Chapter one: Microcomputer Architecture

Microcomputer Architecture

A computer system has three main components: **a Central Processing Unit (CPU) or processor**, **a Memory Unit** and **Input Output Units (devices)**. In any microcomputer system, the component which actually processes data is entirely contained on a single chip called **Microprocessor** (MPU). This MPU can be programmed using assembly language. Writing a program in assembly language requires a knowledge of the computer hardware (or Architecture) and the details of its instruction set.

The main **internal hardware** features of a computer are the processor, memory and **registers** (registers are special processor components for holding address and data).

The **external hardware** features are the computer Input/output devices such as keyboard, monitor…

Software consists of the operating system (O.S) and various programs and data files stored on disk. Inside any computer based on a member of the 8086 family, the basic arrangement of the main components is shown in Figure 1.

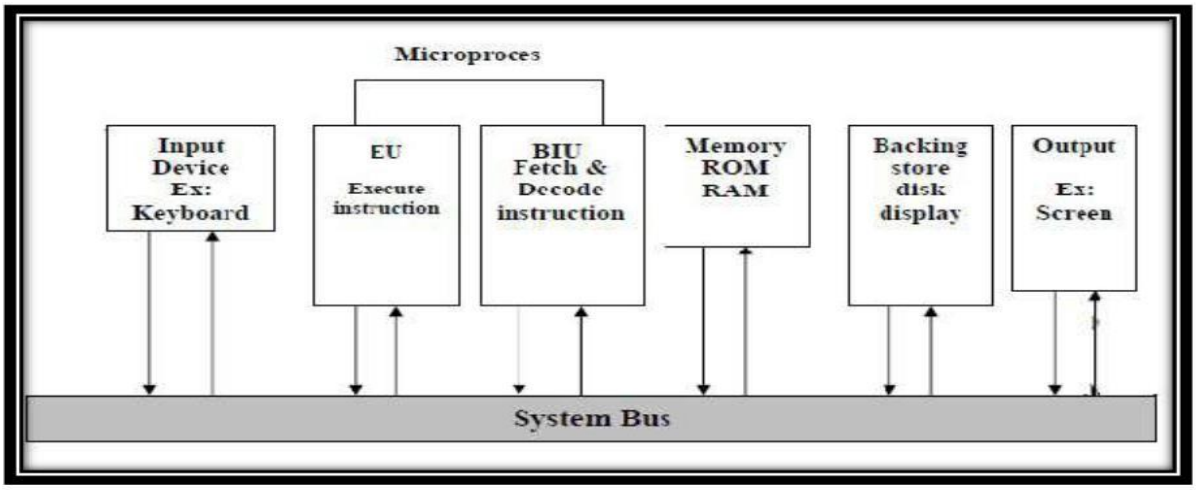

Figure 1: Data flow between the main components of an 8086 family computer.

Information is sent from one main component to another along the communication channel, which is often called the **System Bus**. Both programs and data are stored in the memory.

The **Bus Interface Unit** (BIU) within the MPU fetches new instruction or data as necessary. It is also the BIU jobs to interpret or decode instruction and to route results to their proper destination.

The MPU **Execution Unit** carries out any arithmetic which is required, including memory calculation. Microcomputer memories consist of a collection of chips of two kinds **Read Only Memory** (ROM) and **Random Access Memories** (RAM).

System Bus

The components of the computer system must communicate with each other and with the outside world. Although it may be possible to connect each component to the CPU separately as a practical matter this would require too many physical connects. To keep the number of connections manageable, the processor is connected to memory and all peripherals using a bus.

A **Bus** is a bunch of wires, and electrical path on the printed IC to which everything in the system is connected.

There are three types of Bus:

- **Address Bus (AB)**: the width of AB determines the amount of physical memory addressable by the processor.
- **Data Bus (DB)**: the width of DB indicates the size of the data transferred between the processor and memory or I/O device.

- **Control Bus (CB)**: consists of a set of control signals, typical control signals includes memory read, memory write, I/O read, I/O write, interrupt acknowledge, bus request. These control signals indicates the type of action taking place on the system bus.

Personal Computer (PC) Components

The main component of the PC is its **System Board** (or mother board). It contains the processor, co-processor, main memory, connectors, and expansion slots for optional cards. The slots and connectors provide access to such components as ROM, RAM, hard disk, CD-ROM drive, additional memory, video unit, keyboard, mouse, parallel and serial device, sound adapter and cache memory (the processor use high speed cache memory to decrease its need to access the slower main memory). A bus with wires attached to the system board connect the components. It transfers data between the processor, memory and external devices.

A. The processor

The CPU or processor acts as the controller of all actions or services provided by the system. The operations of a CPU can be reduced to three basic steps: **fetch**, **decode**, and **execute**. Each step includes intermediate steps, some of which are:

- **Fetch the next instruction**:
  - Place it in a holding area called a queue.
  - Decode the instruction.
- **Decode the instruction** :
  - Perform address translation.
  - Fetch operand from memory.

➢ **<u>Execute the instruction</u>**:

- Perform the required calculation.
- Store results in memory or register.
- Set status flag attached to the CPU.

Figure 2 shows a block diagram of a simple imaginary CPU. The CPU is divided into two general parts. **<u>Arithmetic Logic Unit</u>** (ALU) and **<u>Control Unit</u>** (CU).

✓ The **<u>ALU</u>** carry Arithmetic, logical, and shifting operations.

✓ The **<u>CU</u>** fetches data and instruction, and decodes addresses for the ALU

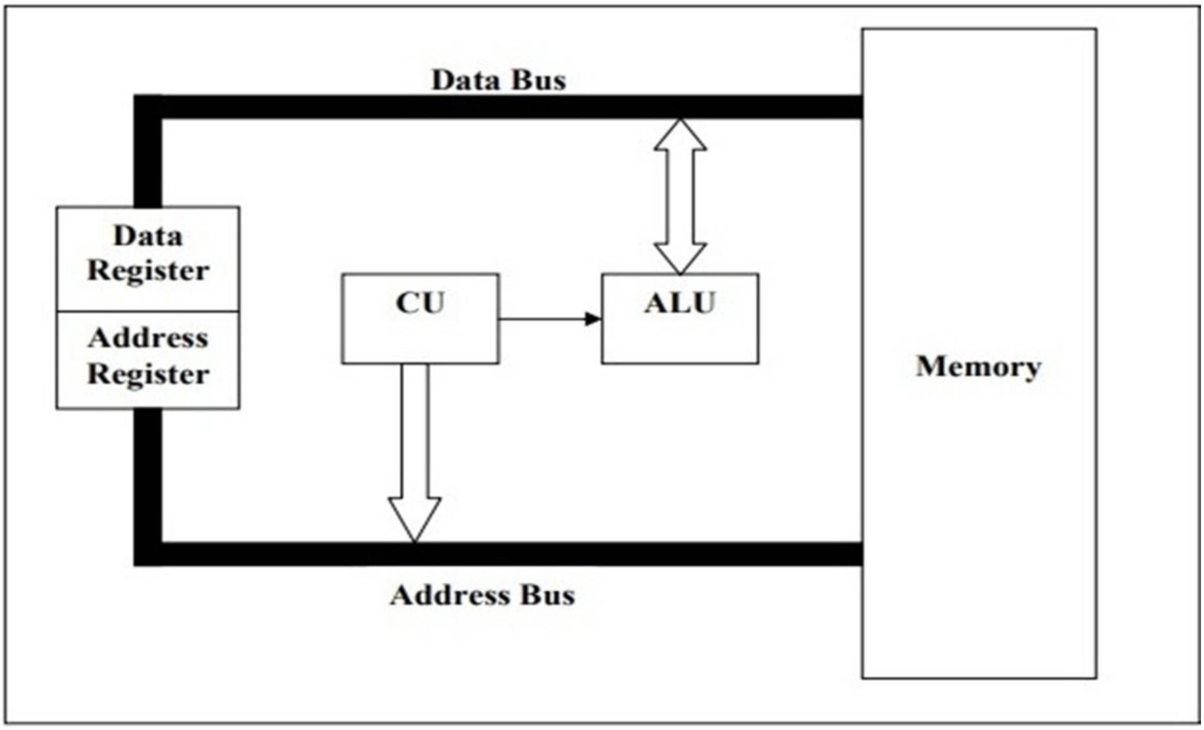

Figure 2: A block diagram of a simple CPU.

B. Memory

The memory of a computer system consist of tiny electronic switches, with each switch set in one of two states: open or close. It is however more convenient to think of these states as **<u>0</u>** and **<u>1</u>**.

Thus each switch can represent a binary digit or bit, as it is known, the memory unit consists of millions of such bits, bits are organized into groups of eight bits called **byte**.

Memory can be viewed as consisting of an ordered sequence of bytes. Each byte in this memory can be identified by its sequence number starting with 0, as shown in Figure 3. This is referred to as memory address of the byte. Such memory is called **byte addressable memory**.

8086 can address up to 1 MB (220 bytes) of main memory this magic number comes from the fact that the address bud of the 8086 has 20 address lines. This number is referred to as the **Memory Address Space** (MAS).

The memory address space of a system is determined by the address bus width of the CPU used in the system. The actual memory in a system is always less than or equal to the MAS.

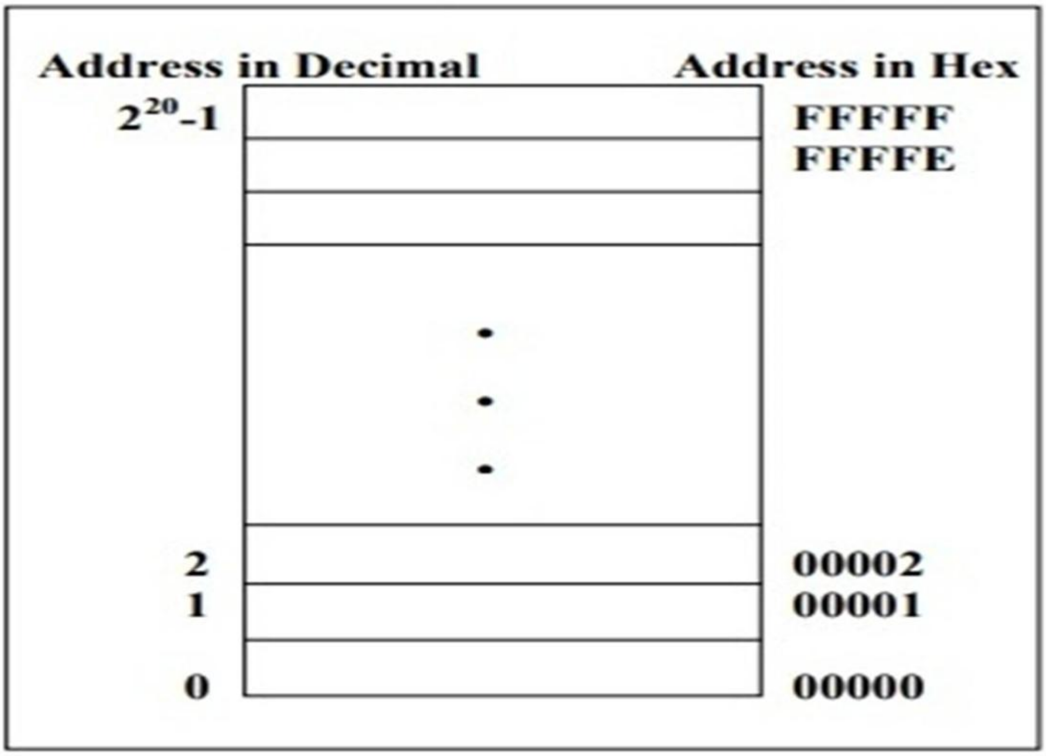

Figure 3: Logical view of the system memory

Two basic memory operations

The memory unit supports two fundamental operations: Read and Write. The read operation read a previously stored data and the write operation stores a value in memory. See Figure 4

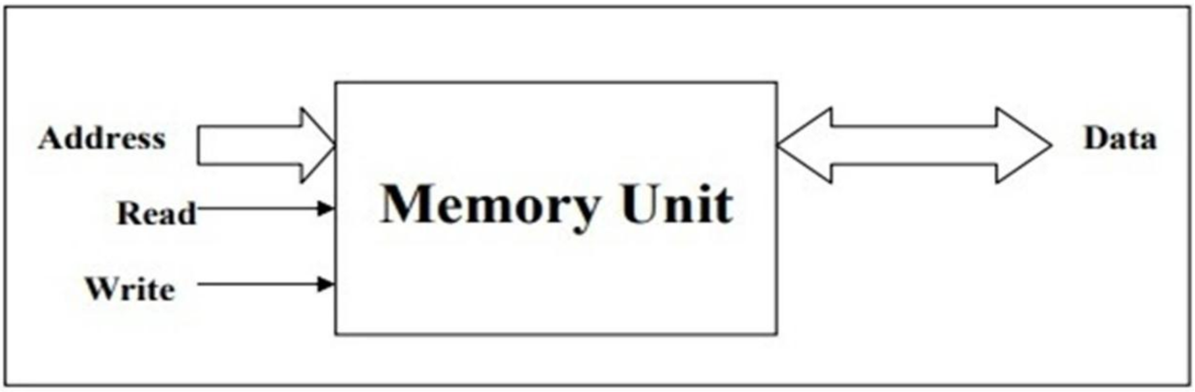

Figure 4: Block diagram of system memory

Steps in a typical read cycle

- ❑ Place the address of the location to be read on the address bus.
- ❑ Activate the memory read control signal on the control bus.
- ❑ Wait for the memory to retrieve the data from the address memory location.
- ❑ Read the data from the data bus.

Drop the memory read control signal to terminate the read cycle.

Steps in a typical write cycle

- o Place the address of the location to be written on the address bus.
- o Place the data to be written on the data bus.
- o Activate the memory write control signal on the control bus.
- o Wait for the memory to store the data at the address location.
- o Drop the memory write control signal to terminate the write cycle.

Addresses

Group of bits which are arranged sequentially in memory, to enable direct access, a number called address is associated with each group. Addresses start at 0 and increase for successive groups. The term location refers to a group of bits with a unique address. Table 1 represents Bit, Byte, and Larger units.

Table1: Bit, Byte, and Larger units.

Name	Number of Byte
Bit	0 or 1
Byte	is a group of bits used to represent a character, typically 8-bit.
Word	2-bytes (16-bit) data item
Double Word	4-byte (32-bits)
Quadword	8-Bytes (64-bit)
Paragraph	16-bytes (128-bit)
KiloByte (KB)	the number 2^{10} = 1024 = 1K for KiloByte, (thus 640K = 640 * 1024 = 655360 bytes)

Types of memory

The memory unit can be implemented using a variety of memory chips- different speeds, different manufacturing technology, and different sizes. The two basic types are RAM and ROM.

1. **Read Only Memories (ROM):** ROMs allow only read operation to be performed. This memory is non-volatile. Most ROMs are programmed and cannot be altered.

This type of ROM is cheaper to manufacture than other types of ROM. The program that controls the standard I/O functions (called BIOS) is kept in ROM, configuration software.

Other types of ROM include

- Programmable ROM (PROM).
- Erasable PROM (EPROM) is read only memory that can be reprogrammed using special equipment.
- EAPROM, Electrically Alterable Programmable ROM is a Read Only Memory that is electrically reprogrammable.

Read/Write Memory

Read/Write memory is commonly referred to as Random Access Memory (RAM), it is divided into **static** and **dynamic**.

Static RAM (SRAM)

- used for implementing CPU registers and cache memories.
- used for special high speed memory called cache memory which greatly improves system performance. Static RAM keeps its value without having to be refreshed.

Dynamic RAM (DRAM)

- the bulk of main memory in a typical computer system consists of dynamic ram.
- main memory, or RAM is where program, data are kept when a program is running. It must be refreshed with in less than a millisecond or losses its contents.

C. INPUT/OUTPUT

Input / Output (I/O) devices provide the means by which the computer system can interact with the outside world. Computers use I/O devices (also called peripheral devices) for two major purposes:

a. To communicate with the outside world and,
b. Store data.

Devices that are used to communicate like, printer, keyboard, modem, Devices that are used to store data like disk drive. I/O devices are connected to the system bus through **<u>I/O controller</u>** (interface) – which acts as interface between the system bus and I/O devices.

There are two main reasons for using I/O controllers

1. I/O devices exhibit different characteristics and if these devices are connected directly, the CPU would have to understand and respond appropriately to each I/O device. This would cause the CPU to spend a lot of time interacting with I/O devices and spend less time executing user programs.
2. The amount of electrical power used to send signals on the system bus is very low. This means that the cable connecting the I/O device has to be very short (a few centimeters at most). I/O controllers typically contain driver hardware to send current over long cable that connects I/O devices. See Figure5.

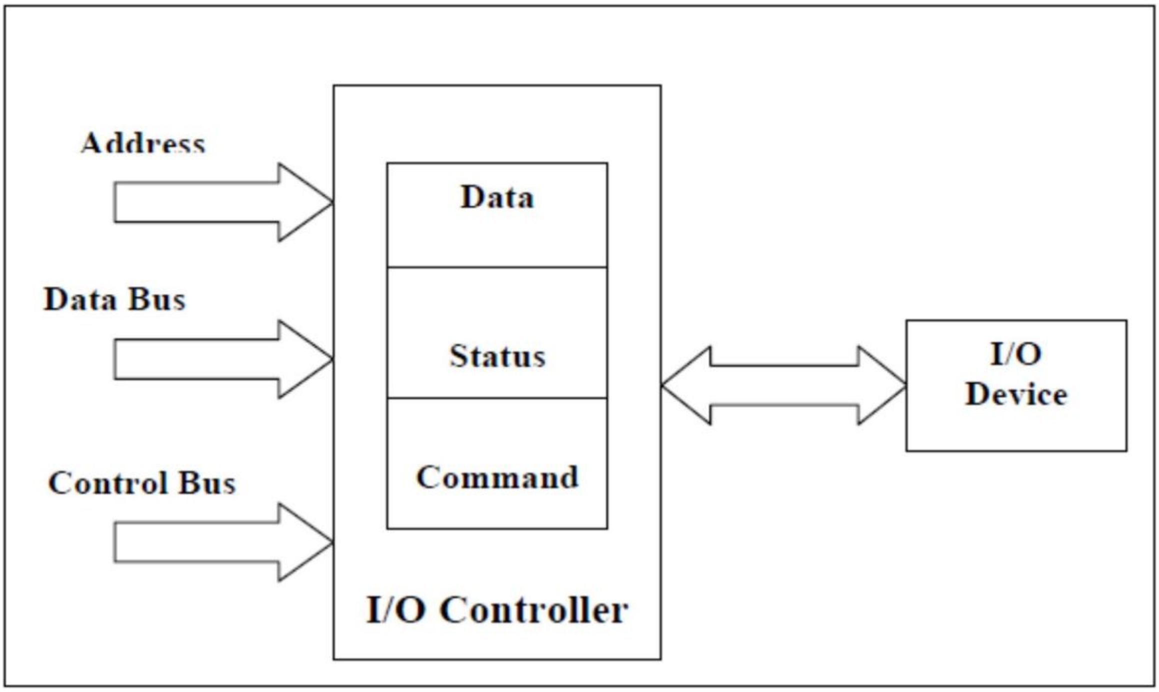

Figure5: Block diagram of a generic I/O device interface.

Evolution of Intel Microprocessor

The principle way in which MPU & microcomputer are categorized in term of the maximum number of binary bit in the data they process that is, their **word length**. Processor vary in their speed, capacity of memory, register and data bus, below are a brief description of various Intel processor in Table 2.

8088 and **8086** functionally identical but **8088** lower performance, **80186** run all **8088** and **8086** software, but have 10 new instructions. **80188** in function is identical to **80186** but lower performance. **80286** run all **8086**, **80186** program, but has extra instruction, more powerful than **8086**. **83086** has various operation mode, which allow it to act as **80286** chip or multiple **8086** chip, as well as a set of instruction capable of 32 bit operations such as arithmetic.

Table 2: Different Microprocessor features descriptions

Microprocessor Name	Features Descriptions		
	Width of (DB)	Width of (AB)	Instruction queue length
8086	16 bit	20 bit	6 Byte
8080	8 bit	20 bit	4 Byte
80186	16 bit	20 bit	6 Byte
80188	8 bit	20 bit	4 Byte
80286	16 bit	24 bit	6 Byte
80386	32 bit	32 bit	6 Byte

Execution Unit and Bus Interface

Unit In the Figure 6, the processor is partitioned into two logical units: an **Execution Unit** (EU) and **Bus Interface Unit** (BIU). The role of the EU is to execute instruction, whereas the BIU delivers instruction and data to EU. The EU contains **ALU**, **CU** and number of **registers**. This feature enables the EU to execute instructions and perform arithmetic and logical operations. The most important function of BIU is to manage the bus control unit, segment registers instruction queue. The BIU controls the busses that transfer data to the EU, to memory, and to external input/output devices, whereas the segment registers control the memory addressing.

Another function of the BIU is to provide access to instructions, because the instructions for a program that is executing are kept in memory, the BIU must access instruction from memory and place them in an instruction queue, which varies in size depending on the processor. This feature enables the BIU to look ahead and prefetch instructions, so that there is always a queue of instructions ready to execute.

The EU and BIU work in parallel, with the BIU keeping one step ahead. The EU notifies the BIU when it needs access to data in memory or I/O devices. Also the EU request machine code instructions from the BIU instruction queue. The top instruction is the currently executable one, and while the EU is occupied executing an instruction, the BIU fetch another instruction from memory. This fetching overlaps with execution and speeds up processing.