UNIT - I

1 8086 16-Bit Microprocessor

1.1 Introduction

- In 1978, Intel came out with the 8086 processor. The Intel 8086 is a 16-bit microprocessor, implemented in N-channel, depletion load, silicon gate technology (HMOS) and packaged it in a 40 pin dual in line package.

1.2 Features of 8086 MSBTE : Winter-16,17, Summer-17

1. **The 8086 is a 16-bit microprocessor.** The term "16-bit" means that its arithmetic logic unit, internal registers and most of its instructions are designed to work with 16-bit binary words.
2. **The 8086 has a 16-bit data bus,** so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time.
3. **The 8086 has a 20-bit address bus,** so it can directly access 2^{20} or 10,48,576 (1 MB) memory locations.
4. The 8086 can generate **16-bit I/O address,** hence it can access 2^{16} = 65536 I/O ports.
5. The 8086 provides **fourteen 16-bit registers.**
6. The 8086 has **multiplexed address and data bus** which reduces the number of pins needed, but does slow down the transfer of data (drawback).
7. The Intel 8086 is designed to operate in two modes, namely the **minimum mode** and the **maximum mode**. When only one 8086 CPU is to be used in a microcomputer system, the 8086 is used in the minimum mode of operation. In this mode the CPU issues the control signals required by memory and I/O devices. In multiprocessor (more than one processor in the system) system 8086 operates in maximum mode. In maximum mode, control signals are generated with the help of external bus controller (8288).
8. The Intel 8086 **supports multiprogramming.** In multiprogramming, the code for two or more processes is in memory at the same time and is executed in a time-multiplexed fashion.
9. An interesting feature of the 8086 is that it **fetches upto six instruction bytes** from memory and queue stores them in order to speed up instruction execution.

Board Questions

1. *List the features of 8086 microprocessor.*
2. *List any four features of 8086.*

MSBTE : Winter-16, 17, Summer-17, Marks 2

1.3 8086 Architecture

MSBTE : Winter-15, 16, 17, Summer-15, 16, 17, 18

- Fig. 1.3.1 shows a block diagram of the 8086 internal architecture.
- It is internally divided into two separate functional units. These are the Bus Interface Unit (BIU) and the Execution Unit (EU).
- These two functional units can work simultaneously to increase system speed and hence the throughput.
- Throughput is a measure of number of instructions executed per unit time. (See Fig. 1.3.1 on next page)

1.3.1 Bus Interface Unit (BIU)

- The bus interface unit is the 8086's interface to the outside world.
- It provides a full 16-bit bi-directional data bus and 20-bit address bus.

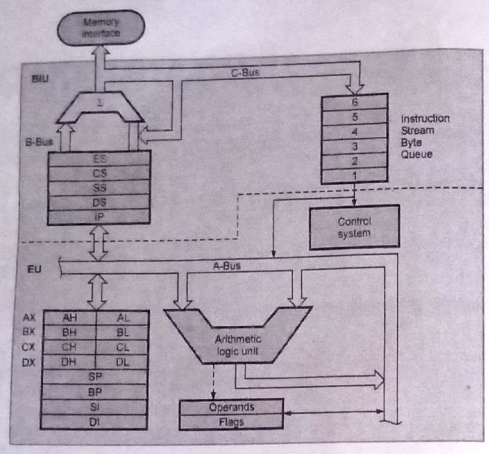


Fig. 1.3.1 8086 internal block diagram

- The bus interface unit is responsible for performing all external bus operations, as listed below.

Functions of Bus Interface Unit

1. It sends address of the memory or I/O.
2. It fetches instruction from memory.
3. It reads data from port/memory.
4. It writes data into port/memory.
5. It supports instruction queuing.
6. It provides the address relocation facility.

- To implement these functions the BIU contains the instruction queue, segment registers instruction pointer, address summer and bus control logic.

1.3.2 Execution Unit (EU)

- The execution unit of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains
  - Control circuitry
  - Instruction decoder
  - Arithmetic Logic Unit (ALU)
  - Register organisation
    - Flag register
    - General purpose registers
    - Pointers and index registers

Control Circuitry, Instruction Decoder, ALU

- The control circuitry in the EU directs the internal operations.
- A decoder in the EU translates the instructions fetched from memory into a series of actions which the EU performs.
- ALU is 16-bit. It can add, subtract, AND, OR, XOR, increment, decrements, complement and shift binary numbers.

transducers - A transducer is a device that converts energy from one form to another form. (energy).

Microprocessors Passive transducers 1-3 Thermocouple, photo diode 8086 16-Bit Microproce

1.3.3 Concepts of Pipelining

- To speed up program execution, the BIU fetches **six instruction bytes** ahead of time from the memory.
- These prefetched instruction bytes are held for the execution unit in a group of registers called **queue**.
- With the help of queue it is possible to fetch next instruction when current instruction is in execution.
- The BIU continues this process as long as the queue is not full.
- Due to this, execution unit gets the ready instruction in the queue and instruction fetch time is eliminated. This is illustrated in Fig. 1.3.2.
- The queue operates on the principle First In First Out (FIFO) so that the execution unit gets the instructions for execution in the order they are fetched. Thus it is also known as **opcode fetch FIFO buffer**.
- In case of JUMP and CALL instructions, instruction already fetched in queue are of no use. Hence, in these cases queue is dumped and newly formed by loading instructions from new address specified by JUMP or CALL instruction.
- Feature of fetching the next instruction while the current instruction is executing is called **pipelining**.

Board Questions

1. *Draw architecture of 8086 and label it. Write the functions of BIU and EU.* **MSBTE : Winter-15, Summer-15,16, Marks 8**
2. *What is pipelining ? State its need and how it is done in 8086.* **MSBTE : Winter-15, Marks 8**
3. *What is pipelining ? How it is implemented in 8086 microprocessor.* **MSBTE : Summer-16, Marks 2**
4. *Draw the functional block diagram of 8086 microprocessor and describe instruction queue in detail.* **MSBTE : Winter-16, Marks 8**
5. *Explain the concept of pipelining in 8086 microprocessor with diagram.* **MSBTE : Summer-17, Marks 4**
6. *Describe concept of pipelining in 8086.* **MSBTE : Winter-17, Summer-17, Marks 4**
7. *State the advantages of pipeline architecture.* **MSBTE : Summer-18, Marks-4**
8. *Explain pipelining in 8086 microprocessor. How is queuing useful in speeding up the operation of 8086 microprocessor.* **MSBTE : Summer-15, Marks 4**

1.4 Register Organization (8086 Programming Model)

MSBTE : Winter-15,16,17, Summer-15,16,17,18

- The 8086 has a powerful set of registers.
- It includes general purpose registers, segment registers, pointers and index registers, and flag register.

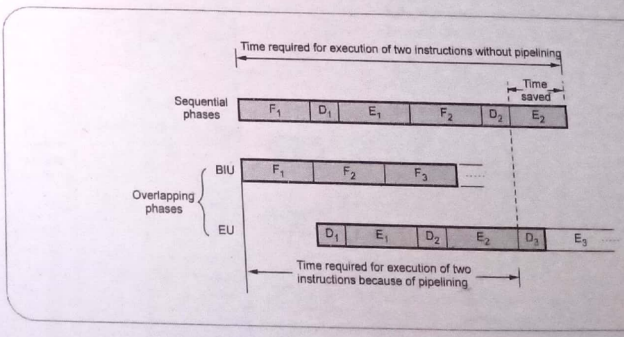


Fig. 1.3.2 Pipelining

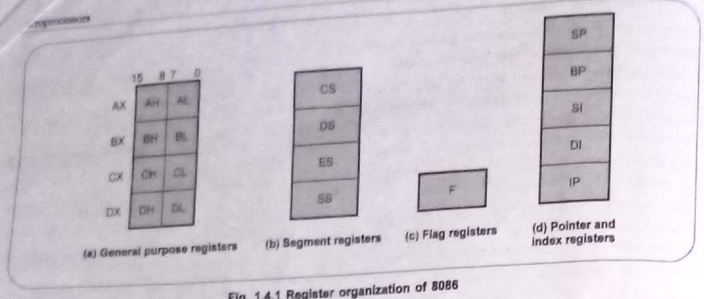


Fig. 1.4.1 Register organization of 8086

- The Fig. 1.4.1 shows the register organization of 8086. It is also known as **programmer's model of 8086.**
- The registers shown in programmer's model are accessible to programmer.
- As shown in the Fig. 1.4.1, all the registers of 8086 are 16-bit registers.

1.4.1 General Purpose Registers

- The 8086 has four 16-bit general purpose registers labeled AX, BX, CX and DX. Each 16-bit general purpose register can be split into two 8-bit registers.
- The letters L and H specify the lower and higher bytes of a particular register. For example, BH means the higher byte (8-bits) of the BX register and BL means the lower byte (8-bits) of the BX register. The letter X is used to specify the complete 16-bit register.
- The general purpose registers are either used for holding data, variables and intermediate results temporarily.
- They can also be used as a counters or used for storing offset address for some particular addressing modes.
- The register AX is used as 16-bit accumulator whereas register AL (lowerbyte of AX) is used as 8-bit accumulator.
- The register BX is also used as offset storage for generating physical addresses in case of certain addressing modes.
- The register CX is also used as a default counter in case of string and loop instructions.
- The DX register is concatenated with AX (DX:AX) to form 32-bit register for some MUL and DIV operations. It is also used to specify port address in IN and OUT operations.

1.4.2 Segment Registers

- The physical address of the 8086 is 20-bits wide to access 1 Mbyte memory locations. However, its registers and memory locations which contain logical addresses are just 16-bits wide. Hence 8086 uses memory segmentation.

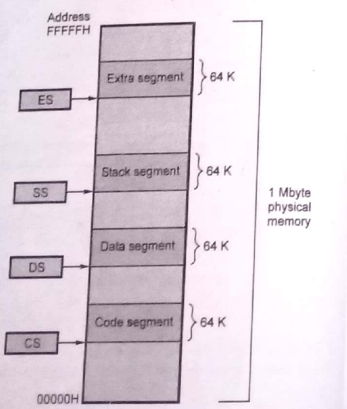


Fig. 1.4.2 Memory segmentation and segment registers

- It treats the 1 Mbyte of memory as divided into segments, with a maximum size of a segment as 64 kbytes. Thus any location within the segment can be accessed using 16 bits.
- The 8086 allows only four active segments at a time, as shown in the Fig. 1.4.2.
- For the selection of the four active segments the 16-bit segment registers are provided by the Bus Interface Unit (BIU) of the 8086.
- The four segment registers are : Code Segment (CS) register, the Data Segment (DS) register, the Stack Segment (SS) register, and the Extra Segment (ES) register.
- These are used to hold the upper 16-bits of the starting addresses of the four memory segments, on which 8086 works at a particular time.
- For example, the value in CS identifies the starting address of 64 k byte segment known as code segment. By "starting address", we mean the lowest addressed byte in the active code segment.
- The starting address is also known as **base address** or **segment base.**
- The BIU always inserts zeros for the lower 4 bits (nibble) in the contents of segment register to generate 20-bit base address. For example, if the code segment register contains 348AH, then code segment will start at address 348A0H.

Functions of Segment Registers

1. The CS register holds the upper 16-bits of the starting address of the segment from which the BIU is currently fetching the instruction code byte.
2. The SS register is used for the upper 16-bits of the starting address for the program stack (all stack related instructions will operate on stack).
3. ES register and DS register are used to hold the upper 16-bits of the starting address of the two memory segments which are used for data.

1.4.3 Pointers and Index Registers

- All segment registers are 16-bit wide. But it is necessary to generate 20-bit address (physical address) on the address bus.
- To get 20-bit physical address one or more pointer or index registers are associated with each segment register.
- The pointer registers IP, BP and SP are associated with code, data and stack segments, respectively. They hold the offset within the code, data and stack segments, respectively.
- The index registers DI and SI are used as a general purpose registers as well as for offset storage in case of indexed, based indexed and relative based indexed addressing modes.

1.4.4 Flag Register

- A flag is a flip-flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU.
- The flag register contains nine active flags as shown in the Fig. 1.4.3.

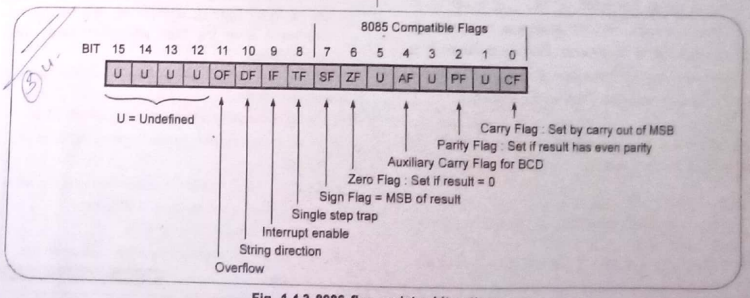


Fig. 1.4.3 8086 flag register bit pattern

- Six of them are used to indicate some condition produced by instruction.

1. **Carry Flag (CF) :** In case of addition this flag is set if there is a carry out of the MSB. The carry flag also serves as a borrow flag for subtraction. In case of subtraction it is set when borrow is needed.
2. **Parity Flag (PF) :** It is set to 1 if result of byte operation or lower byte of the word operation contain an even number of ones; otherwise it is zero.
3. **Auxiliary Flag (AF) :** This flag is set if there is an overflow out of bit 3 i.e., carry from lower nibble to higher nibble (D_3 bit to D_4 bit). This flag is used for BCD operations and it is not available for the programmer.
4. **Zero Flag (ZF) :** The zero flag sets if the result of operation in ALU is zero and flag resets if the result is nonzero. The zero flag is also set if a certain register content becomes zero following an increment or decrement operation of that register.
5. **Sign Flag (SF) :** After the execution of arithmetic or logical operations, if the MSB of the result is 1, the sign bit is set. Sign bit 1 indicates the result is negative; otherwise it is positive.
6. **Overflow Flag (OF) :** This flag is set if result is out of range. For addition this flag is set when there is a carry into the MSB and no carry out of the MSB or vice-versa. For subtraction, it is set when the MSB needs a borrow and there is no borrow from the MSB, or vice-versa.

Ex. 1.4.1 *Give the contents of the flag register after execution of following addition.*

```
  0110  0101  1101  0001
+ 0010  0011  0101  1001
  1000  1001  0010  1010
```

Sol. : SF = 1, ZF = 0, PF = 1, CF = 0, AF = 0 , OF = 1

Ex. 1.4.2 *Give the contents of the flag register after execution of following subtraction*

```
  0110  0111  0010  1001
- 0011  0101  0100  1010
  0011  0001  1101  1111
```

Sol. : SF = 0, ZF = 0, PF = 1, CF = 0, AF = 1, OF = 0

- The three remaining flags are used to control certain operations of the processor.

1. **Trap Flag (TF) :** One way to debug a program is to run the program one instruction at a time and see the contents of used registers and memory variables after execution of every instruction. This process is called '**single stepping**' through a program. Trap flag is used for single stepping through a program. If set, a trap is executed after execution of each instruction, i.e. interrupt service routine is executed which displays various registers and memory variable contents on the display after execution of each instruction. Thus programmer can easily trace and correct errors in the program.
2. **Interrupt Flag (IF) :** It is used to allow/prohibit the interruption of a program. If set, a certain type of interrupt (a maskable interrupt) can be recognized by the 8086; otherwise, these interrupts are ignored.
3. **Direction Flag (DF) :** It is used with string instructions. If DF = 0, the string is processed from its beginning with the first element having the lowest address. Otherwise, the string is processed from the high address towards the low address.

Board Questions

1. *Draw and explain programmer's model of 8086.*
2. *What is the purpose of the segment registers in the 8086 ?*
3. *Explain with suitable example working of all segment and offset registers in 8086 µp.*
4. *Explain the flag register of 8086.*
5. *List all 16 bit registers in 8086 and write their functions.* **MSBTE : Summer-15, Marks 4**

6. *List all the 16 bit registers of 8086 and write their function.* **MSBTE : Winter-15, Marks 4**
7. *Draw flag register structure of 8086 and describe operation of each flag.* **MSBTE : Winter-15, Marks 8**
8. *Draw labelled flag register format of 8086 microprocessor.* **MSBTE : Summer-16, Marks 2**
9. *Name the general purpose registers of 8086 giving brief description of each.* **MSBTE : Winter-16, Marks 4**
10. *State the function of following registers of 8086 microprocessor : i) General purpose register ii) Segment register* **MSBTE : Summer-17, Marks 4**
11. *Describe register organization of 8086.* **MSBTE : Winter-17, Marks 4**
12. *State the use of OF, TF, AF and PF flags in 8086.* **MSBTE : Summer-18, Marks 4**
13. *State the names of segment registers in 8086 microprocessor.* **MSBTE : Summer-18, Marks 4**
14. *Name the general purpose register of 8086, give brief description of each.* **MSBTE : Summer-18, Marks 4**

1.5 8086 Memory Segmentation

MSBTE : Winter-15, 17, 18, Summer-15, 16, 17, 18

- Two types of memory organisations are commonly used. These are **linear addressing** and **segmented address**. g.
- In linear addressing the entire memory space is available to the processor in one linear array.
- In the segmented addressing, on the other hand, the available memory space is divided into "chunks" called segments. Such a memory is known as **segmented memory**.
- In 8086 system the available memory space is 1Mbytes.
- This memory is divided into number of logical segments.
- Each segment is 64 K bytes in size and addressed by one of the segment registers.
- The 16-bit contents of the segment register gives the starting/base address of a particular segment, as shown in Fig. 1.5.1.

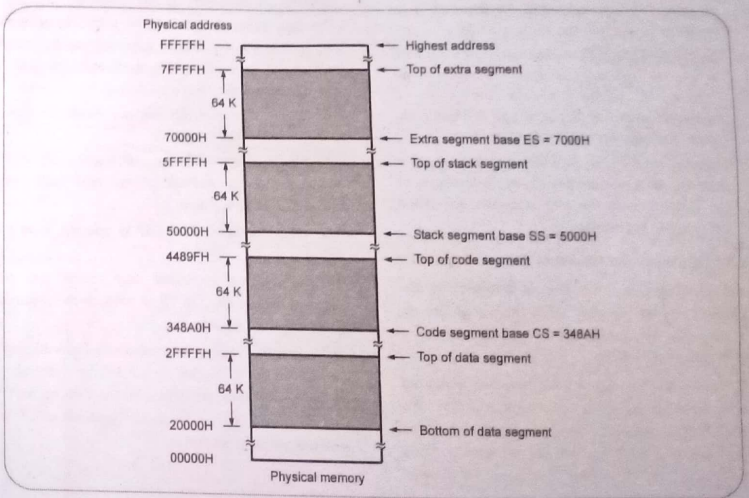


Fig. 1.5.1 Memory segmentation

- To address a specific memory location within a segment we need an offset address.
- The offset address is also 16-bits wide and it is provided by one of the associated pointer or index register.

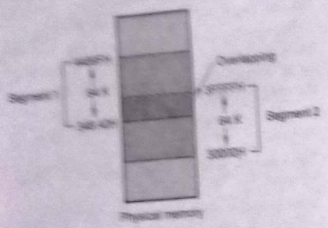


Fig. 1.5.1 (a)

1.5.1 Rules for Memory Segmentation

1. The four segments can overlap for small programs. In a minimum system all four segments can start at the address 00000H.
2. The 16-byte separation between segment base (due to the 4-bit shift) is called a paragraph and therefore, a segment always begins on a paragraph boundary. A paragraph is 16-bytes in size so the segment address is divisible by 16.
3. Segment overlapping is allowed. When another segment starts within the 64 kbytes locations of the first segment, the two segments are called **overlapping segments**.

1.5.2 Minimum and Maximum Size of Segment

- The maximum segment size is decided by the pointer or index registers. Since these registers are 16-bit wide, the maximum size of segment is 2^{16} = 64 kbytes.
- Consider that two segment registers are initialized with successive addresses. For example, 1000H and 1001H. The segment base addresses for these segments are 10000H and 10010H respectively. Thus the size of the first segment is 16 bytes (10000H - 1000FH). This is the minimum size of the segment.

1.5.3 Advantages of Memory Segmentation

1. It allows the memory addressing capacity to be 1 Mbyte even though the address associated with individual instruction is only 16-bit.
2. It allows instruction code, data, stack, and portion of program to be more than 64 kB long by using more than one code, data, stack segment, and extra segment.
3. It facilitates use of separate memory areas for program, data and stack.
4. It permits a program or its data to be put in different areas of memory, each time the program is executed i.e. program can be relocated which is very useful in multiprogramming.

1.5.4 Physical Memory Address Generation

- To access a specific memory location from any segment we need 20-bit physical address.
- The 8086 generates this address using the contents of segment register and the offset register associated with it. Let us see how 8086 access code byte within the code segment.
- We know that the CS register holds the base address of the code segment.
- The 8086 provides an Instruction Pointer (IP) which holds the 16-bit address of the next code byte within the code segment.
- The value contained in the IP is referred to as an **offset**.
- This value must be offset from (added to) the segment base address in CS to produce the required 20-bit physical address.
- The contents of the CS register are multiplied by 16. i.e. shifted by 4 position to the left by inserting 4 zero bits and then the offset i.e. the contents of IP register are added to the shifted contents of CS to generate physical address.

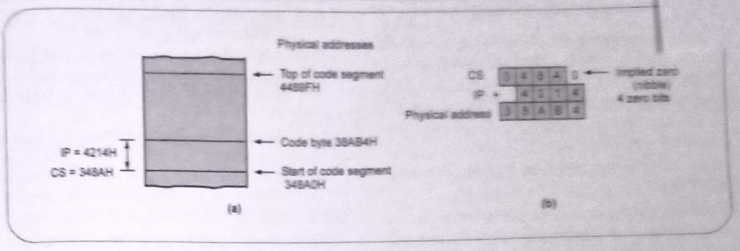


Fig. 1.5.2

- As shown in the Fig. 1.5.2, the contents of CS register are 348AH, therefore the shifted contents of CS register are 348A0H.
- When the BIU adds the offset of 4214H in the IP to this starting address, we get 38AB4H as a 20-bit physical address of memory. This is illustrated in Fig. 1.5.2 (b).
- We have seen that how 20-bit physical address is generated within the code segment. In the similar way the 20-bit physical address is generated in the other segments.
- It is important to note that each segment requires particular segment register and offset register to generate 20-bit physical address.

Ex. 1.5.1 *Calculate the physical address for the given CS = 3420H, IP = 689AH* **MSBTE : Summer-18, Marks 4**

Sol. :

CS	3	4	2	0	0 ← Implied zero
IP +		6	8	9	A
Physical Address	3	A	A	9	A (3AA9A)H

Ex. 1.5.2 *Describe the generation of physical address in 8086. If CS = 2000 H, and IP = 1122 H, calculate the physical address generation.* **MSBTE : Winter-15, Marks 4**

Sol. :

CS	2	0	0	0	0 ← Implied zero
IP +		1	1	2	2
Physical Address	2	1	1	2	2 (21122)H

Ex. 1.5.3 *Calculate physical address is CS = 2308 H and IP = 76A9 H.* **MSBTE : Summer-16, Marks 4**

Sol. :

		1				Carry
CS	2	3	0	8	0	← Implied zero
IP +		7	6	A	9	
Physical Address	2	A	7	2	9	(2A729)H

Ex. 1.5.4 *Calculate the physical address for given :*
i) DS = 73A2H SI = 3216H
ii) CS = 7370H IP = 561EH **MSBTE : Summer-17, Marks 4**

Sol. : (i)

DS	7	3	A	2	0	← Implied zero
SI +		3	2	1	6	
Physical Address	7	6	C	3	6	(76C36)H

(ii)

CS	7	3	7	0	0	← Implied zero
IP +		5	6	1	E	
Physical Address	7	8	D	1	E	(78D1E)H

Ex. 1.5.5 *Calculate the physical addres for the given CS = 3420H, IP = 689AH.* **MSBTE : Summer-18, Marks 2**

Sol. :

CS	3	4	2	0	0	← Implied zero
IP +		6	8	9	A	
Physical Address	3	A	A	9	A	(3AA9A)H

1.5.5 Pointers and Index Registers

- All segment registers are 16-bit. But it is necessary to put 20-bit address (physical address) on the address bus.
- To get 20-bit physical address one more register is associated with each segment register the way IP is associated with CS.
- These additional registers belong to the pointer and index group.
- The pointer and index group consists of Instruction Pointer (IP), Stack Pointer (SP), Base Pointer (BP), Source Index (SI) and Destination Index (DI) registers.
- **Stack Pointer (SP) :** The Stack Pointer (SP) register contains the 16-bit offset from the start of the segment to the top of stack.
- For stack operation, physical address is produced by adding the contents of stack pointer register to the segment base address in SS. To do this the contents of the stack segment register are shifted four bits left and the contents of SP are added to the shifted result.
- If the contents of SP are 9F20H and SS are 4000H then the physical address is calculated as follows. (Refer Fig. 1.5.3)

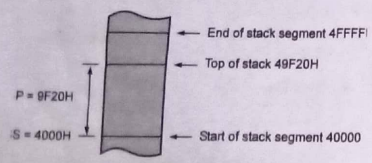


Fig. 1.5.3 Stack and stack pointer

SS = 4000H after shifting four bits left SS

= 40000H

Now

SS	40000H
+ SP	9F20H
Physical address	49F20H

1.5.6 Base Pointer, Source Index and Destination Index (BP, SI and DI)

- These three 16-bit registers can be used as general purpose registers. However, their main use is to hold the 16-bit offset of the data word in one of the segments.
- **Base pointer :** We can use the BP register instead of SP for accessing the stack using the based addressing mode. In this case, the 20-bit physical stack address is calculated from BP and SS. Addressing modes are discussed in later section.
- **Source index :** Source Index (SI) can be used to hold the offset of a data word in the data segment. In this case, the 20-bit physical data address is calculated from SI and DS.
- **Destination index :** The ES register points to the extra segment in which data is stored. String instructions always use ES and DI to determined the 20-bit physical address for the destination.

1.5.7 Default and Alternate Register Assignments

- Table 1.5.1 shows that some memory references and their default and alternate segment definitions.
- For example, instruction codes can only be stored in the code segment with IP used as an offset.
- For stack operations only SS and SP or BP registers can be used to give segment and offset addresses respectively.
- For accessing general data, string source, data pointed by BX and BP registers; it is possible to use alternate segments by using segment override prefix. See examples given after Table 1.5.1.

Type of memory reference	Default segment	Alternate segment	Offset (Logical address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective address
BP used as pointer	SS	CS, ES, DS	Effective address

Table 1.5.1 Default and alternate register assignments

For the following examples we have assumed

CS = 1000H, DS = 2000H, SS = 3000H, ES = 4000H, BP = 0010 H, BX = 0020H, SP = 0030H, SI = 0040H, DI = 0050H

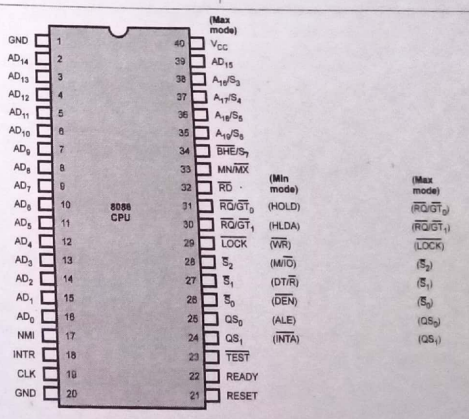
Board Questions

1. *Describe how 20 bit physical address is generated in 8086 microprocessor. Give one example.* **MSBTE : Summer-15, Marks 4**
2. *Describe memory segmentation in 8086 microprocessor and list it's four advantages.* **MSBTE : Summer-15, Marks 4**
3. *What is memory segmentation ? How it is done in 8086 microprocessor ?* **MSBTE : Winter-15, Marks 4**
4. *Explain the concept of segmentation with diagram.* **MSBTE : Summer-17, Marks 4**
5. *Describe concept of memory segmentation of 8086.* **MSBTE : Winter-16, 17, Marks 4**
6. *With the help of diagram, describe physical memory address generation of 8086.* **MSBTE : Summer-15,17,18, Winter-17, Marks 4**
7. *List the steps in physical address generation 8086 microprocessor.* **MSBTE : Summer-18**

1.6 Pin Description of 8086

MSBTE : Winter-15, 16, 17, Summer-15, 16, 17, 18

- In order to implement many situations in the microcomputer system the 8086 has been designed to work in two operating modes :
  1. Minimum mode 2. Maximum mode
- The minimum mode is used for a small systems with a single processor and maximum mode is for medium size to large systems, which often include two or more processors.
- Fig. 1.6.1 shows the pin diagram of 8086 in minimum as well as maximum mode.
- The 8086 signals can be categorised in three groups :
  - Signals having common functions in both minimum and maximum modes.
  - Signals having special functions for minimum mode.
  - Signals having special functions for maximum mode.



Pin diagram of 8086

Fig. 1.6.1

1.6.1 Signals with Common Functions in Both Modes

1. **AD_{15}-AD_0** : Acts as address bus during the first part of machine cycle and data bus for the remaining part of the machine cycle.
2. **A_{19}/S_6-A_{16}/S_3** : During the first part of machine cycle these are used to output upper 4-bits of address. During remaining part of the machine cycle these are used to output status, which indicates the type of operation to be performed in that cycle. S_5 and S_4 indicate the segment register being used as follows :

S_4	S_3	Register
0	0	ES
0	1	SS
1	0	CS or none
1	1	DS

S_5 gives the current setting of the Interrupt Flag (IF) and S_6 is always zero.

3. **$\overline{BHE}/S_7$: BHE (Bus High Enable)** : Low on this pin during first part of the machine cycle, indicates that at least one byte of the current transfer is to be made on higher order byte AD_{15}-AD_8, otherwise the transfer is made on lower order byte AD_7-AD_0.

$\overline{BHE}$	A_0	Data accesses
0	0	Word
0	1	Upper byte from odd address
1	0	Lower byte from even address
1	1	None

Status S_7 is output during the later part of the machine cycle, but, presently, S_7 has not been assigned a meaning.

4. **NMI** : It is a positive edge triggered nonmaskable interrupt request.
5. **INTR** : It is a level triggered maskable interrupt request. It is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt service routine.
6. **CLK** : 8086 requires clock signal (with 33 % duty cycle) from some external, crystal controlled generator to synchronize internal operations. Clock frequency depends on the version of 8086.

Processor	Required clock signal
8086	5 MHz
8086-2	8 MHz
8086-1	10 MHz

7. **RESET** : It clears PSW, IP, DS, SS, ES, and the instruction queue. It then sets CS to FFFFH. This signal must be high for at least 4 clock cycles. When RESET is removed, 8086 will fetch its next instruction from physical address FFFF0H.
8. **READY** : If this signal is low the 8086 enters into wait state. This signal is used primarily to synchronize slower peripherals with the microprocessor.
9. **$\overline{TEST}$ (Input)** : This signal is only used by the WAIT instruction. The 8086 enters into a wait state after execution of the WAIT instruction until a LOW signal on the $\overline{TEST}$ pin. $\overline{TEST}$ signal is synchronized internally during each clock cycle on the leading edge of the clock cycle.
10. **$\overline{RD}$ (Output)** : $\overline{RD}$ is low whenever the 8086 is reading data from memory or an I/O device.
11. **MN/$\overline{MX}$ (Input)** : The 8086 can be configured in either minimum mode or maximum mode using this pin. This pin is tied high for minimum mode.

1.6.2 Signal Definitions (24 to 31) for Minimum Mode

INTA (Interrupt Acknowledge) output : This indicates recognition of an interrupt request. It consists of two negative going pulses in two consecutive bus cycles. The first pulse informs the interface that its request has been recognized and upon receipt of the second pulse, the interface is to send the interrupt type to the processor over the data bus.

ALE (Address Latch Enable) output : This signal is provided by 8086 to demultiplex the AD_0-AD_{15} into A_0-A_{15} and D_0-D_{15} using external latches.

$\overline{DEN}$ (Data Enable) output : This signal informs the transceivers that the CPU is ready to send or receive data.

DT/$\overline{R}$ (Data Transmit / Receive) output : This signal is used to control data flow direction. High on this pin indicates that the 8086 is transmitting the data and low indicates that the 8086 is receiving the data.

M/$\overline{IO}$ output : It is used to distinguish memory data transfer, (M/$\overline{IO}$ = HIGH) and I/O data transfer (M/$\overline{IO}$ = LOW).

$\overline{WR}$: Write output : $\overline{WR}$ is low whenever the 8086 is writing data into memory or an I/O device.

HOLD input, HLDA output : A HIGH on HOLD pin indicates that another master (DMA) is requesting to take over the system bus. On receiving HOLD signal processor outputs HLDA signal HIGH as an acknowledgment. At the same time, processor tristates the system bus. A low on HOLD gives the system bus control back to the processor. Processor then outputs low signal on HLDA.

1.6.3 Signal Definitions (24 to 31) for Maximum Mode

1. **QS_1, QS_0 (output)** : These two output signals reflect the status of the instruction queue. This status indicates the activity in the queue during the previous clock cycle.

QS_1	QS_0	Status
0	0	No operation (queue is idle)
0	1	First byte of an opcode
1	0	Queue is empty
1	1	Subsequent byte of an opcode

2. **$\overline{S}_2$, $\overline{S}_1$, $\overline{S}_0$ (output)** : These three status signals indicate the type of transfer to be take place during the current bus cycle.

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Machine cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Instruction fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Inactive-Passive

3. **$\overline{LOCK}$** : This signal indicates that an in with a LOCK prefix is being executed and the bus is not to be used by another processor.
4. **$\overline{RQ}/\overline{GT}_1$ and $\overline{RQ}/\overline{GT}_0$** : In the maximum mode, HOLD and HLDA pins are replaced by $\overline{RQ}$ (Bus request)/$\overline{GT}_0$ (Bus Grant), and $\overline{RQ}/\overline{GT}_1$ signals. By using bus request signal another master can request for the system bus and processor communicate that the request is granted to the requesting master by using bus grant signal. Both signals are similar except the $\overline{RQ}/\overline{GT}_0$ has higher priority than $\overline{RQ}/\overline{GT}_1$.

Board Questions

1. *State the function of the following pins of 8086.*
   i) NMI ii) $\overline{TEST}$
   iii) $\overline{DEN}$ iv) MN/$\overline{MX}$
   MSBTE : Summer-15,16, Marks 4
2. *State the functions of the following pin of 8086 microprocessor*
   i) ALE ii) DT/$\overline{R}$
   iii) HOLD iv) in/$\overline{IO}$ MSBTE : Winter-15, Marks 4
3. *State all the control signal generated by S_0, S_1, S_2 with their function by 8086 microprocessor.* MSBTE : Summer-16, Marks 4
4. *State the functions of following pins of 8086.*
   1) ALE 2) $\overline{WR}$ MSBTE : Winter-16, Marks 2
5. *Explain the function of following pins of 8086 microprocessor :*
   i) MN/$\overline{MX}$ ii) READY
   iii) ALE iv) DT/$\overline{R}$
   MSBTE : Summer-17, Marks 4
6. *Describe functions of folowing pins of µP 8086 :*
   a) MN/MX b) ALE MSBTE : Winter-17, Marks 2
7. *State all control signal generated by S_0, S_1, S_2 with their function by 8086.* MSBTE : Winter-17, Summer-16,18, Marks 4
8. *State all the control signals generated by S_0, S_1, S_2 with their functions.* MSBTE : Summer-18, Marks 4
9. *State the function of following pins of 8086 microprocessor.*
   i) DT/$\overline{R}$ ii) NM1
   iii) $\overline{RD}$ iv) $\overline{DEN}$ MSBTE : Summer-18, Marks 4
10. *Explain maskable and non maskable interrupt used in 8086.* MSBTE : Summer-15, Marks 4

1.7 Minimum Versus Maximum Mode

MSBTE : Winter-15, 16, Summer-

Sr. No.	Minimum Mode	Maximum Mode
1.	MN/$\overline{MX}$ signal is connected to V_{CC}(+5 V).	MN/$\overline{MX}$ signal is connected to ground.
2.	It is a single processor system configuration.	It is a multi-processor system configuration.
3.	The 8086, itself generates system control signals.	External bus controller (8288) is required to generate system control signals.
4.	In minimum mode, signals from 24 through 31 are : $\overline{INTA}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, M/$\overline{IO}$, $\overline{WR}$, HLDA and HOLD, respectively.	In maximum mode, signals from 24 through 31 are : $Q\bar{S}_1$, $Q\bar{S}_0$, $\bar{S}_0$, $\bar{S}_1$, $\bar{S}_2$, LOCK, $\overline{RQ}/\overline{GT}_1$, and $\overline{RQ}$ / $\overline{GT}_0$, respectively.

Board Question

1. *Compare minimum and maximum mode of 8086.* MSBTE : Winter-15, 16, Summer-18, Marks 4