Faculty of Engineering and Technology
University of Lucknow
Er. Priyanka Singh
B.Tech –IV Sem
Fundamentals of Microprocessor (EC-403)

8086 MICROPROCESSOR

The microprocessor 8085 followed by 8080, with a few more added features to it's architecture, which resulted in a functionally complete microprocessor.

• The main limitations of 8-bit microprocessor were

– Low speed,

– Low memory addressing capability

– Limited number of general purpose registers

– Less powerful instruction set

• All these limitations lead to the launching of 8086 microprocessor.

• In the family of 16 bit microprocessors, Intel's 8086 was the first one to be launched in 1978.

•The 8086 microprocessor has a much more powerful instruction set along with the architectural developments which imparts substantial programming flexibility and improvement in speed over the 8-bit microprocessor.

• The peripheral chips designed earlier for 8085 were compatible with microprocessor 8086 with slight or no modifications.

Register organization of 8086

• 8086 has a powerful set of registers known as general purpose registers and special purpose registers.

• All of them are 16 bit registers.

• The general purpose registers can be used as either 8 bit registers or 16 bit registers.

• They may be either used for holding data, variables and intermediate results temporarily or other purposes like a counter or for storing offset address for some particular addressing modes etc.

• The special purpose registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes.

• The register set is categorized into four groups, as follows:

– General data registers

– Segment registers

– Pointers and index registers

– Flag register

General data registers:

• Figure 1.1 shows the register organization of 8086.

• The registers AX, BX, CX and DX are the general purpose 16 bit registers. AH, AL

• AX is used as 16 bit accumulator (AH, AL)

• AL can be used as an 8 bit accumulator for 8 bit operations. This is the most important general purpose register having multiple functions.

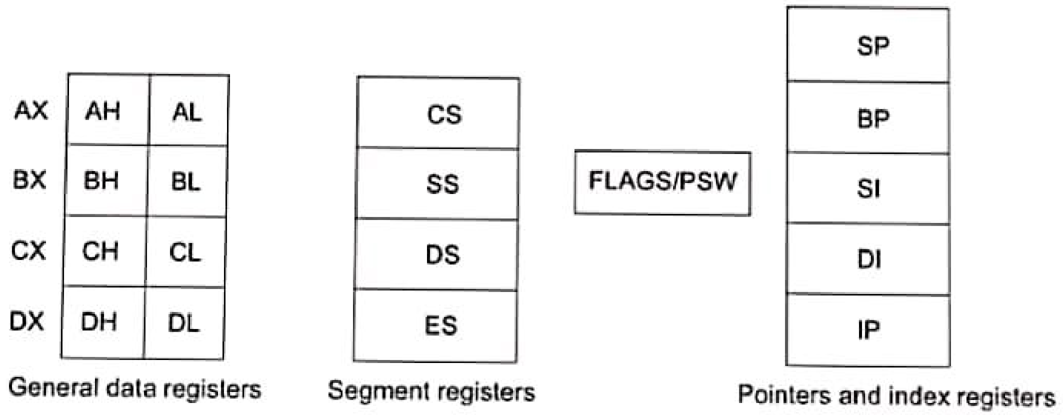


Fig. 1.1 *Register organisation of 8086*

• Usually L and H specify the lower and higher bytes of a particular register.

• AX – accumulator,

BX – offset storage,

CX – counter,

DX – to store data

Segment Registers:

• Unlike 8085, the 8086 addresses segmented memory.

• The complete 1 megabyte memory, which the 8086 addresses, is divided into 16 logical segments.

• Each segment thus contains 64 k bytes of memory.

• There are 4 segment registers:-

Code segment register (CS)	Code,
Data segment register (DS)	Data,
Extra segment register (ES)	data ,
Stack segment register (SS)	Stack related data

• The CPU uses the stack for temporarily storing important data.

• While addressing any location in the memory bank, the physical address is calculated from two parts, the first is segment address and the second is offset.

• The segment registers contain 16 bit segment base addresses, related to different segments

• Any of the pointers and index registers or BX may contain the offset of the location to be addressed

• The advantage of this scheme is that instead of maintaining a 20 bit register for a physical address, the processor just maintains two 16 bit registers which are within the word length capacity of the machine.

• It may be noted that all these segments are logical segments

Pointers and Index registers:

• The pointers contain offset within the particular segments.

• The pointers IP, BP and SP usually contain offsets within the code (IP), and stack (BP & SP) segments.

• The index registers are used as general purpose registers as well as for offset storage in case of indexed, based indexed and relative based indexed addressing modes.

• The register SI is generally used to store the offset of the source data in data segment while the register DI is used to store the offset of the destination in data or extra segment.

• The index registers are particularly useful for string manipulations.

Architecture of 8086

The architecture of 8086 supports a 16 bit ALU, a set of 16 bit registers and provides the segmented memory addressing capability, a rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.

• The internal block diagram, shown in Fig 1.2, describes the overall organization of different units inside the chip.

• The complete architecture of 8086 can be divided into two parts.

– Bus interface unit

– Execution Unit

• The bus interface unit contains the circuit for physical address calculations and a predecoding instruction byte queue (6 bytes long)

• The bus interface unit makes the system's bus signals available for external interfacing of the devices.

• In other words, this unit is responsible for establishing communications with external devices and peripherals including memory via the bus.

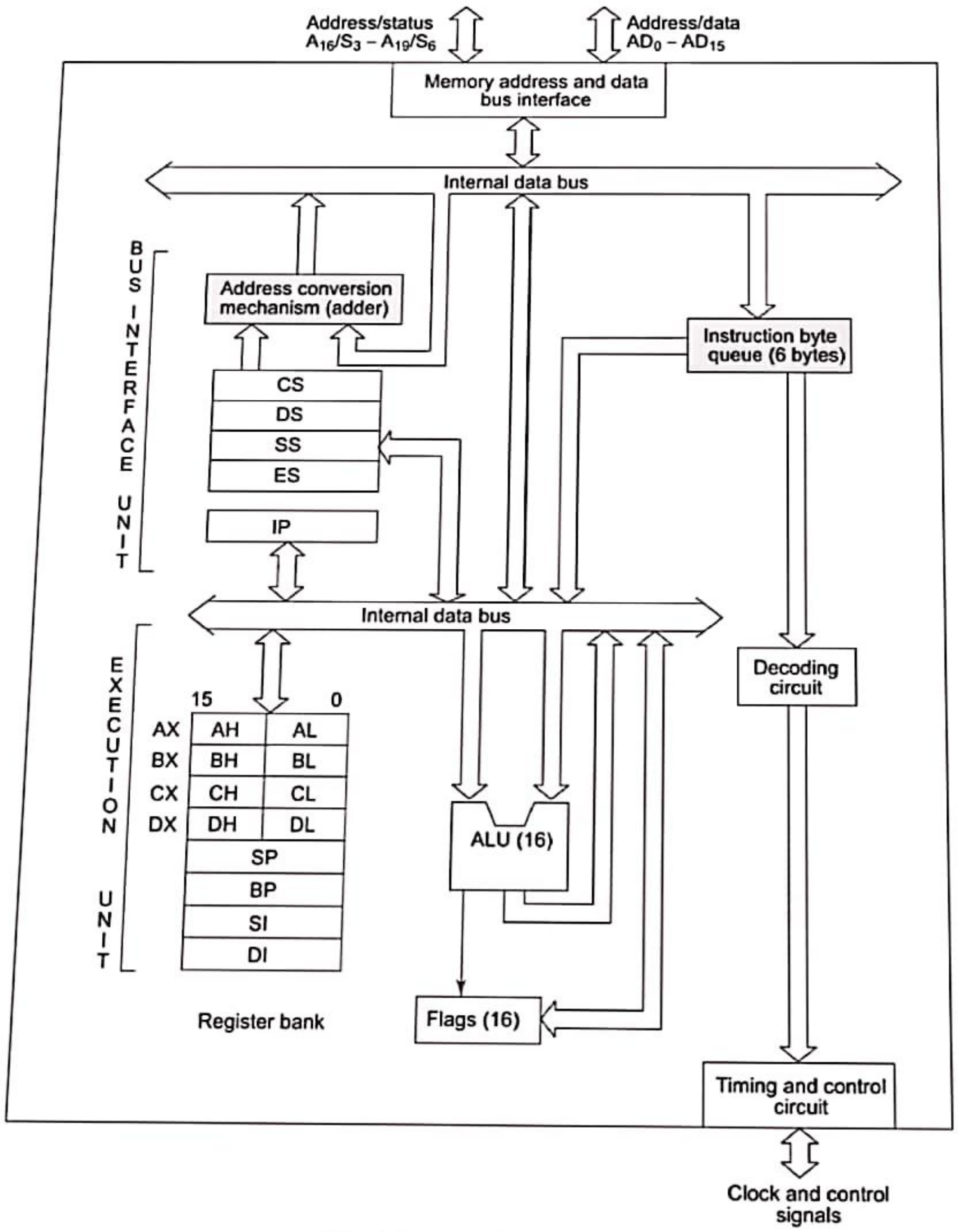
• As already stated, the 8086 addresses a segmented memory. The complete physical address which is 20 bits long is generated using segment and offset registers, each 16 bit long.

• For generating a physical address from contents of these two registers, the content of a segment register also called as segment address is shifted left bit-wise four times and to this result, content of an offset register also called as offset address is added, to produce a 20 bit physical address.

• For example, if the segment address is 1005H and the offset is 5555H, then the physical address is calculated as below 1005H◊segment address 5555H◊offset address Physical address = 1005 * 10 + 5555 = 155A5H

• Thus, the segment addressed by the segment value 1005H can have offset values from 0000H to FFFFH within it i.e. maximum 64 K locations may be accommodated in the segment.

• Thus, the segment register indicates the base address of a particular segment, while the offset indicates the distance of the required memory location in the segment from the base address.



Address/status $A_{16}/S_3 - A_{19}/S_6$
Address/data $AD_0 - AD_{15}$
Memory address and data bus interface
Internal data bus
BUS INTERFACE UNIT
Address conversion mechanism (adder)
Instruction byte queue (6 bytes)
CS
DS
SS
ES
IP
Internal data bus
EXECUTION UNIT
15
0
AX AH AL
BX BH BL
CX CH CL
DX DH DL
SP
BP
SI
DI
ALU (16)
Decoding circuit
Register bank
Flags (16)
Timing and control circuit
Clock and control signals

Fig. 1.2 *8086 Architecture*

• Since the offset is a 16-bit number, each segment can have a maximum of 64k locations.

• The bus interface unit has a separate adder to perform this procedure for obtaining a physical address while addressing a memory.

• The segment address value is to be taken from an appropriate segment register depending upon whether code, data or stack are to be accessed, while the offset may be the content of IP, BX, SI, DI, SP, BP or an immediate 16-bit value, depending upon the addressing mode.

• In case of 8085, once the op-code is fetched and decoded, the external bus remains free for some time, while processor internally executes the instruction.

• The time slot is utilized in 8086 to achieve the overlapped fetch and execution cycles.

• While the fetched instruction is executed internally, the external bus is used to fetch the machine code of the next instruction and arrange it in a queue known as pre-decoded instruction byte queue. It is a 6 byte long, first-in first-out structure.

• The instructions from the queue are taken for decoding sequentially.

• Once a byte is decoded, the queue is rearranged by pushing it out and the queue status is checked for the possibility of the next op-code fetch cycle.

• While the op-code is fetched by the interface unit (BIU), the execution unit (EU) executes the previously decoded instruction concurrently.

• The BIU along with EU thus forms a pipeline.

• The bus interface unit, thus manages the complete interface of execution unit with memory and I/O devices, of-course, under the control of the timing and control unit.

• The execution unit contains the register set of 8086 except segment register and IP.

• It has a 16-bit ALU, able to perform arithmetic and logical operation.

• The 16-bit flag register reflects the results of execution by the ALU.

• The decoding unit decodes the op-code bytes issued from the instruction byte queue.

• The timing and control unit derives the necessary control signals to execute the instruction op-code received from the queue, depending upon the information made available by the decoding circuit.

• The execution unit may pass the results to bus interface unit for storing them in memory.

Memory Addresses

• As 8086 has got 20 address lines, it's addressing capability is 1 M Byte memory locations.

• The physical address is calculated from segment address and offset address as given below Physical address=10*segment addr + offset addr

Memory Segmentation

• The memory in an 8086 based system is organized as segmented memory.

• In this scheme, the complete physically available memory may be divided into a number of logical segments.

• Each segment is 64 Kbytes in size and is addressed by one of the segment register.

• The 16 bit contents of the segment register actually point to the starting location of a particular segment.

• To address a specific memory location within a segment, we need an offset address.

• The offset address is also 16 bit long so that the maximum offset value can be FFFFH, and the maximum size of any segment is thus 64 K locations.

• To emphasize this segmented memory concept, we will consider an example of a housing colony containing say, 100 houses

– Numbering the houses sequentially

– Numbering the houses matrix wise (rows X columns) 10 X 10

• In the second scheme, the efforts required for finding the same house will be too less.

• This second scheme in our example is analogous to the segmented memory scheme, where the addresses are specified in terms of segment addresses analogous to rows and offset addresses analogous to columns

• The CPU 8086 is able to address 1 Mbytes of physical memory.

• The complete 1 Mbytes memory can be divided into 16 segments, each of 64 Kbytes size.

• The offset address values are from 0000H and FFFFH so that the physical addresses range from 00000H to FFFFFH.

• In the above said case, the segments are called non-overlapping segments which are shown in Figure 1.3a.

• In some cases, however, the segments are overlapping.

• Suppose a segment starts at a particular address and its maximum size can be 64 Kbytes

• But, if another segment starts before this 64 kbytes locations of the first segment, the two segments are said to be overlapping segments.

• The area of memory from the start of the second segment to the possible end of the first segment is called an overlapped segment area • Figure 1.3b explains the phenomenon more clearly.

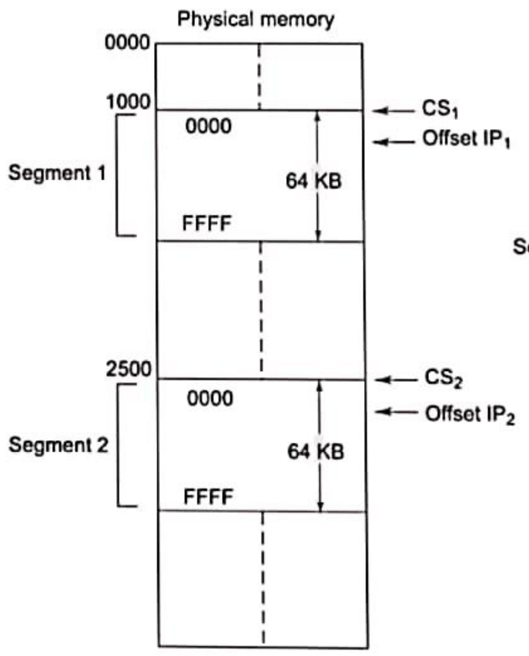


Fig. 1.3(a) *Non-overlapping Segments*

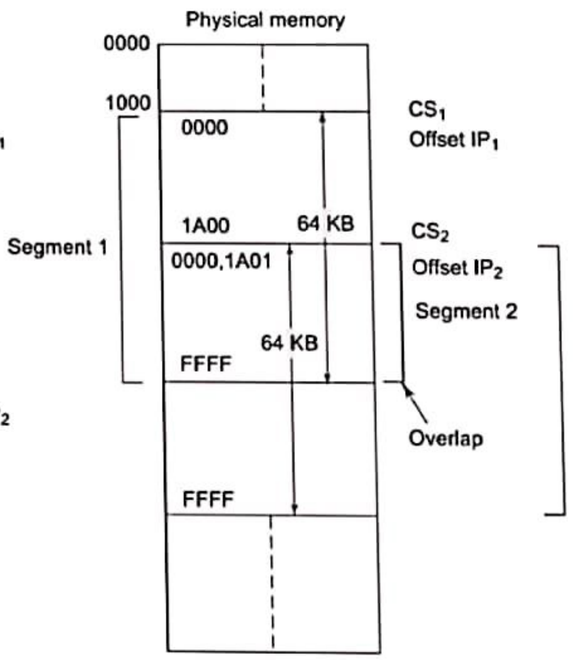


Fig. 1.3(b) *Overlapping Segments*

• The locations lying in the overlapped area may be addressed by the same physical address generated from two different sets of segment and offset addresses.

• The main advantages of the segmented memory scheme are as follows

– 1 Allows the memory capacity to be 1 Mbytes although the actual addresses to be handled are of 16 bit size.

– 2 Allows the placing of the code, data, and stack portions of the same program in different parts of memory for data and code protection.

– 3 Permits a program and/or its data to be put into different areas of memory each time the program is executed i.e. provision for relocation is done.

• In the overlapped Area locations physical address =CS1+ IP1 = CS2 + IP2 where + indicates the procedure of physical address formation

Flag Register:

• The 8086 flag register contents indicate the results of computations in the ALU. It also contains some flag bits to control the CPU operation.

• 8086 has a 16-bit flag register which is divided into two parts,

– Condition code or status flags

– Machine control flags

• The condition code flag register is the lower byte of the 16bit flag register along with the overflow flag

• This flag is identical to the 8085 flag register with an additional overflow flag, which is not present in 8085

• This part of the flag register of 8086 reflects the results of the operations performed by ALU

• The control flag register is the higher byte of the flag register of 8086.

• It contains three flags,

– Direction flag (D)

– Interrupt flag (I)

– Trap flag (T)

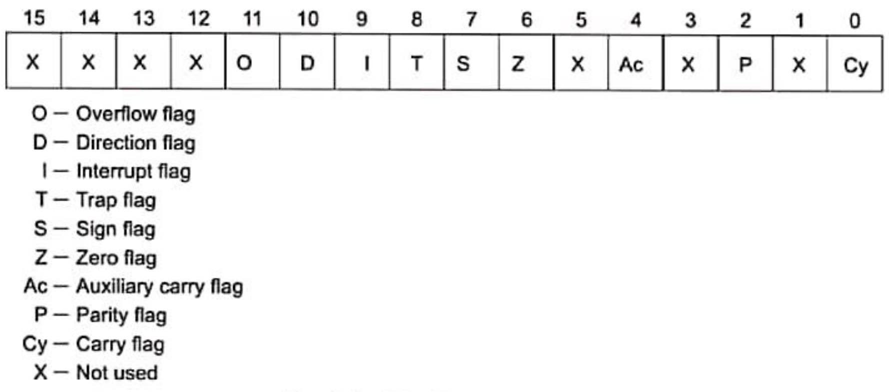


Fig. 1.4 ***Flag Register of 8086***

• T - Trap flag When it is set, the processor enters the single step execution mode.

• I – Interrupt Flag mode If this flag is set, the maskable interrupts are recognized by the CPU, otherwise they are ignored.

• D – Direction flag This flag is used by string manipulations instruction. If this flag bit is 0, the string is processed beginning from the lowest address to the highest address ie auto increment mode. Otherwise, the string is processed from the highest address towards the lowest address ie auto decrement mode.

• O – Overflow flag – this flag is set if an overflow occurs i.e. if the result of a signed operation is large enough to be accommodated in a destination register.

• For example, in case of the addition of two signed numbers, if the result overflows into the sign bit ie the result is of more than 7 bits in size in case of 8-bit signed operations and more than 15 bits in size in case of 16 bit signed operations, then overflow flag will be set.

Signal Descriptions of 8086

• The microprocessor 8086 is a 16 bit CPU available in 3 clock rates 5,8 and 10 MHz, packed in a 40 pin CERDIP or plastic package.

• The 8086 operates in single processor or multiprocessor configurations to achieve high performance.

• The pin configuration is shown in fig 1.5

• Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multi processor mode) configuration

• The 8086 signals can be categorized in three groups.

– Signals having common functions in minimum as well as maximum mode

– Signals which have special functions for minimum mode

– Signals which have special functions for maximum mode.

• The following signal descriptions are common for both the minimum and maximum modes.

AD15 – AD0:

• These are the time multiplexed memory I/O address and data lines

• Address remains on lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4.

• Here T1, T2, T3, T4 and Tw are the clock states of a machine cycle.

• Tw is a wait state.

• These lines are active high and float to a tri-state during interrupt acknowledge and local bus hold acknowledge cycles.

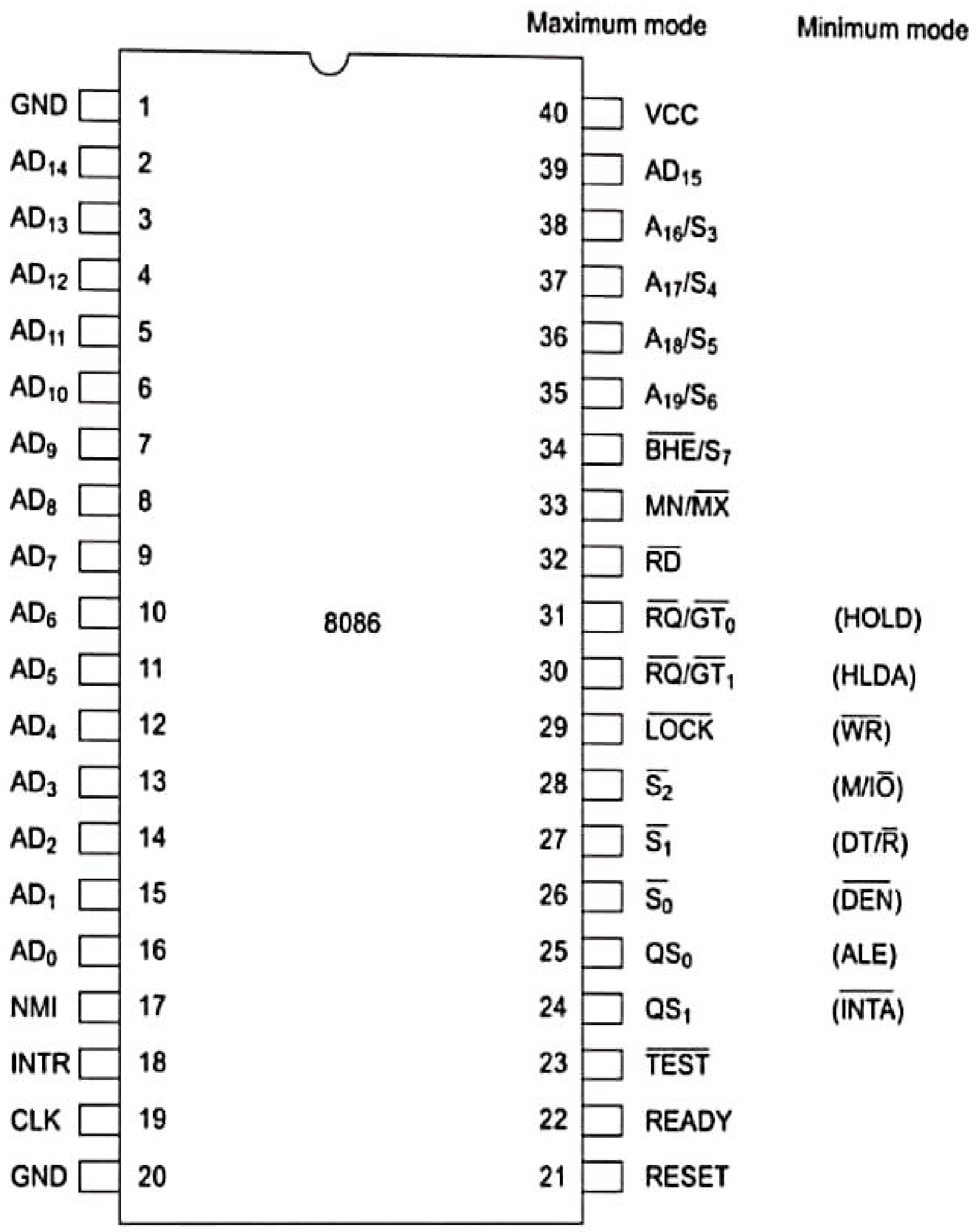


Fig. 1.5 *Pin Configuration of 8086*

A19/S6, A18/S5, A17/S4, A16/S3:

• These are the time multiplexed address and status lines.

• During T1, these are the most significant address lines for memory operations.

• During I/O operations, these lines are low.

• During memory or I/O operations, status information is available on those lines for T2, T3, Tw and T4.

• The status of the interrupt enable flag bit (displayed on S5) is updated at the beginning of each clock cycle.

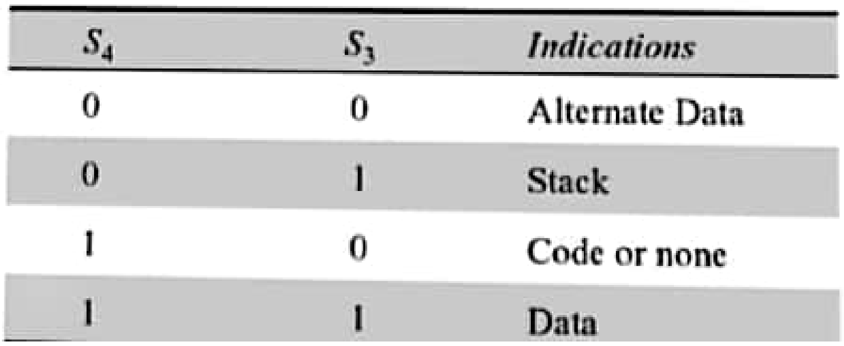
• The S4 and S3 together indicate which segment register is presently being used for memory accesses, as shown in table 1.1.

• These lines float to tri-state off (tri-stated) during the local bus hold acknowledge

• The status line S6 is always low (logical).

• The address bits are separated from the status bits using latches controlled by the ALE signal

Table 1.1 ***Status***



S_4	S_3	*Indications*
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

$\overline{\text{BHE}}$/S7 – Bus High Enable/Status:

• The bus high enable signal is used to indicate the transfer of data over the higher order (D15 – D8) data bus as shown in Table 1.2

• It goes low for the data transfers over D15 – D8 and is used to derive chip selects of odd address memory bank or peripherals.

• $\overline{\text{BHE}}$ is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on the higher byte of the data bus.

• The status information is available during T2, T3 and T4.

• The signal is active low and is tri-stated during “Hold”.

• It is low during T1 for the first pulse of the interrupt acknowledge cycle. S7 is not currently used.

Table 1.2 *Bus High Enable and A_0*

$\overline{BHE}$	A_0	*Indication*
0	0	Whole word (2 bytes)
0	1	Upper byte from or to odd address.
1	0	Lower byte from or to even address
1	1	None

$\overline{\text{RD}}$ – READ

• Read signal, when low, indicates the peripherals that the processor is performing a memory or I/O read operation.

• $\overline{\text{RD}}$ is active low and shows the state for T2, T3, Tw of any read cycle.

• The signal remains tri-stated during the 'HOLD Acknowledge'

READY

• This is the acknowledgement from the slow devices or memory that they have completed the data transfer.

• This is an input signal to the 8086.

• This signal is active high.

INTR – Interrupt request

• This is a level triggered input

• This is sampled during the last clock cycle of each instruction to determine the availability of the request

• If any interrupt request is pending the processor enters the interrupt acknowledge cycle.

• This can be internally masked by resetting the interrupt enable flag.

• This signal is active high.

$\overline{\text{TEST}}$

• This input is examined by a 'wait' instruction.

• If the $\overline{\text{TEST}}$ input goes low, execution will continue, else, the processor remains in an idle state.

NMI – Non-maskable Interrupt.

• This is an edge triggered input which causes a type 2 interrupt.

• The NMI is not maskable internally by software.

• A transition from low to high initiates the interrupt response at the end of the current instruction.

RESET

• This input causes the processor to terminate the current activity and start execution from FFFF0H

• The signal is active high and must be active for at least four clock cycles.

CLK – Clock Input

• The clock input provides the basic timing for processor operation and bus control activity.

• It's an asymmetric square wave with 33% duty cycle.

• The range of frequency for different 8086 versions is from 5 MHz to 10 MHz.

Vcc

• 8086 requires +5V power supply for the operation of the internal circuit.

GND

• This is the ground for the internal circuit.

$\overline{\text{MX}}$/MN.

• The logic level at this pin decides whether the processor is to operate in either minimum (Single processor) or maximum (multi processor) mode.

The following pin functions are for the minimum mode operation of 8086.

M/$\overline{\text{IO}}$ – Memory/IO

• This is a status line logically equivalent to $\overline{S2}$ in the maximum mode.

• Low – I/O operation High – Memory operation.

• This line becomes active in the previous T4 and remains active till final T4 of the current cycle.

• It is tri-stated during local bus “hold acknowledge”

$\overline{\text{INTA}}$ – Interrupt acknowledge

• This signal is used as a read strobe for interrupt acknowledge cycles

• In other words, when it goes low, it means that the processor has accepted the interrupt.

• It is active low during T2,T3 and Tw of each interrupt acknowledge cycle.

ALE – Address Latch Enable

• This output signal indicates that availability of the valid address on the address / data lines, and is connected to latch enable input of latches

• This signal is active high and is never tri-stated.

DT/$\overline{\text{R}}$ – Data Transmit / Receive

• This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers).

• Data transmission – signal is high. data receiving – signal is low.

• Logically, this is equivalent to S1 in maximum mode.

• It’s timing is the same as M/$\overline{\text{I/O}}$.

• This is tri-stated during ‘hold acknowledge’

$\overline{\text{DEN}}$ – Data Enable

• This signal indicates the availability of valid data over the address / data lines.

• It is used to enable the transreceivers to separate the data from the multiplexed address / data signal.

• It is active from the middle of T2 until the middle of T4.

• $\overline{\text{DEN}}$ is tri-stated during hold acknowledge cycle

HOLD, HLDA – Hold, Hold Acknowledge

• When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access.

• The processor, after receiving the HOLD request, issues the HOLD acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle. At the same time, the processor floats the local bus and control lines.

• When the processor detects the HOLD line low, it lowers the HLDA signal.

The following pin functions are applicable for maximum mode operation of 8086.

S2, S1, S0 – status lines

• These are the status lines which indicate the type of operation, being carried out by the processor.

• These become active during T4 of the previous cycle and remain active during T1 and T2 of the current bus cycle.

• The status lines return to passive state during T3 of the current bus cycle so that they may again become active for the next bus cycle during T4.

• The various operations indicated by these status lines are given in the table 1.3.

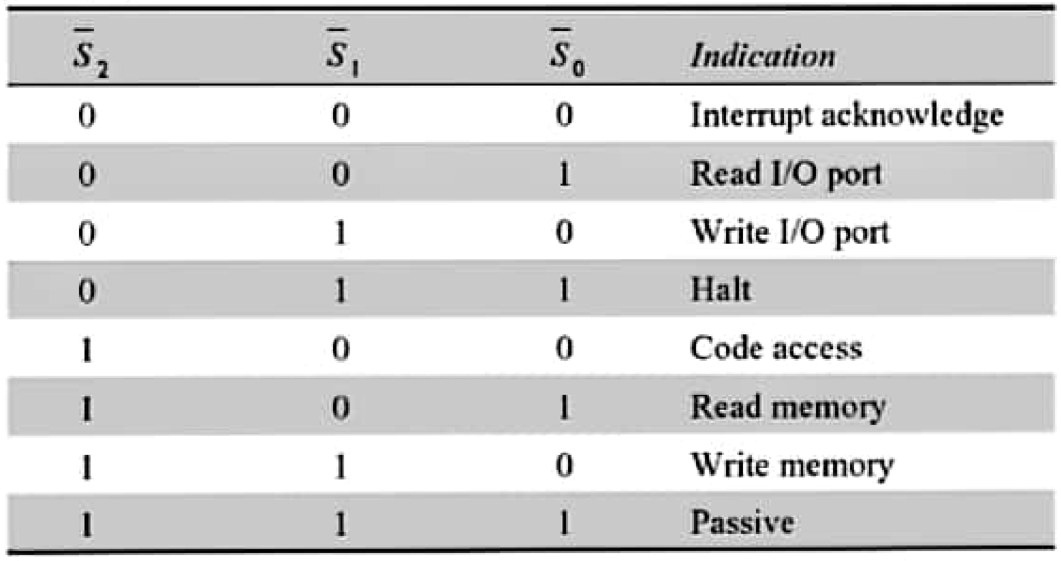


Table 1.3

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	*Indication*
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

$\overline{\text{LOCK}}$

• This output pin indicates that other system bus masters will be prevented from gaining the system bus, while LOCK signal is low.

• The $\overline{\text{LOCK}}$ signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction.

• This floats to tri-state off during 'hold acknowledge'

• When CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensure that other processors connected in the system will not gain the control of the bus.

QS1, QS0 – Queue Status

• These lines give information about the status of the code-prefetch queue.

• These are active during the CLK cycle after which the queue operation is performed

• These lines indicate various operations as indicated in the table 1.4.

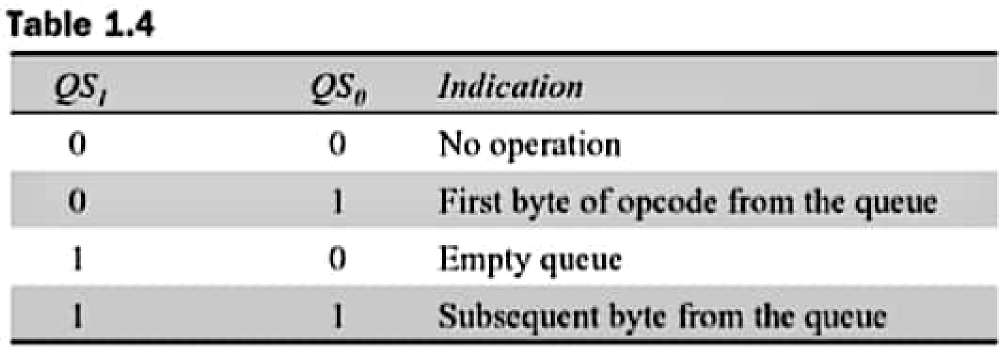


Table 1.4

QS_1	QS_0	*Indication*
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty queue
1	1	Subsequent byte from the queue

• This simultaneous fetching and executing of instructions is called pipe-lining. This results in faster execution.

$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$ – REQUEST/GRANT

• These pins are used by other local bus masters in maximum mode, to force the processor to release local bus at the end of processor's current bus cycle.

• Each of the pins is bidirectional with $\overline{RQ}/\overline{GT0}$ having the higher priority than $\overline{RQ}/\overline{GT1}$.

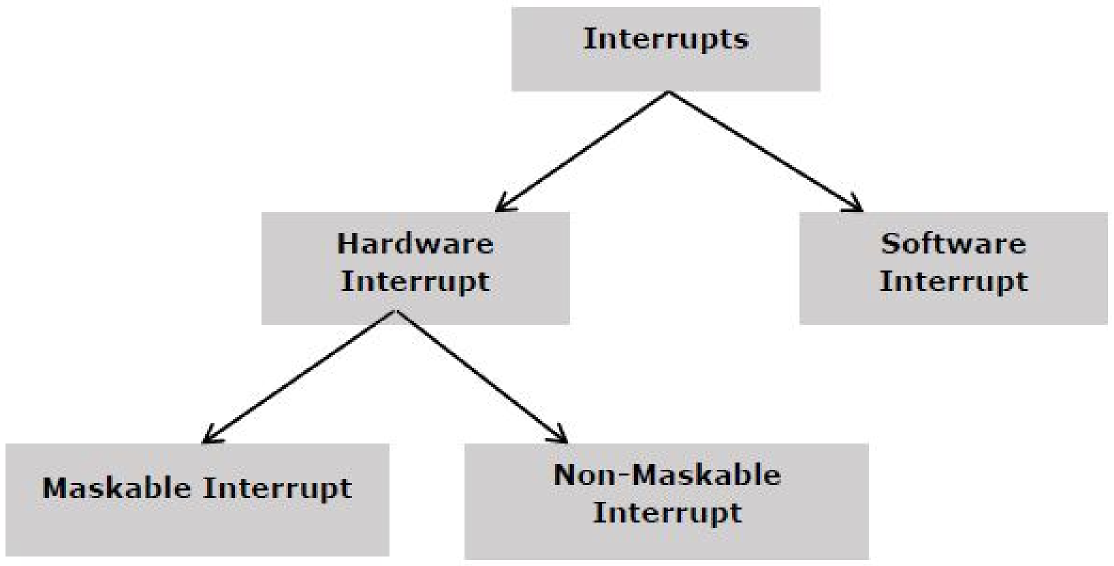
• The request and grant pulses are active low.

• $\overline{RQ}/\overline{GT}$ pins have internal pull up resistors and may be left unconnected.

8086 Interrupt

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

The following image shows the types of interrupts we have in a 8086 microprocessor –



Hardware Interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice. The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0.

Software Interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location ‘type number’ × 4
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. –

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

  The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

INT 3-Break Point Interrupt Instruction

It is a 1-byte instruction having op-code is CCH. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3×4 = 0000CH
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on overflow instruction

It is a 1-byte instruction and their mnemonic **INTO**. The op-code for this instruction is CEH. As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps −

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0