1.4 Von Neumann and Harvard Architecture

→ (MU - Dec. 2015, May 2016, Dec. 2016)

Q. What are the functions of following registers ?
(i) PC (ii) SP (iii) MAR (iv) MDR (v) IR
Dec. 15, 5 Marks

Q. With the help of diagram, explain Von-Neumann architecture. **May 16, Dec. 16, 5 Marks**

There are two ways of memory interfacing architectures for a processor depending on the processor design. The first one is called Von Neumann architecture and later Harvard architecture.

1.4.1 Von Neumann Architecture

→ (MU - Dec. 2015)

Q. Define Stored Program Concept and draw Von-Neumann's architecture. **Dec. 15, 5 Marks**

Q. Explain von Neumann's system. **(5 Marks)**

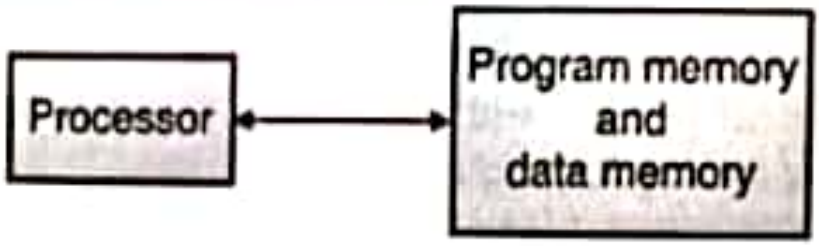


Fig. 1.4.1

- Fig. 1.4.1 shows the connection for Von Neumann architecture of computer.
- The name is derived from the mathematician and early computer scientist John Von Neumann.
- The computer has a common memory for data as well as code to be executed.

- The processor needs two clock cycles to complete an instruction, first to get an instruction and second to get the data.

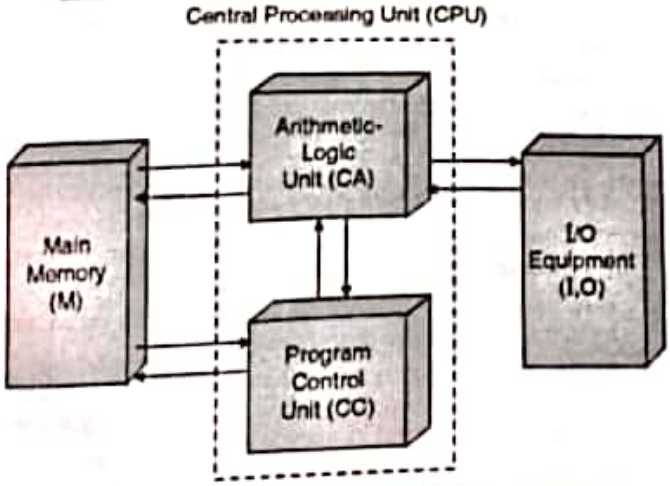


Fig. 1.4.2 : Von Neumann Architecture of a computer

This system has three units CPU, Memory and I/O devices. The CPU has two units Arithmetic Unit and Control unit. Let us discuss these units in detail.

➜ **1. Input Unit**

A computer accepts inputs from the user through these devices i.e. input devices. The commonly used input devices are keyboard and mouse. Besides that, there are devices like joystick, camera, scanner etc. which are also input devices. The devices input the data accepted from the user in a proper coded form understood by the computer.

➜ **2. Output Unit**

The result is given back by the computer to the user through an output device. Input devices and output devices are also called as human interface devices, because they are used to interface the human to the computer. The mainly used output devices are monitor and printer. But there are many other output devices like plotter, speaker etc.

➜ **3. Arithmetic and Logic Unit (ALU)**

Arithmetic or logic operations like multiplication, addition, division, AND, OR, EXOR etc. are performed by ALU. Operands are brought into the ALU, where the necessary operation is performed.

➜ **4. Control Unit**

The control unit as we know is the main unit that controls all the operations of the system, inside and outside the processor. The memory or I/O devices have to be controlled by the computer to perform the operation according to the instruction given to it.

➜ **5. Memory Unit**

➜ **(MU - May 2014, Dec. 2014, May 2015)**

Q. What is stored program concept? **May 14, 3 Marks**

Q. Define stored program concept and draw Von Neumann's Architecture. **Dec. 14, 5 Marks**

Q. What is stored program concept in digital computer ? **May 15, 3 Marks**

- Memory is used to store the programs and data for the computer. The instructions from the programs are taken by the processor, decoded and executed accordingly.

- The data is also stored in the memory. The data is taken from memory and the operation is performed on that data, as well as the results are stored in the memory.
- In some cases the input to an operation and the result may also be from input and output devices.
- Memory in the Von Neumann system has a special organization wherein the data and instructions are stored in the same memory. We will see about this in the subsequent section.

☞ Key Features of a Von Neumann machine

- The Von Neumann machine uses stored program concept. The program and data are stored in the same memory unit.
- Each location of the memory has a unique address i.e. no two locations have the same memory address.
- Execution of instruction in Von Neumann machine is carried out in a sequential manner (unless explicitly altered by the program itself) from one instruction to the next.

☞ Detailed structure of the CPU

- The block diagram of the computer proposed by Von Neumann have a minimal number of registers along with the above blocks.
- This computer has a small set of instruction and an instruction was allowed to contain only one operand address.
- Fig. 1.4.3 gives the detailed structure of IAS CPU. The structure shown in Fig. 1.4.3 consists of the following registers.

☞ Accumulator (AC)

It normally provides one of the operand to ALU and stores the result.

☞ Data Register (DR)

It acts as buffer storage between the CPU and the main memory or I/O devices.

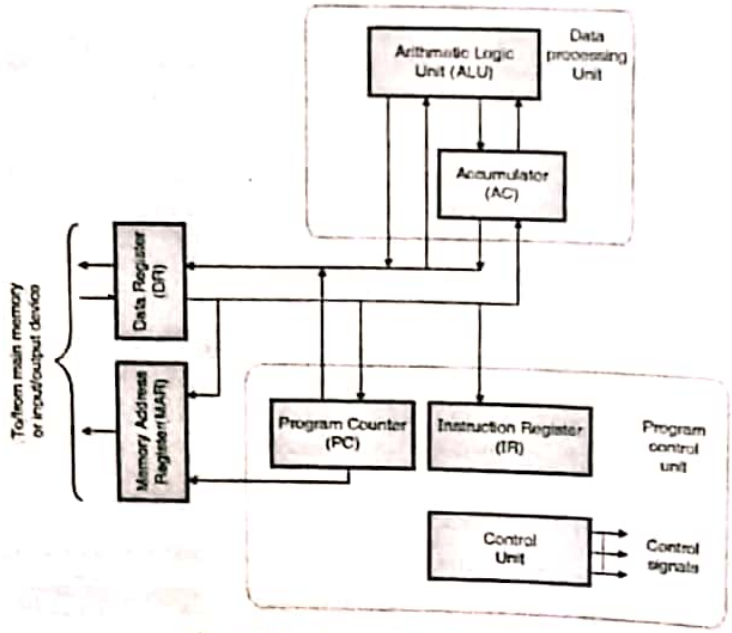


Fig. 1.4.3 : Structure of the CPU

☞ Program Counter (PC)

It always contains the address of the next instruction to be executed.

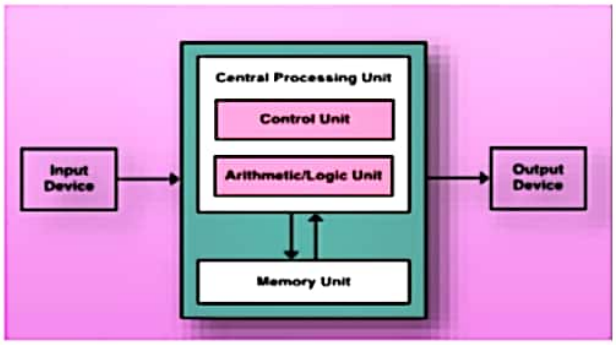
☞ Instruction Register (IR)

It holds the current instruction i.e. the opcode and operand of the instruction to be executed.

VON NEUMANN ARCHITECTURE

- Four components of the architecture

- Memory
- Control Unit
- Arithmetic Logic Unit(ALU)
- Input/Output



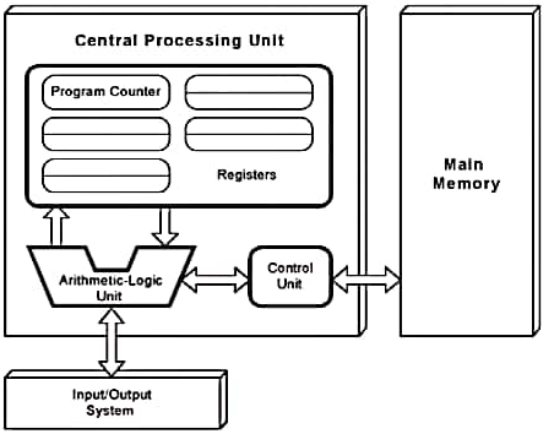
VON NEUMANN

- The von neumann architecture has been incredibly successful, with most modern computers following the idea.
- Von neumann architecture is also called as "Stored–Program Architecture"
- The most important feature is memory.
- It is better for desktop computers, laptops, high performance computers.

- Key Concepts :
- Data and instructions are stored in a single read-write memory.
- The content of this memory are addressable by locations.
- Execution occurs in a sequential fashion from one instruction to the next.

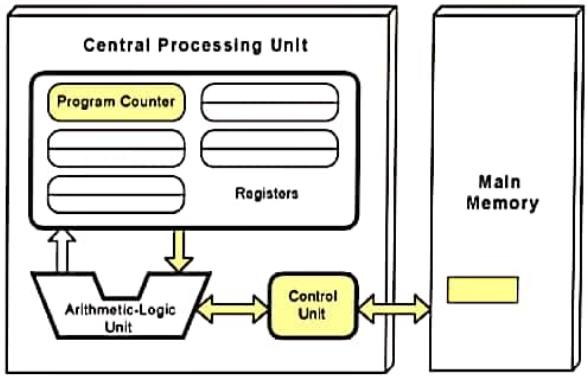
THE VON NEUMANN MODEL

- These computers employ a
- fetch-decode-execute cycle to run programs as follows .



THE VON NEUMANN MODEL

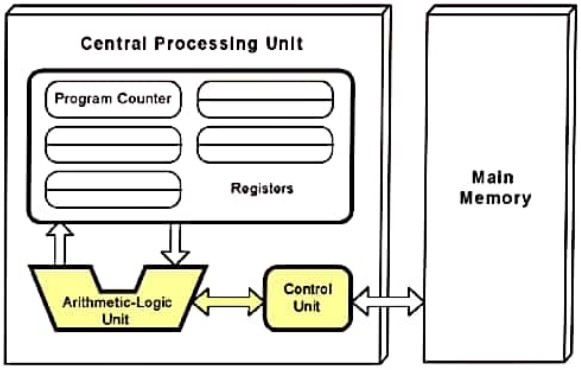
- The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.



32

THE VON NEUMANN MODEL

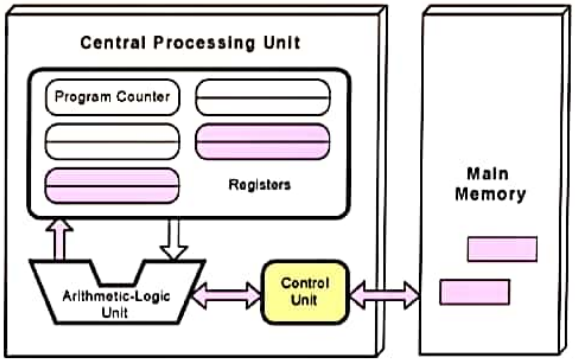
- The instruction is decoded into a language that the ALU can understand.



33

THE VON NEUMANN MODEL

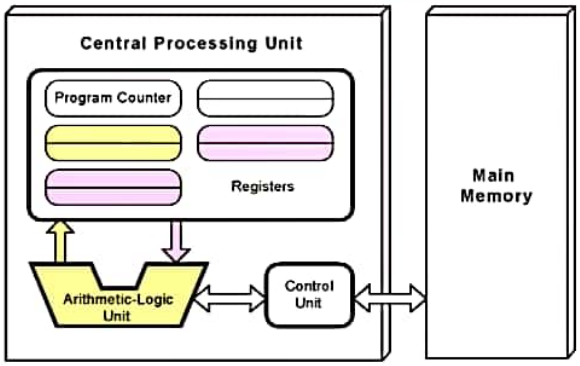
- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.



34

THE VON NEUMANN MODEL

- The ALU executes the instruction and places results in registers or memory.



Program Counter –
Register that contains the address of current instruction.
With each fetched instruction counter increments by 1. After fetching each instruction PC Points to next instruction.

ADVANTAGES

- **ADVANTAGES**

  Simplicity

  Flexibility: any well coded program can be executed.

DRAWBACKS

Speed efficiency- Not efficient due to sequential programming.

Resource efficiency: only one part of the hardware resources is required for the execution of instruction. The rest remains idle.

Memory Access : Memories are slower than the processor.

DISADVANTAGES

- **Bottleneck is an issue because only one bit of information can be accessed at once**
- **Confusion between data and instructions can lead to a system crash.**
- **Instructions stored as the data can be accidently rewritten by an error in a program.**
- **Serial instruction processing does not allow parallel execution of the program. Parallel execution are simulated later by the Operating system.**
- **Only handles one task at a time**

CHAPTER 1

Overview of Computer Architecture & Organization

Module I

llabus

roduction of computer organization and architecture, Basic organization of computer and block level description of e fictional units, Evolution of computers, Von Neumann model, Performance measure of computer architecture, chitecture of 8086 family, 8086 Hardware design, Minimum mode and maximum mode of operation, Study of bus ntroller 8288 and its use in Maximum mode.

Syllabus Topic : Introduction to Computer Organization and Architecture

Introduction to Computer Organization and Architecture

Compare Computer Architecture and organization. **(5 Marks)**

There are various people involved in making of a computer. The chip designer who designs and manufactures the chip, he system designer who designs the system using the nanufactured chip or microprocessor and the programmer who makes software using the system.

Those attributes of a computer that are necessary to be known o a system designer or a programmer are called as the rchitectural features of the computer. Hence the people nanufacturing the chip have to reveal certain things about the rocessor to the system designer and the programmer using he datasheets for their processor chips.

Those attribute of a computer or moreover the processor, that re just used for the designing purposes of the processor and re not revealed are called as the organizational features of he processor.

Computer architecture	Computer organization
It refers to those attributes of a system visible to the programmer.	It refers to the implementation of these features and is mostly not known to the user.
Instruction set, number of bits used for data representation, addressing techniques etc. form the part of computer architecture.	Control signals, interfaces, memory technology etc. form the part of the computer organization.
For example is there a multiply instruction?	For example is there a dedicated hardware multiply unit or it is done by repeated addition?
4. All INTEL 80x86 microprocessors share the same basic architecture.	All INTEL 80x86 microprocessors differ in their organization.

Syllabus Topic: Basic Organization of Computer and Block Level Description of Functional Units

1.2 Basic Organization of Computer and Block Level Description of Functional Units

1.2.1 Structural Components of a Computer

Q. Explain the structural overview of a computer. **(5 Marks)**

- It is the way in which components related to each other. Fig. 1.2.1 shows the structure of a digital computer.
- It is made up of three main components namely the Central Processing Unit (CPU) or the processor, the memory to store the programs and data, and the Input / Output (I/O) devices.
- The functions of these are explained below :

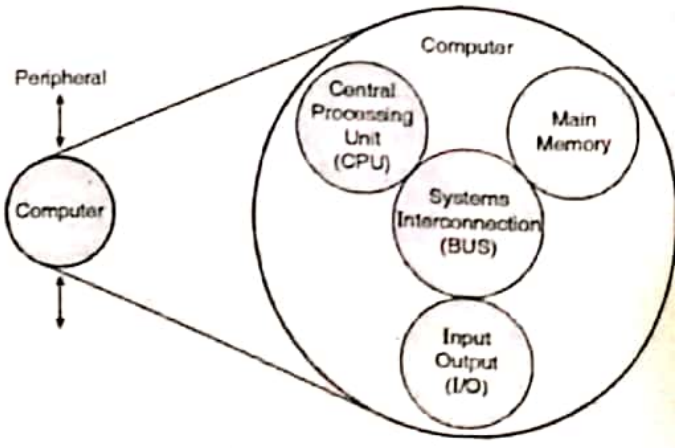


Fig. 1.2.1 : Structure of a computer

• This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.

• These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.

• After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.

• The BIU also contains a dedicated adder which is used to generate the 20 bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.

• For example, the physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.

• The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write

The BIU uses a mechanism known as an instruction stream queue to implement a pipeline architecture.

This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.

These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.

After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.

The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.

These intervals of no bus activity, which may occur between bus cycles are known as **Idle state**.

If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write Cycle.

The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.

- These instruction bytes are stored in the 6 byte Queue on the first-in-first-out (FIFO) basis.
- When EU completes the execution of the existing instruction, and becomes ready for the next instruction, it simply reads the instruction bytes in the sequence 1, 2, ... from the queue.
- Thus the Queue will always hold the instruction bytes of the next instructions to be executed by the EU.

☞ Pipelining

- The process of fetching the next instruction, when the present instruction is being executed is called as pipelining.
- Pipelining has become possible due to the use of queue.
- BIU fills in the queue, until the entire queue is full.
- BIU restarts filling in the queue when at least two locations of queue are vacant.

☞ Advantage of pipelining

- The EU always reads the next instruction byte from the queue in BIU. This is much faster than sending out an address to the memory and waiting for the next instruction byte to come.
- In short pipelining eliminates the waiting time of EU and speeds up the processing.
- The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue.8086 BIU normally obtains two instruction bytes per fetch.

☞ Behavior of queue in JMP and CALL instructions

- If a Jump (JMP) or CALL instruction appears in the main program, then all the existing instruction bytes in the Queue are flushed out and Queue is made empty.
- Then it is reloaded with the new instruction bytes which correspond to the new locations mentioned in the JMP or CALL instructions.
- The refilling of the queue then continues from the new locations corresponding to the in main program.

☞ Why is the 8086 Queue only six byte long?

- Because the longest instruction, in the instruction set of 8086 is six byte long. Hence with a six byte long queue it is

1.13 8086 Internal Architecture

Q. Explain the architecture of 8086. **(5 Marks)**

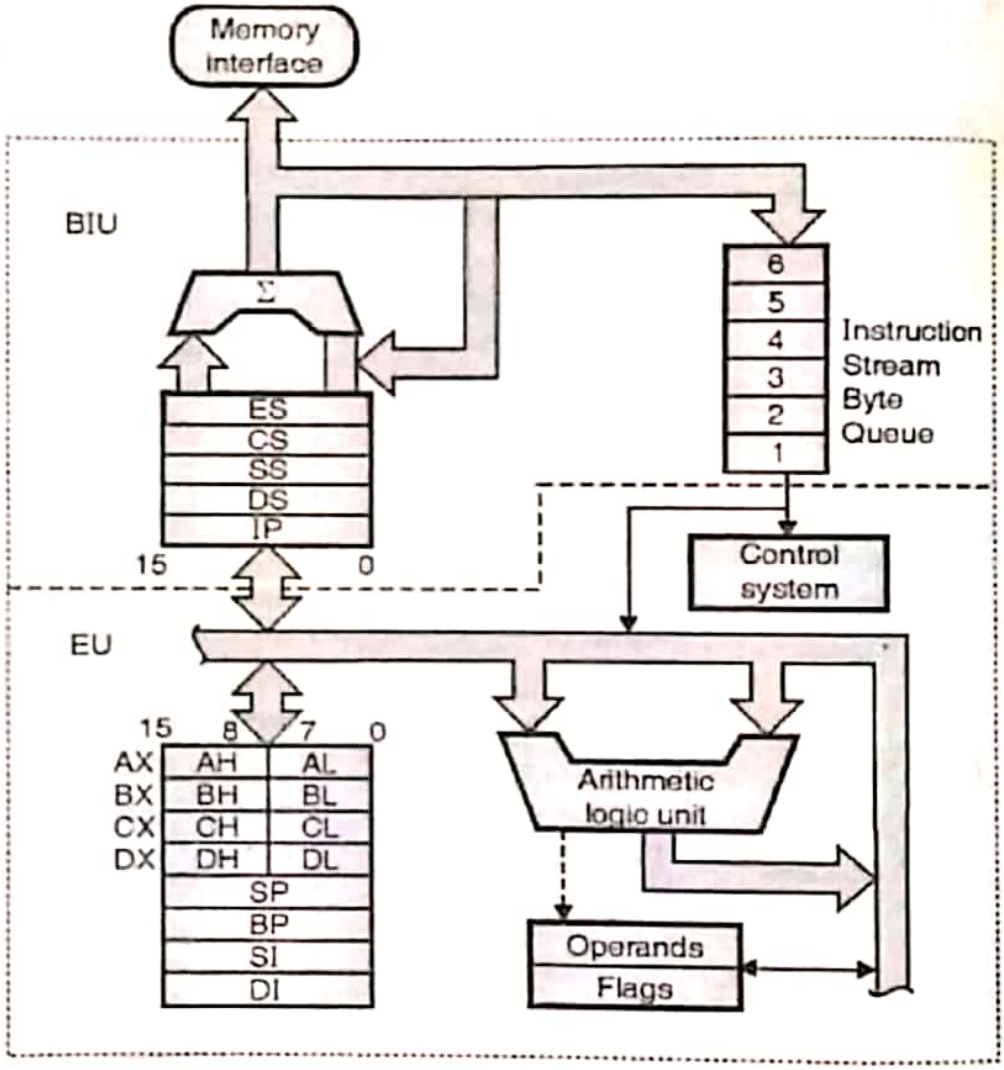


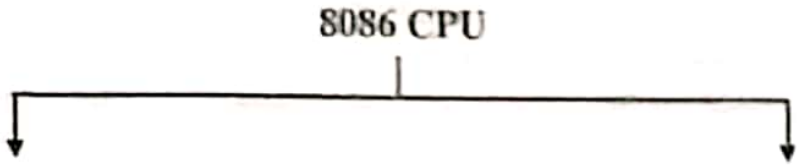
Fig. 1.13.1 : 8086 internal architecture

- Before talking about programming, we need to discuss the special features and internal architecture of Intel 8086. Fig. 1.13.1 shows the block schematic of the internal structure of Intel 8086.
- As shown in Fig. 1.13.1, the 8086 CPU is divided into two units viz :

  1. The bus interfacing unit (BIU)
  2. The execution unit (EU).

- These two units are completely independent of each other and they share the work of CPU. Such a work division speeds up the processing and reduces the processing time. This is called as pipelining.

The work of 8086 CPU is divided between BIU and EU in the following manner.



Bus Interfacing Unit (BIU)

Tasks of BIU are

- It sends out addresses.
- It fetches instructions from memory
- It reads data from memory and ports.
- It also writes data from memory and ports.
- So BIU takes care of all the address and data transfers on the buses.
- Hence it is called the external world interface of the processor.
- It works in synchronous with machine cycles.

Execution Unit (EU)

Tasks of EU are

- It tells the BIU, from where to fetch the instruction or data.
- It decodes the fetched instructions.
- It executes the decoded instructions.
- EU takes care of performing operations on the data.
- EU is called as the execution heart of the processor.
- It works in synchronous with t-states.

1.14 The Execution Unit (EU)

Main function of EU is **decoding** and **execution** of the instructions. In order to carry out its tasks it has the following units

1. Arithmetic Logic Unit (ALU)
2. Flag register.
3. General purpose registers.
4. Control unit
5. Decoder
6. Pointer and index registers

1.14.1 Arithmetic Logic Unit (ALU)

- The ALU in the EU is a 16 bit unit i.e. it can perform 16-bit operation simultaneously.
- It is capable of performing a variety of arithmetic and logic operations such as add, subtract, AND, OR, NOT, EX-OR, increment, decrement, shift etc.

1.14.3 General Purpose Registers

- As shown in Fig. 1.14.2, the execution unit (EU) has four general purpose 16-bit registers.
- Each one of them can be used for temporary storage of 8 bit data, 16-bit data or 32-bit data.
- 8-bit registers - AH, AL, BH, BL, CH, CL, DH, DL. Any of these registers can be used as an 8-bit operand.
- 16-bit registers - AX, BX, CX, DX, SI, DI, SP, BP. Any of these registers can be used as a 16-bit operand.
- 32-bit registers – DX : AX together can be used for 32-bit operand.
- These registers can be used for general purpose computing when their other specialized functions do not interfere.

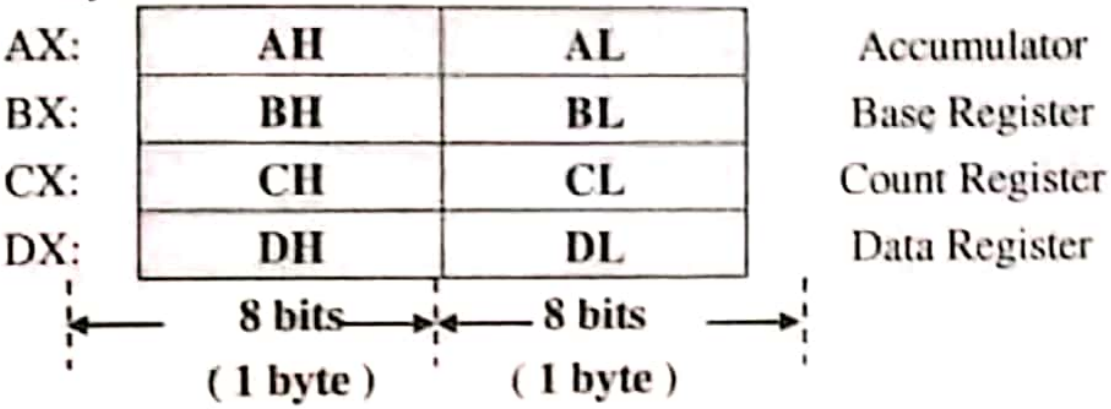
☞ **Can we store 16 bit data words in the general purpose registers?**

- The answer is yes. We can use certain pairs of the general purpose registers to store 16 bit data words.
- Such register pairs are AH and AL, BH and BL, CH and CL, DH and DL.

- The above mentioned register pairs are referred to as follows.

Pair of general purpose registers	Referred to as
1. Pair of AH and AL	AX register
2. Pair of BH and BL	BX register
3. Pair of CH and CL	CX register
4. Pair of DH and DL	DX register

- In short, the four general-purpose 16-bit data registers: AX, BX, CX, and DX, each of these is a combination of two 8-bit registers which are separately accessible as AL, BL, CL, DL (the "low" bytes) and AH, BH, CH, and DH (the "high" bytes).



- For Example, if AX contains the 16-bit number 1234H, then AL contains 34H and AH contains 12H.
- The upper and lower halves of the data registers are separately addressable. This means that each data register can be used as a single 16-bit register or as two 8-bit registers.

☞ Special functions of general purpose register

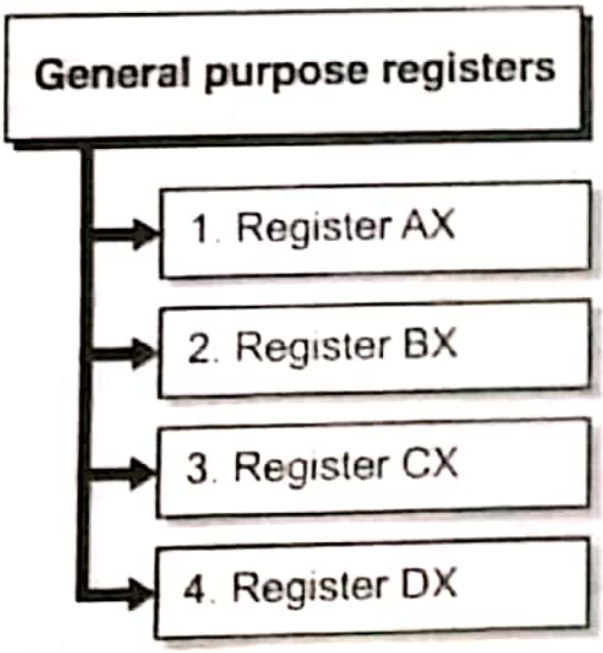


Fig. 1.14.2 : General purpose registers

Although the first four registers AX, BX, CX, and DX are called as "general purpose", each of them is designed to play a particular role in common use :

➜ 1. Register AX

AX is the "16-bit accumulator" while AL is "8-bit accumulator".

Accumulator has the following special functions :

(i) Some of the operations, such as Multiplication and Division, require that one of the operands be in the accumulator and also the result is stored in accumulator.

Some other operations, such as Addition and Subtraction, may be applied to any of the registers (that is, any of the eight general- and special-purpose registers) but are more efficient when working with the accumulator.

(ii) It works as a via register for I/O accesses i.e. a data is routed through accumulator for the communication of the processor and I/O devices.

For OUT instruction the data in accumulator (AL for 8-bit data and AX for 16-bit data) can only be given to the output device.

For IN instruction the data taken from the input device can be taken only in accumulator (AL for 8-bit data and AX for 16-bit data).

(iii) It also works as a via register for string instructions. Whenever a data is to be brought from memory or given to memory in case of string operations it is routed through accumulator only.

➜ **2. Register BX**

BX is the "base" register.

(i) It is the only general-purpose register which may be used for indirect addressing. (Various addressing modes are discussed in next chapter).

(ii) For Example, the instruction MOV [BX], AX causes the contents of AX to be stored in the memory location whose address is given in BX.

➜ **3. Register CX**

CX is the "count" register. It works as a default counter register for three instructions viz:

(i) The looping instructions (LOOP, LOOPE, and LOOPNE), to indicate the number of iterations.

(ii) The shift and rotate instructions (RCL, RCR, ROL, ROR, SHL, SHR, and SAR), to indicate number of shifts or rotations (Here only CL is used and not entire CX).

(iii) The string instructions (with the prefixes REP, REPE, and REPNE) to indicate the size of the string block.

➜ **4. Register DX**

DX is the "data" register

(i) It is used together with AX for the word-size MUL and DIV operations, when the operand size is greater than the register AX i.e. operand is 32-bit.

(ii) It also holds the port number for the IN and OUT instructions. For 16-bit address accesses of I/O ports only DX can be used as a pointer.

Summary of Implicit use of General Purpose Registers

Registers	Operations
AX	Word multiply, Word divide, Word I/O and Word string.
AL	Byte multiply, byte divide, byte I/O, byte string and decimal / ASCII arithmetic.
BX	Store address information.
CX	Counter for String operations and loops.
CL	Counter for Variable shift and rotate.
DX	Word multiply, word divide, Indirect I/O.

1.14.4 Control Circuitry

The control circuit is a part of EU. It is used for directing the internal operations.

1.14.5 A Decoder

- "The process of translation from instructions into action is known as decoding".
- A decoder in the execution unit (EU) is used for translating the instructions fetched from the memory into a series of actions.
- The EU will actually carry out these actions.

1.14.2 Flag Register

Q. Explain the flag register of 8086.

- Flag register is a part of EU. It is a 16-bit register with each bit corresponding to a flip-flop.
- A flag is a flip-flop. It indicates some condition produced by the execution of an instruction. For example the zero flag (ZF) will set if the result of execution of an instruction is zero.
- A flag can control certain operations of EU.
- Fig. 1.14.1 shows the details of the 16 bit flag register of 8086 CPU. As shown, it consists of nine active flags out of sixteen.
- The remaining seven flags marked "U" are undefined flags.

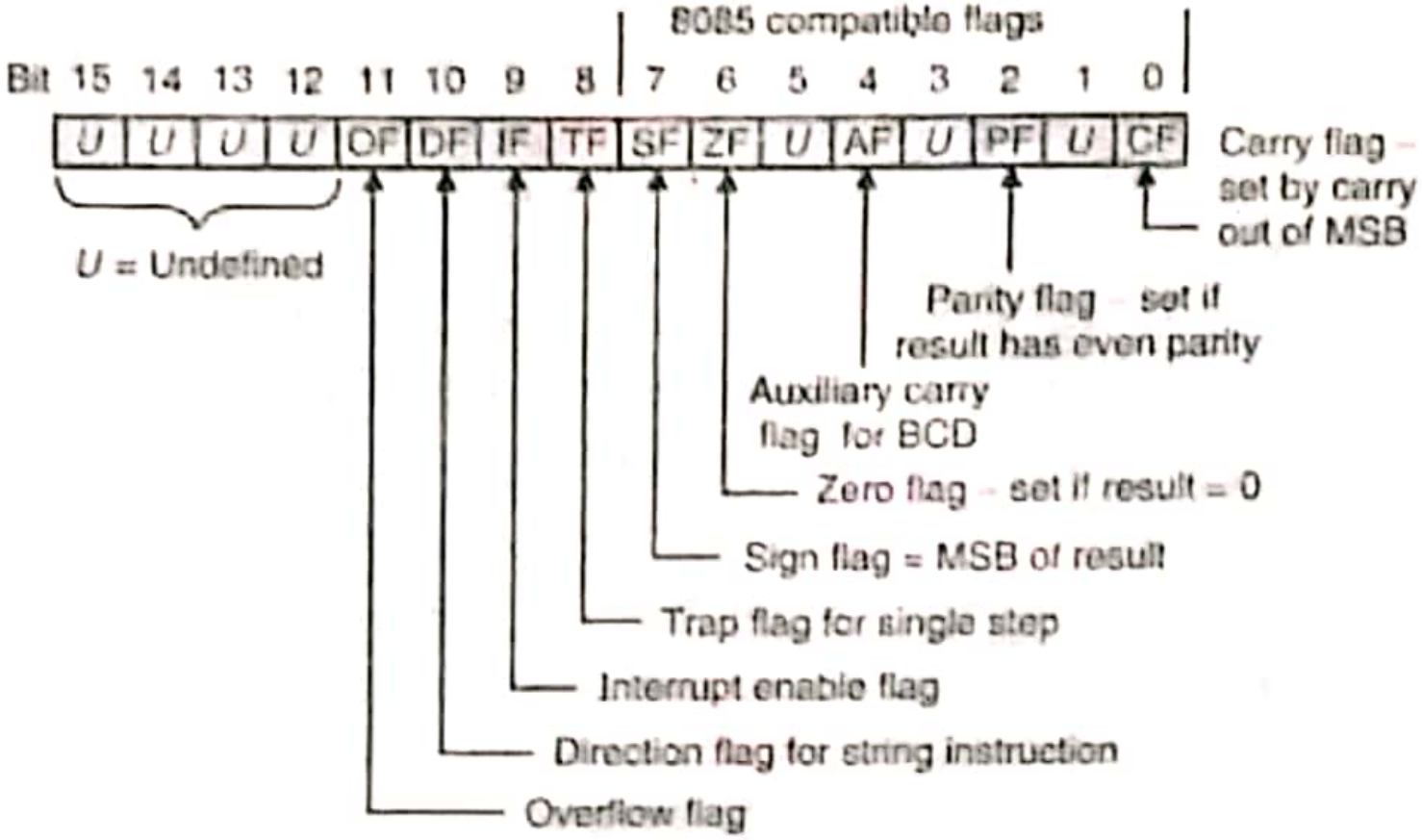


Fig. 1.14.1 : 8086 flag register format

☞ Active flags

There are nine active flags out of 16, in the 8086 flag register. The remaining are undefined flags.

☞ Control flags

- Out of the nine active flags, six are conditional (status) flags and the remaining three are called as the control flags, because they are used to control certain operations of the processor.
- The three control flags are :

1. The Trap Flag (TF)
2. The Interrupt Flag (IF)
3. The Direction Flag (DF)

1.15.2 Segmentation in 8086

Q. List and explain the various pointer registers of 8086. **(5 Marks)**

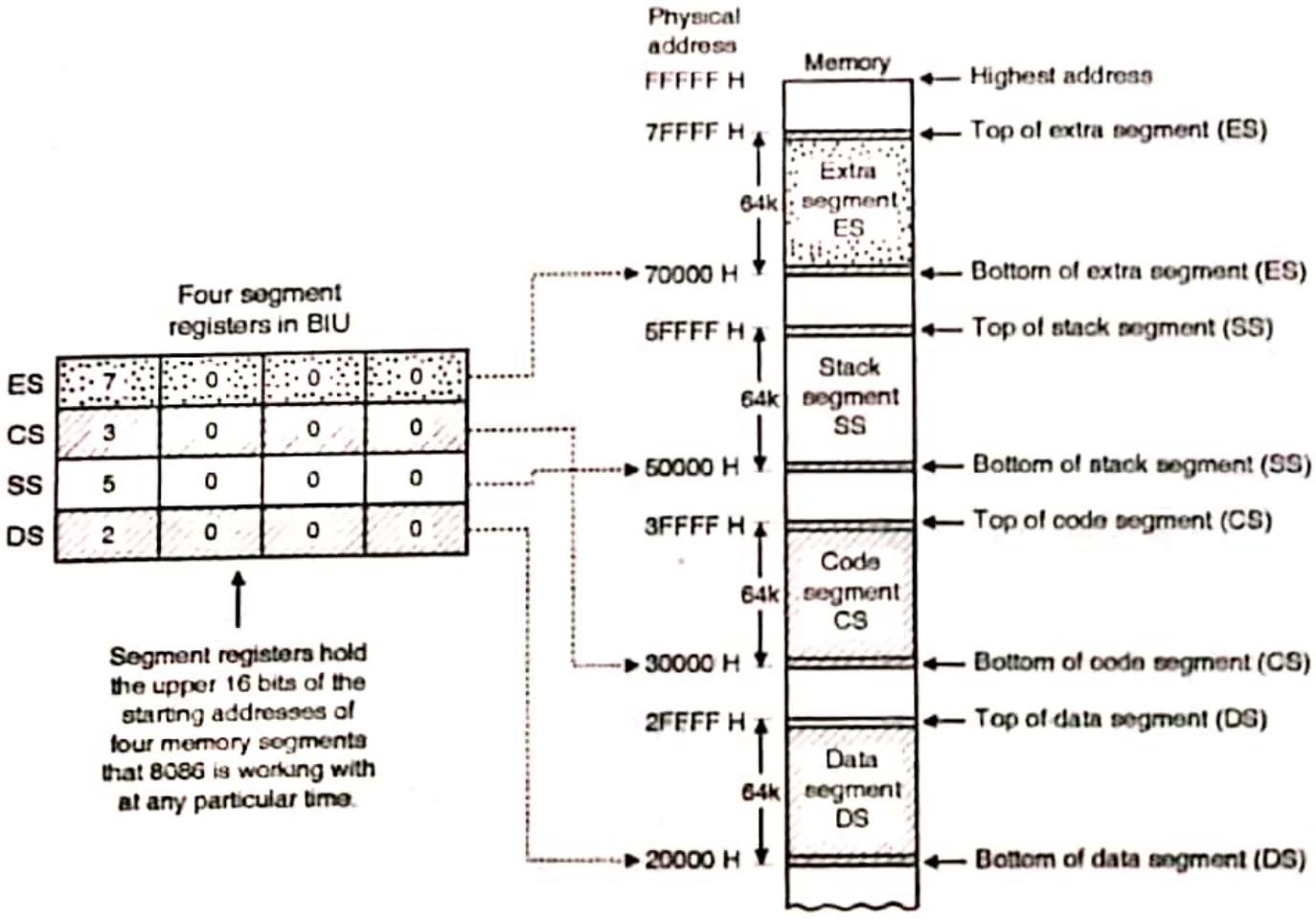
Q. Explain the implementation of segmentation of 8086. **(5 Marks)**

- Memory can be thought of as a vast collection of bytes. These bytes need to be organized in some efficient manner in order to be of any use.
- Segmentation in 8086 refers to division of the 1MB main memory into segments or blocks of 64KB each. This is done so as to access a segment of the memory using the 16-bit address.
- The BIU contains four special purpose registers called as segment registers. They are :

1. The Code Segment (CS) register
2. The Stack Segment (SS) register
3. The Extra Segment (ES) register and
4. The Data Segment (DS) register

The purpose of using these segment registers can be explained as follows :

- All these are 16 bit registers.
- The number of address lines in 8086 is 20. So the 8086 BIU will send out a 20 bit address in order to access one of the 1,048,576 or 1 Mb memory locations.
- But it is interesting to note that the 8086 does not work the whole 1,048,576 byte (1M.byte) memory at any given time. However it works with only four 65,536 (64k-byte) segments within the whole 1 M-byte memory.



. 1.15.2 : One way of positioning four 64 k byte segments within the 1 M byte memory space of an 8086

- The four segment registers actually hold (contain) the upper 16 bits of the starting addresses of the four memory segments of 64 k byte each with which the 8086 is working at that instant of time.
- Note that these starting addresses will always be changing. They are not fix.
- This concept can be clearly understood by referring to Fig. 1.15.2.
- Fig. 1.15.2 shows one of the possible ways to position the four 64 k byte segments within the 1-M byte memory space of 8086. There is no restriction on the locations of these segments in the memory.
- Note that these segments can be separate from each other as shown in Fig. 1.15.2 or they can overlap.
- Note that the starting address or base address of the data segment is 20000H. The upper 16-bits of this i.e. 2000 are loaded into the data segment register (DS).
- Similarly upper 16 bits of the starting addresses of the other segments are stored into the corresponding segment registers.
- The segment overlapping usually takes place for small programs which do not need all the 64 k bytes in each segment.
- The segments can adjacent, disjoint, partially overlapping or fully overlapping.
- In the users program there can be many segments. But 8086 can deal with only four of them at any given time, because it has only four segment registers.
- Whenever the segment orientation is to be changed, we have to change the base addresses and load the upper 16 bits into the corresponding segment registers.