8086/8088 processors – Architecture, Pin Diagrams & Timing Diagrams

→ The 8085 microprocessor was a fully functional complete microprocessor

→ The main limitations of 8085 microprocessor are

- → low speed
- → low memory addressing capability
- → limited no. of general purpose registers
- → less powerful instruction set

→ The above limitations of the 8085 microprocessor pushed the designer to build more powerful microprocessors in terms of advanced architecture, more processing capability, larger memory addressing capability, a more powerful instruction set.

8086 was the result of such developmental design efforts.

→ Intel 8086 – 16 bit microprocess → 1978

→ The peripheral chip earlier designed for 8085 were compatible with 8086 with slight or no modifications.

→ Memory interfacing techniques were similar, but includes use of a few additional signals.

Register Organization of 8086

→ 8086 has a powerful set of register known as general purpose and special purpose registers.
All of them are 16 bit registers.

→ The general purpose registers are used as either 8 bit registers or 16 bit registers. They can be used for holding data, variables and results (intermediate) temporarily or for other purposes like a counter or for storing offset address for some particular addressing mode etc.

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

Registers are categorized into four groups:

(1) General Data Registers (2) Segment Registers

(3) Pointers and Index Registers (4) Flag Registers.

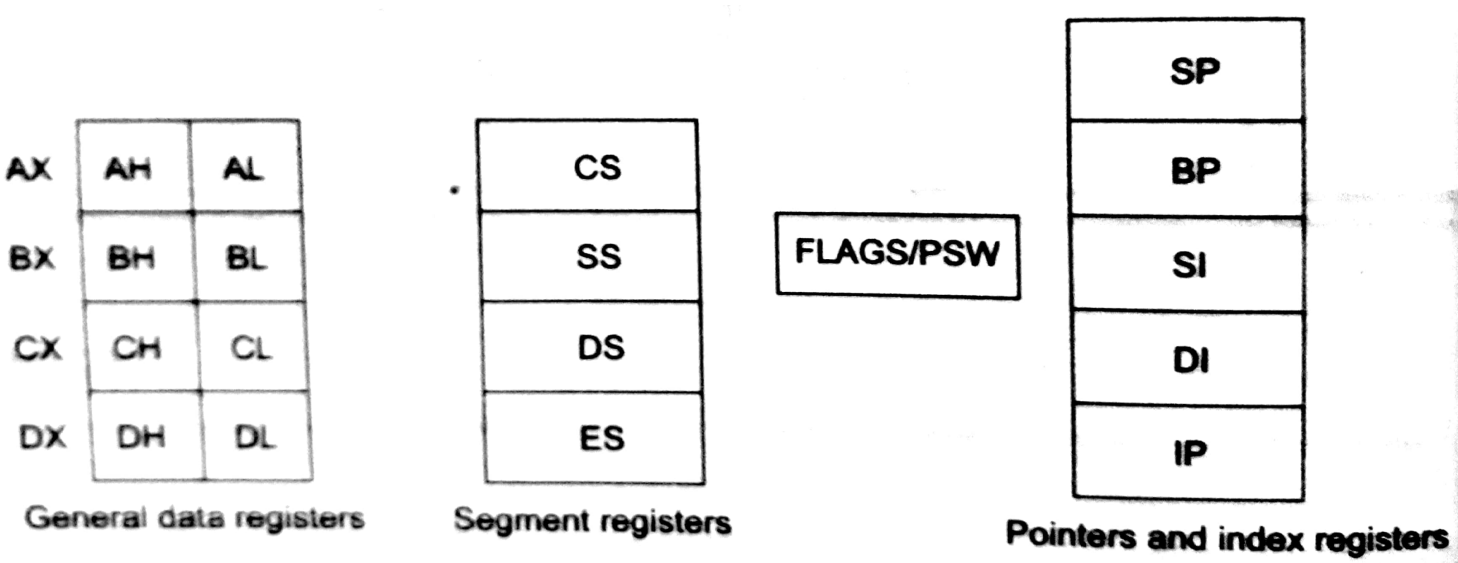


Fig. 1.1 ***Register organisation of 8086***

(1) General data Register

→ letters L, H specify the lower or higher bytes of a particular register.

CH → higher 8 bits of CX register

CL → lower 8 bit of CX register

X → used to specify the complete 16 bit Register.

→ CX → used as default counter in case of string and loop instructions.

→ BX used as an offset storage for forming the physical address in case of certain addressing modes.

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

→ AX is used as 16 bit accumulator, with lower 8 bits designated as AL and higher 8 bits as AH.

AL can be used as an 8 bit accumulator for 8 bit operations.

This is the most important general purpose register having multiple functions.

(2) Segment Register

→ 8086 addresses a segmented memory.

→ The 1M Byte of memory of 8086 microprocessor, which it can address is divided into 16 logical segments.

Each segment thus contains 64 k Bytes of memory.

→ There are four segment registers:

- Code Segment Register (CS)
- Data Segment Register (DS)
- Extra Segment Register (ES)
- Stack Segment Register (SS)

→ The CS register is used for addressing a memory location in code segment of the memory, where executable program is stored.

→ The Data segment register points to the data segment of memory where the data resides.

→ The extra segment also refers to a segment which essentially is another data segment of memory. Thus extra segment also contains data.

→ The stack ~~segment~~ register is used to ~~store~~ for addressing stack segment of memory. The CPU uses stack for temporarily storing important data like contents of CPU registers etc.

→ 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 the 16 bit base addresses related to different segments.

Any of the pointers, 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.

→ Thus, CS, DS, ES, SS registers contain the segment addresses for code, data, extra and stack segments of memory.

→ All these segments are logical segments. They may or may not be physically seperated. In other words, a single segment may require more than one memory chip or more than one segment may be accomodated in a single memory chip.

③ Pointers and Index Registers:

→ The pointers contain offset within the particular segment.

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

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

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

→ The index registers are particularly useful for string manipulation.

④ Flag Register:

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

Architecture

8086 architecture provides a number of improvements over 8085 architecture.

1. Support a 16 bit ALU
2. contains a set of 16 bit registers
3. segmented memory addressing capability
4. rich instruction set
5. powerful interrupt structure
6. fetched instructions queue for overlapped fetching and execution etc.

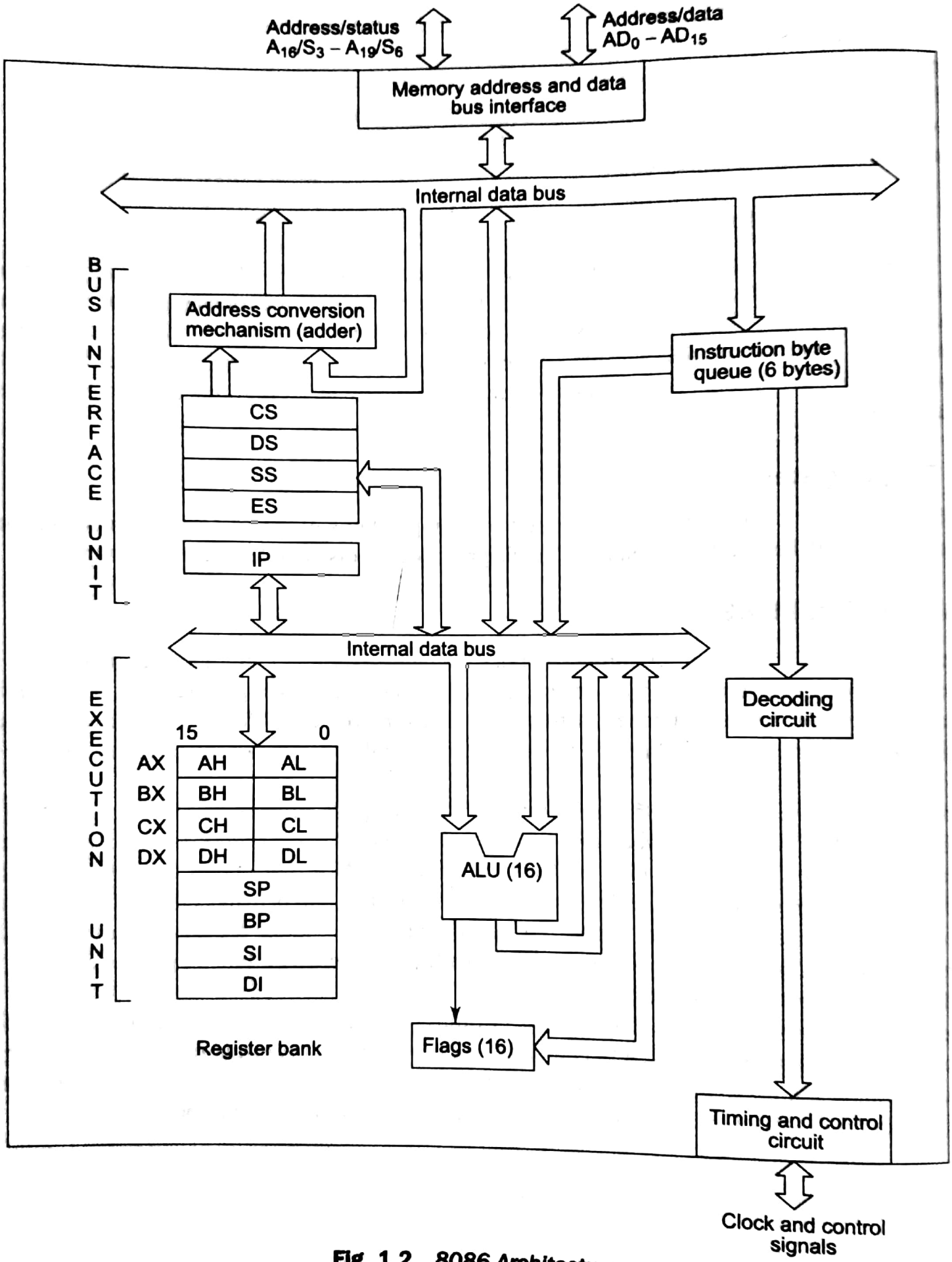


Fig. 1.2 *8086 Architecture*

→ The complete architecture of 8086 can be divided into two parts

(a) Bus Interface Unit (BIU) and

(b) Execution Unit (EU)

The bus interface unit makes the system's bus signals available for external interfacing of the device. In other words, this unit is responsible for establishing communications with external devices and peripherals including memory via the bus.

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

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

→ For generating physical address from content of these two registers, the content of segment register also called segment address is shifted left bitwise four times and to this result, the content of an offset register also called offset address is added to get a 20 bit physical address.

Ex:-

Segment address → 1005H

offset address → 5555H

Segment address → 1005H → 0001 0000 0000 0101

shifted by 4 bit position → 0001 0000 0000 0101 0000

+ 0101 0101 0101 0101

offset address

0001 0101 0101 0101 0101

1 5 5 A 5

∴ physical address is 155A5 H.

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

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

→ The Bus Interface Unit (BIU) has a separate adder to perform this procedure for obtaining a physical address while addressing memory.

→ The segment address value is to be taken from the appropriate segment register depending on 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 on the addressing mode.

→ In the case of 8085, once the opcode is fetched and decoded, the external bus remains free for some time, while the processor internally executes the instruction. This time slot is utilized in 8086 to achieve the overlapped fetch and execute 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 predecoded instruction byte queue. It is 6 bytes long FIFO structure.

→ The instructions from the queue are taken for decoding sequentially. Once a byte a decoded, the queue is rearranged by pushing it out and the queue status is checked for the possibility of read opcode fetch cycle.

→ While the opcode is fetched by the BIU, the execution Unit (EU) executes the previously decoded instruction concurrently.

→ The BIU unit along with EU thus form a pipeline.

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

It has a 16 bit ALU, able to perform arithmetic and logic operations.

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

→ The decoding unit decodes the opcode bytes issued from instruction byte queue.

→ The timing and control unit derives the necessary control signals to execute the instruction opcode received from the queue.

Memory Segmentation:

→ 8086/8088 based systems use segmented memory organization.

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

Each segment is 64K bytes in size and addressed by one of the segment registers.

The 16 bit contents of a segment register actually point to a 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 64K locations.

→ The CPU 8086 is able to address 1M bytes of physical memory.

$$1M \text{ bytes} = 2^{20}$$

$$2^{20} = 2^4 \times 2^{16}$$

$$= 2^4 \times 2^6 \times 2^{10}$$

$$= 16 \times 64K$$

∴ 1M bytes can be divided into 16 segments, each of 64K bytes size.

∴ The address of 16 segments can be assigned as 0000H to F000H.

The offset address can range from 0000H - FFFFH

∴ the physical addresses range from 00000H to FFFFFH

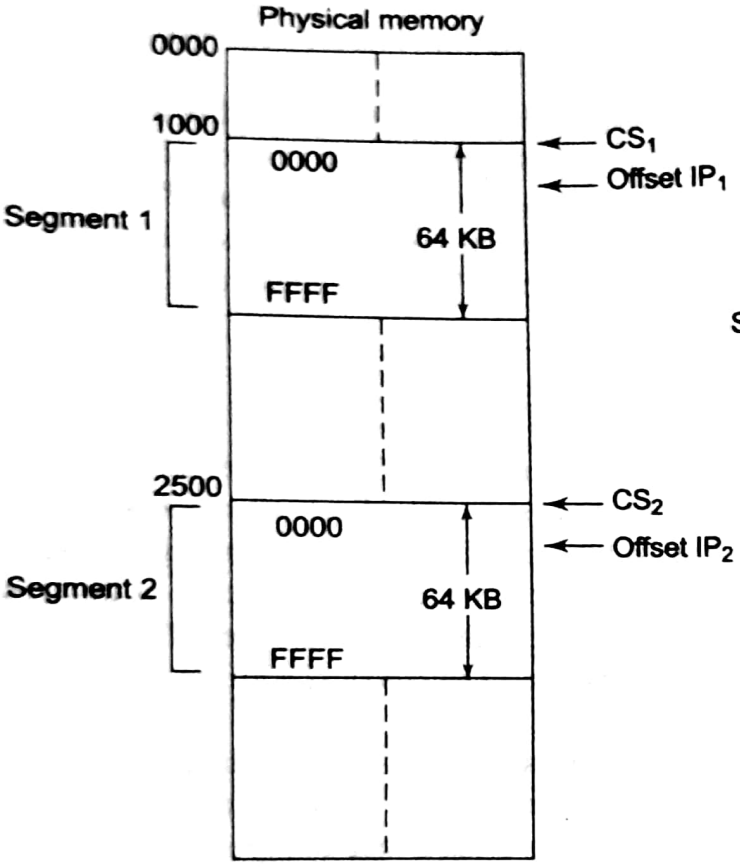


Fig. 1.3(a) *Non-overlapping Segments*

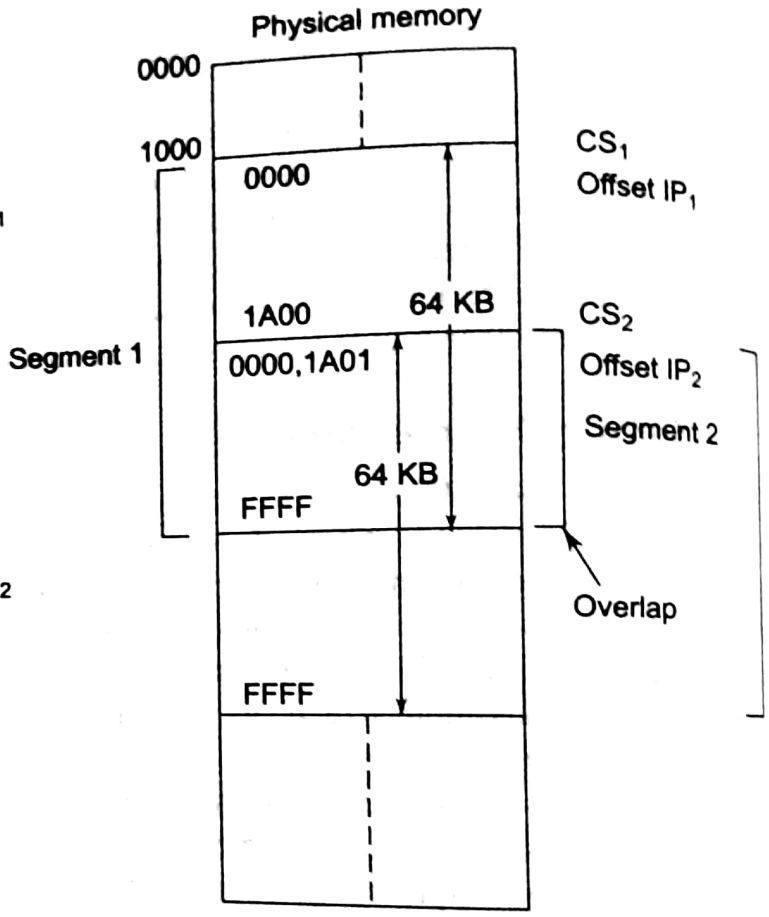


Fig. 1.3(b) *Overlapping Segments*

→ In the above said case, the segments are called non-overlapping segments.

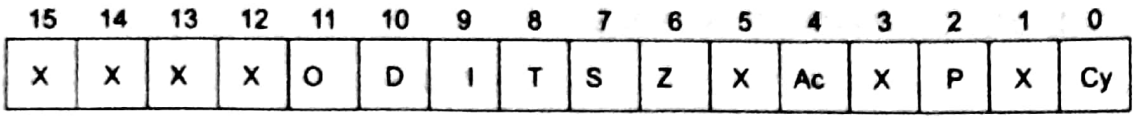
→ In some cases, the segments may be overlapping. Suppose a segment starts at a particular address and its max. size can be 64 Kbytes.
But if another segment starts before this 64 K byte locations of the first segment, the two segments are said to be overlapping segments.
The area of memory from the start of second segment to the possible end of first segment is called overlapped segment area.

→ 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.

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 1Mbytes although the actual addresses to be handled are of 16-bit size
2. Allows the placing of code, data and stack portions of the same program in different parts (segments) 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 = $CS_1 + IP_1 = CS_2 + IP_2$, where '+' indicates the procedure of physical address formation.



O – Overflow flag
D – Direction flag
I – Interrupt flag
T – Trap flag
S – Sign flag
Z – Zero flag
Ac – Auxiliary carry flag
P – Parity flag
Cy – Carry flag
X – Not used

Fig. 1.4 *Flag Register of 8086*

The description of each flag bit is as follows:

S-Sign Flag This flag is set when the result of any computation is negative. For signed computations, the sign flag equals the MSB of the result.

Z-Zero Flag This flag is set if the result of the computation or comparison performed by the previous instruction/instructions is zero.

P-Parity Flag This flag is set to 1 if the lower byte of the result contains even number of 1s.

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. For example, when two numbers are added, a carry may be generated out of the most significant bit position. The carry flag, in this case, will be set to '1'. In case, no carry is generated, it will be '0'. Some other instructions also affect or use this flag and will be discussed later in this text.

T-Trap Flag If this flag is set, the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

I-Interrupt Flag If this flag is set, the maskable interrupts are recognised 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. *autoincrementing mode*. Otherwise, the string is processed from the highest address towards the lowest address, i.e. *autodecrementing mode*. We will describe string manipulations later in chapter 2 in more detail.

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-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, i.e. 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 the overflow flag will be set.

Signal Descriptions of 8086:

→ The 8086 microprocessor is a 16 bit CPU available in three clock rates i.e. 5, 8 and 10MHz packaged in a 40 pin DIP package.

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

→ Of the 40 pins, some of the pins serve a particular function in minimum mode (single processor mode) and others functions in maximum mode (multiprocessor mode).

→ The 8086 signals are categorized into three groups.

1. Signals having common functions in minimum as well as maximum mode
2. Signals having special functions for minimum mode
3. Signals having special function for maximum mode.

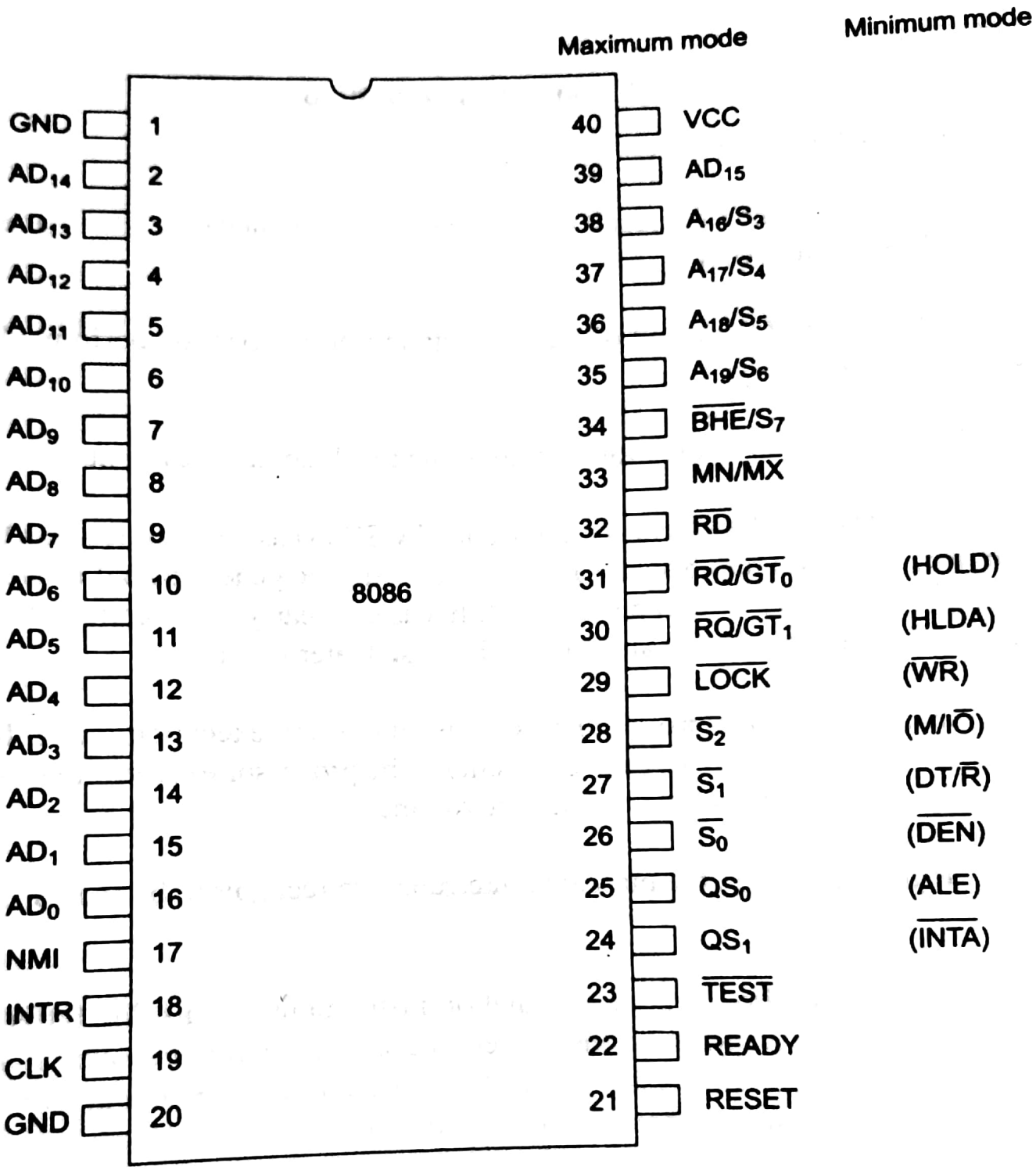


Fig. 1.5 *Pin Configuration of 8086*

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

$AD_{15} - AD_0$: These are time multiplexed memory I/O Address and data lines

Address remains on the line during T_1 state, while data is available during T_2, T_3, T_W, T_4.

T_1, T_2, T_3, T_W are clock states of a machine cycle.

T_W → wait state.

These lines are active high and float to a ~~Hi~~ hi state during interrupt acknowledge and local bus hold acknowledge cycles.

$A_{19}|S_6$, $A_{18}|S_5$, $A_{17}|S_4$, $A_{16}|S_3$: These are time multiplexed address and status lines.

During T_1, 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 info is available in T_2, T_3, T_W & T_4

The status of interrupt enable flag bit (S_5) is updated at the beginning of each clock cycle.

The S_4, S_3 together indicate which segment register is being used for memory access.

S_4	S_3	Indications
0	0	Alternate Data
0	1	Stack
1	0	code or none
1	1	Data

The address bits ~~and~~ are seperated from status bits using latches controlled by the ALE signal.

$\overline{BHE}/S_7$ – Bus high enable/Status.

The bus high enable signal is used to indicate the transfer of data over $D_{15}-D_8$ data bus. It goes low for data transfers over $D_{15}-D_8$ and is used to derive chip selects of odd address memory bank or peripherals.

$\overline{BHE}$ is low during T_1 for R/W, interrupt acknowledge cycles

Whenever byte is to be transferred over high order lines.

State information is available
during T_2, T_3 and T_4.

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

$\overline{RD}$ - Read → $\overline{RD}$ signal when low indicates the peripherals that the processor is performing memory or I/O Read operation.

$\overline{RD}$ is active low and shows the state for T_2, T_3, T_W of any read cycle.

Signal is tristated during "hold acknowledge".

READY → This is the acknowledgement from slow devices or memory that they have completed the data transfer

The signal made available by the devices is synchronized by 8284A clock generator to provid ready input to 8086.

The signal is active high.

INTR → Interrupt Request

This is a level triggered input

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

If any 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 and internally synchronized.

$\overline{TEST}$ → This input is examined by a WAIT instruction. If the $\overline{TEST}$ input goes low, execution will continue else processor remains in an idle state.
The input is synchronized internally during each clock cycle on leading edge of clock.

NMI — Non maskable interrupt:
This is an edge triggered input which causes a Type 2 interrupt. NMI is not maskable by software. A transition from low to high initiates the interrupt response at the end of the current instruction. This input is internally synchronized.

RESET → This input causes the processor to terminate the current activity and start execution from FFFF0H.
The signal is active high and must be high for atleast four clock cycles.
It restarts execution when the RESET returns low.

CLK → clock input

This provides the basic timing for processor operation and bus control activity.

It is an asymmetric square wave with 33% duty cycle.

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

Vcc +5V power supply for the operation of internal circuit.

GND ground for internal circuit.

MN/MX The logic level at this pin decides whether the processor is to operate either in minimum (single processor) or maximum (multiprocessor) mode.

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

M/$\overline{IO}$ — Memory/IO :

→ This is a status line logically equivalent to $\overline{S_2}$ in maximum mode.
When it is low, it indicates CPU having an I/O operation
When it is high, it indicates CPU having memory operation.
This line becomes active in previous T_4 and remains active till final T_4 of current cycle.
It is tristated during local bus "hold acknowledge".

$\overline{INTA}$ — Interrupt Acknowledge :

→ This signal is used as a read strobe for interrupt acknowledge cycles
when it goes low, it means that the processor has accepted the interrupt.
It is active low during T_2, T_3 and T_W of each interrupt acknowledge cycle.

ALE – Address Latch Enable :

This output signal indicates the availability of valid address on the address/data lines and is connected to the latch enable input of latches.
The signal is active high and is never tristated.

DT/$\overline{R}$ – Data transmit/Receive :

This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers).
When the processor sends out data, this signal is high.
When the processor is receiving data, this signal is low.
Logically this is equivalent to $\overline{S_1}$ in maximum mode.
This is tristated during "hold acknowledge".
Timing is same as M/$\overline{IO}$

$\overline{DEN}$ – Data Enable :

This signal indicates availability of valid data over the address/data lines.
It is used to enable the transreceivers to seperate the data from multiplexed address/data signal.
It is active from middle of T_2 to middle of T_4.
It is tristated during "hold acknowledge".

HOLD, HLDA – Hold / Hold Acknowledge :

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

→ The processor, after receiving the HOLD request issues hold acknowledge (HLDA) signal on HLDA pin, in the middle of next clock cycle after completing the current bus (instruction) cycle.

→ At the same time the processor floats the local bus and control lines.

→ When the processor detects HOLD line low, it lowers HLDA signal. HOLD is an asynchronous input and it should be externally synchronized.

→ When DMA request is made while CPU is performing memory or I/O cycle, it will release local bus during T_4 provided:

1. Request occurs on or before T_2 state of current cycle
2. The current cycle is not operating over the lower byte of a word.
3. The current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A lock instruction is not being executed.

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

$\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ - Status lines:

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

These become active during T_4 of the previous cycle and remain active during T_1 and T_2 of the current bus cycle.

The status lines become passive during T_3 of current bus cycle so that they may again become active for the next bus cycle during T_4.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	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.

$\overline{LOCK}$ →

This output pin indicates that other system bus masters will be prevented from gaining access to system, while $\overline{LOCK}$ is low.

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

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

QS_1, QS_0 → 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.

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 modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of pipelined processing of instructions.

→ The 8086 architecture has a 6 byte instruction prefetch queue. Thus, even the largest (6 bytes) instruction can be prefetched from memory and stored in the prefetch queue. This results in faster execution of instructions.

In 8085, an instruction is fetched, decoded and executed and only after the execution of this instruction, next one is fetched.

By prefetching the instruction in 8086, there is a considerable speeding up of instruction execution in 8086. This scheme is known as Instruction pipelining.

$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ — Request / Grant :

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

Each of pins is bidirectional, with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$.

The request grant sequence is as follows :

1. A pulse one clock wide from another bus master request bus access to 8086.
2. During T_4 (current) or T_1 (next) clock cycle, a pulse one clock wide from 8086 to the requesting master indicates that 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state on next clock cycle.
3. A one clock wide pulse from another master indicates to 8086 that 'hold' request is about to end and 8086 may regain control of local bus at next clock cycle.

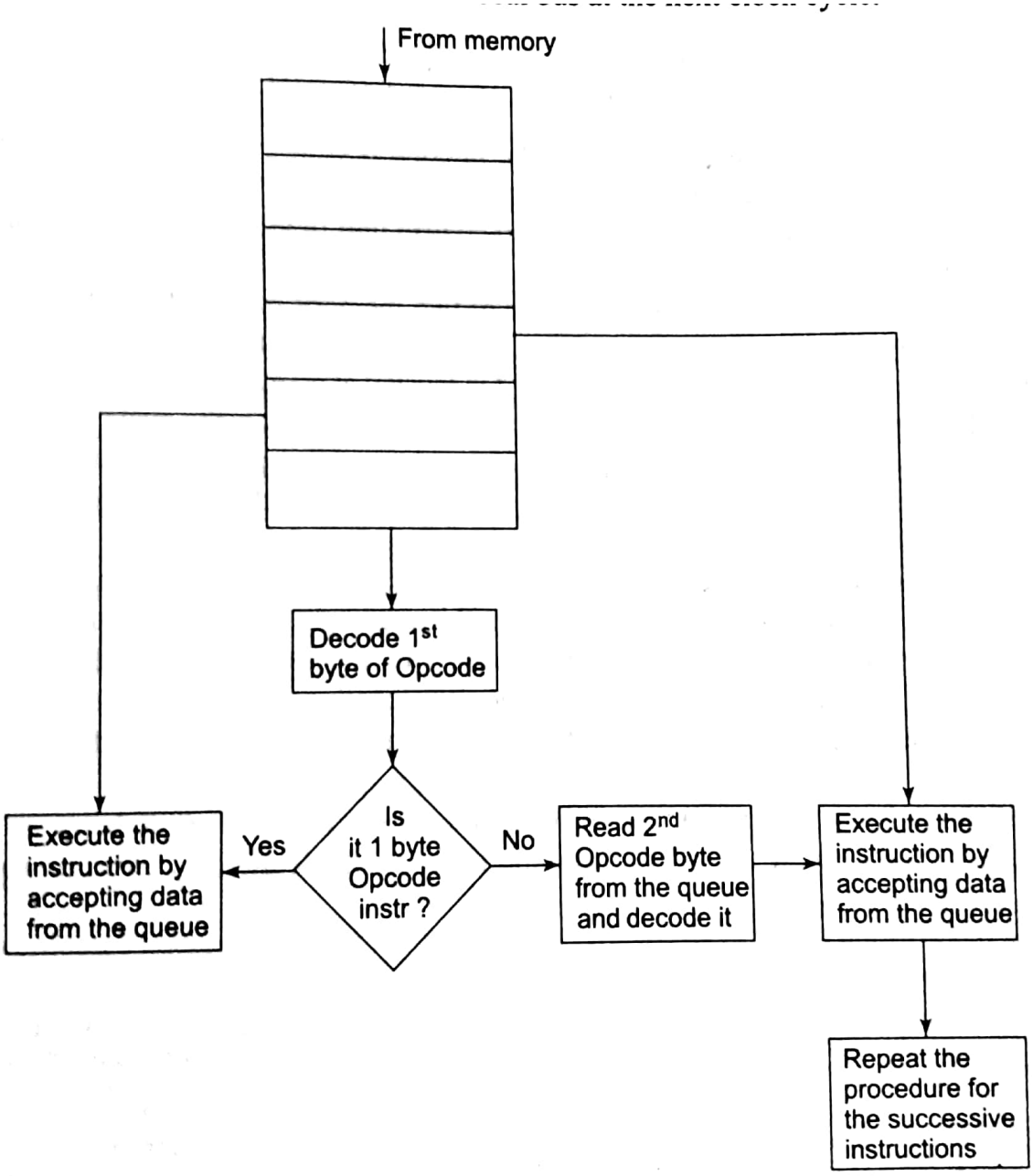


Fig. 1.6 *The Queue Operation*

Physical Memory Organization:

→ In an 8086 based system, the 1M byte memory is physically organized as an odd bank and an even bank, each of 512 K bytes, addressed in parallel by the processor.

→ Byte data with an even address is transferred on $D_7 - D_0$, while byte data with an odd address is transferred on $D_{15} - D_8$ bus lines.

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

$\overline{BHE}$	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

→ If the processor fetches a word (consecutive two bytes) from memory, there are possibilities like

1. Both the bytes may be data operands
2. Both the bytes may be opcode bits
3. One of the bytes may be opcode while the other may be data.

The above possibilities are taken care by the decoder circuit of the microprocessor. The opcodes and operands are identified by the internal decoder circuit, which further derives signals that act as input to the timing and control unit. The timing and control unit then derives all the signals required for execution of the instruction.

A map of an 8086 memory system starts at 00000H and ends at FFFFFH. 8086 being a 16-bit processor is expected to access 16-bit data to / from 8-bit commerically available memory chips in parallel, as shown below in Fig. 1.7.

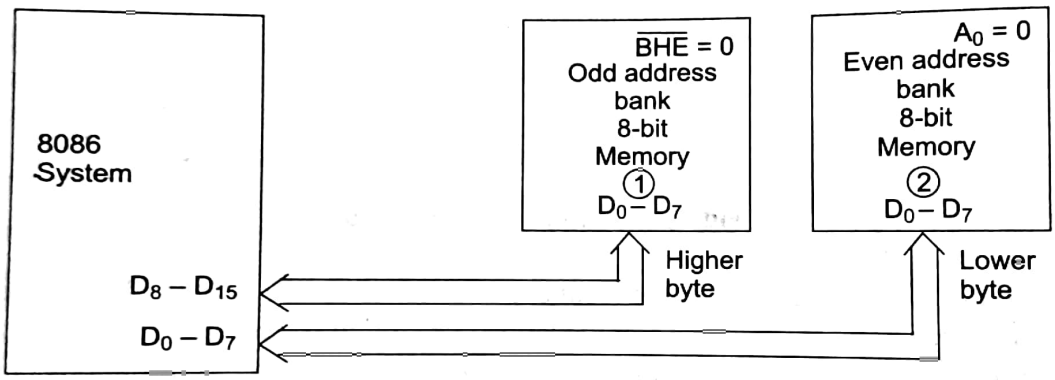


Fig. 1.7 *Physical Memory Organisation*

→ While referring to word data, the BIU requires one or two memory cycles depending on whether the starting byte is located at an even or odd address.

→ It is always better to locate the word data at an even address. To read/write a complete word from/to memory, if it is located at an even address, one read or write cycle is required. If the word is located at an odd address, the first read or write cycle is required for accessing the lower byte while second cycle is used for accessing upper byte.

→ It should be kept in mind, that while initializing the structures like stack, they should be initialized at an even address for efficient operation.

→ Thus, if it is imagined that the complete memory map of 16 bit 8086 is filled with 16 bit data, all the lower bytes ($D_0 - D_7$) will be stored at even addresses memory bank (1) and all higher order bytes ($D_8 - D_{15}$) to be stored at odd addresses memory bank (2).

→ If 8086 transfers a 16 bit data to/from memory, both of these banks must be selected for 16 bit operation.

→ To maintain an upward compatibility with 8085, 8086 must be able to implement 8 bit operations.

In which case, there can be two possibilities:

1. an 8 bit operation with even address memory bank
2. an 8 bit operation with odd address memory bank i.e. with odd address.

The two signals A_0 and $\overline{BHE}$ solve the problem of selection of appropriate memory banks.

problem 8

→ Certain locations in memory are reserved for specific CPU operations.

① The locations from FFFF0H to FFFFFH are reserved for operations including jump to initialization programme and I/O processor initialization.

② The locations 00000H to 003FFH are reserved for interrupt vector table. The interrupt structure provides space for a total of 256 interrupt vectors.
The vectors, CS and IP for each interrupt routine require 4 bytes for storing it in the interrupt

vector table.

Hence 256 types of interrupts require
256 x 4 = 1K bytes locations
Hence the final address of 03FFH for
the complete interrupt vector table.