BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2103

COMPUTER ORGANIZATION

Unit 4

Processor Organization

Table of Contents

1. INTRODUCTION

In the last unit, we studied about the different types of instructions and their characteristics, different types of operation, and different types of addressing modes. You must be aware that the microprocessor is the main functional component of the computer system. The processor executes the instructions of a computer program, to perform the key logical, arithmetical, and input/output operations of the computer. This unit deals with various aspects of the processor. The unit starts with the discussion of parallelism and computer arithmetic. Then we will move on to the 8086 processor and study various aspects of it such as floating-point, programmers model, and max/min mode. You will learn about the register organization of the 8086 processors.

In the later part of the unit, you will learn about the instructional cycles and read-write cycles. It is very essential to understand the use of the addressing modes before you start writing assembly language programs. Therefore, in the end, we will cover addressing modes.

1.1 Objectives:

After studying this unit, you should be able to:

- *Discuss Parallelism and Computer Arithmetic*
- *Explain in detail the floating-point and programmer's model of 8086*
- *Discuss the max/min mode and register organization of 8086*
- *Explain the Instruction cycle and the Read Write cycle*
- *Discuss in detail the various types of addressing modes*

2. PARALLELISM AND COMPUTER ARITHMETIC

You will be surprised to know that with the advancement in multi-core architectures, parallel computing is increasingly gaining importance for software developers, scientists, engineers, and computer system designers. Parallel processing, which was once considered an exotic technology, has now become quite common. In this section, we review the role of parallelism in computing and introduce the parallel processing that willserve as the basis for subsequent discussion.

2.1 Parallelism

Generally, most PCs use one microprocessor for every calculation, performing one instruction after another. In order to increase the speed of execution, the concept of parallelism is used. The parallelism technology involves multiple microprocessors cooperating simultaneously to accomplish one task. Figure 4.1 depicts the concept of parallelism. Parallelism canincrease the operating time of battery-powered computers by decreasing power consumption.

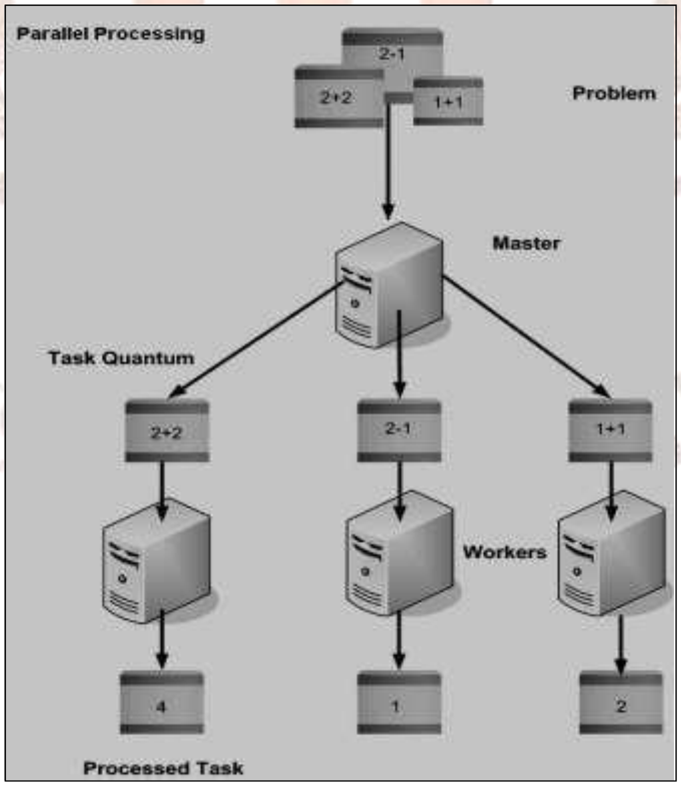


Fig 4.1: An Example of Parallelism

2.2 Computer Arithmetic

Computer arithmetic covers methods of representing integers and real values in digital systems and effective algorithms for controlling such numbers by way of software or hardware circuits. The hardware aspect of computer arithmetic includes adders, subtractors, dividers, multipliers, square-rooters, and circuit techniques. On the software side, complexity, stability, error characteristics, and certifiability of computational algorithms are covered. Performance in many computer applications is critically dependent on the speed of arithmetic operations.

2.3 Computer Arithmetic Associativity

Programs have typically been written first to run sequentially before being rewritten to run concurrently, so a natural question is "do the two versions get the same answer?" If the answer is no, you presume there is a bug in the parallel version that you need to track down. This approach assumes that computer arithmetic does not affect the results when going from sequential to parallel. This assumption holds for two's complement integers even if the computation overflows. Another way to say this is the integer addition is associative. But as floating-point numbers are approximations of real numbers and because arithmetic has limited precision, floating-point addition is not associative.

A more confusing version of this pitfall occurs on a parallel computer where the operating system scheduler may use a different number of processors depending on what other programs are running on a parallel computer. The unaware parallel programmer may be confused by his/her programs getting slightly different answers each time it is run for the same identical code and input, as the varying number of processors for each run would cause the floating-point sums to be calculated in different orders.

Given this quandary, programmers who write parallel code with floating-point numbers need to verify whether the results are credible even if they don't give the same exact answers as the sequential code. The field that deals with such issues are called numerical analysis.

Self-Assessment Questions – 1

1. Expand the following abbreviations:
   a) SIMD
   b) MIMD
2. Floating-point addition is not associative. (True/False)

3. FLOATING POINT IN THE 8086

Before studying the floating-point in 8086 let us discuss the 8086 model and its features briefly.

The Intel 8086 is a 16-bit microprocessor developed by Intel. The phrase "16-bit" signifies that its arithmetic logic unit, majority of its instructions, and internal registers are designed to perform 16-bit binary words. It possesses not only a 16-bit data bus but also a 20-bit address bus. You can see the internal architecture of 8086 in figure 4.2.

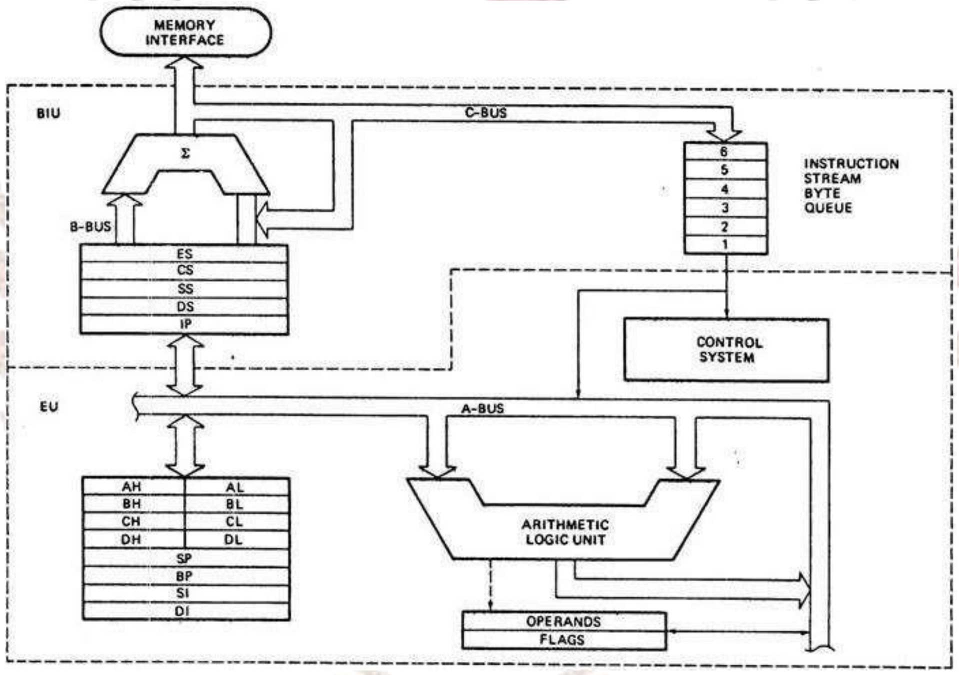


Fig 4.2: Internal Architecture of 8086

As shown in figure 4.2, the 8086 CPU is divided into two separate functional parts, the Execution Unit or EU and the Bus Interface Unit or BIU. The BIU handles all data and addresses on the buses for the execution unit. It reads data from memory and ports, writes data to memory and ports, fetches instructions from memory, and sends out addresses.

The execution unit (EU) tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions.

Some of the important features of the 8086 microprocessor are as follows:

- It is a 16-bit microprocessor

- 8086 possesses a 20-bit address bus and is able to access up to 220 memory locations (1 MB)
- It is capable of supporting up to 64K I/O ports
- It utilizes a 16-bit data transfer bus
- It possesses multiplexed address and data bus AD0- AD15 and A16 – A19
- It needs a single-phase clock along with a 33% duty cycle to offer internal timing
- It is planned to work in two modes, viz., minimum and maximum.
- It can pre-fetch up to 6 instruction bytes from memory and line them up to accelerate instruction execution
- It needs a +5V power supply
- A 40 pin dual inline package

We will discuss in detail about 8086 model in the subsequent topics.

The bus architecture of 8086: The 80x86 family consists of three major busses: data, address, and control bus.

The functions of the three buses are listed below:

- The 80x86 processors utilize the data bus to move data between the several components in the computer system.
- Address bus carries the address of the instruction to be executed by the processor.
- The data bus on an 80x86 family processor moves information between an I/O device or a particular memory location and the CPU.

Floating point: In order to add hardware/microcode-based floating-point performance, the 8086/8088 can be connected to a mathematical coprocessor.

The Intel 8087 was the standard math coprocessor for the 8086 and 8088. When the 8086 was first introduced, semiconductor technology was not so advanced that Intel could assign floating-point instructions straightaway on the 8086 CPU. Consequently, they invented a system through which they could employ a second chip to execute the floating-point calculations. They introduced their floating-point chip- 8087 in the year 1980. This specific floating-point unit (FPU) worked with the 8086, 8088, 80186, and 80188 CPUs. After the introduction of the 80286 CPU, Intel brought out the 80287 FPU chip to go with it.

The 80486 CPU was the first Intel CPU to include an on-chip FPU. Soon after the introduction of 80486, Intel brought out 80486sx CPU that was 80486 without the built-in FPU. To acquire floating-point capacities on this chip, you had to add an 80487 chip.

Each processor in the 80x86 families has a corresponding coprocessor with which it is compatible. 8087 is a coprocessor used for the 8086 processor. It consists of a control unit and an execution unit. CU is used to synchronize the operation of the coprocessor and the processor. The numeric execution unit performs all operations that access and manipulate the numeric data in the coprocessor's registers.

Pipelining: Every task is split into a series of pieces in pipelining and every piece is tackled by a separate (specialized) functional unit. Figure 4.3depicts an example of the fundamental difference between implementing four subtasks of a specified instruction utilizing pipelining and sequential processing. In this case, the four subtasks may be fetching (F), decoding (D), execution (E), and writing the results (W).

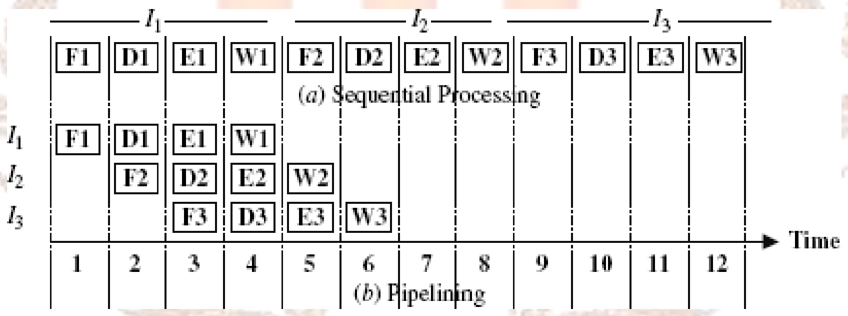


Fig 4.3: Pipelining vs. Sequential Processing

It is obvious from the figure that in case four-stage pipelining is utilized, the total time needed to process three instructions (I1, I2, I3) is just six-time units. On the contrary, sequential processing is utilized, 12-time units are required.

Pipeline tries to boost instruction output by overlapping the implementation of multiple instructions. Pipelining provides remarkable speedup.

Self-Assessment Questions – 2

3. Intel 8086 consists of___bit address bus.
4. 8086 is a 16-bit microprocessor designed by IBM. (True/ False)
5. 8087 is a coprocessor used for the 8086 processor. (True/ False)

4. PROGRAMMER'S MODEL OF 8086

Let us know that the programmer's model of 8086 is the most important concept to study in the 8086 model. Figure 4.4 shows the register the organisation of 8086. This is termed the Programmer's model of 8086. The registers shown in the programmer's model are the ones accessible to the programmer. The register set is the most visible component of the CPU. The 80x86 chips have a set of on-board powerful set of registers.

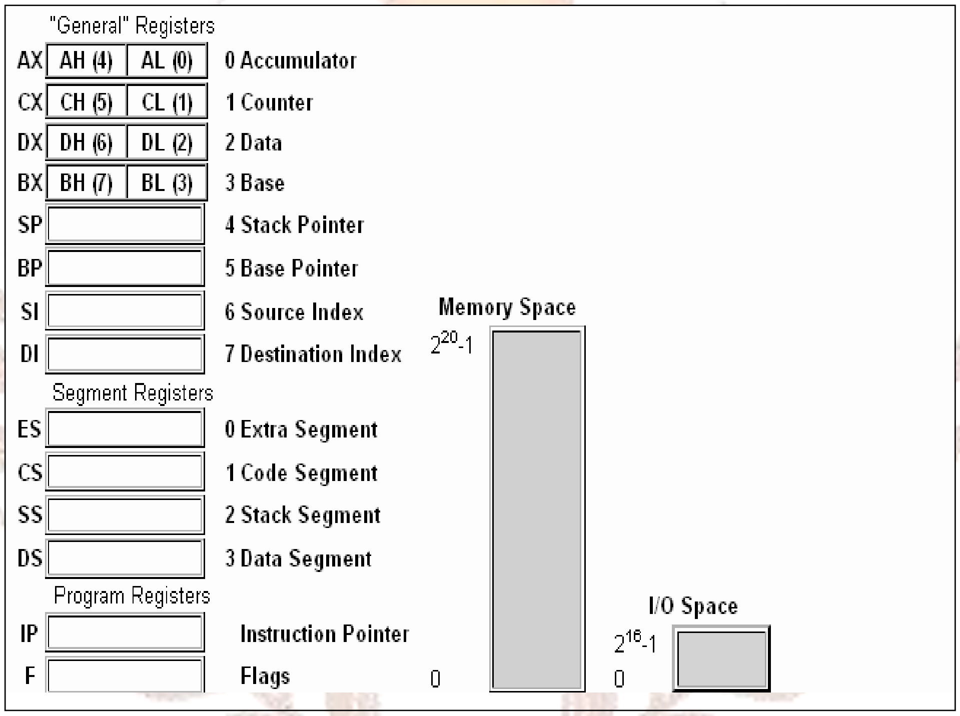


Fig 4.4: Register Organisation of 8086

8086 register set includes:

- General-purpose registers
- Segment registers
- Pointers and index registers
- Flag registers

Now let us study the programmer model in detail:

1. The general-purpose registers are those registers which may seem like operands of the logical, arithmetic, and associated instructions. There are 16 general purpose registers which can be split into two 8-bit registers. The general-purpose registers have been

named as AX, BX, CX, and DX. These registers can be either lower order (L) and High order (H).

2. The 8086 processor utilizes the segment registers to retrieve blocks of memory known as segments. Since there are four 4 memory segments in an 8086 model i.e. code, data, stack, and extra segments therefore there are four segment registers to hold the upper 16-bit addresses of the four memory segments.
3. Pointers and index registers are useful in addressing functions. Since the segment registers are 16-bits but the address bus requires 20 bits, hence pointer registers IP, BP and SP are used to hold the offset within the code, data, and stack segments respectively. The index registers SI and DI are used for offset storage in certain types of addressing modes. They can be utilized as general-purpose registers as well.
4. The flags register is a 32-bit register called EFLAGS. The flags operate particular operations and depict the status of the 80386.
5. The last class of 8086 registers is the special/miscellaneous registers such as control registers, debug registers, test registers, descriptor registers, task registers, and model-specific registers.

Activity 1

Find out more about the coprocessor used for 8086 model.You may use the following links:

http://train-srv.manipalu.com/wpress/?p=15507

http://nptel.iitm.ac.in/courses/Webcourse-contents/IISc-BANG/Microprocessors%20and%20Microcontrollers/pdf/Teacher_Slides/mod4/M4L1.pdf

Self-Assessment Questions – 3

6. The most noticeable component of the CPU is the____________.set.
7. There are four major types of registers set in 8086 model. (True/False)

5. MAX/MIN MODE

You must know that the 8086 can be organised to operate in either of two modes:

- Minimum Mode
- Maximum Mode

Minimum mode is the first of two different hardware modes of the Intel 8088 and 8086 processors (CPU). The other is the maximum mode. Mode selection is acquired by the way the chip is hard-wired in the circuit. Particularly, pin #33 (MN/MX) is utilized to choose the mode, depending on whether it is wired to the ground or to voltage.

5.1 Minimum Mode

The Minimum mode is utilized for a single processor system, where 8086/8088 directly produces all the essential control signals. It is acquired by linking the mode selection pin MN/MX to +5.0V (Refer figure 4.5).

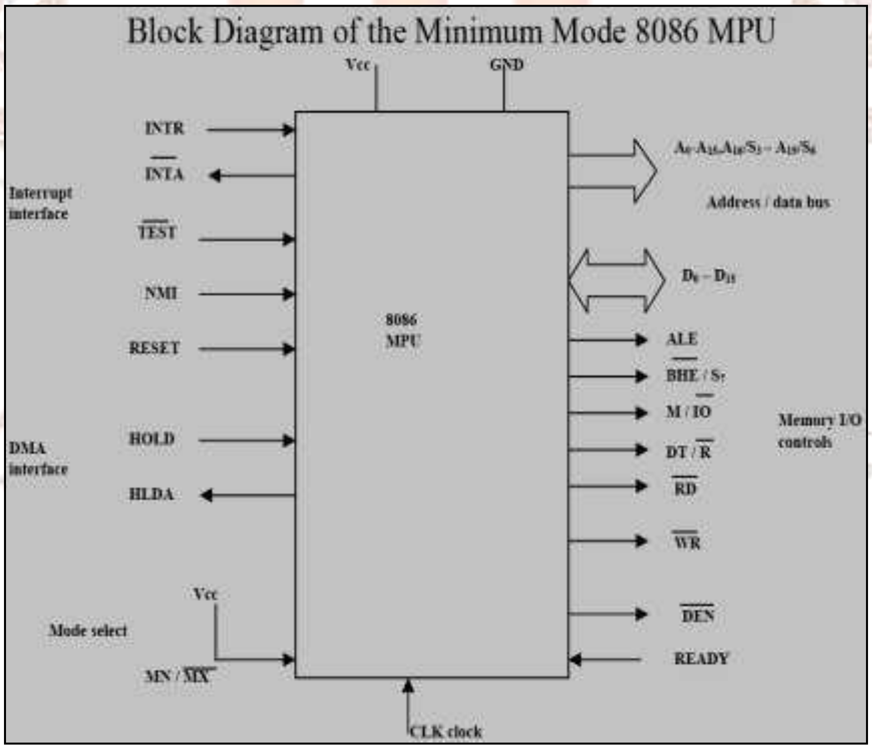


Fig 4.5: Block Diagram of the Min Mode 8086 MPU

In such a mode, all the control signals are distributed by the microprocessor chip itself. A single microprocessor is present in the minimum mode system. The rest of the components in the system are clock generator, latches, transceivers, memory, and I/O devices. Latches are usually buffered output D-type flip-flops such as 74LS373 or 8282. Transceivers are the bidirectional buffers.

5.2 Maximum Mode

Maximum mode is planned for multiprocessor systems, where an extra“Bus- controller” IC is needed to produce the control signals. (Refer figure 4.6) The processors monitor the Bus-controller utilizing status codes. Maximum mode operation is new and specially designed for the operation of the 8086 arithmetic coprocessor. It is selected by grounding MN/MX. It is different from a minimum as few of the control signals should be externally produced. In this way, it requires an external bus controller, an 8288 bus controller like the 8086 arithmetic coprocessor.

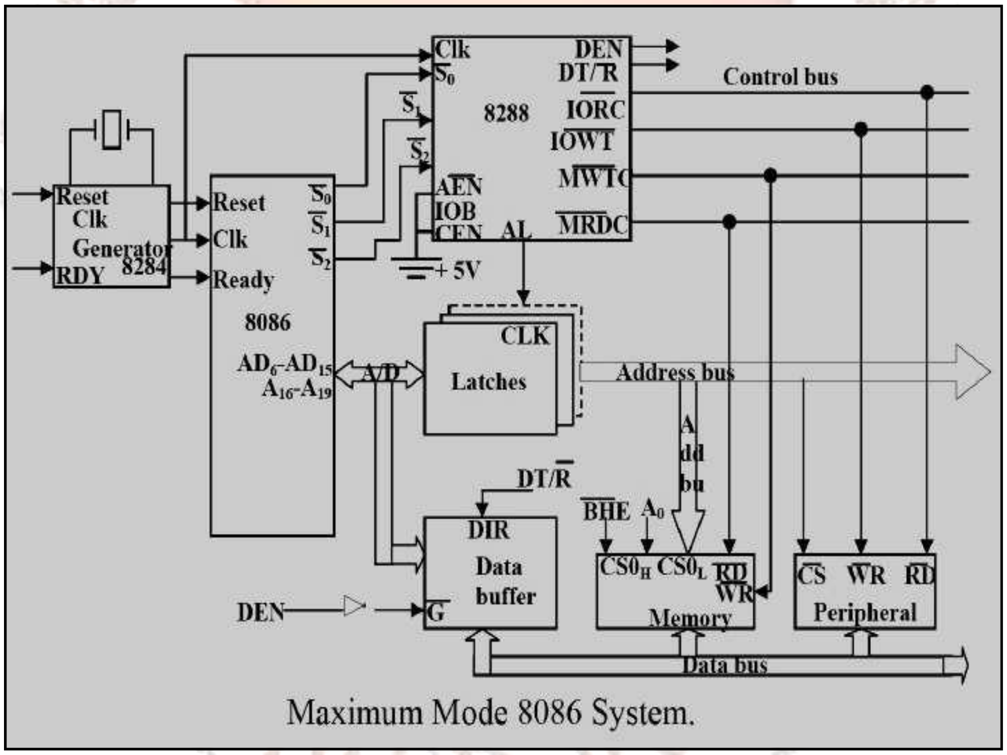


Fig 4.6: Block Diagram of the Max Mode 8086 MPU

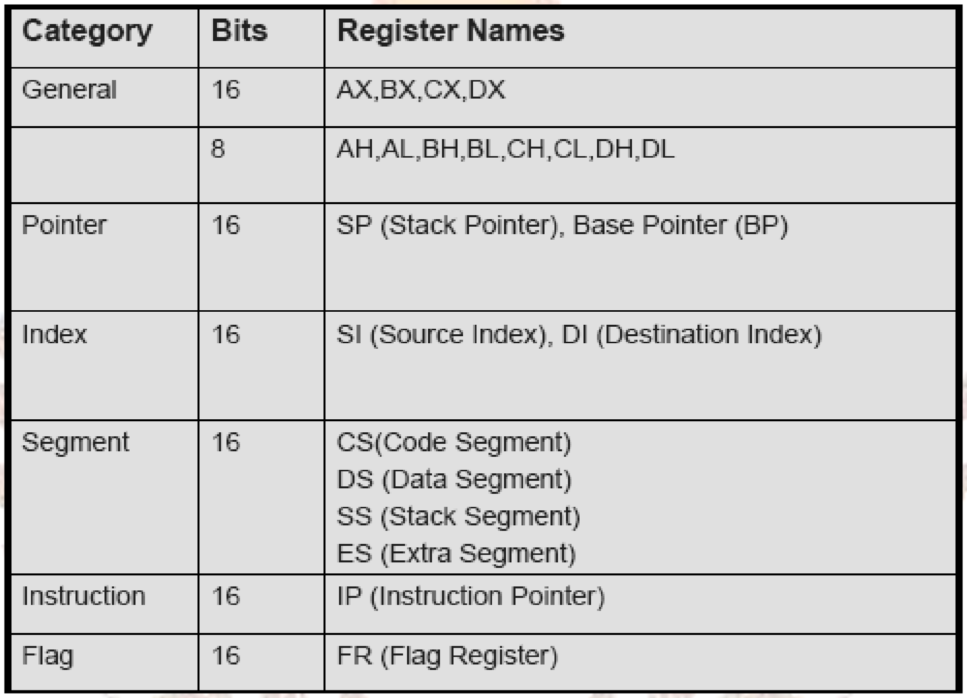
Self-Assessment Questions – 4

8. A single microprocessor is present in the____________mode system.
9. The 8086 can be organised to work in any of the two modes.(True/False)

6. REGISTER ORGANISATION

In this section, you will learn about the register organization of the Intel 8086 microprocessor. Here we will discuss the role and functions of the various registers in detail. Table 4.1 below summarises the register organization of the 8086 model.

Table 4.1: Register Organisations of 8086



Category	Bits	Register Names
General	16	AX,BX,CX,DX
	8	AH,AL,BH,BL,CH,CL,DH,DL
Pointer	16	SP (Stack Pointer), Base Pointer (BP)
Index	16	SI (Source Index), DI (Destination Index)
Segment	16	CS(Code Segment) DS (Data Segment) SS (Stack Segment) ES (Extra Segment)
Instruction	16	IP (Instruction Pointer)
Flag	16	FR (Flag Register)

6.1 8086 General Purpose Registers

8086 processor consists of eight 16-bit general-purpose registers which are named as **ax, bx, cx, dx, si, di, bp,** and **sp.** Whereas you are able to utilize most of such registers in an exchanged way in a computation, most of the instructions perform more competently or completely need a specific register from such a group.

- The **ax** register is known as the **accumulator**. It is a register where the majority of the logical and arithmetic computations occur.
- The **bx** register stands for base register. It also has few particular uses as an accumulator register. It is mostly utilized to contain indirect addresses.

- The **cx** register stands for count register. cx register is commonly used to state the number of characters in a string or to add up the number of iterations in a loop.
- The **dx** register means the data register. The Data register serves two main principles: firstly it carries the overflow from specific arithmetic processes and secondly that it contains I/O addresses at the time of retrieving data on the 80x86 I/O bus.
- The **di** and **si** registers stand for the Destination Index and Source Index registers respectively. Such registers are utilized as pointers (quite similar to the **bx** register) to not access memory directly.
- The **bp** register means the Base Pointer. It is much like the **bx** register.
- The **sp** register stands for the Stack Pointer. Its main purpose is to maintain the program stack.

Besides these eight 16-bit registers, the 8086 CPUs also possess eight 8-bit registers (lower-order (LO) and high order (HO) byte registers).

The figure 4.7 below shows the general-purpose register set.

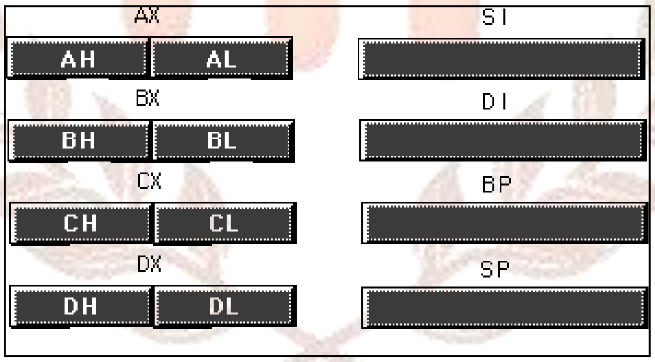


Fig 4.7: General Purpose Register Set

You may note that an independent register set is not formed by the eight-bit registers. Changing all will modify the value of **ax**; so will changing **ah.** The value of al precisely matches with bits 0 through 7 of **ax.** The value of ah matches with bits 8 through 15of **ax.** Hence, any changes to **al** or **ah** will change the value of **ax.** Likewise changing **ax** will also modify both **al** as well as **ah**. Remember, however, that modifying al will not influence the value of **ah**, and vice versa. Such a statement likewise applies to **bx/bl/bh, cx/cl/ch**, and **dx/dl/dh** also. The **si, di, bp,** and **sp** registers depicted in figure 4.6 are just 16-bits. There is no method to directly retrieve the individual bytes of such registers just as you are able to retrieve the high and low order bytes of **ax, bx, cx**, and **dx** registers.

6.2 8086 Segment Registers

The 8086 has four exclusive **segment registers** known as **cs, ds, es**, and **ss**. **cs** stand for Code Segment, **ds** stands for Data Segment, es stands for Extra Segment, and **ss** stands for Stack Segment, respectively. All these registers are 16 bits wide. They handle choosing blocks (segments) of main memory. A segment register (e.g., cs) aims at the starting of a segment in memory.The maximum length of the 8086 segment register is 65,536 bytes. The "64K segment limitation" has proved to be difficult for many programmers. The cs register is utilized to aim at the segment holding the present machine instructions to be implemented. Remember that, in spite of the 64K segment limitation, 8086 programs can have a greater length than 64K. You just require multiple code segments in memory. As the users are able to change the value of the cs register, you have the option to switch to a new code segment whenever you which to implement the code placed there.

ds, the data segment register, is commonly used to aim at the program's global variables. As similar to the above cs register, you're restricted to 65,536 bytes of data available in the data segment; however, you can anytime modify the value of the ds register to retrieve extra data in the remaining segments.

Thees stands for extra segment register. 8086 programs frequently utilize this segment register to attain access to segments as soon as it becomes impossible or problematic to change the other segment registers.

The ss register aims at the segment holding the 8086 **stacks**. The stack is a block of memory locations booked to perform as temporary memory. The stack essentially is a memory unit containing an address register which can just count (once an initial value is stacked into it). The **stack** is placed where the 8086 saves procedure parameters, significant machine state information, local variables, and subroutine return addresses.

6.3 8086 Special Purpose Registers

8086 processor consists of two special purpose registers; the flags register and the instruction pointer (ip). These registers cannot be accessed the same way as other 8086 registers. Rather, the CPU usually directly controls these registers. The **ip** register of the 8086 processor is equal to the **ip** register on the x86 processors because it holds the address of the recently executing instruction. ip register is a 16-bit register that offers a pointer into the recent code segment (16 bits allows you to select any one of 65,536 varied memory positions).

The flags register is quite different from the other registers on the 8086. The other registers contain 8or 16-bit values while the flags register holds 1-bit value that assist in deciding the present condition of the processor, the 8086 utilizes only nine out of the 16 bits possessed by the flags register. Out of these flags, you can utilize four flags every time: carry, zero, overflow, and sign. Such four flags are called the 8086 **condition codes**. The flags register are shown below in figure 4.8.

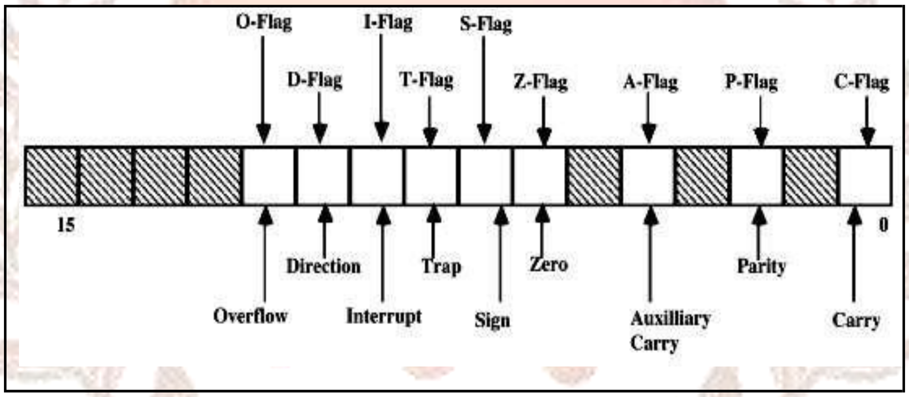


Fig 4.8: 8086 Flag Register Set

The functions of various flag registers are given below in the table:

Carry Flag (CF): As soon as there is an unsigned overflow, it is positioned to 1.

Parity Flag (PF): As soon as there exists an even number of 1-bit in the outcome, as well as to 0 at the time there exists an odd number of 1-bit, it is positioned to 1.

Auxiliary Flag (AF): As soon as there is an unsigned overflow for a low nibble (4 bits), it is positioned to 1

Zero Flag (ZF): As soon as the result is zero, it is positioned to 1. For non-zero outcomes, this flag is positioned to 0.

Sign Flag (SF): As soon as the result is negative, it is positioned to 1. As soon as the result is positive, it is positioned to 0. (This flag acquires the value of the most important bit)

Trap Flag (TF): It is utilized for on-chip debugging.

Interrupt enable Flag (IF): As soon as this flag is positioned to 1, the CPU responds to interrupts from external devices.

Direction Flag (DF): If the processing is conducted forward, this flag is positioned to 0. When the processing is conducted backward; this flag is positioned to 1

Overflow Flag (OF): As soon as there is a signed overflow, this flag is set to 1.

Self-Assessment Questions – 5

10. There are two special- purpose registers on the 8086 CPU: the ____________ register and the instruction pointer (IP).
11. The 8086 has four special segment registers. (True/False)
12. How many general-purpose registers are there in 8086?
    a) 8
    b) 16
    c) 24
    d) 12

7. INSTRUCTION CYCLES

Now let us study about instruction cycles. An instruction cycle means the time duration in which a single instruction is obtained from memory and implemented as soon as a computer is provided with instruction in machine language. It is also called as a fetch-and-execute cycle, fetch-decode-execute cycle, or FDX). It is the procedure through which a computer obtains program instruction from its memory, chooses the type of actions the instruction needs, and thereafter transmits out those actions. The instruction cycle constantly operates from the time the computer starts to the time the computer is shut down.

The numerous registers utilized in the CPU during the cycle are:

- **Program Counter (PC):** The program counter, PC, is a special-purpose register which is utilized by the processor to contain the address of the subsequent instruction to be implemented. As each instruction is implemented, PC is automatically incremented.
- **Memory Address Register (MAR):** The MAR saves the physical memory address on which the subsequent instruction is placed or the subsequent piece of data will be written. It contains the address of a memory block to be written to or read from.
- **Memory Data Register (MDR):** Memory Data Register is a two-way register which contains data accessed from memory (and all set for the CPU to process) or data awaiting storage in memory. Hence, MDR can stack its data from:
    i. One of the CPU registers (for storing data)
    ii. The data bus (for reading data)
- **Instruction register (IR):** The instruction register stores the instruction presently being executed.
- **Control Unit (CU):** After choosing machine resources like a specific arithmetic operation and a data source register, the CU is utilized to decode the program instruction in the IR. It synchronizes the initiation of those resources.
- **Arithmetic logic unit (ALU):** ALU works on the logical and operations.

Every instruction cycle is further divided into a series of sub phases or cycles as shown in figure 4.9.

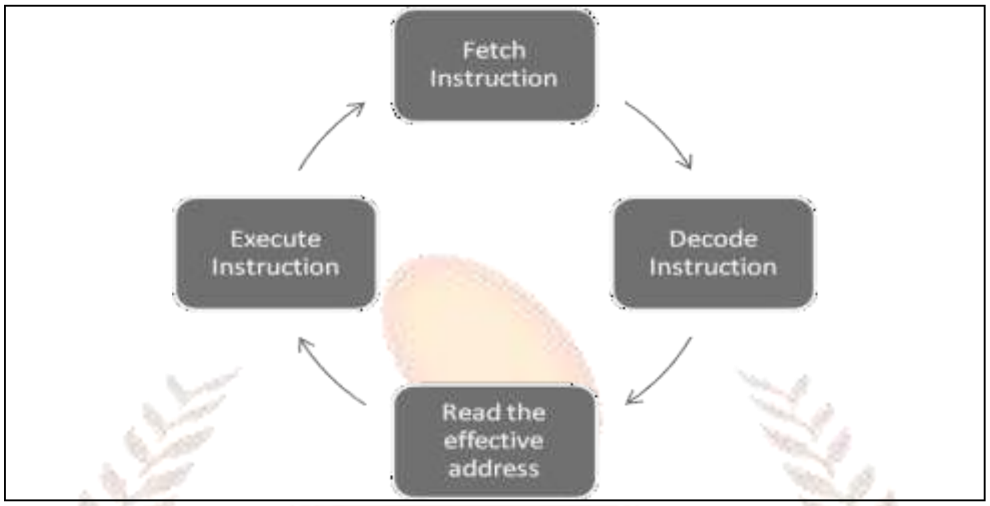


Fig 4.9: Instruction Cycle

Every instruction cycle in the basic computerhas the parts as mentioned below:

1. **Fetch the instruction:** Fetching the instruction to be implemented is included in the first phase of an instruction cycle. The subsequent instruction is extracted from the memory address which is at present saved in the Program Counter (PC), as well as saved in the Instruction Register (IR).
2. **Decode the instruction:** The decoder interprets the instruction when the instruction has been obtained. In this phase, the instruction present in the instruction register (IR) gets decoded.
3. **Read Effective Address:** When the instruction contains an indirect address, the actual address is read from the main memory, and any needed data is obtained from the main memory and thereafter positioned in data registers (Clock Pulse: T3).
4. **Execute the instruction:** The Control Unit of CPU forwards the decoded information in the form of a sequence of control signals to the applicable functional units of the CPU to perform the actions needed by the instruction like reading values from registers, and forwarding them to the ALU to conduct logical or mathematical functions on them as well as writing the outcome back to a register.

Subsequent to the end of step 4, the control returns to step 1 to fetch, decode and execute the subsequent instruction. This process carries on till a HALT instruction comes across.

Self-Assessment Questions – 6

13. ____________is an incrementing counter that maintains trackof the memory address of which instruction is to be implemented subsequently.
14. Which register is utilised for acting as a base for the instruction that hasjust been obtained from memory?
    a) Instruction register
    b) Memory address register
    c) Memory data register
    d) All of the above

8. READ WRITE CYCLES

After understanding of the instruction cycle of the processor, now let us learn about the Read-Write cycles. **Read Cycles** Data Transfer from Memory or Input/Output (I/O) to CPU. **Write Cycle** is Data Transfer from CPU to Memory or I/O.

8.1 Read Cycle

Read Cycle consists of several steps (see figure 4.10) which are discussed below.

1) The processor starts a read bus cycle by floating the address of the memory location on the address lines.
2) After the address lines are steady, the processor declares the address strobe signal on the bus. The address strobe signals are utilized to verify the validity of the address lines.
3) The Processor thereafter positions the Read/Write signal to high, i.e. read.
4) Subsequently, the processor declares the data strobe signal. Data strobe gives a signal to the memory that the processor is all set to read data.
5) The memory subsystem decodes the address and puts the data on the data lines.
6) The memory subsystem thereafter declares the data acknowledge signal. Data acknowledgment is utilised to signal to the processor that valid data can at this point be latched in.
7) The processor latches in the data and cancels outthe data strobe. This gives a signal to the memory that the data has been latched by the processor.
8) The processor also cancels out the address strobe signal.
9) The memory subsystem now cancels out the data acknowledgment signal. This acts as a signal to the termination of the read bus cycle.

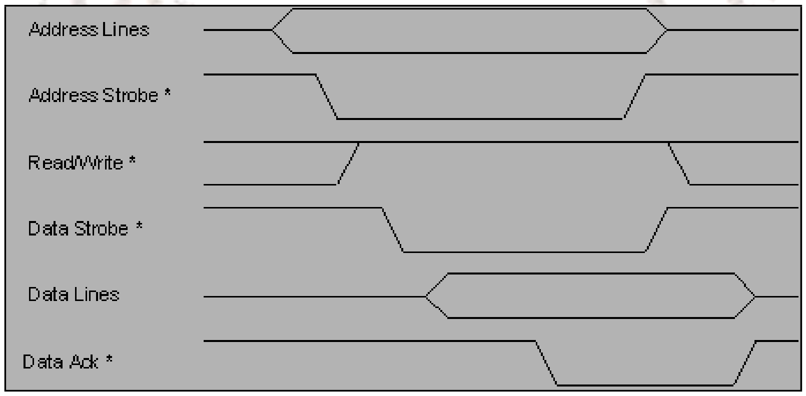


Fig 4.10: Read Cycle

8.2 Write Cycle

The various steps included in the Write Cycle are as given below:

1. The processor starts a write bus cycle by floating the address of the memory position on the address lines.
2. After the address lines are steady, the processor declares the address strobe signal on the bus. The address strobe gives a signal to the validity of the address lines.
3. Processor thereafter positions the Read/Write signal to low, i.e. write.
4. After this, the processor puts the data on the data lines.
5. At this point, the processor declares the data strobe signal. This provides a signal to the memory that the processor contains valid data needed for the memory write operation.
6. The memory subsystem decodes the address and writes the data into the addressed memory location.
7. The memory subsystem thereafter declares the data acknowledge signal. This gives a signal to the processor that data has been written to the memory.
8. Then the processor cancels out the data strobe, giving the signal that the data is not valid anymore.
9. The processor also cancels out the address strobe signal.
10. The memory subsystem at this point cancels out the data acknowledgment signal, giving the signal for termination of the write bus cycle.

Self-Assessment Questions – 7

15. The processor sets the Read/Write signal to low for a ___________ operation.
16. Processor starts a______________cycle by floating the addressof the memory location on the address lines.

9. ADDRESSING MODES

We all know that an instruction may perform upon innumerable operands Addressing mode stresses a rule for changing or interpreting the address field of the instruction prior to the actual reference of the operand.

Addressing mode is a code which informs the control unit on the way to acquire the efficient address from the displacement. The process by which the operands are selected during program execution depends upon the addressing mode of the instruction

The operand is the portion of a machine instruction which references a peripheral device or data. It can be of three types:

- Implicit
- Explicit
- Both Implicit and Explicit

Implicit operands denote that the instruction by definition has some particular operands. The programmers do not choose these operands. Explicit operands signify the instruction functions on the operands indicated by the programmer. The position of an operand value in memory space is known as the Effective Address (EA).

The various 8086 addressing modes can be divided into groups as given below:

1. Implied Mode
2. Immediate Mode
3. Direct Addressing Mode
4. Indirect Addressing Mode
5. Register Addressing Mode
6. Register Indirect Mode
7. Relative Addressing Mode
8. Indexed Addressing Mode

Now, let us understand these modes in detail.

Implied Mode

The operands are indicated perfectly in the meaning of the instruction in this mode. There is no operand in the instruction.

Example: CLC which clears the carry flag to zero.

Immediate Mode

The operand is indicated in the instruction itself in this mode. The address part in instruction is not memory/register address but the data or number (constant) to be processed.

Example:

MOV AX, 2550H (Moves the 16-bit data 2550H into AX) MOV CX, 0625H (Movesthe 16-bit data 0625H into CX)

Effective Address (EA) is the memory address attained from the computation stated by the specified addressing mode. It is the actual address where the data is located in memory. The instruction (IR) contains the operand without referring to memory.

Direct Addressing Mode

The effective address is equivalent to the address portion of the instruction. The operand exists in memory and its address is offered directly by the address portion of the instruction. Normally used in branch-type instruction.

Example:

MOV DL, [2400]; move contents of DS: 2400H into DL Effective address = address portion of an instruction.

Indirect addressing mode

The address field of the instruction provides the address in which the effective address is saved in memory. The control extracts the instruction from memory as well as utilizes its address portion to retrieve memory one more time to read the data.

This mode is used for an array in which the content of the address in the instruction is a pointer to an array of data.

Example:

MOV CL,	[SI]	(move contents of DS: SI into CL)
MOV [DI],	AH	(move the contents of DS: SI into CL)
MOV [SI],	AX	

Register addressing mode

The address field of the instruction means that a register holds the effective address of the operand. One memory reference is utilized by register addressing mode to acquire the operand. In the register addressing mode, the operands may be:

- reg16: 16-bit general registers: AX, BX, CX, DX, SI, DI, SP or BP.
- reg8: 8-bit general registers: AH, BH, CH, DH, AL, BL, CL, or DL.
- Sreg: segment registers: CS, DS, ES, or SS.

Note: There is an exception: CS cannot be a destination.

For register addressing modes, there is no need to compute the effective address. The operand is in a register and to get the operand there is no memory access involved.

Example:

MOV AX, BX;

Register Indirect Mode

The instruction indicates a register in the CPU whose content offers the address of data in memory or the effective address. (See figure 4.11)This mode is utilized for an array in which the effective address is a pointer. The advantage of utilizing such a mode is that the address field of the instruction utilizes lesser bits to choose a register than bits of memory.

Example:

MOV A, M

SUB M

DCR M

MOV A, M instruction will move the contents of the memory location to the accumulator.

M represents the address present in the H-L register pair. So when MOV A, M is executed, the contents of the address specified in H-L register pair are moved to the accumulator, as shown in the figure given below.

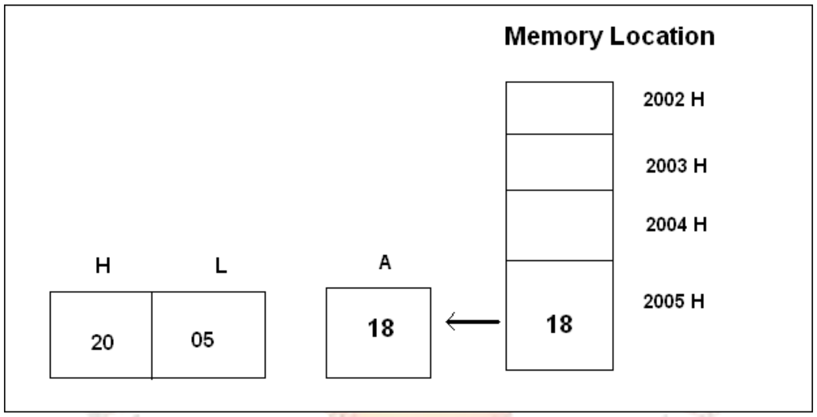


Fig 4.11: Pictorial Representation of Register Indirect Mode

Relative Addressing Mode

The program counter's content is combined to the address portion of the instruction in order to attain the effective address.

Effective address = Program Counter [PC] + address part in instruction.

Example:

12	PC

It is often used with branch-type instruction as soon as the branch address is in the region adjacent to the instruction word only. The address part in the instruction is actually the displacement needed from the instruction to the data.

Indexed Addressing Mode

The index register's content is combined with the address portion of the instruction to acquire the EA in this mode. Effective address = content of index register + address in instruction.The index register is a special CPU register that contains an index value. This mode is used for an array and the address field of the instruction decides the starting address of the data array in memory.

Example:

3 XR

TABLE = 185←beginning of array

Activity 2

Using internet find out some more types of addressing modes used in8086 model apart from those mentioned in this unit.

You may make use of following links:

http://www.dcc.unicamp.br/~celio/mc404s2-03/addr_modes/intel_addr.html

http://www.angelfire.com/elx215/ELX215-3a.pdf?q=segi

Self-Assessment Questions – 8

17. For register addressing modes, there is no need to compute theeffective address. (True/False)
18. __________forms of this addressing mode are present on the 8086.
19. Intel means the [si] and [di] addressing modes like _____________ addressing modes.
20. The based indexed addressing modes are just arrangements of the __________addressing modes.

10. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- The 8086 contains fourteen 16-bit registers.
- The 8086 has two operating modes: the minimum mode and the maximum mode.
- The minimum mode is chosen by employing logic 1 to the MN / MX# input pin. The maximum mode is chosen by employing logic 0 to the MN / MX# input pin.
- The instruction cycle is divided into three main phases.
- There are mainly 12 addressing modes in the 8086 model, which can be classified into five groups.

11. GLOSSARY

Instruction cycle: A procedure through which a computer accesses a program instruction from its memory, decides the type of actions that the instruction needs, and performs those actions.

Addressing mode: The addressing mode states rule for changing or interpreting the address field of the instruction prior to the actual reference of the operand.

Shift Register: It has the ability to shift its binary information either to the left or to the right and is therefore called a shift register.

12. TERMINAL QUESTIONS

1. Explain the programmer's model of 8086.
2. Describe various data addressing modes of 8086 with the help of examples.
3. Discuss the register organisation of 8086.
4. Explain various Addressing Modes
5. Explain in detail the instruction cycle of a processor along with the circuits used.
6. Discuss briefly the read and write cycle of an 8086 processor.
7. In how many modes can 8086 microprocessor operate? Briefly describe.

13. ANSWERS

Self-Assessment Questions

1. Single Instruction stream, Multiple data stream
2. True 3. 20
3. True
4. True
5. Register
6. False
7. Minimum
8. True
9. Flag
10. False 12. 16
11. Program Counter(PC)
12. Instruction Register(IR)
13. Write
14. Read bus
15. True
16. Four
17. Indexed
18. Register indirect

Terminal Questions

1. Programmers model of 8086 refers to the basic register organisation of 8086.Refer to section 4
2. There are mainly four types of data addressing modes in 8086 model i.e. Register, Immediate, Memory and I/O. Refer to section 9.
3. The 8086 contains fourteen 16-bit registers. Refer to section 6.
4. a) The operand is indicated in the instruction itself in Immediate Addressing mode. Refer to section 9.

   b) Register addressing mode utilises a single memory reference to acquire the operand. Refer to section 9.
5. The instruction cycle refers to the fetch – execute cycle. Refer to section 7.

6. Read and Write cycle consists of a series of steps. Refer to section 8.
7. An 8086 microprocessor can operate in two modes: max and min mode. Refer to section 6.

References:

- Sajjan G. Shiva; *Computer Design and Architecture*; Marcel Dekker
- Silvia Melitta Mueller, Wolfgang J. Paul; *Computer Architecture*; Springer
- Joseph D. Dumas II; *Computer Architecture*; CRC Press
- Nicholas P. Carter; *Schaum's outline of computer Architecture*; Mc. Graw-Hill Professional

E-references:

- nptel.iitm.ac.in/courses/Webcourse-contents/IISc.../M1L3.pdf
- http://www.csi.ucd.ie/staff/jcarthy/home/alp/alp-05.pdf