Chapter 1 Microcomputer Systems

PREPARED BY

AHMED AL MAROUF

LECTURER, DEPT. OF CSE

DAFFODIL INTERNATIONAL UNIVERSITY

Outline

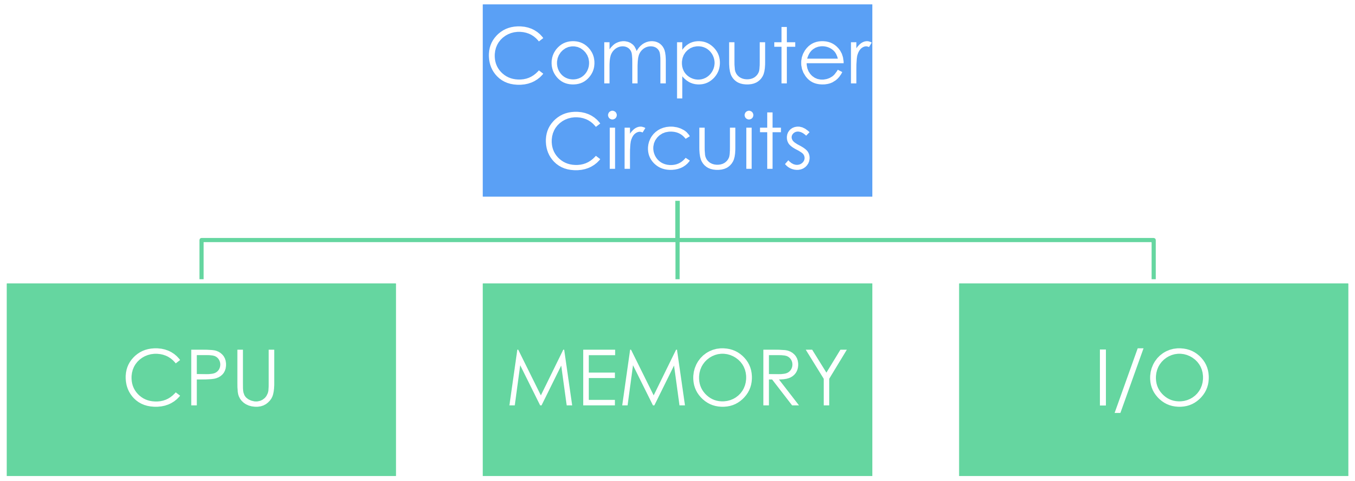
- The Components of Microcomputer Systems
- CPU and System Board
  - Memory
  - Bit Position
  - Memory Operations
  - RAM & ROM
  - Bus
  - I/O Ports
- Instruction Execution
- I/O Devices
- Programming Languages
  - Machine Language
  - Assembly Language
  - High-Level Language

The Components of Microcomputer Systems

- ***Computer system examples:*** ATM Machine, Mobile Phones, Desktop, Laptop computers.
- ***Integrated Circuits (IC)*** are the minimalist and important component of any computer systems.
- Each IC may contains hundreds or even thousands of transistors. IC circuits are known as ***digital circuits***.
- These digital circuits operate on ***discrete voltage signal levels***, typically, a ***high voltage (1)*** and a ***low voltage (0)***.
- These voltage level symbols are called ***binary digits*** or ***bits***.

CPU

- A ***central processing unit (CPU)*** is the electronic circuitry within a computer that executes instructions that make up a computer program.
- The CPU performs basic arithmetic, logic, controlling, and input/output (I/O) operations specified by the instructions in the program.

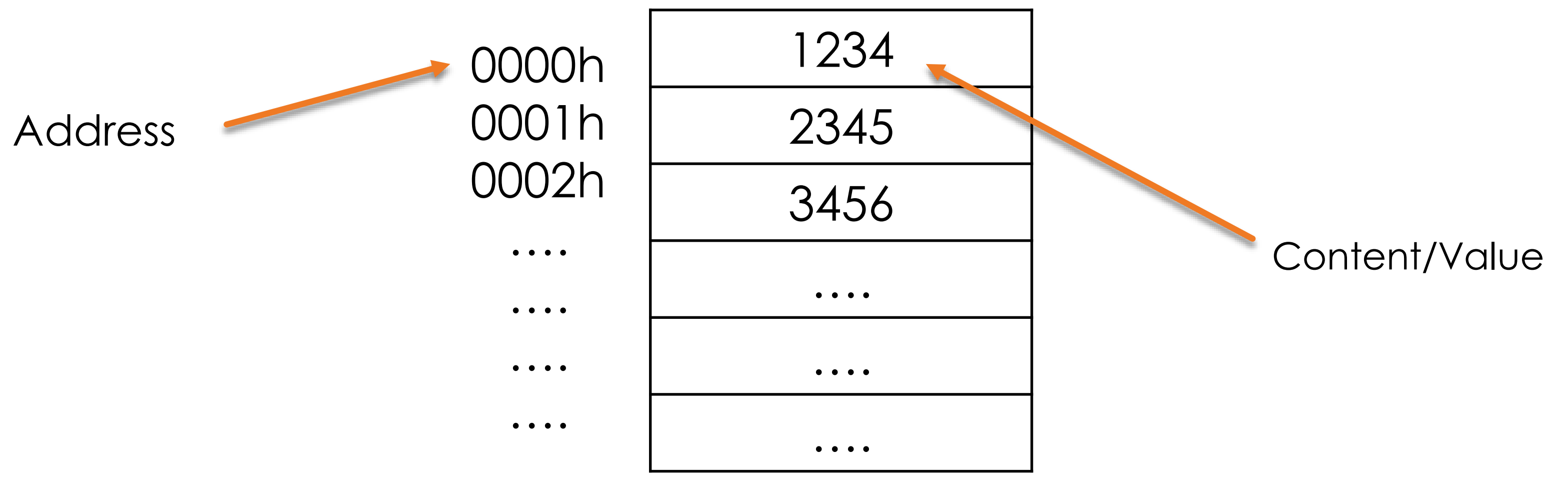


System Board

- Inside the system unit, there is a main board, which is called ***system board***.
- System board is commonly known as ***Motherboard***.
- It is called motherboard, as it contains the expansion slots, which are the connectors for additional circuit boards called add-in boards or add-in cards.
- Example of add-in cards: Modem, TV card, Sound Card, Graphics Card etc.
- I/O circuits are usually located on the add-in cards.

Memory

- Information processed by the computer is stored in its ***memory***.
- Each memory byte is identified by a number that is called its ***address***.
- The first memory byte has address zero (0).



Bit, Byte, Word and many more

- Bit is the smallest unit of memory, which is of size 1.
- 1 Nibble = 4 bits
- 1 Byte = 2 Nibbles = 8 bits
- 1 Word = 2 Bytes = 16 bits
- 1 Double-word = 2 Words = 4 Bytes = 32 bits
- 1 Quad word = 2 Double-word = 4 Words = 8 Bytes = 64 bits
- 1 KB (Kilo Bytes) = 1000 Bytes = 8000 bits
- 1 MB = 1024 KB = 8,19,20,000 bits

So many MB you have spend while enjoying YouTube?

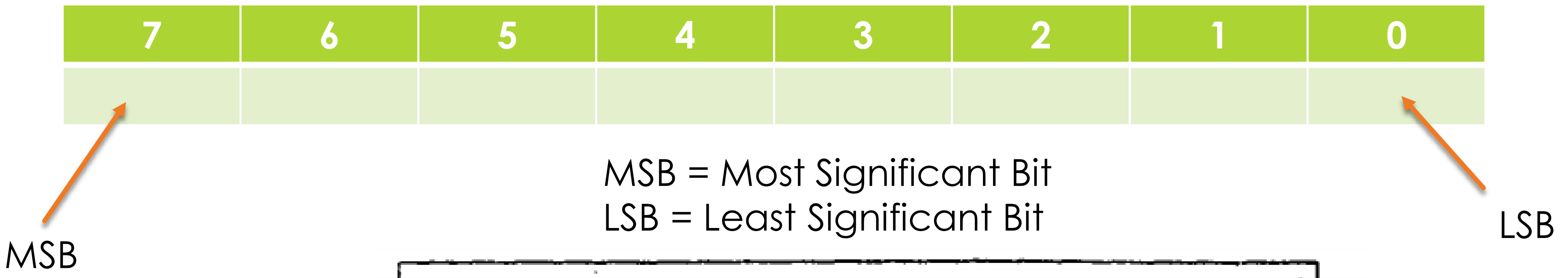
Example 1.1

- Suppose a processor uses 20 bits for an address. How many memory bytes can be accessed?

Solution:

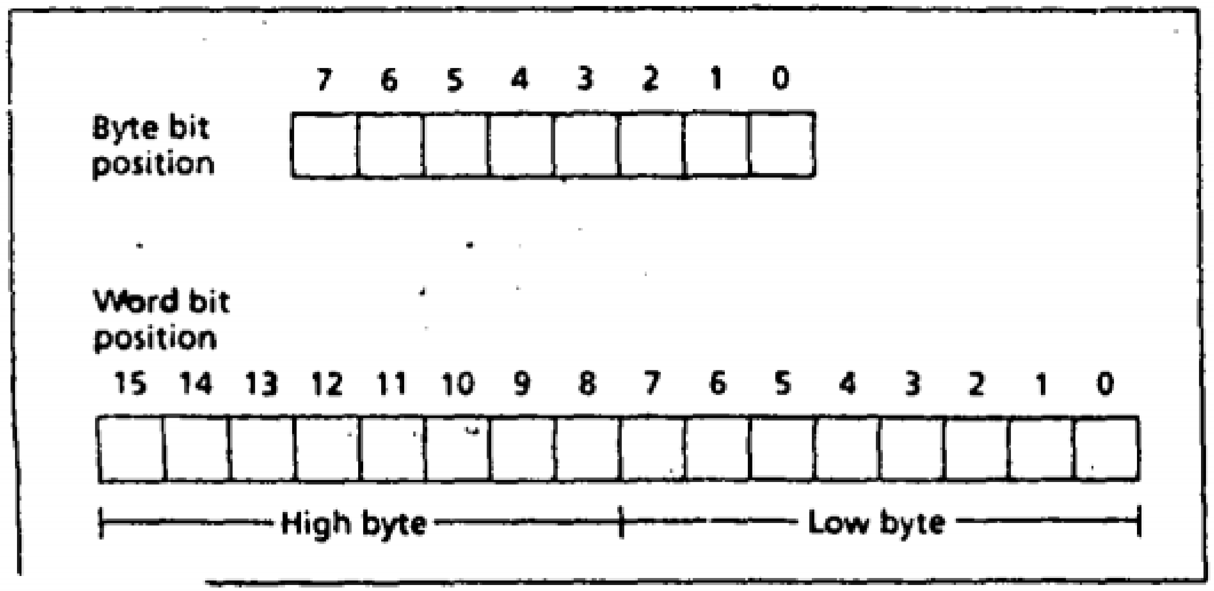
A bit can have two possible values (0 or 1), so in a **20-bit address there can be 2^{20} = 1,048,576 different values**, with each value being the potential address of a memory byte. In computer terminology, the number 2^{20} is called ***1 mega***. Thus, a 20-bit address can be used to address ***1 megabyte or 1 MB***.

Bit Position



7	6	5	4	3	2	1	0

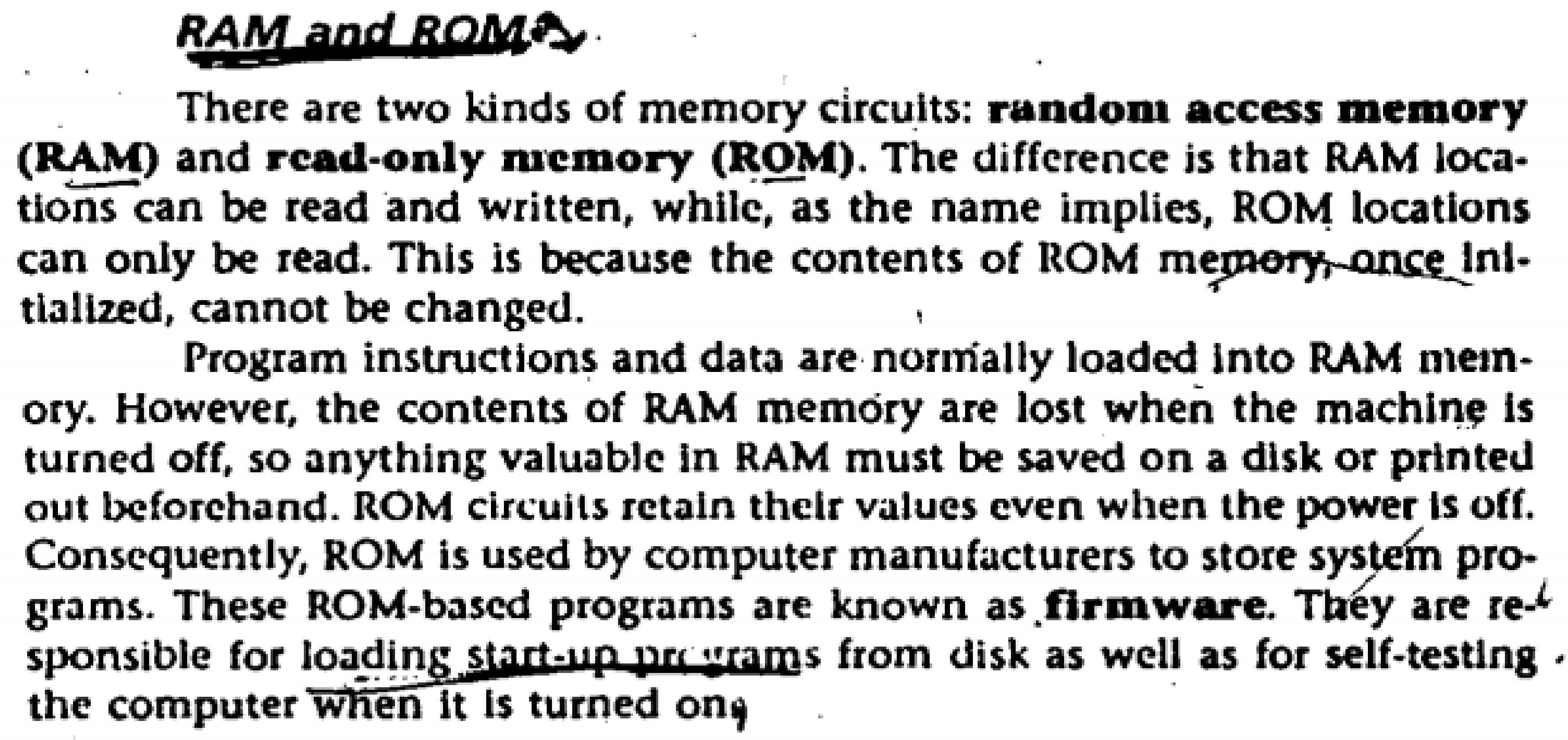
MSB = Most Significant Bit
LSB = Least Significant Bit



Memory Operations

- The processor can perform two operations on memory.
- ***Read (fetch)*** the contents of a location and
- ***Write (store)*** data at a location.
- In a read operation, the processor only gets a copy of the data; the original contents of the location are unchanged.
- I a write operation, the data written become the new contents of the location; the original contents are thus lost.

RAM vs. ROM

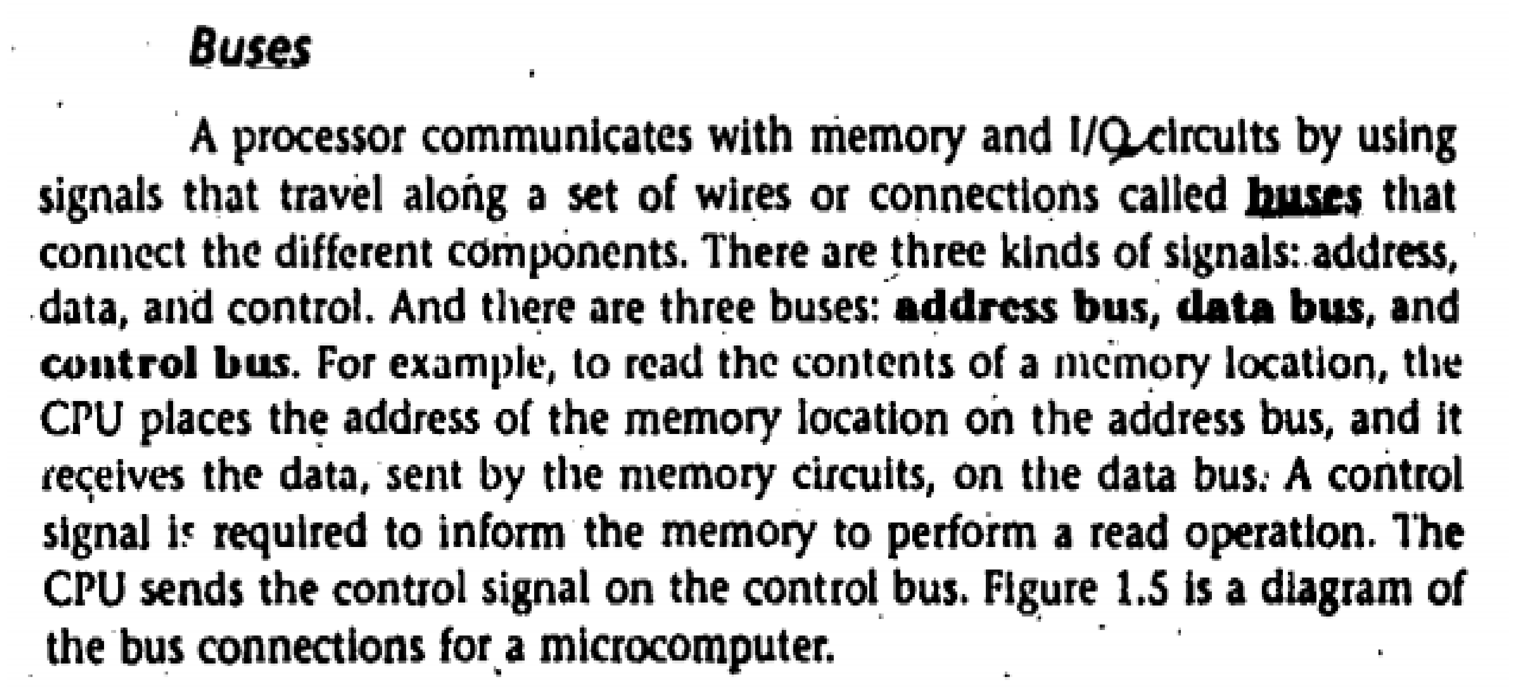


RAM and ROM

There are two kinds of memory circuits: **random access memory (RAM)** and **read-only memory (ROM)**. The difference is that RAM locations can be read and written, while, as the name implies, ROM locations can only be read. This is because the contents of ROM memory, once initialized, cannot be changed.

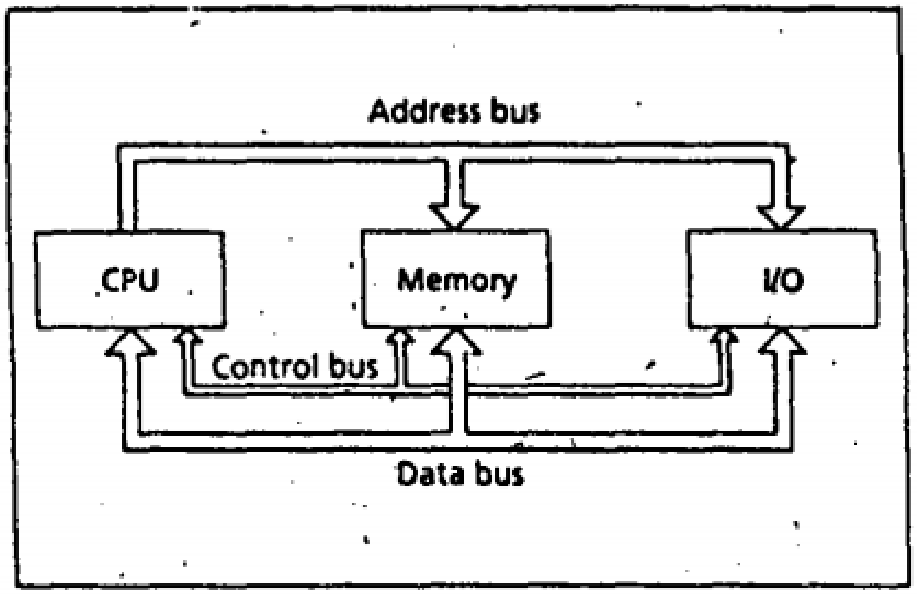
Program instructions and data are normally loaded into RAM memory. However, the contents of RAM memory are lost when the machine is turned off, so anything valuable in RAM must be saved on a disk or printed out beforehand. ROM circuits retain their values even when the power is off. Consequently, ROM is used by computer manufacturers to store system programs. These ROM-based programs are known as **firmware**. They are responsible for loading start-up programs from disk as well as for self-testing the computer when it is turned on.

BUS

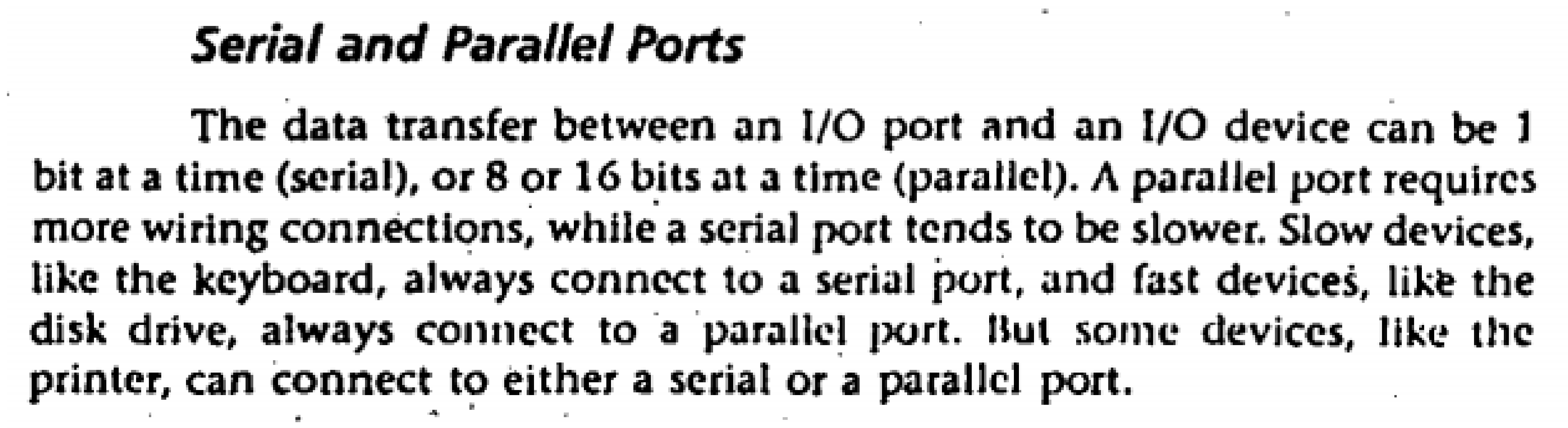


Buses

A processor communicates with memory and I/O circuits by using signals that travel along a set of wires or connections called **buses** that connect the different components. There are three kinds of signals: address, data, and control. And there are three buses: **address bus, data bus,** and **control bus**. For example, to read the contents of a memory location, the CPU places the address of the memory location on the address bus, and it receives the data, sent by the memory circuits, on the data bus. A control signal is required to inform the memory to perform a read operation. The CPU sends the control signal on the control bus. Figure 1.5 is a diagram of the bus connections for a microcomputer.



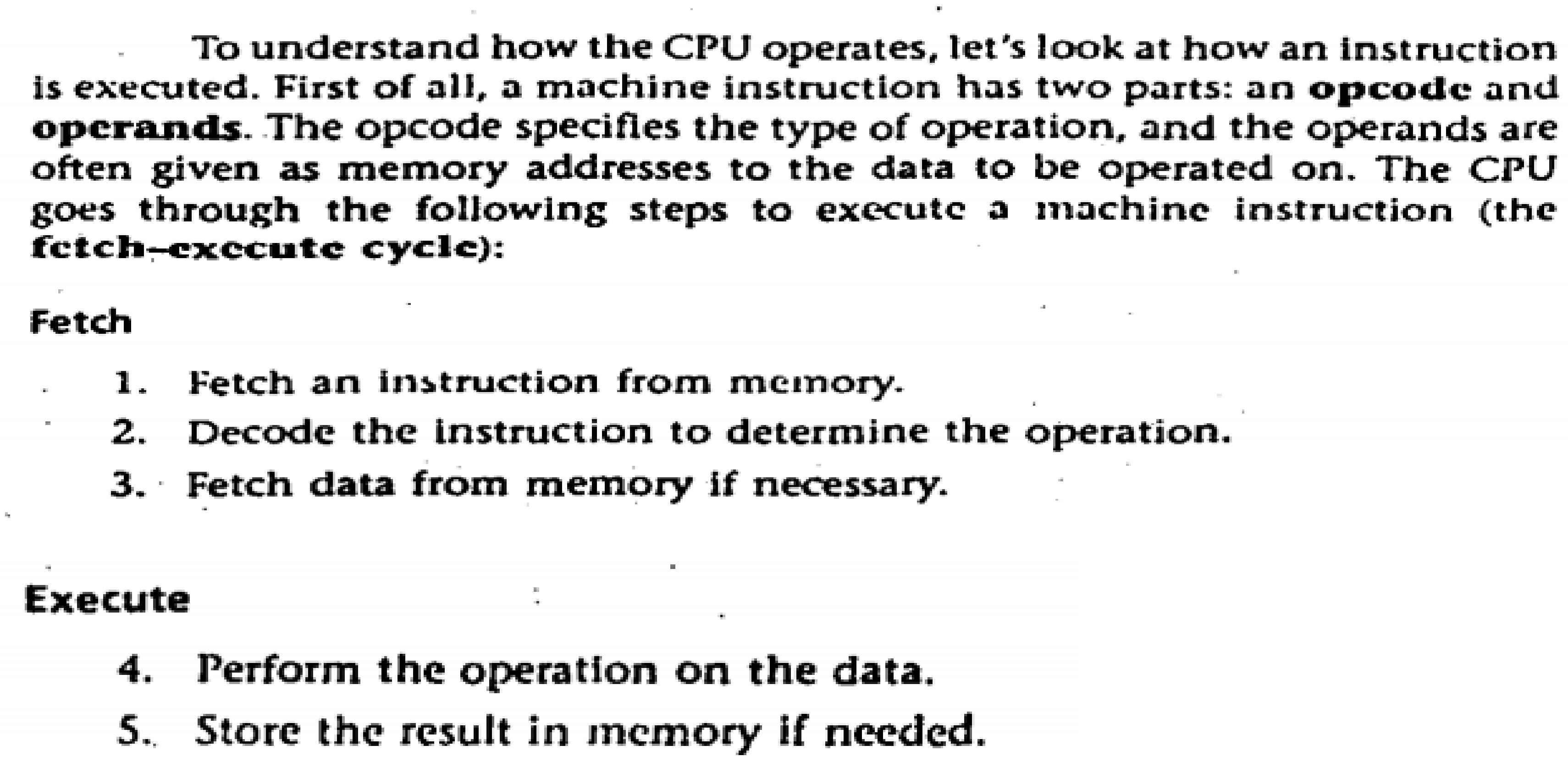
I/O ports



Serial and Parallel Ports

The data transfer between an I/O port and an I/O device can be 1 bit at a time (serial), or 8 or 16 bits at a time (parallel). A parallel port requires more wiring connections, while a serial port tends to be slower. Slow devices, like the keyboard, always connect to a serial port, and fast devices, like the disk drive, always connect to a parallel port. But some devices, like the printer, can connect to either a serial or a parallel port.

Instruction Execution



To understand how the CPU operates, let's look at how an instruction is executed. First of all, a machine instruction has two parts: an **opcode** and **operands**. The opcode specifies the type of operation, and the operands are often given as memory addresses to the data to be operated on. The CPU goes through the following steps to execute a machine instruction (the **fetch–execute cycle**):

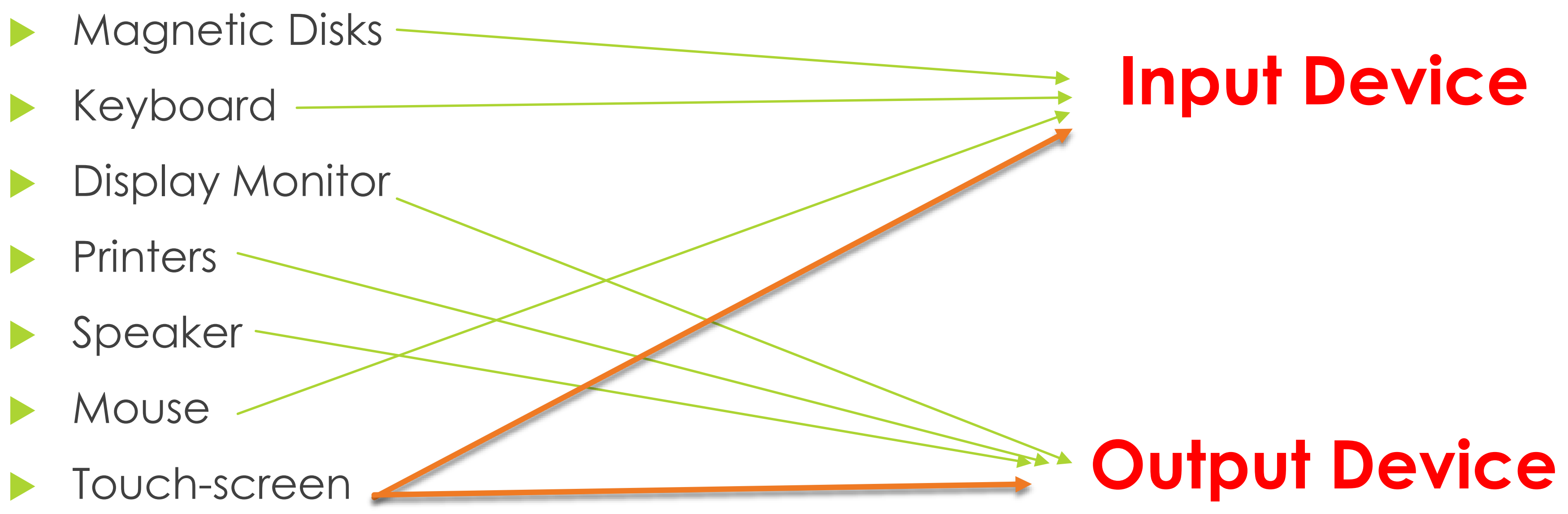
Fetch

1. Fetch an instruction from memory.
2. Decode the instruction to determine the operation.
3. Fetch data from memory if necessary.

Execute

4. Perform the operation on the data.
5. Store the result in memory if needed.

I/O Devices

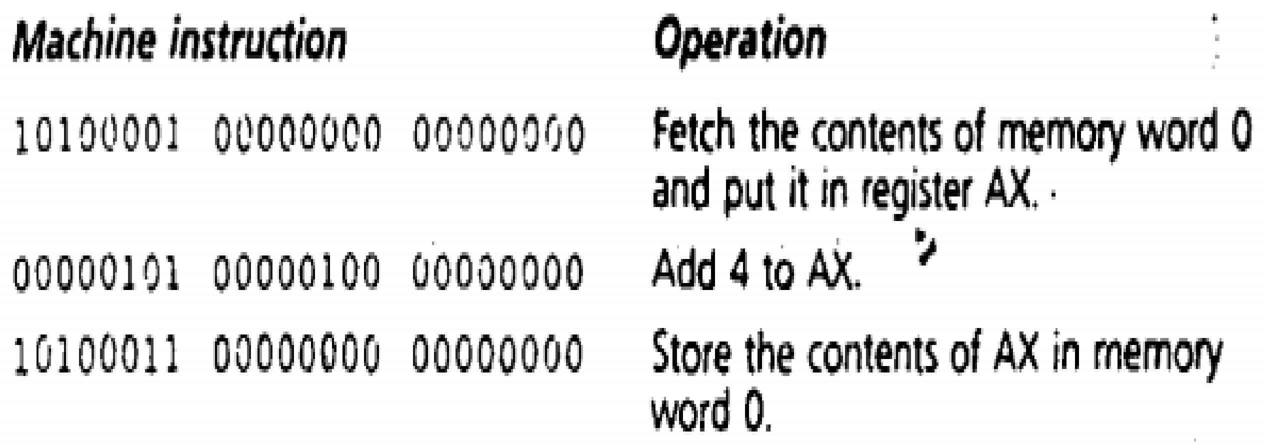

Magnetic Disks
Keyboard
Display Monitor
Printers
Speaker
Mouse
Touch-screen
Input Device
Output Device

Programming Languages

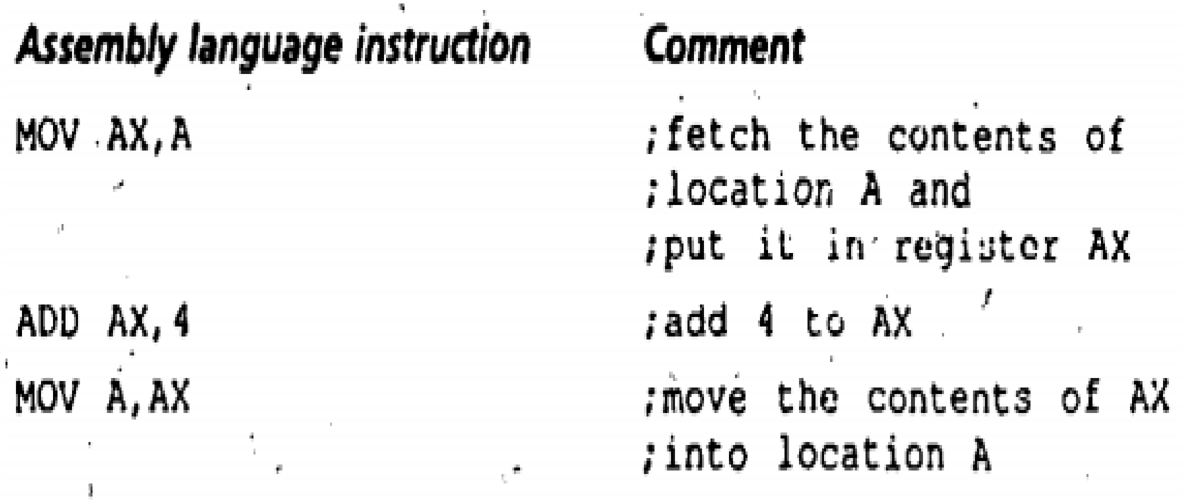
- Machine Language
  - A CPU can only execute machine language instruction.
- Assembly Language
  - A more convenient language
  - A program written in assembly language must be converted to machine language before the CPU can execute it.
  - Assembler translates each assembly language state into a single machine language instruction.
- High-level Language
  - Easier to write, as human understands easily
  - Examples: C, C++, C#, JAVA, Python, FORTRAN, Pascal etc.
  - A compiler is needed to translate a high-level language into machine code

Examples of Language

Both the programs fetches the content of location A, adds 4 with it and stores the new values back to the same location.



Machine instruction	*Operation*
10100001 00000000 00000000	Fetch the contents of memory word 0 and put it in register AX.
00000101 00000100 00000000	Add 4 to AX.
10100011 00000000 00000000	Store the contents of AX in memory word 0.



Assembly language instruction	*Comment*
MOV AX,A	;fetch the contents of ;location A and ;put it in register AX
ADD AX,4	;add 4 to AX
MOV A,AX	;move the contents of AX ;into location A

Thank you