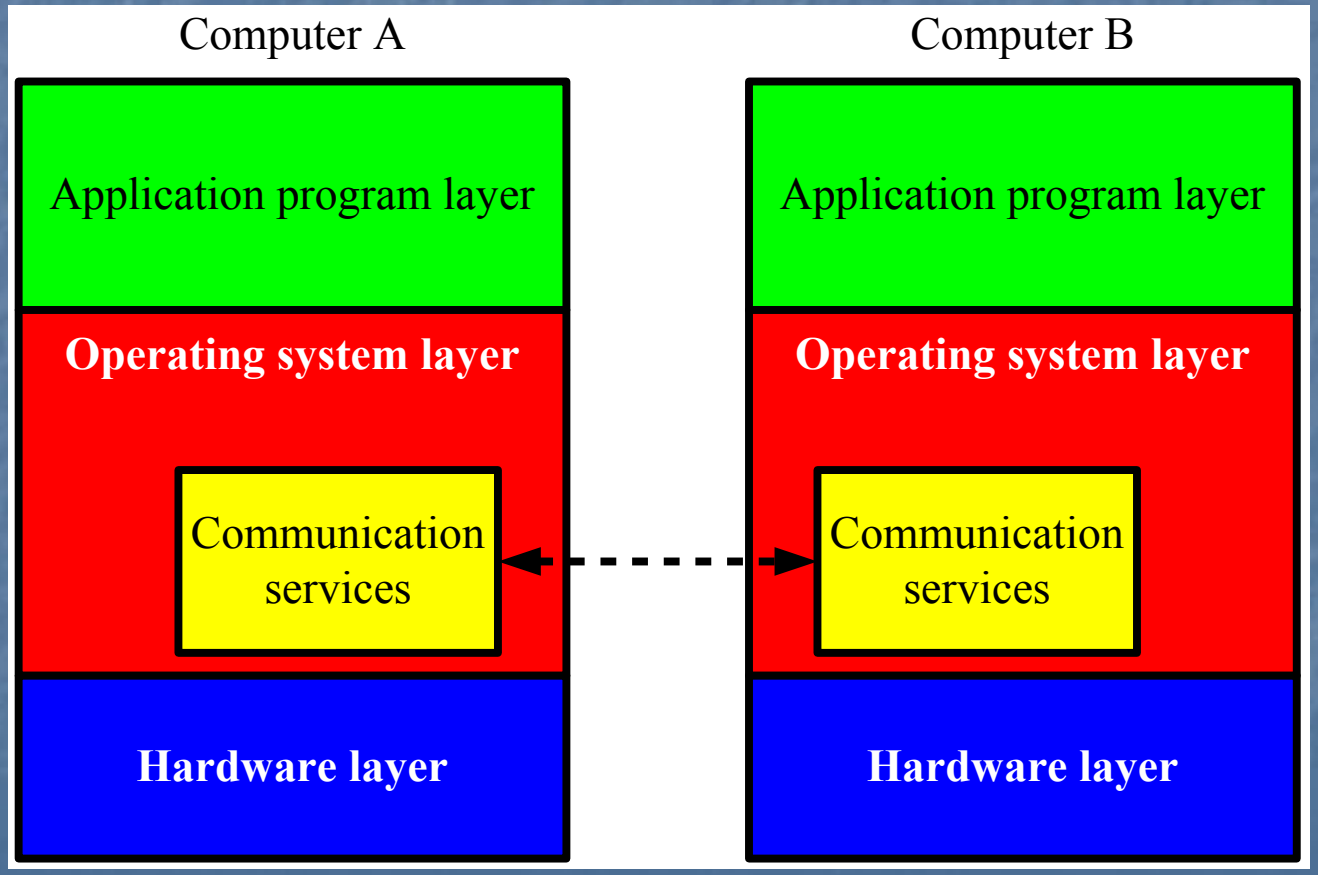
- CPU
- Memory (RAM
- Storage (Secondary Memory of all types)
- GPU (Graphics Processing Unit)
- Bandwidth
- Print Resources

Opening a File

- The user selects *Open.*
- The shell interprets the command.
- File management finds the file.
- Device management reads the file from disk.



Communication between two computers is enabled by communication services installed on both computers.



Advantages of Layering

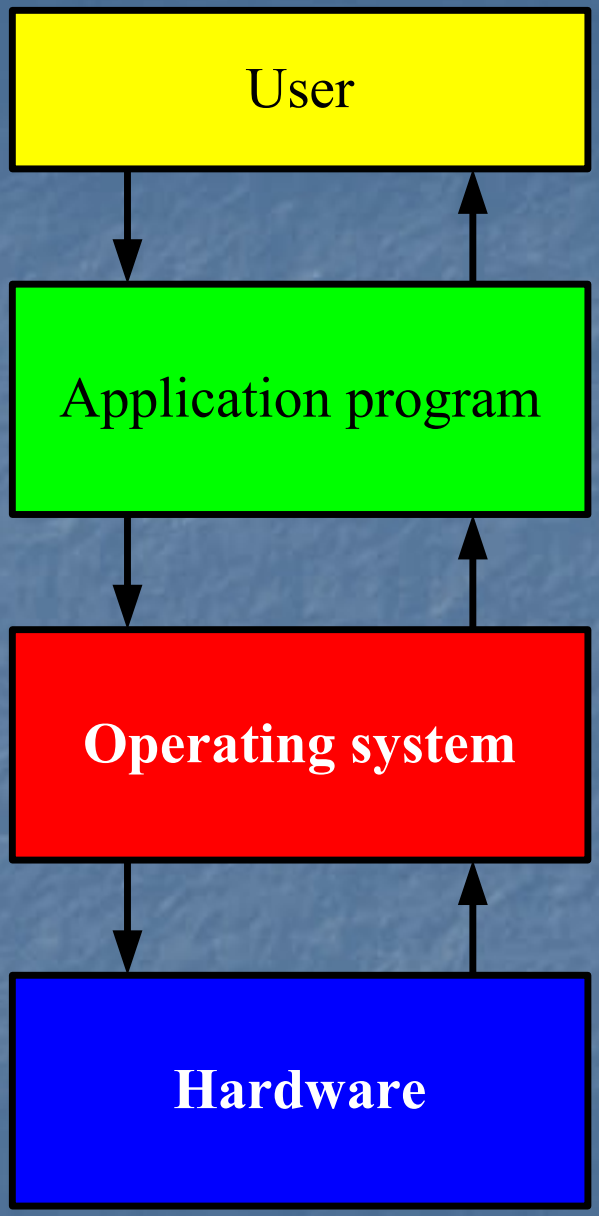
- Viewing the system as a set of layers simplifies:
  - Maintenance
  - System assembly
- Stacking layers is similar to stacking blocks.

Important Takeaways

- Open source
  - UNIX and Linux
  - Open, published source code
- Proprietary
  - Apple Macintosh
  - Closed source code
- Hybrid
  - Microsoft
  - Source code open for some elements and closed for others

Hardware

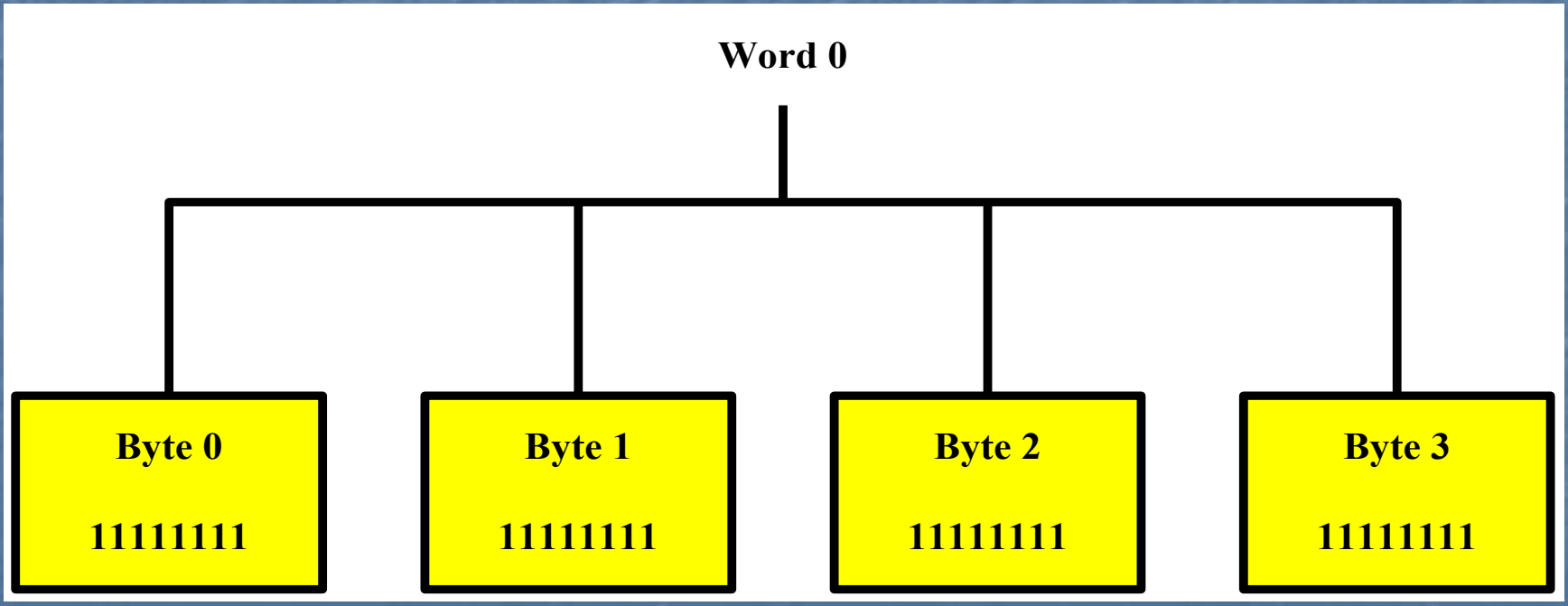
This section focuses on the key components of the hardware layer.



Memory

- Holds active programs and data.
- Contents:
  - Bit: One binary digit—0 or 1
  - Byte: Eight bits—1 character
  - Word: A group of bytes
- Write is a destructive operation.
- Read is not a destructive operation.

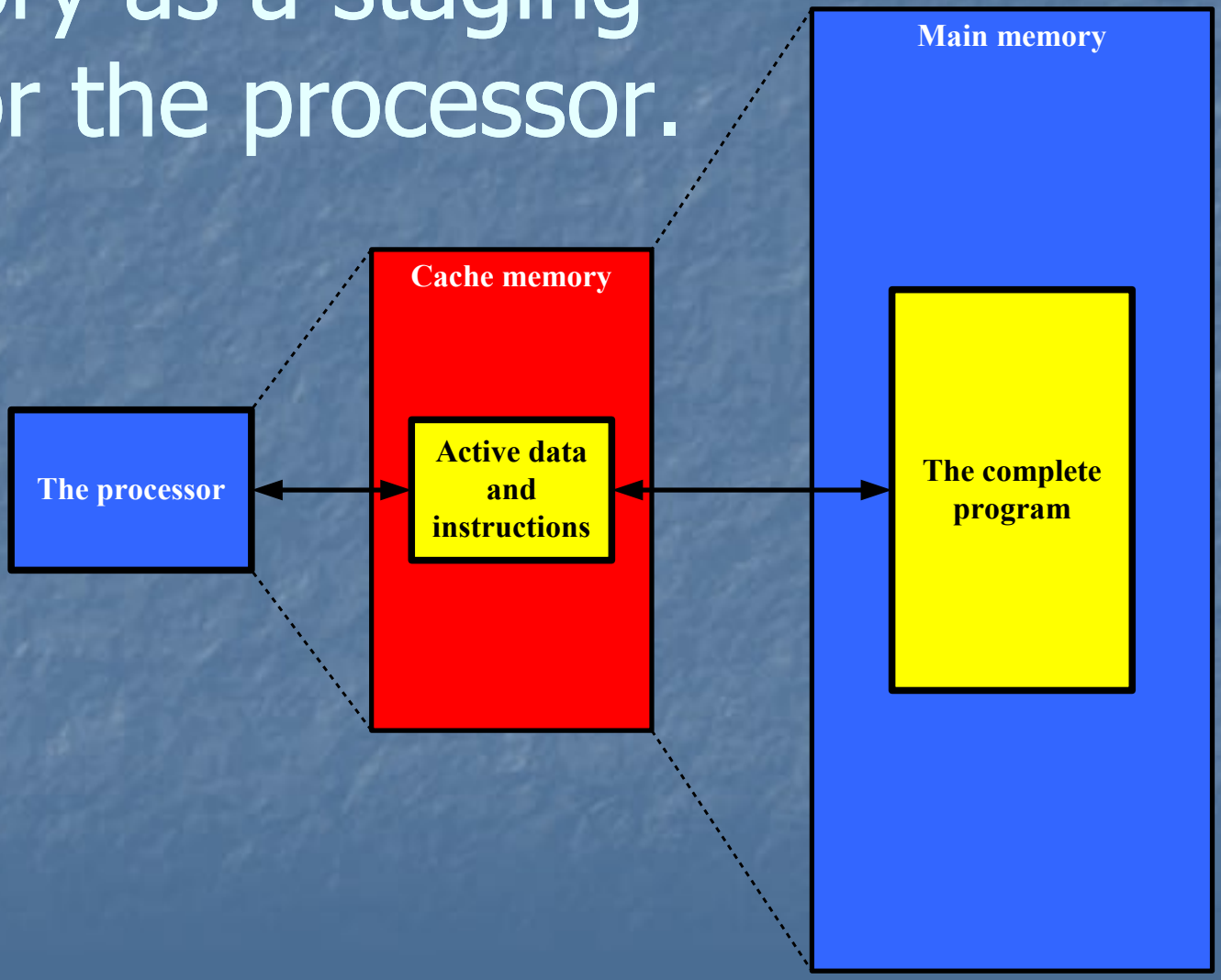
In the memory of a computer, bits are grouped to form bytes, which, in turn, are grouped to form words.



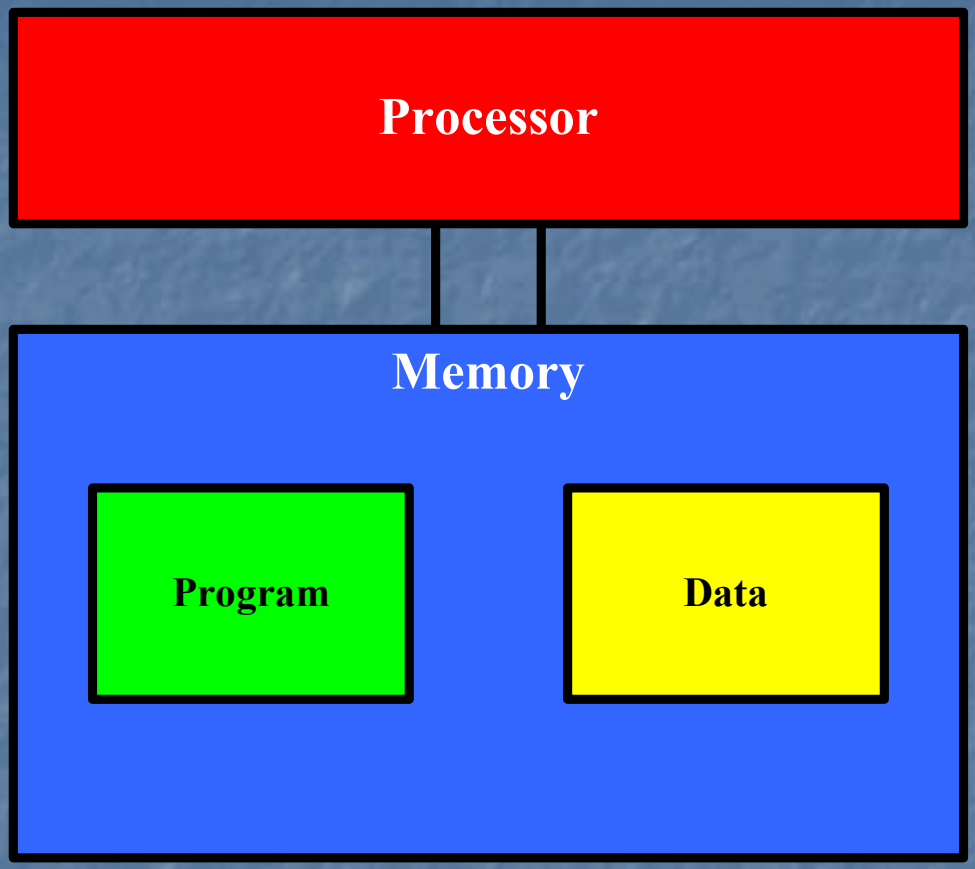
Addressing Memory

- Bytes or words are the basic addressable units of memory.
- Each byte or word is assigned a unique address.
- Bytes or words are numbered sequentially.

Think of cache memory as a staging area for the processor.



The processor manipulates the data stored in memory under the control of a program stored in the memory.

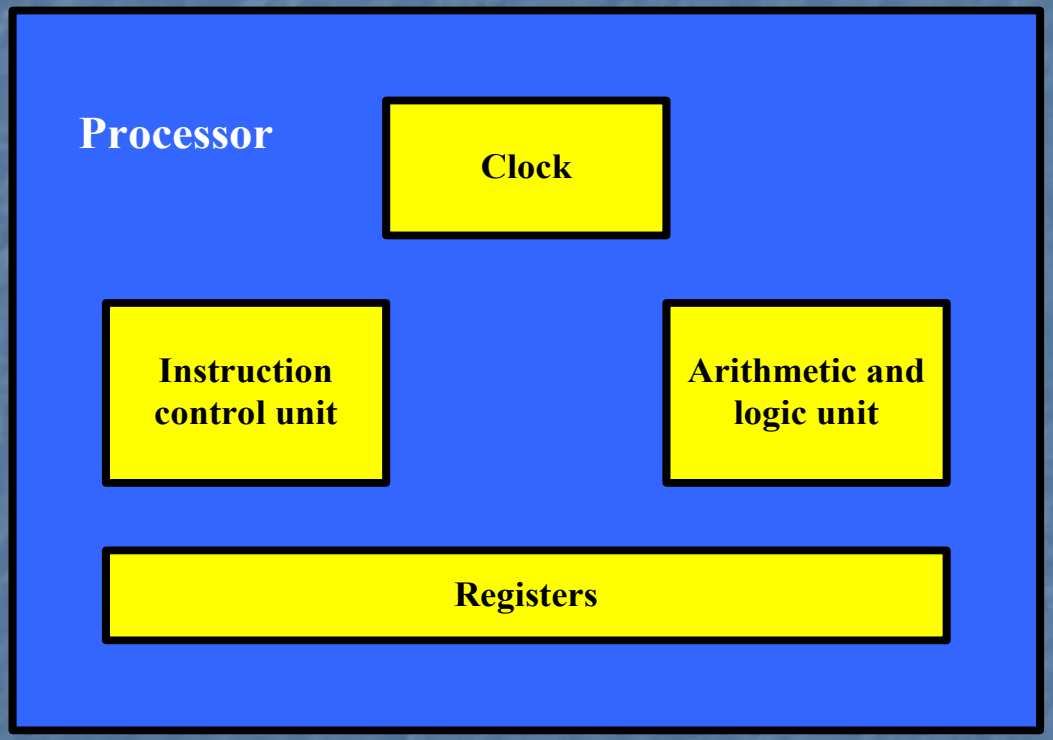


Each instruction has an operation code and one or more operands.

Operation code	Operands
ADD	*1000,1004*

Program: A series of instructions.

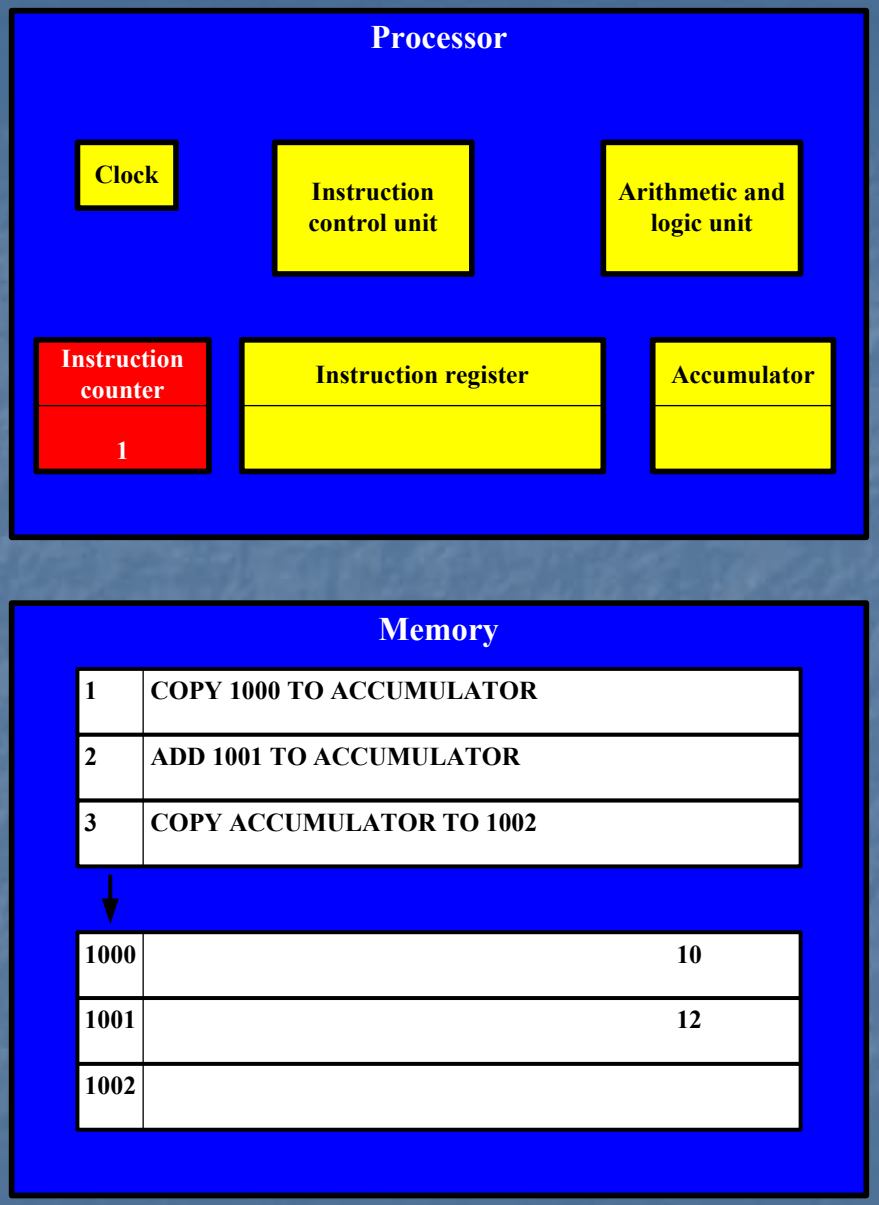
Key Components of a Processor



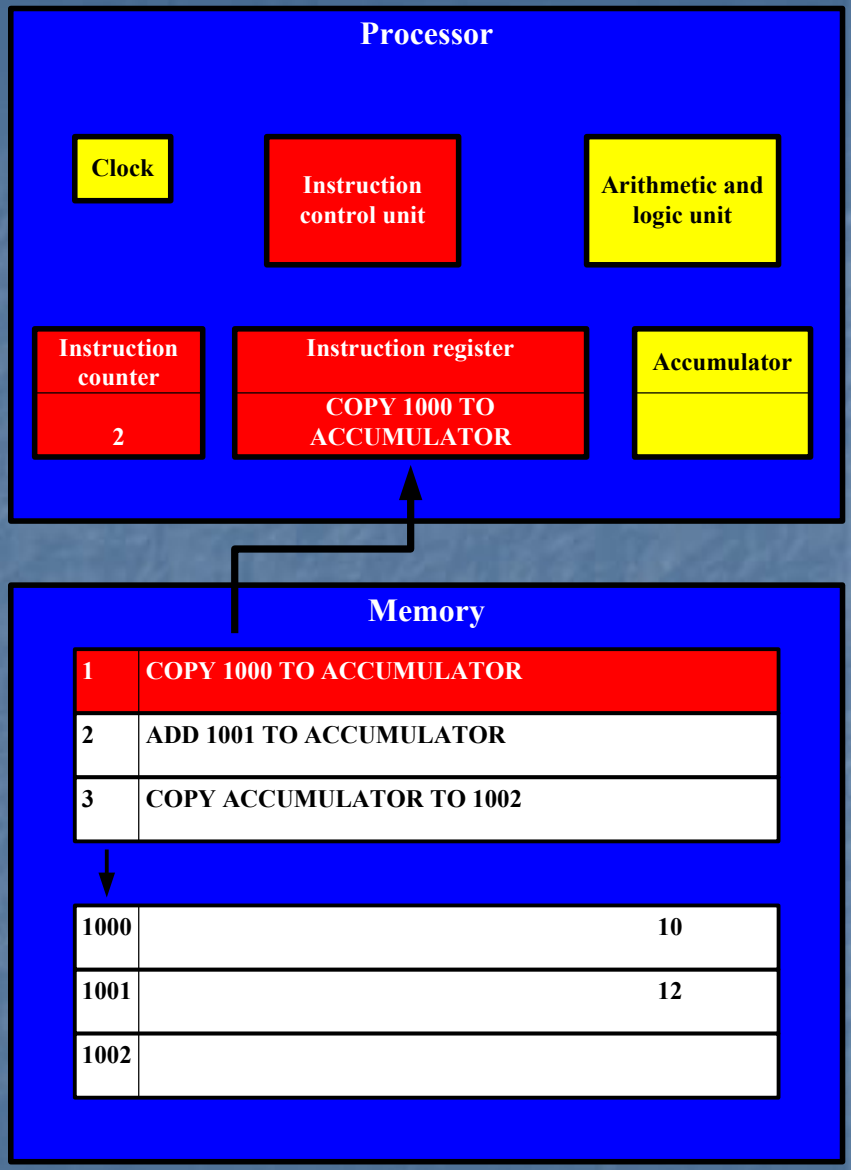
Machine Cycle

A computer executes instructions by following the basic machine cycle.

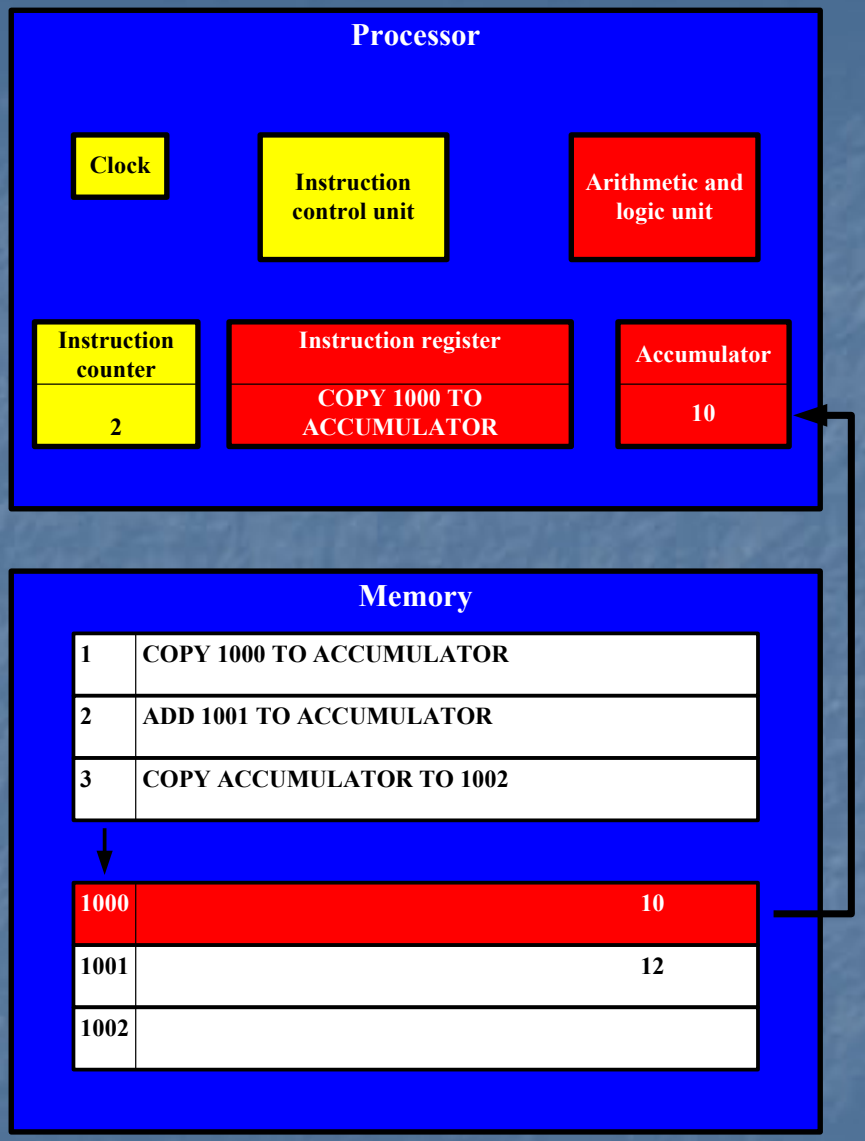
a. In this example, memory holds both instructions and data initially. The instruction counter points to the first instruction to be executed.



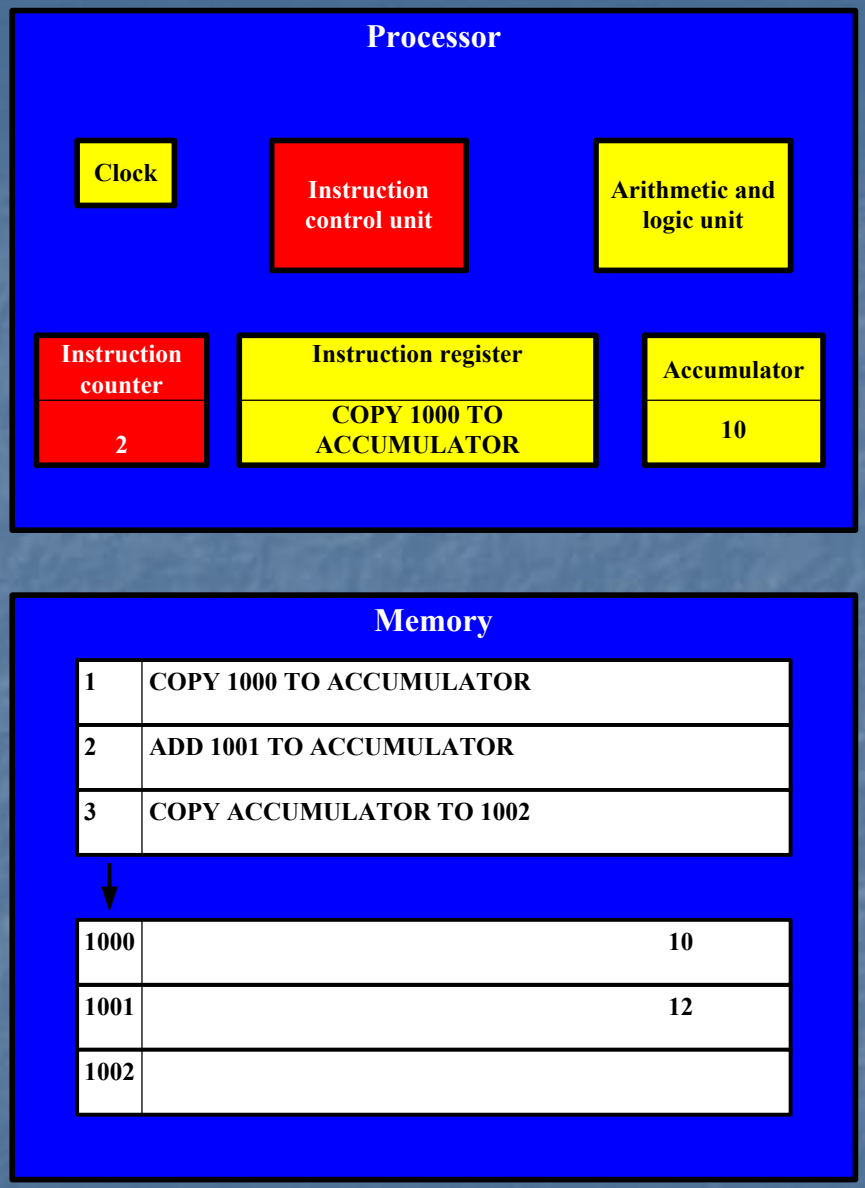
b. The first instruction is fetched from memory and stored in the instruction register. Keep in mind that the instruction counter points to the next instruction.



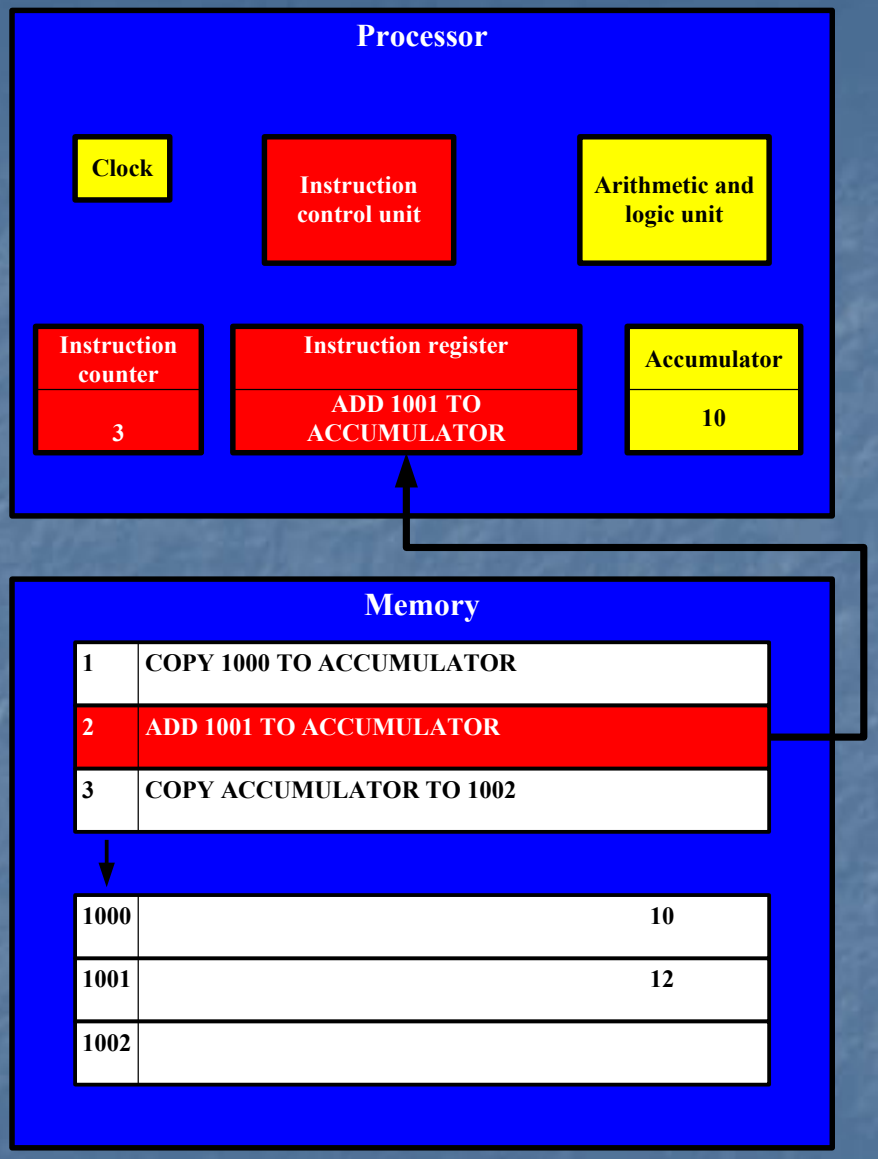
c. The arithmetic and logic unit executes the instruction in the instruction register.



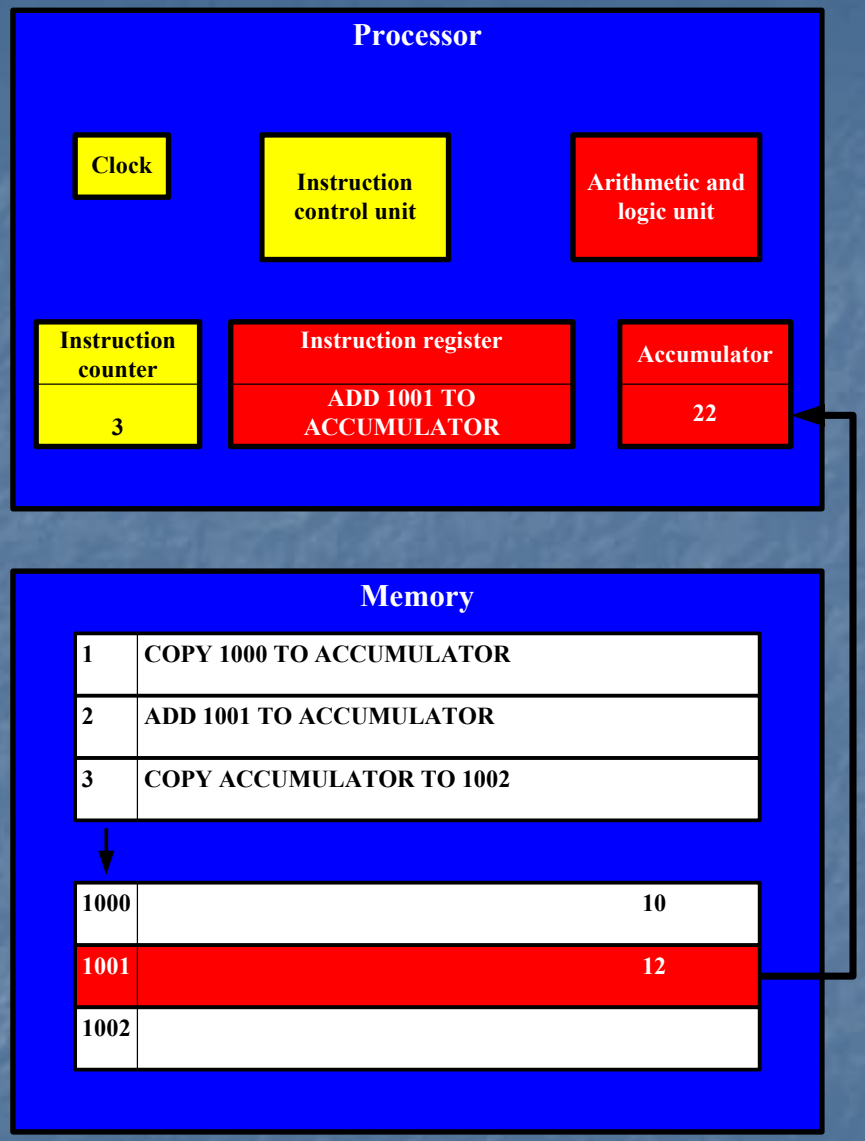
d. The instruction control unit once again looks to the instruction counter for the address of the next instruction.



e. The next instruction is fetched into the instruction register. Keep in mind that the instruction counter points to the next instruction.

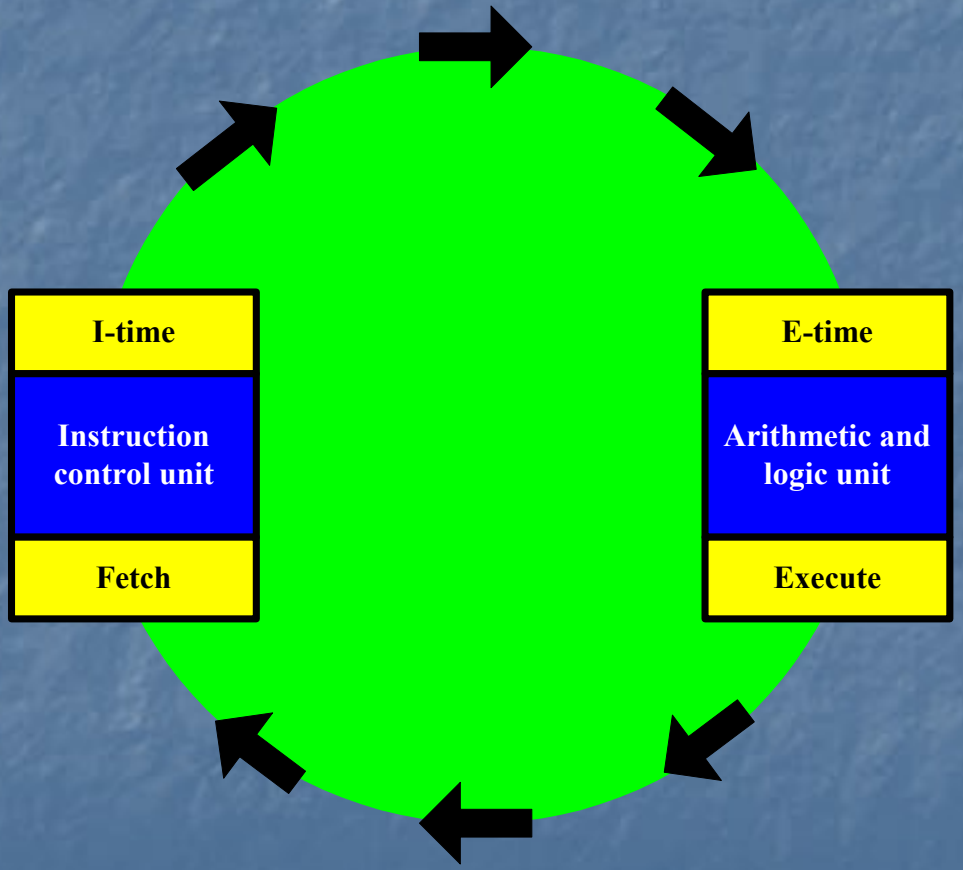


f. The arithmetic and logic unit executes the instruction in the instruction register.



During a machine cycle, an instruction is fetched during

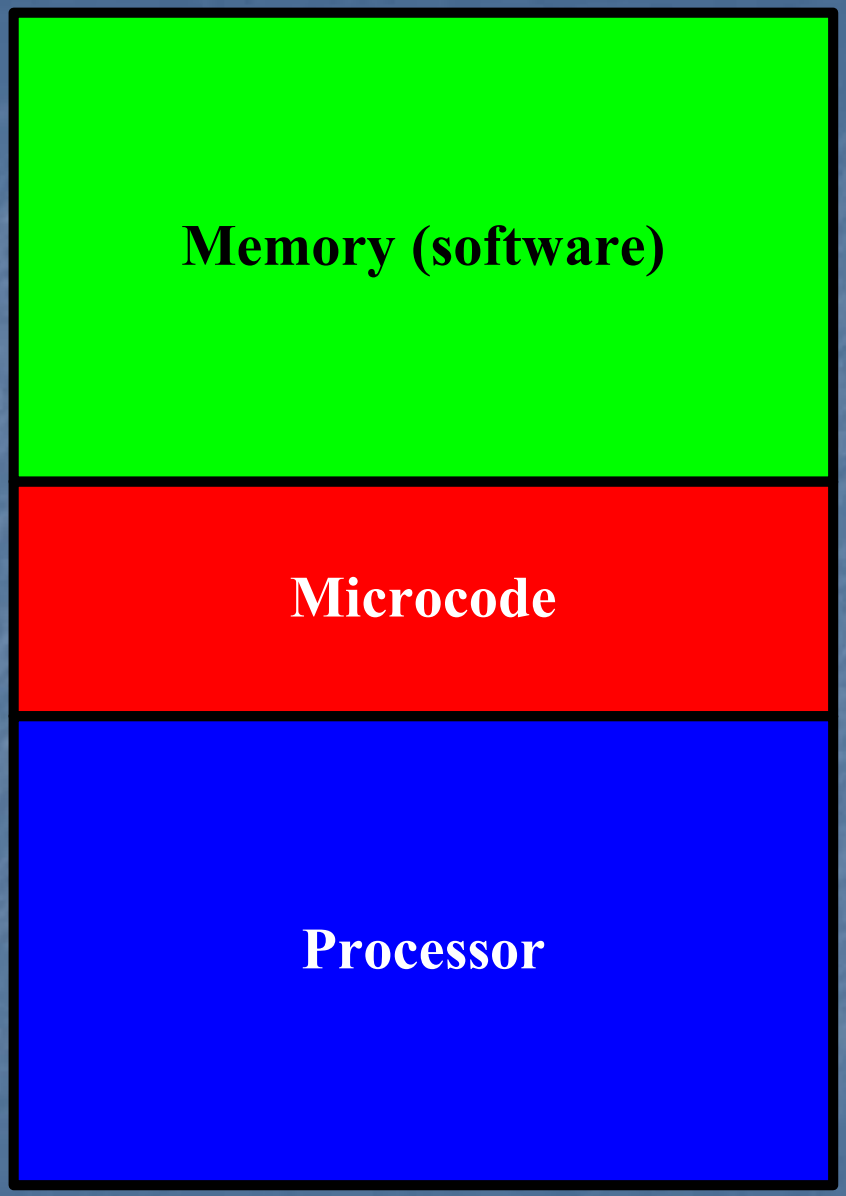
I-time and executed during E-time.



- I-time → fetch
- E-time → execute
- Clock speed
  - Cycles per second
  - Measured in MHz or GHz

A layer of microcode lies between the memory and the processor.

Microcode is also known as firmware.



Input and Output

- Input
  - The act of transferring data into memory from a peripheral device.
- Output
  - The act of transferring data from memory to a peripheral device.
- Input and Output
  - Support the interaction between the user and the computer.

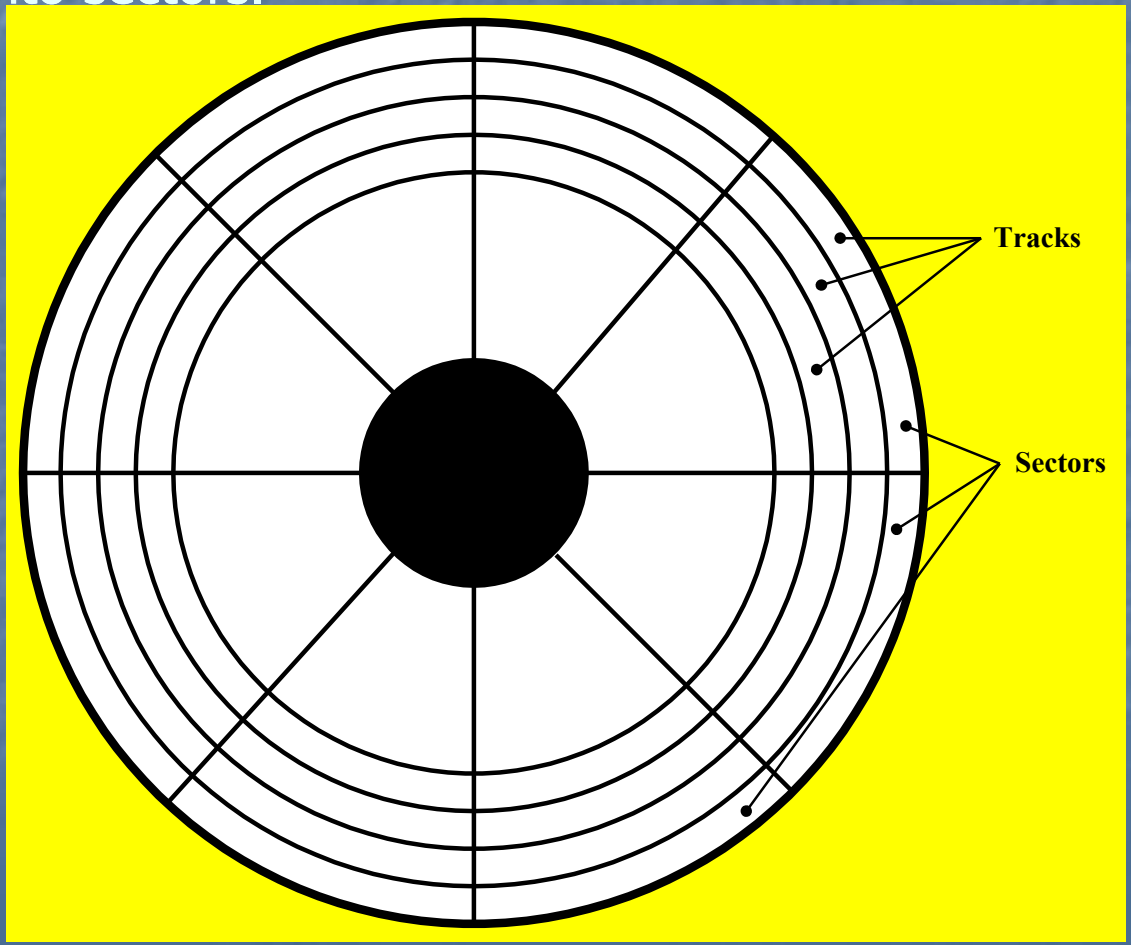
Secondary Storage

- Extension of main memory
  - Fast, accurate, and inexpensive
  - High-capacity and nonvolatile
- Long-term storage
  - Contents must be transferred into main memory.
- Machine-readable only
  - People cannot read contents directly.
- Media
  - Hard disk and floppy disk
  - Magnetic tape
  - CD-ROM and DVD

Tracks and Sectors

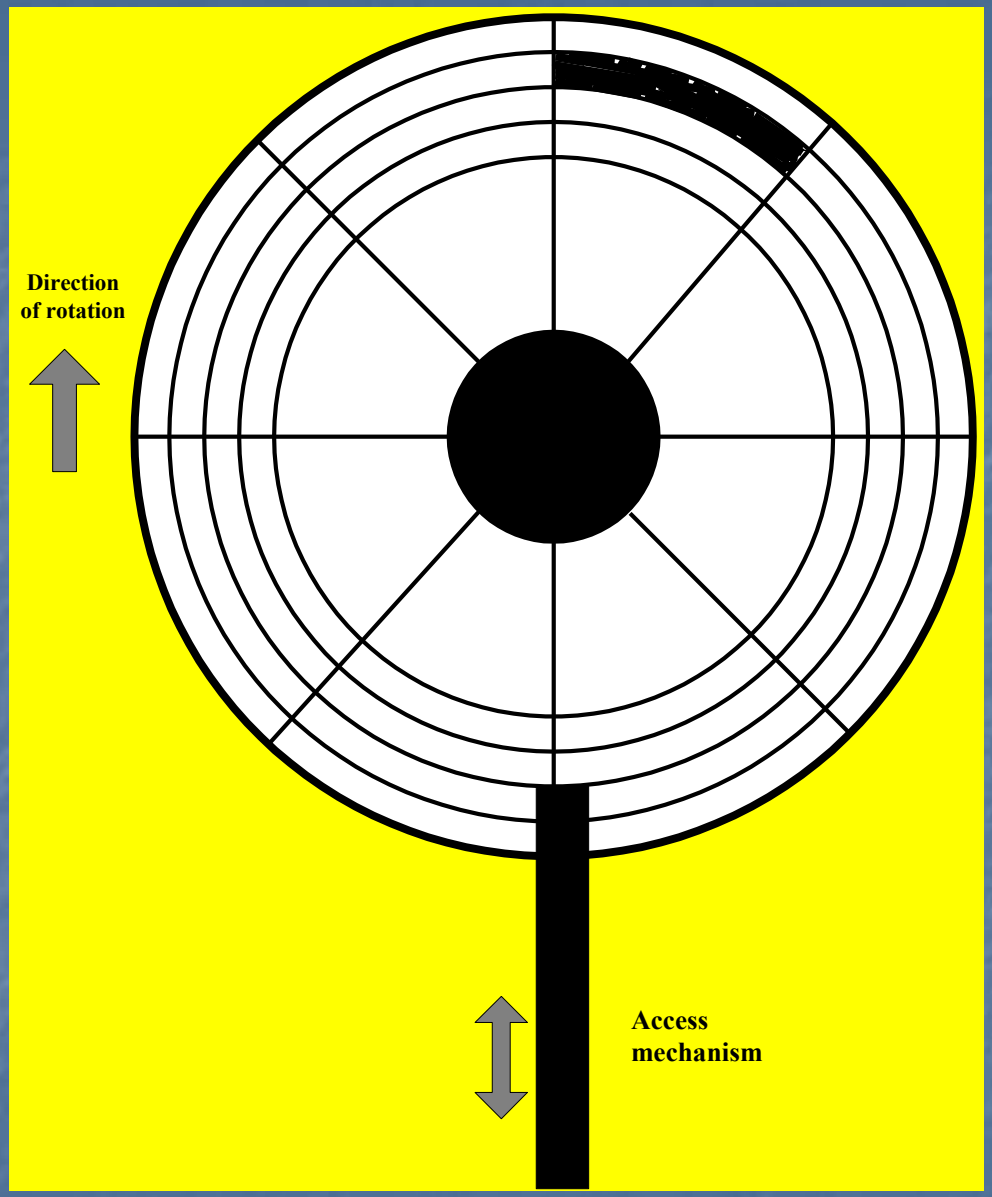
The data on a disk is recorded on a series of concentric circles called tracks.

Tracks are subdivided into sectors.

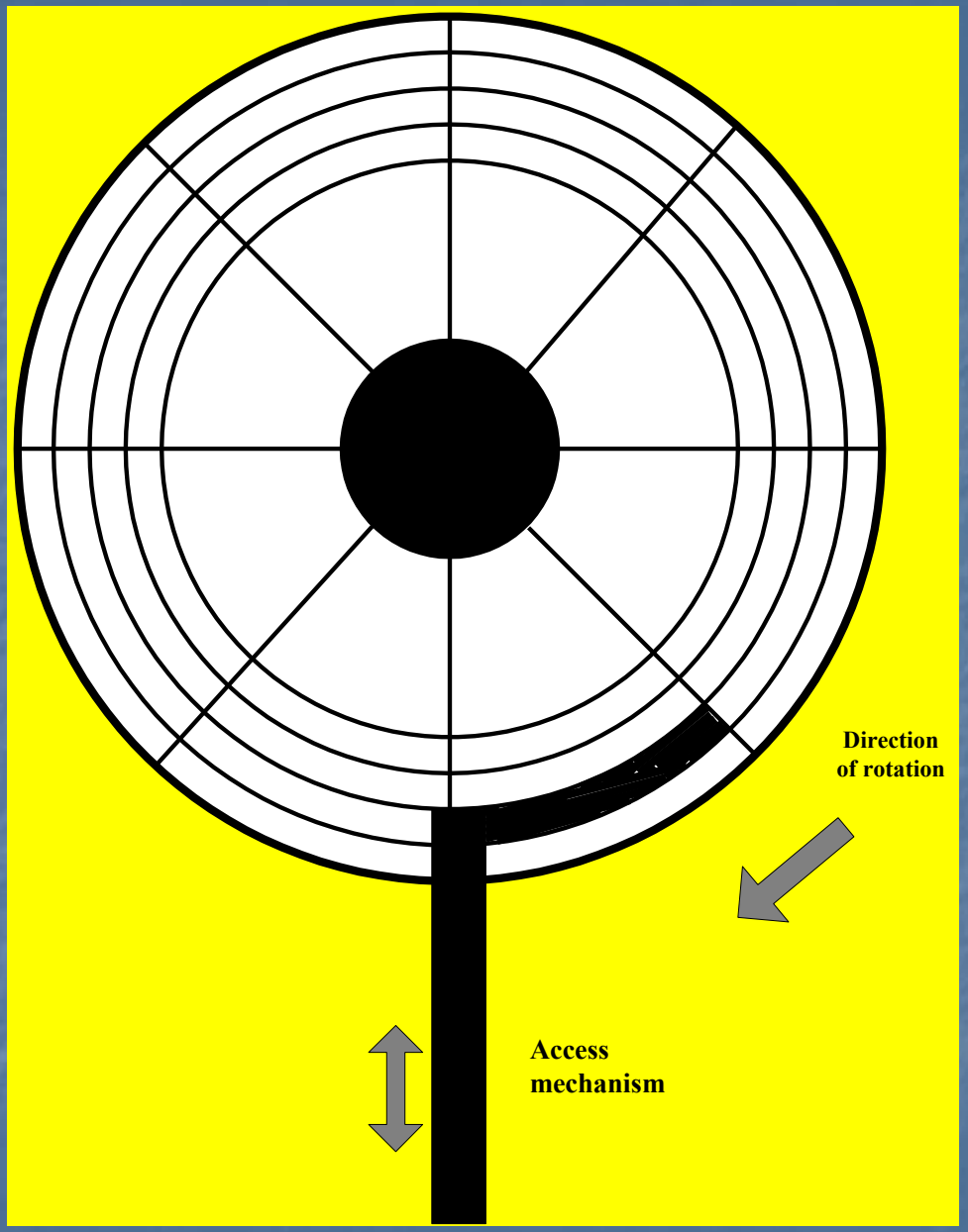


Reading a Sector from the Disk

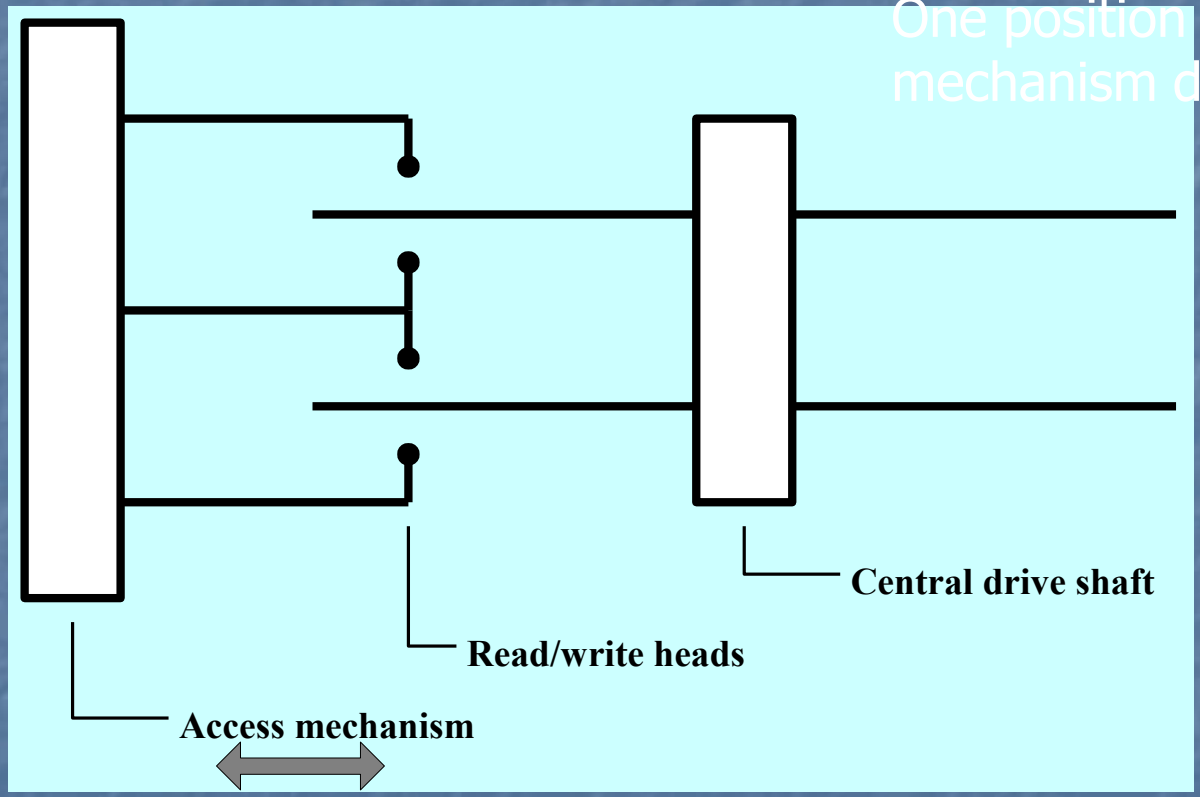
a. During seek time, the access mechanism is positioned over the track that holds the required data.



b. The system waits while the sector rotates to the read/write head, causing rotational delay, and the data is transferred into memory.

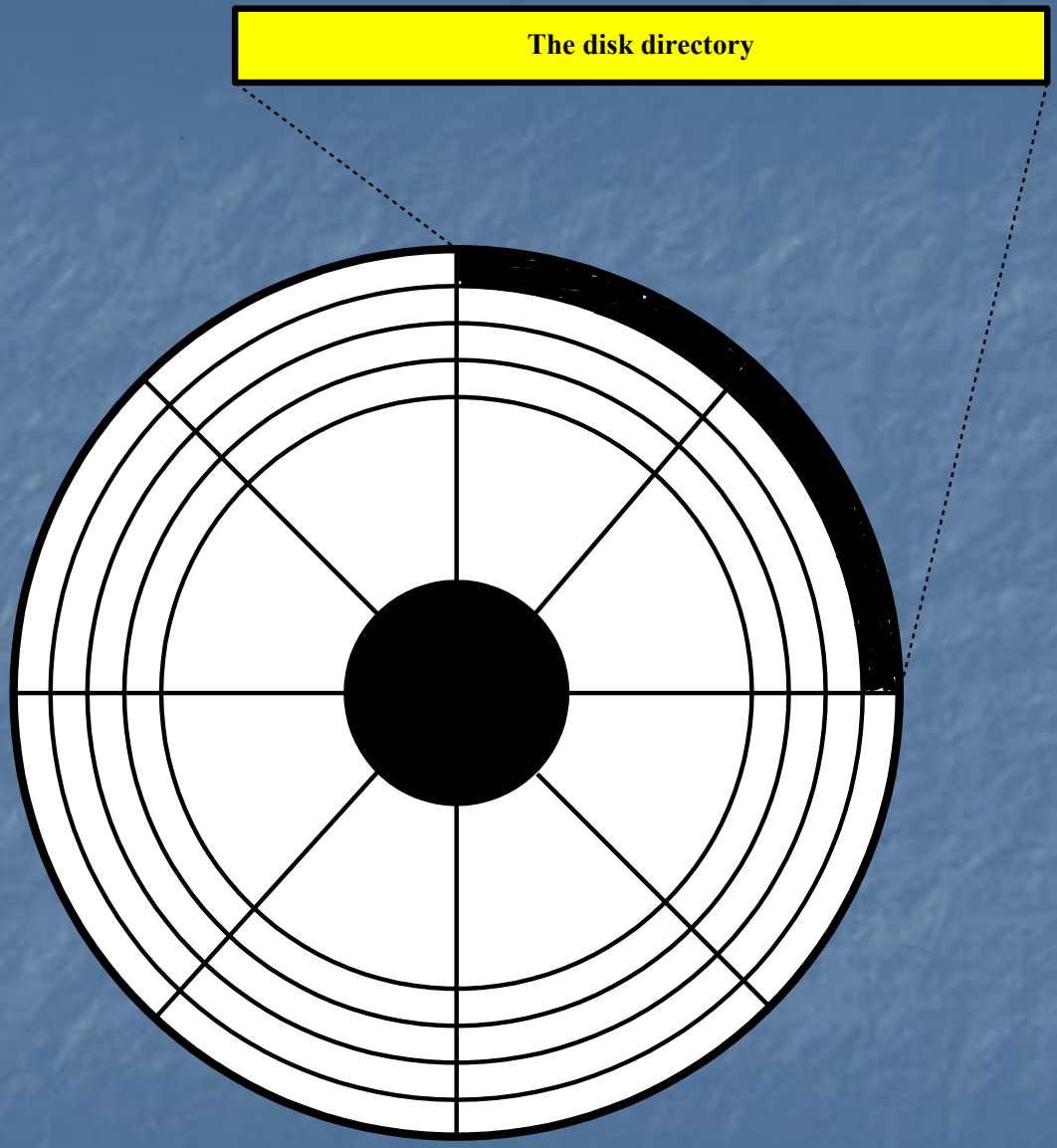


Each surface on a disk pack has its own read/write head.



One position of the access mechanism defines a cylinder.

The programs and data files stored on a disk are listed in the disk directory.



Communication Hardware

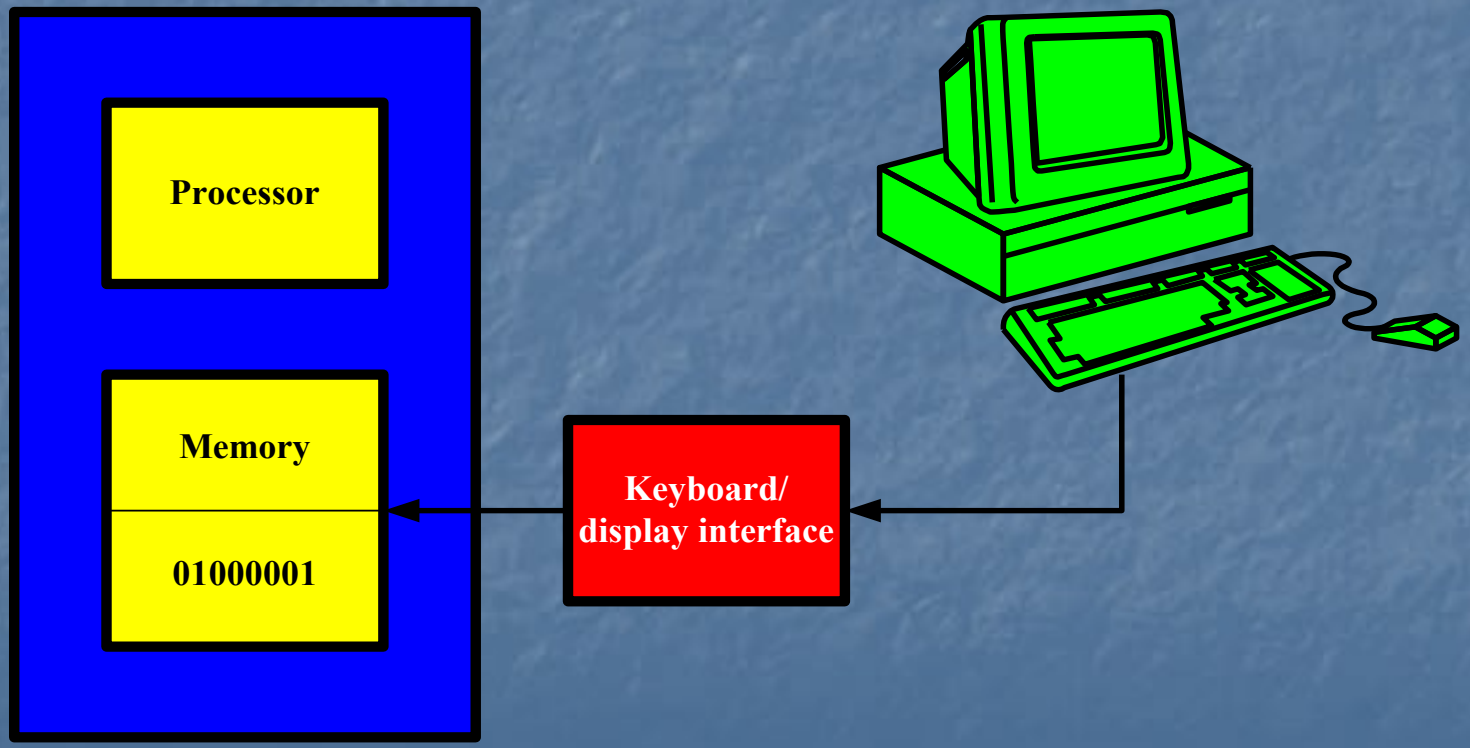
- Modem
  - A device used to communicate over standard telephone lines
- Cable modem
  - A device used to communicate over high-speed cable

Linking the Components

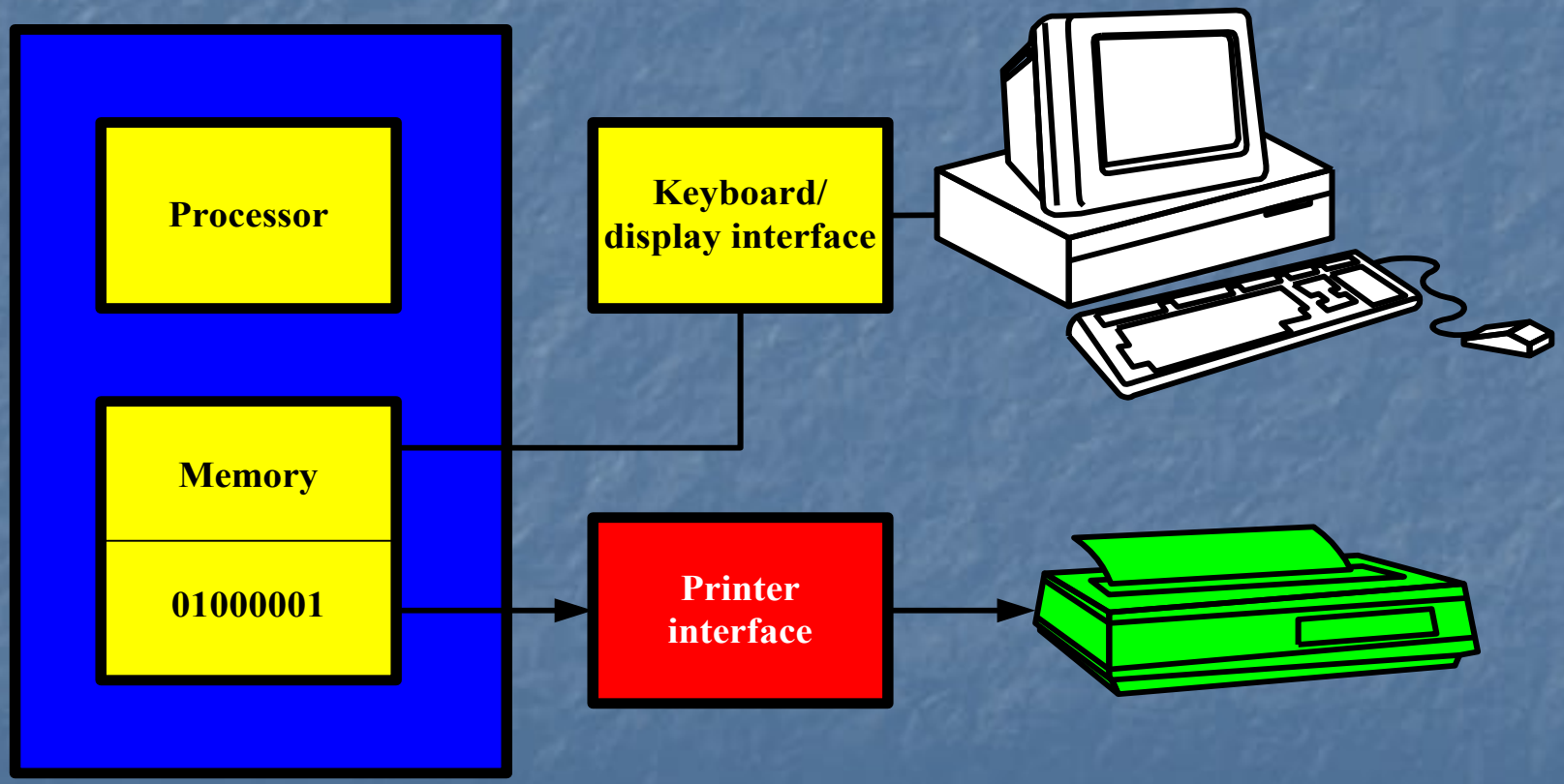
- Interface
  - Translates between internal and external formats
- Buffer
  - Temporary memory used to adjust for the speed differential between adjacent devices

Functions of an Interface Board

The keyboard or display interface converts input from the keyboard to an internal format.



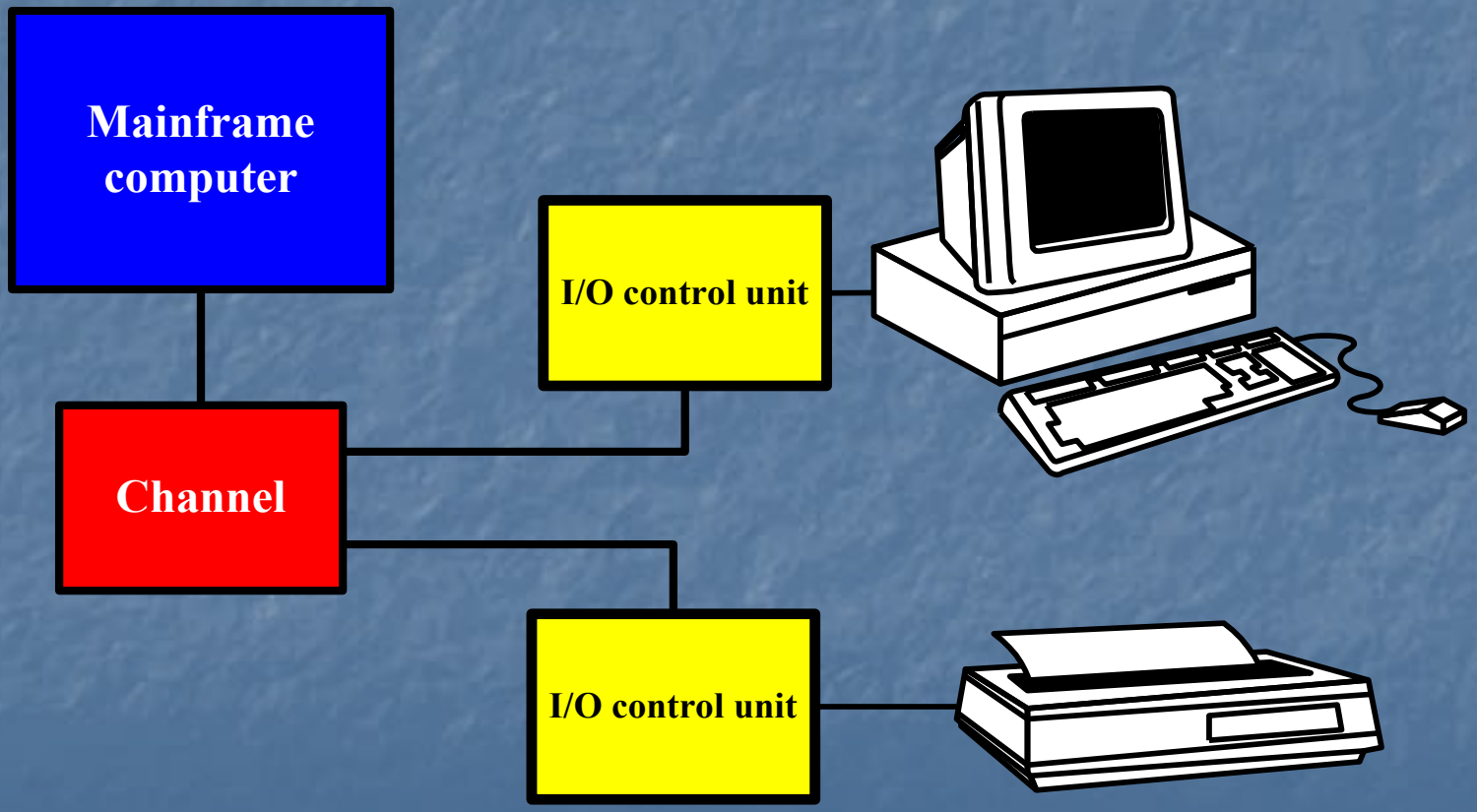
Data stored in memory is sent to the printer interface, converted to printable format, and printed.



Channels and Control Units

- Channel
  - Performs peripheral, device-independent functions on large computers
- I/O control unit
  - Performs device-dependent functions

On a mainframe, peripheral devices are linked to the system through a channel and an I/O control unit.



Number Systems

Introduction

- In this section, you are introduced to the basic number systems that are currently used by most operating systems.
- We will start with an introduction to both Binary and Hexadecimal number systems.
- You will then be shown how we use digital gates to interpret these binary values.
- The objective is to provide you with a foundation in the use of these number systems and digital gates that form the basis of all computer operations.

Numbering systems

- Decimal
  - Digit values – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Place values – 10^4, 10^3, 10^2, 10^1, 10^0
- Binary
  - Digit values – 0, 1
  - Place values – 2^4, 2^3, 2^2, 2^1, 2^0
- Value of any number is derived by multiplying each digit by its place value and adding these products.

Each octal digit is equivalent to three binary digits.

Octal	Binary	Octal	Binary
0	**000**	4	**100**
1	**001**	5	**101**
2	**010**	6	**110**
3	**011**	7	**111**

Each hexadecimal digit is equivalent to four binary digits.

Hex	Binary	Hex	Binary
0	**0000**	8	**1000**
1	**0001**	9	**1001**
2	**0010**	A	**1010**
3	**0011**	B	**1011**
4	**0100**	C	**1100**
5	**0101**	D	**1101**
6	**0110**	E	**1110**
7	**0111**	F	**1111**

What does the number 0 represent?

- Off, No, or False
- The open state of a switch, indicating no current flow through the transistor
- The value zero

What does the number 1 represent?

- On, Yes, or True

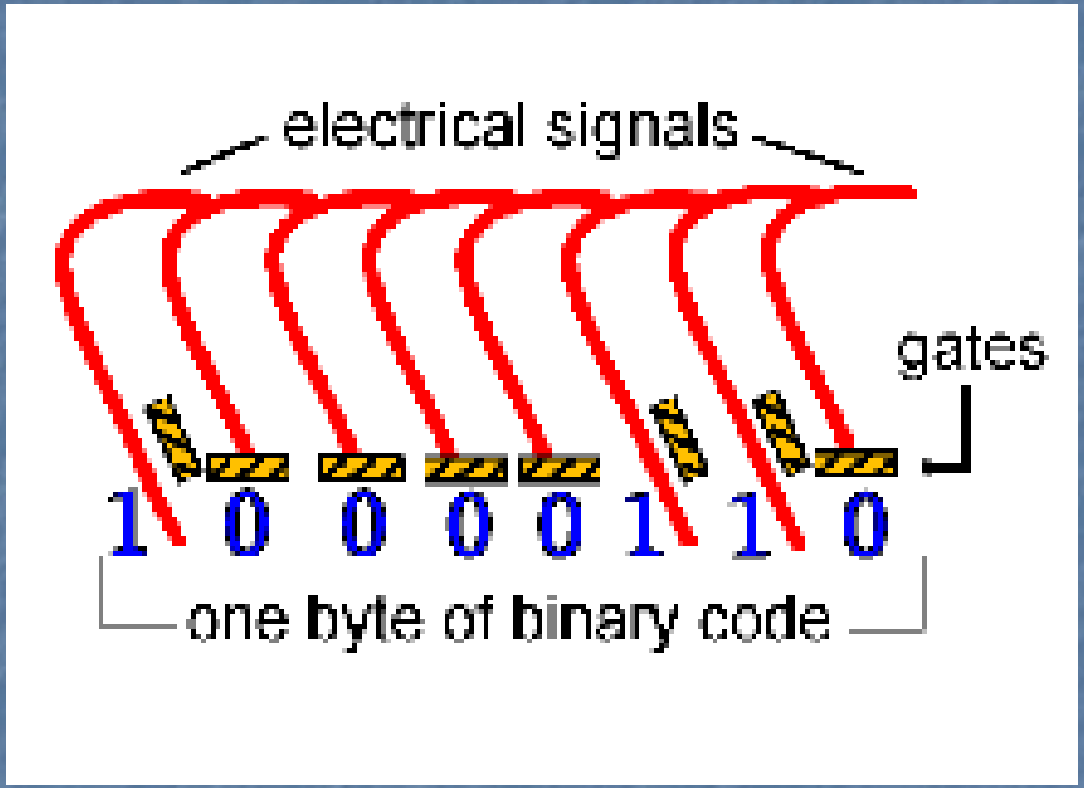
- The closed state of a switch, indicating current flow through the transistor

Binary Number System

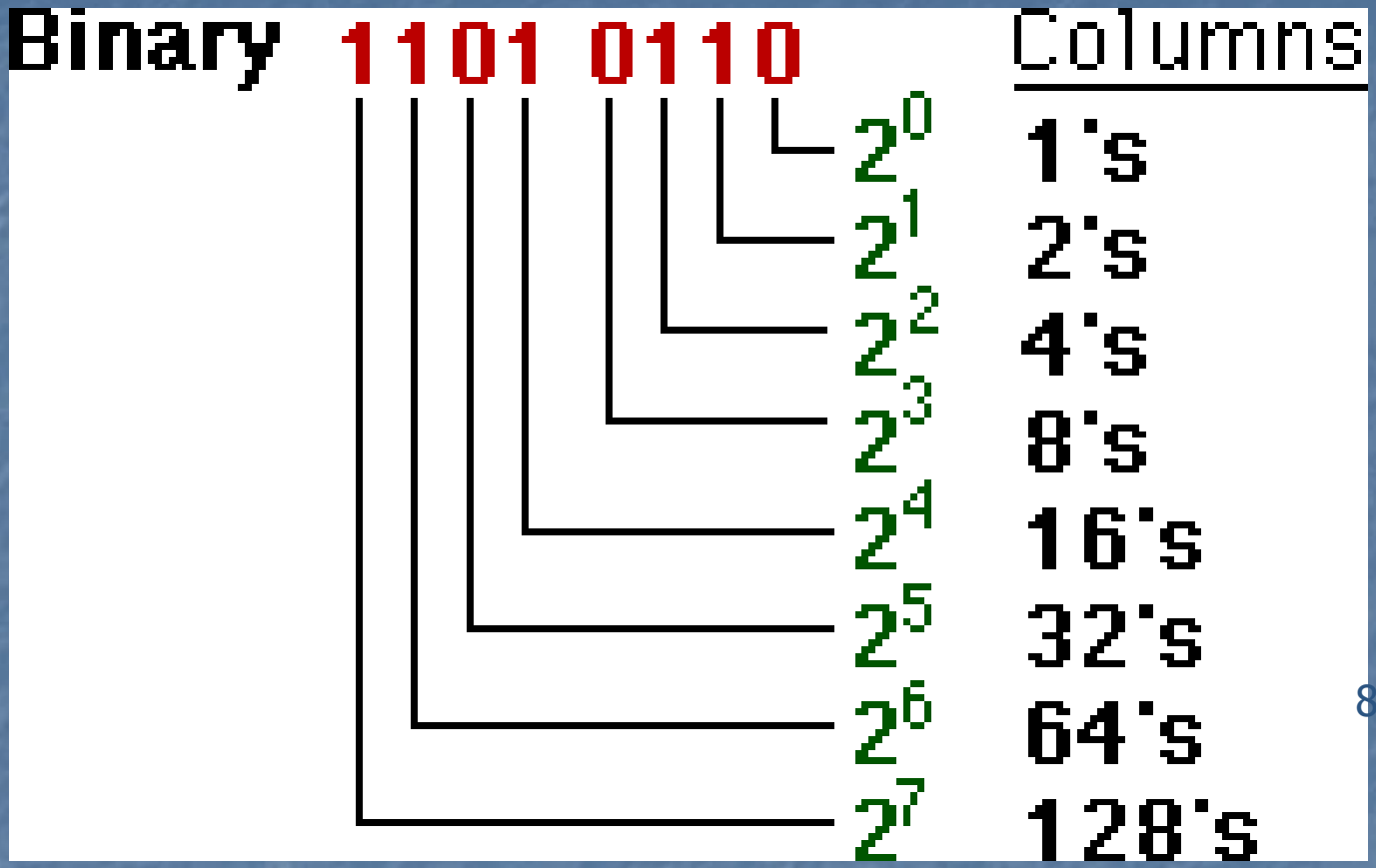
In terms of the hardware, think of the memory in your computer as a set of thousands of switches called bits that can be turned on or off.

The value 1 represents on, and the value 0 represents off.

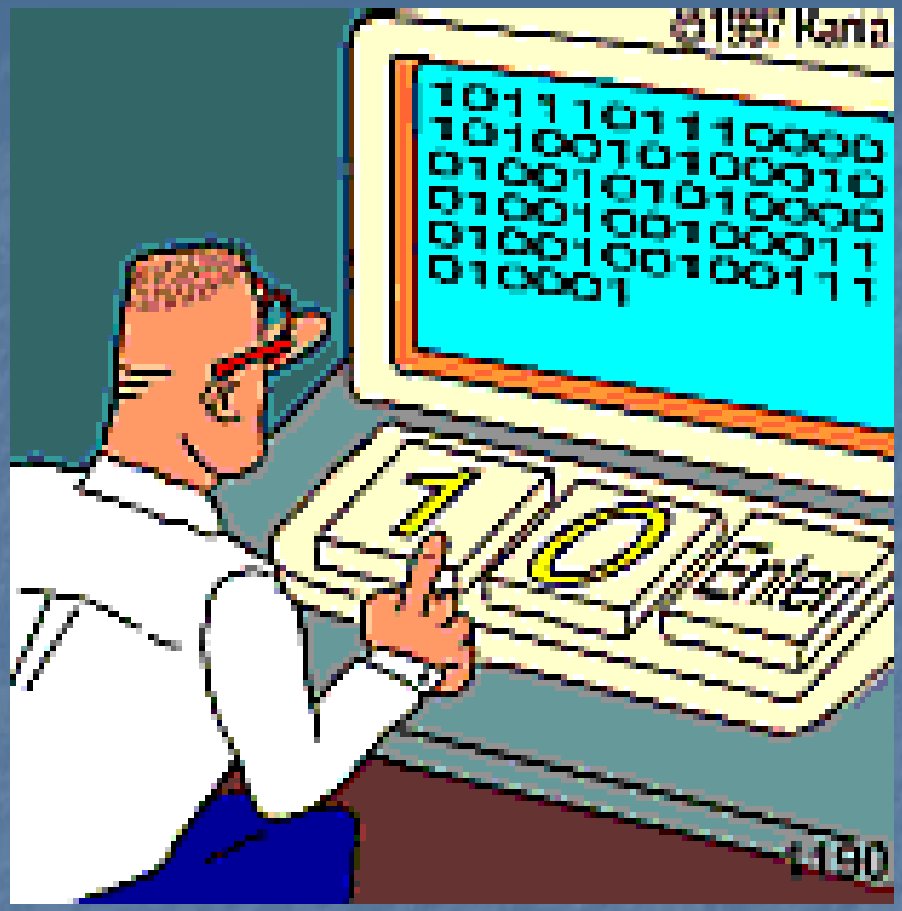
Eight bits are grouped together to form a byte; every number in binary is made up of 8 bits.



An example to represent the number 5 in binary format:



There are 8 bits in a binary number, so picture 00000000. Now, 5 expressed in base 2 is 2^2 plus 2^0.



Say you want to write 10 in binary. Break it down into base 2. To get 10, you need 2^3 plus 2^1. 2^1 is 00000010 and 2^3 is 00001000. Add them together and you get 00001010.

- Each letter is assigned an ASCII value, which is a set of binary numbers. The computer interprets the binary numbers to obtain the letter.

Bits

- A bit is a number in the set of binary numbers.
- A bit can be either 0 or 1.
- The bigger the number, the more bits the number contains.

Byte

- A byte is 8 bits or 2 nibbles or 4 bits.
- Bytes are used to represent a character.
- Bytes are the smallest data item in the microprocessor.
- The format of a byte is:

b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0

Where,

- b_7 is the high order bit.
- b_0 is the low order bit.

Words

The size of a word depends on the processing capability of the central processing unit (CPU), but a word usually contains 16 bits or 32 bits.

Example: A 16-bit processor can process only those words that are up to 16 bits long.

Binary Number System

Name	*Size (bits)*	*Example*
Bit	1	1
Nibble	4	0101
Byte	8	0000 0101
Word	16	0000 0000 0000 0101

Other:

- A kilobyte is equal to 1024 bytes (2^{10}).
- A megabyte is equal to about a million bytes.

Why Hexadecimal?

- The hexadecimal system is appropriate for matching 4 bits.
- There are 16 hexadecimal values, which means 16 4-bit possibilities.
- Four bits can be represented by using one hexadecimal value.
- Eight bits can be represented by using two hexadecimal values.
- Remember that color is represented using 24 bits or 3 bytes.

Dec.	Hex.	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

Dec.	Hex.	Binary
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Binary Equivalents of Hexadecimal Numbers (Contd...)

Dec.	Hex.	Binary	Dec.	Hex.	Binary
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

Hex	Binary	Hex	Binary	Hex	Binary
12	0001 0010	3C		99	
AB		1A		00	
02		B4		7D	
	0111 0111		1000 1111		1111 1111
	0000 0010		1100 1001		0101 1100

Binary Equivalents of Hexadecimal Numbers

Dec.	Hex.	Binary	Dec.	Hex.	Binary
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

Hex	Binary	Hex	Binary	Hex	Binary
12	0001 0010	3C	0011 1100	99	1001 1001
AB	1010 1011	1A	0001 1010	00	0000 0000
02	0000 0010	B4	1011 0100	7D	0111 1101
77	0111 0111	8F	1000 1111	FF	1111 1111
02	0000 0010	C9	1100 1001	5C	0101 1100

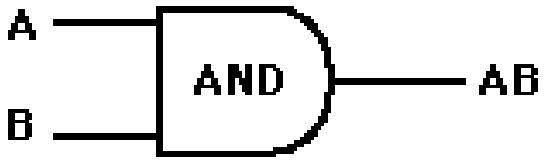
Hexadecimal Code for Colors

Dec.	Hex.	Binary	Dec.	Hex.	Binary
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

- We said the hexadecimal RGB code for Carrot Orange is #FF8E2A or in decimal (255, 142, 42)
- Remember that this is using 24 bits for color or 3 bytes.
- Carrot Orange is
- In binary: **1111 1111 , 1000 1110 , 0010 1010**
- In hexadecimal: **F F , 8 E , 2 A**
- In decimal: **255 , 142 , 42**

The AND Gate

- The output is high only when both input A and B are high.

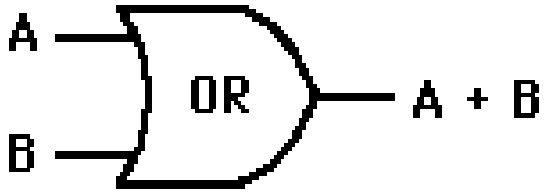


A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

The AND operation will be signified by AB or A·B. Other common mathematical notations for it are $A \wedge B$ and $A \cap B$, called the intersection of A and B.

The OR Gate

- The output is high when either or both input A and input B are high.

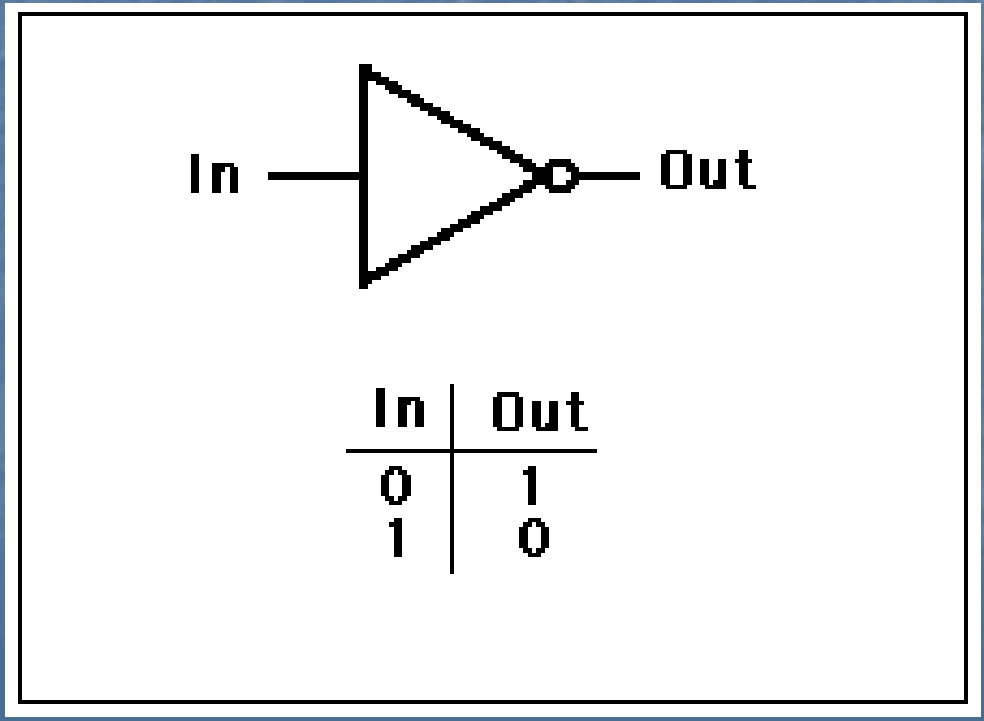


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

The OR operation will be signified by A + B. Other common mathematical notations for it are $A \vee B$ and $A \cup B$, called the union of A and B.

Inverter

- A single-input device, which produces an output state opposite to the input state.
- High input produces low output, and vice versa.
- Commonly referred to as a NOT gate.



In	Out
0	1
1	0

Adding Binary Numbers

A key requirement of digital computers is the ability to use logical functions to perform arithmetic operations. The basis of this is addition; if we can add two binary numbers, we can just as easily subtract them, or get a little fancier and perform multiplication and division. How, then, do we add two binary numbers? Let's start by adding two binary bits. Each bit has only two possible values, 0 or 1; therefore, there are only four possible combinations of input values.

These four possibilities and the resulting sums are:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Binary addition produces a sum and a carry. The truth table for binary addition is:

INPUTS		OUTPUT	
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The Carry is a simple AND function.
The Sum is an Exclusive-OR function.
You can use two gates to add two bits.

Negative Numbers and Binary Subtraction

We have seen how simple logic gates can be used to perform the process of binary addition. It is only logical to assume that a similar circuit can perform binary subtraction.
If we look at the possibilities involved in subtracting one 1-bit number from another, we can quickly see that three of the four possible combinations are easy and clear. The fourth combination involves a bit more calculation:

0 - 0 = 0 1 - 0 = 1 1 - 1 = 0 0 - 1 = 1, with a borrow bit.

The borrow bit is just like a borrow in decimal subtraction. It subtracts from the next higher order of magnitude in the overall number. Let's see what the truth table looks like.

INPUTS		OUTPUTS	
A	B	BORROW	A - B
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

The difference, A-B, is an Exclusive-OR function.

The borrow is an AND function, but is A'B instead of AB.

To simplify binary subtraction, convert B to its negative equivalent and then use the basic adder.

Summary

In this unit, you discussed:

- Basics of operating systems
- Interfaces for operating systems
- Layers of abstraction
- Differences between open source and proprietary operating systems
- Bits, bytes, and words
- Machine cycle
- Key components of a processor
- Common input and output devices
- Binary, Decimal, and Hexadecimal numbering systems
- Digital Gates