

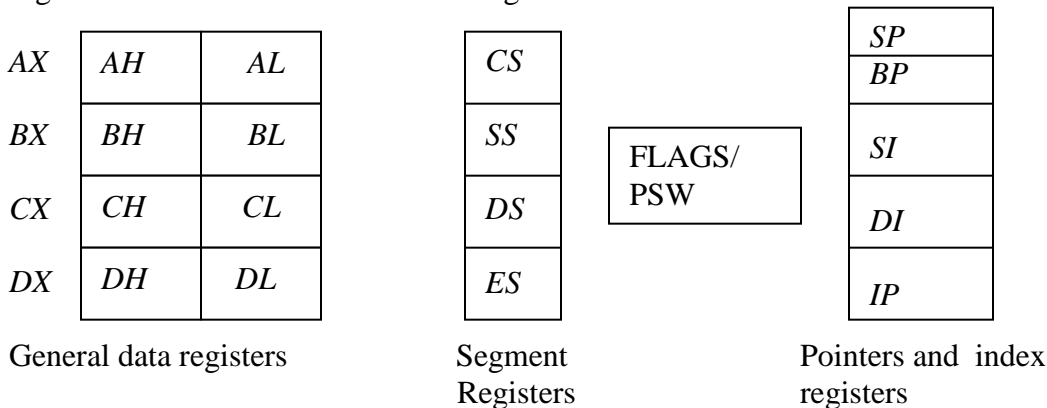
## Intel 8086 Microprocessor

Intel 8086 was the first 16-bit microprocessor introduced by Intel in 1978.

### Register Organization of 8086

All the registers of 8086 are 16-bit registers. The general purpose registers can be used as either 8-bit registers or 16-bit registers.

The register set of 8086 can be categorized into 4 different groups. The register organization of 8086 is shown in the figure.



Register organization of 8086

### General Data Registers:

The registers *AX*, *BX*, *CX* and *DX* are the general purpose 16-bit registers.

*AX* is used as 16-bit accumulator. The lower 8-bit is designated as *AL* and higher 8-bit is designated as *AH*. *AL* can be used as an 8-bit accumulator for 8-bit operation.

All data register can be used as either 16 bit or 8 bit. *BX* is a 16 bit register, but *BL* indicates the lower 8-bit of *BX* and *BH* indicates the higher 8-bit of *BX*.

The register *CX* is used default counter in case of string and loop instructions.

The register *BX* is used as offset storage for forming physical address in case of certain addressing modes.

*DX* register is a general purpose register which may be used as an implicit operand or destination in case of a few instructions.

### Segment Registers:

The 8086 architecture uses the concept of segmented memory. 8086 able to address to address a memory capacity of 1 megabyte and it is byte organized. This 1 megabyte

memory is divided into 16 logical segments. Each segment contains 64 kbytes of memory.

There are four segment register in 8086

- Code segment register (CS)
- Data segment register (DS)
- Extra segment register (ES)
- Stack segment register (SS)

Code segment register (CS): is used for addressing memory location in the code segment of the memory, where the executable program is stored.

Data segment register (DS): points to the data segment of the memory where the data is stored.

Extra Segment Register (ES) : also refers to a segment in the memory which is another data segment in the memory.

Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

While addressing any location in the memory bank, the physical address is calculated from two parts:

- The first is segment address, the segment registers contain 16-bit segment base addresses, related to different segment.
- The second part is the offset value in that segment.

The advantage of this scheme is that in place of maintaining a 20-bit register for a physical address, the processor just maintains two 16-bit registers which is within the memory capacity of the machine.

### Pointers and Index Registers.

The pointers contain offset within the particular segments.

- The pointer register *IP* contains offset within the code segment.
- The pointer register *BP* contains offset within the data segment.
- The pointer register *SP* contains offset within the stack segment.

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

The register *SI* is used to store the offset of source data in data segment.

The register *DI* is used to store the offset of destination in data or extra segment.

The index registers are particularly useful for string manipulation.

### Flag Register

The 8086 flag register contents indicate the results of computation in the *ALU*. It also contains some flag bits to control the *CPU* operations.

### 8086 Architecture:

The 8086 architecture supports

- a 16-bit ALU.
- a set of 16 bit registers
- provides segmented memory addressing scheme
- a rich instruction set.
- Powerful interrupt structure
- Fetched instruction queue for overlapped fetching and execution step.

The internal block diagram units inside the 8086 microprocessor is shown in the figure.

FIG

The architecture of 8086 can be divided into two parts

- (a) Bus Interface unit (BIU)
- (b) Execution unit (EU)

The bus interface unit is responsible for physical address calculations and a predecoding instruction byte queue ( 6 bytes long).

The bus interface unit makes the system bus signal available for external devices.

The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers, each 16-bits long.

### Generating a physical address:

- The content of segment register (segment address) is shifted left bit-wise four times.
- The content of an offset register (offset address) is added to the result of the previous shift operation.

These two operations together produce a 20-bit physical address.

For example, consider the segment address is 2010H and the offset address is 3535H.

The physical address is calculated as:

Segment Address	2010H	0010	0000	0001	0000
Shifted left by					

4 bit positions	0010 0000 0001 0000 0000
	+
offset address	0011 0101 0011 0101
	-----
physical address	0010 0011 0110 0011 0101
	2        3        6        3        5

The segment address by the segment value 2010H can have offset value from 0000H to FFFFH within it, ie. Maximum 64K locations may be accommodated in the segment.

The physical address range for this segment is from 20100H to 300FFH.

The segment register indicates the base address of a particular segment and CS, DS, SS and ES are used to keep the segment address.

The offset indicates the distance of the required memory location in the segment from the base address, and the offset may be the content of register IP, BP, SI, DI and SP.

Once the opcode is fetched and decoded, the external bus becomes free while the Execution Unit is executing the instruction.

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 called as predecoded instruction byte queue.

This is a 6 byte long queue, works in first-in first-out policy.

While the opcode is fetched by the bus interface unit (BIU), the execution unit (EU) executes the previously decoded instruction concurrently.

The execution unit contains.

- the register set of 8086 except segment registers and IF.
- a 16-bit ALU to perform arithmetic & logic operation
- 16-bit flag register reflects the results of execution by the ALU.
- the decoding units decodes the op-code bytes issued from the instruction byte queue.
- the timing and control unit generates the necessary control signals to execute the instruction op-code received from the queue.

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

Memory Segmentation:

The size of address bus of 8086 is 20 and is able to address 1 Mbytes ( $2^{20}$ ) of physical memory.

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

The addresses of the segment may be assigned as  $0000H$  to  $F000H$  respectively.

The offset values are from  $0000H$  to  $FFFFH$ .

If the segmentation is done as per above mentioned way, the segments are called non-overlapping segments.

In some cases segment may overlap also. Suppose a segment starts at a particular address and its maximum size can go up to 64 Kbytes. But if another segment starts before this 64 Kbytes location of the first segment, the two segments are said to be overlapping segment.

The area of memory from the start of the second segment to the possible end of the first segment is called as overlapped segment.

Non-overlapping segments and overlapping segments are shown in the figure.

Fig

#### Non-Overlapping Segments

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

- a) Allows the memory capacity to be 1 Mbyte although the actual addresses to be handled are of 16-bit size
- b) Allows the placing of code data and stack portions of the same program in different parts (segments) of memory, for data and code protection.
- c) Permits a program and/ or its data to be put into different areas of memory each time program is executed, ie, provision for relocation may be done.

#### Flag Register

A 16 flag register is used in 8086. It is divided into two parts .

- (a) Condition code or status flags
- (b) Machine control flags

The condition code flag register is the lower byte of the 16-bit flag register. The condition code flag register is identical to 8085 flag register, with an additional overflow flag.

The control flag register is the higher byte of the flag register. It contains three flags namely direction flag ( $D$ ), interrupt flag ( $I$ ) and trap flag ( $T$ ).

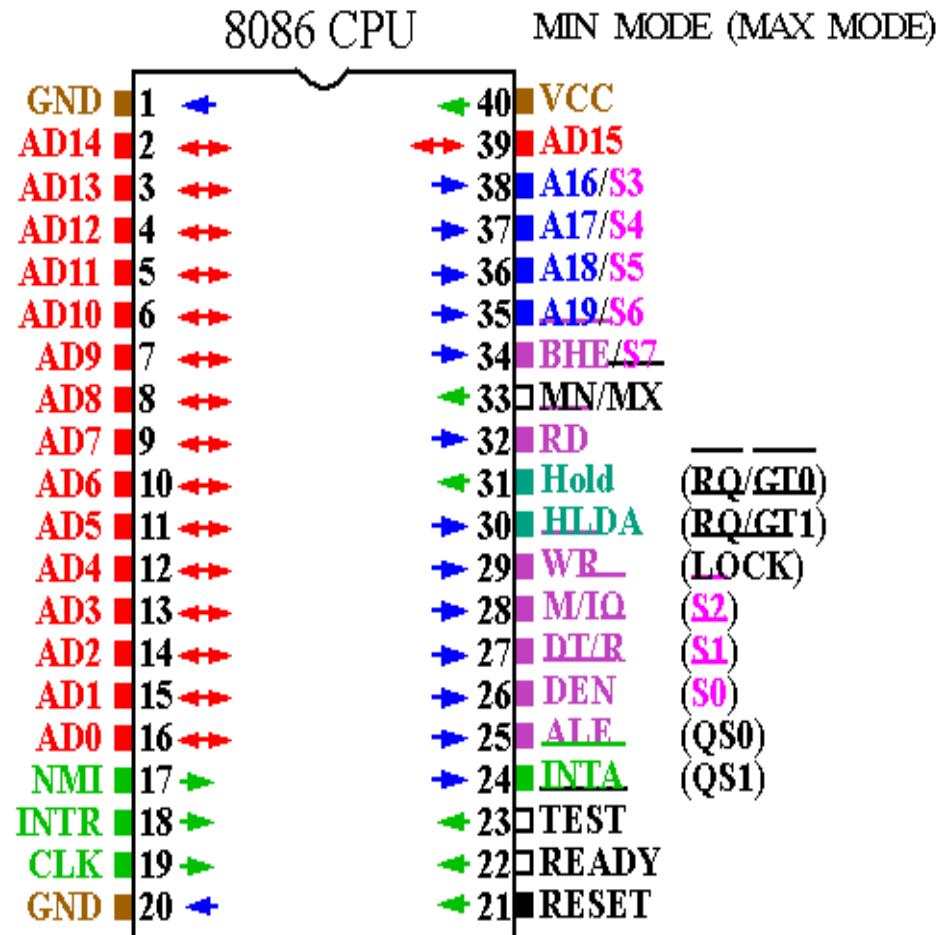
The complete bit configuration of 8086 is shown in the figure.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	0	D	I	T	S	Z	X	AC	X	P	X	CY

- S*- Sign Flag : This flag is set, when the result of any computation is negative.
- Z*- Zero Flag: This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.
- P*- Parity Flag: This flag is set to 1, if the lower byte of the result contains even number of 1's.
- C*- Carry Flag: This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.
- T*- Tarp Flag: If this flag is set, the processor enters the single step execution mode.
- I*- Interrupt Flag: If this flag is set, the maskable interrupt are recognized by the CPU, otherwise they are ignored.
- D*- Direction Flag: This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto decrementing mode.
- AC*-Auxiliary Carry Flag: This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.
- O*- Over flow Flag: This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

The 8086 is a 16-bit microprocessor. This microprocessor operates in single processor or multiprocessor configurations to achieve high performance. The pin configuration of 8086 is shown in the figure. Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multiprocessor mode).

FIGURE



#### Description of signals of 8086

$AD_7 - AD_0$  The address/ data bus lines are the multiplexed address data bus and contain the right most eight bit of memory address or data. The address and data bits are separated by using *ALE* signal.

$AD_{15} - AD_8$  The address/data bus lines compose the upper multiplexed address/data bus. This lines contain address bit  $A_{15} - A_8$  or data bus  $D_{15} - D_8$ . The address and data bits are separated by using *ALE* signal.

$A_{19} / S_6 - A_{18} / S_3$  The address/status bus bits are multiplexed to provide address signals  $A_{19} - A_{16}$  and also status bits  $s_6 - s_3$ . The address bits are separated from the status bits using the ALE signals. The status bit  $s_6$  is always a logic 0, bit  $s_5$  indicates the condition of the interrupt flag bit. The  $s_4$  and  $s_3$  combinedly indicate which segment register is presently being used for memory access.

$s_4$	$s_3$	Funtion
0	0	Extra segment
0	1	Stack segment
1	0	Code or no segment
1	1	Data Segment

$\overline{BHE} / S_7$  The bus high enable (BHE) signal is used to indicate the transfer of data over the higher order ( $D_{15} - D_8$ ) data bus. It goes low for the data transfer over  $D_{15} - D_8$  and is used to derive chip select of odd address memory bank or peripherals.

$\overline{BHE}$	$A_0$	Indication
0	0	Whole word
0	1	Upper byte from or to odd address
1	0	Lower byte from or to even address
1	1	None

$\overline{RD}$  : Read : whenever the read signal is at logic 0, the data bus receives the data from the memory or  $\pi 0$  devices connected to the system

**READY** :This is the acknowledgement from the slow devices or memory that they have completed the data transfer operation. This signal is active high.

**INTR**: Interrupt Request: Interrupt request is used to request a hardware interrupt of **INTR** is held high when interrupt enable flag is set, the 8086 enters an interrupt acknowledgement cycle after the current instruction has completed its execution.

$\overline{TEST}$  : This input is tested by “WAIT” instruction. If the **TEST** input goes low; execution will continue. Else the processor remains in an idle state.



**NMI-** Non-maskable Interrupt: The non-maskable interrupt input is similar to *INTR* except that the *NMI* interrupt does not check for interrupt enable flag is at logic 1, i.e, *NMI* is not maskable internally by software. If *NMI* is activated, the interrupt input uses interrupt vector 2.

**RESET:** The reset input causes the microprocessor to reset itself. When 8086 reset, it restarts the execution from memory location *FFFF0H*. The reset signal is active high and must be active for at least four clock cycles.

**CLK :** Clock input: The clock input signal provides the basic timing input signal for processor and bus control operation. It is asymmetric square wave with 33% duty cycle.

$V_{cc}$  +5V power supply for the operation of the internal circuit

**COND** Ground for the internal circuit

**$\overline{MN} / \overline{MX}$  :** The minimum/maximum mode signal to select the mode of operation either in minimum or maximum mode configuration. Logic 1 indicates minimum mode.

**Minimum mode Signals:** The following signals are for minimum mode operation of 8086.

**$\overline{M} / \overline{IO}$  -Memory/*IO***  $\overline{M} / \overline{IO}$  signal selects either memory operation or *IO* operation. This line indicates that the microprocessor address bus contains either a memory address or an *IO* port address. Signal high at this pin indicates a memory operation. This line is logically equivalent to  $\overline{S_2}$  in maximum mode.

**$\overline{INTA}$  - Interrupt acknowledge:** The interrupt acknowledge signal is a response to the *INTR* input signal. The  $\overline{INTA}$  signal is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.

**ALE-** Address Latch Enable: This output signal indicates the availability of valid address on the address/data bus, and is connected to latch enable input of latches.

**$\overline{DT} / \overline{R}$  :** Data transmit/Receive: This output signal is used to decide the direction of data flow through the bi-directional buffer.  $\overline{DT} / \overline{R} = 1$  indicates transmitting and  $\overline{DT} / \overline{R} = 0$  indicates receiving the data.

**$\overline{DEN}$**  Data Enable: Data bus enable signal indicates the availability of valid data over the address/data lines.

**HOLD:** The hold input request a direct memory access(DMA). If the hold signal is at logic 1, the micro process stops its normal execution and places its address, data and control bus at the high impedance state.

**HLDA:** Hold acknowledgement indicates that 8086 has entered into the hold state.

**Maximum mode signal:** The following signals are for maximum mode operation of 8086.

$\overline{s_2}, \overline{s_1}, \overline{s_0}$  - Status lines: These are the status lines that reflect the type of operation being carried out by the processor.

These status lines are encoded as follows

$\overline{s_2}$	$\overline{s_1}$	$\overline{s_0}$	Function
0	0	0	Interrupt Acknowledge
0	0	1	Read $\pi o$ port
0	1	0	Write $\pi o$ port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	passive

$\overline{LOCK}$  : The lock output is used to lock peripherals off the system, ie, the other system bus masters will be prevented from gaining the system bus.

$\overline{QS_1}$  and  $\overline{QS_0}$  - Queue status: The queue status bits shows the status of the internal instruction queue. The encoding of these signals is as follows

$\overline{QS_1}$	$\overline{QS_0}$	Function
0	0	No operation, queue is idle
0	1	First byte of opcode
1	0	Queue is empty
1	1	Subsequent byte of opcode

$\overline{RQ/GT1}$  and  $\overline{RQ/GT0}$  - request/Grant: The request/grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processors current bus cycle. These lines are bi-directional and are used to both request and grant a *DMA* operation.  $\overline{RQ/GT0}$  is having higher priority than  $\overline{RQ/GT1}$

### Physical Memory Organization

In an 8086 based system, the 1Mbyte memory is physically organized as odd bank and even bank, each of 512kbytes, addressed in parallel by the processor.

Byte data with even address is transferred on  $D_7 - D_0$  and byte data with odd address is transferred on  $D_{15} - D_8$ .

The processor provides two enable signals,  $\overline{BHE}$  and  $A_0$  for selecting of either even or odd or both the banks.

FIG			
$\overline{BHE}$	$A_0$	Function	$D_{15} - D_0 / A_{15} - A_0$ are the common signal line in 8086 as $AD_{15} - AD_0$
0	0	Whole word	
0	1	Upper byte/ odd address	
1	0	Lower byte/even address	
1	1	none	

If the address bus contains  $FFFFO_H$

If  $\overline{BHE} = 0$ , then it reads the 16 bits data from memory location  $FFFFO_H$  and  $FFFF1_H$ .

If  $\overline{BHE} = 1$ , it reads the 8 bits data from memory location  $FFFFO_H$