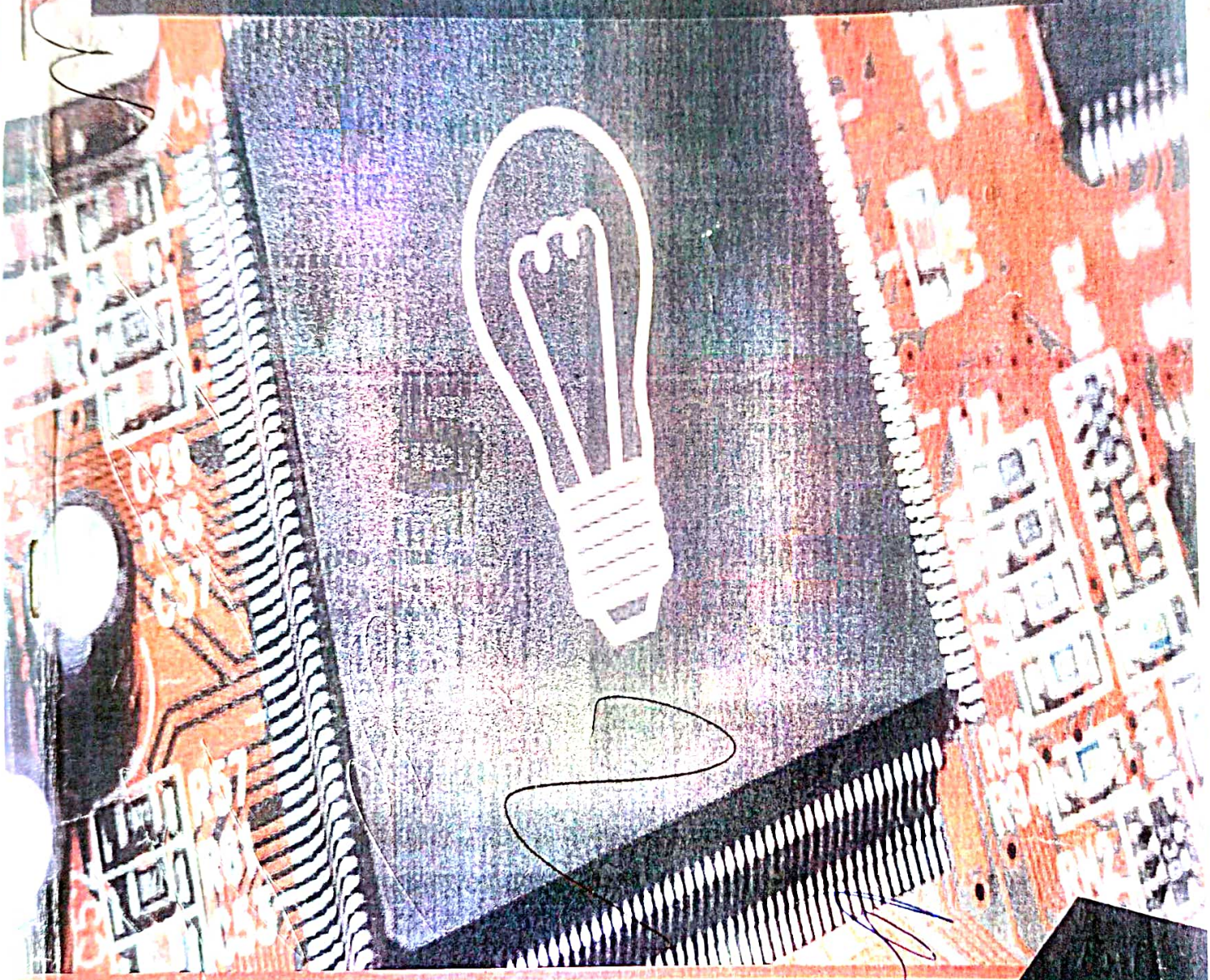
TOPPER'S SOLUTIONS

....In Search of Another Topper



MICROPROCESSOR

(COMPUTER)

5 SEM

As per Revised Syllabus w.e.f 2018-19

Jul 2019 Edition

TOPPER'S SOLUTIONS

....In Search of Another Topper

There are many existing paper solution available in market, but Topper's Solution is the one which students will always prefer if they refer... ;) Topper's Solutions is not just paper solutions, it includes many other important questions which are important from examination point of view. Topper's Solutions are the solution written by the Toppers for the students to be the upcoming Topper of the Semester.

It has been said that **"Action Speaks Louder than Words"** So Topper's Solutions Team works on same principle. Diagrammatic representation of answer is considered to be easy & quicker to understand. So our major focus is on diagrams & representation how answers should be answered in examinations.

Why Topper's Solutions:

- Point wise answers which are easy to understand & remember.
- Diagrammatic representation for better understanding.
- Additional important questions from university exams point of view.
- Covers almost every important question.
- In search of another topper.

"Education is Free.... But its Technology used & Efforts utilized which we charge"

It takes lot of efforts for searching out each & every question and transforming it into Short & Simple Language. Entire Community is working out for betterment of students, do help us.

Thanks for Purchasing & Best Luck for Exams

---- In Association with BackkBenchers Community ----

BackkBenchers
Community

Syllabus:

Exam	TT-1	TT-2	AVG	Term Work	Oral/Practical	End of Exam	Total
Marks	20	20	20	25	25	80	150

#	Module	Details Contents	No.
1.	The Intel Microprocessors 8086/8088 Architecture.	▪ 8086/8088 CPU Architecture, Programmer's Model ▪ Functional Pin Diagram ▪ Memory Segmentation ▪ Banking in 8086 ▪ Demultiplexing of Address/Data bus ▪ Study of 8284 Clock Generator ▪ Study of 8288 Bus Controller ▪ Functioning of 8086 in Minimum mode and Maximum mode ▪ Timing diagrams for Read and Write operations in minimum and maximum mode	01
2.	Instruction Set and Programming.	▪ Addressing Modes ▪ Instruction set – Data Transfer Instructions, String Instructions, Logical Instructions, Arithmetic Instructions, Transfer of Control Instructions, Processor Control Instructions ▪ Assembler Directives and Assembly Language Programming, Macros, Procedures ▪ Mixed Language Programming with C Language and Assembly Language. ▪ Programming based on DOS and BIOS Interrupts (INT 21H, INT 10H)	20
3.	8086 Interrupts.	▪ Types of interrupts ▪ Interrupt Service Routine ▪ Interrupt Vector Table ▪ Servicing of Interrupts by 8086 microprocessor ▪ Programmable Interrupt Controller 8259 – Block Diagram, Interfacing the 8259 in single and cascaded mode, Operating modes, programs for 8259 using ICWs and OCWs	33
4.	Peripherals and their interfacing with 8086.	▪ Memory Interfacing - RAM and ROM Decoding Techniques – Partial and Absolute. ▪ 8255-PPI – Block diagram, Functional PIN Diagram, CWR, operating modes, interfacing with 8086. ▪ 8253 PIT - Block diagram, Functional PIN Diagram, CWR, operating modes, interfacing with 8086. ▪ 8257-DMAC – Block diagram, Functional PIN Diagram, Register organization, DMA operations and transfer modes	41
5.	Intel 80386DX Processor.	▪ Architecture of 80386 microprocessor. ▪ 80386 registers – General purpose Registers, EFLAGS and Control registers. ▪ Real mode, Protected mode, virtual 8086 mode. ▪ 80386 memory management in Protected Mode – Descriptors and selectors, descriptor tables, the memory paging mechanism	64
6.	Pentium Processor.	▪ Pentium Architecture Superscalar Operation, Integer & Floating Point Pipeline Stages, Branch Prediction Logic, Cache Organization and MESI Model	79

Marks Distribution:

#	Chapter	Dec 2018	May 2019
1	The Intel Microprocessors 8086/8088 Architecture	25	25
2	Instruction Set and Programming	25	35
3	8086 Interrupts	15	15
4	Peripherals and their interfacing with 8086	25	25
5	Intel 80386DX Processor	15	15
6	Pentium Processor	15	15
-	**Repeated Questions**	-	**50**

CHAP - 1: THE INTEL MICROPROCESSORS 8086/8088 ARCHITECTURE

Q1. Explain segmentation of 8086 microprocessor. Give its advantages.

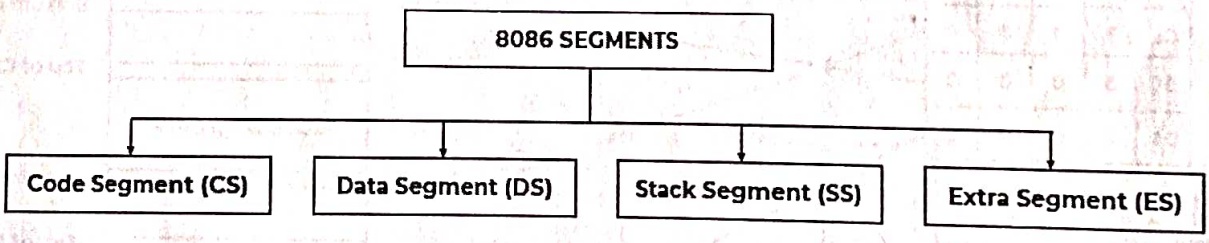
Ans: **[5-10M | Dec18 & May19] [P | High]**

SEGMENTATION:

1. Memory is considered as vast collection of bytes, so it must be organized in efficient manner.
2. The total memory size is divided into segments of various sizes.
3. A segment is just **an area in memory**.
4. The process of dividing memory this way is called **Segmentation**.

8086 SEGMENTATION:

1. In memory, data is stored as bytes.
2. Each byte has a specific address.
3. Intel 8086 has 20 lines address bus.
4. With 20 address lines, the memory that can be addressed is 2^{20} bytes.
5. 2^{20} = 1,048,576 bytes (1 MB).
6. 8086 can access memory with address ranging from 00000 H to FFFFF H.
7. In 8086, memory has **four** different types of segments.
8. These are: Code Segment (CS), Data Segment (DS), Stack Segment (SS) & Extra Segment (ES).



9. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations.
10. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.

Code Segment: It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

Data Segment: It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address.

Stack Segment: It handles memory to store data and addresses during execution.

Extra Segment: ES is additional data segment, which is used by the string to hold the extra destination data.

NEED FOR SEGMENTATION:

1. The number of address lines in 8086 is 20.
2. 8086 BIU will send 20 bit address, so as to access one of the 1MB memory locations.
3. The four segment registers actually contain the upper 16 bits of the starting addresses of the four memory segments of 64 KB each.
4. A segment is a logical unit of memory that may be up to 64 kilobytes long.
5. Each segment is made up of contiguous memory locations.
6. It is independent, separately addressable unit.
7. Starting address will always be changing.
8. It will not be fixed.
9. Below figure 1.1 is the one way of positioning four 64 kilobyte segments within the 1M byte memory space of an 8086.

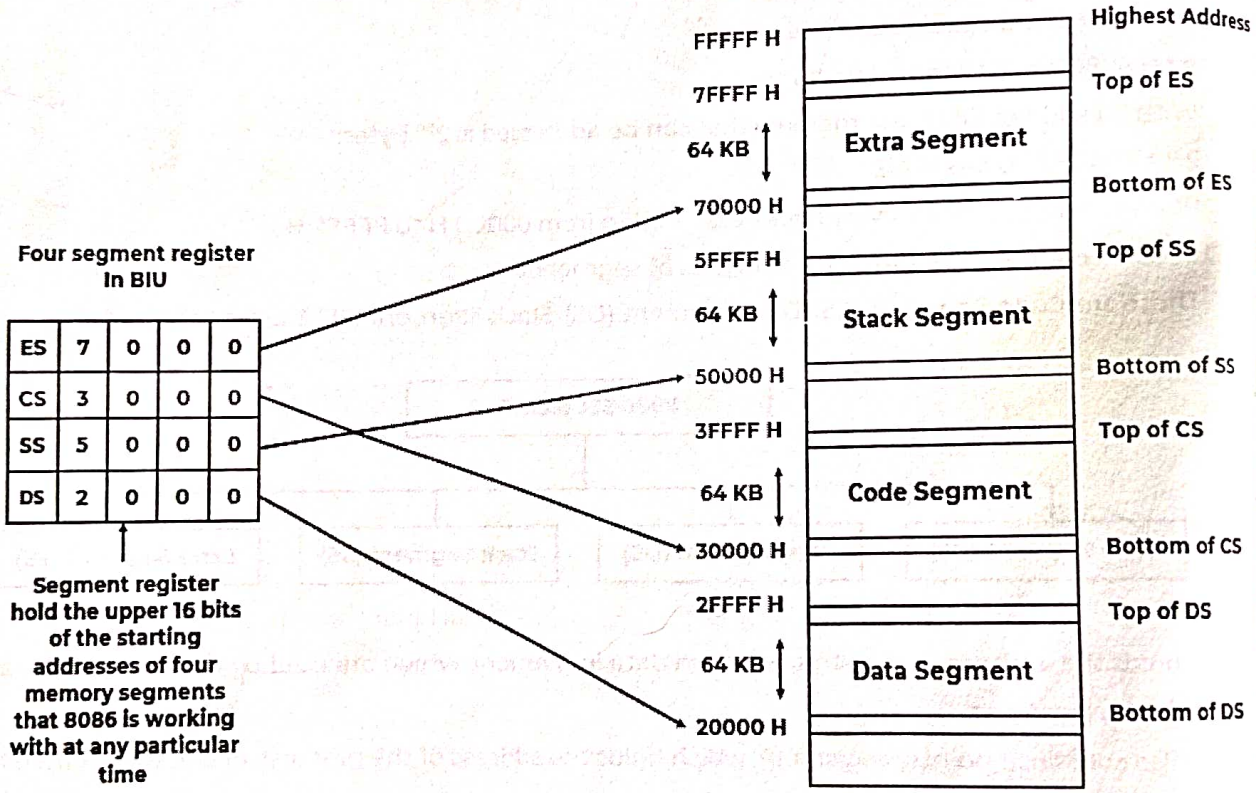


Figure 1.1: Segment Registers Segmentation.

TYPES OF SEGMENTATION:

1. **Overlapping Segment:**
   a. A segment starts at a particular address and its maximum size can go up to 64kilobytes.
   b. But if another segment starts along this 64kilobytes location of the first segment, then the two are said to be Overlapping Segment.
2. **Non-Overlapped Segment:**
   a. A segment starts at a particular address and its maximum size can go up to 64kilobytes.
   b. But if another segment starts before this 64kilobytes location of the first segment, then the two segments are said to be Non-Overlapped Segment.

ADVANTAGES OF MEMORY SEGMENTATION:

1. Segmentation provides a **powerful memory management mechanism.**
2. Segments provide a way to easily implement object oriented programs.
3. Using memory segmentation the data and code can be stored separately allowing for more flexibility.
4. Segmentation makes it possible to separate the memory areas for stack, code and data.
5. Segmentation makes it possible to write programs which are position independent or dynamically re-locatable.
6. Segmented structure of 8086 memory space supports modular software design.

DISADVANTAGES:

It becomes complicated for the programmer as multiple register to access a memory location.

Q2. Explain the maximum mode configuration of 8086 microprocessor

Ans:

[10M | Dec18] [P | Medium]

BLOCK DIAGRAM:

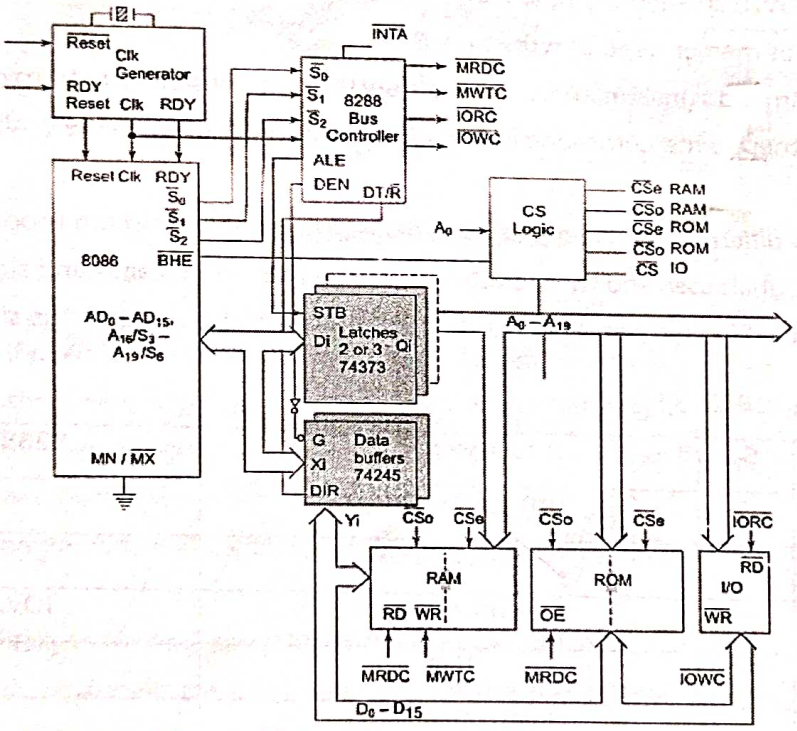


Figure 1.2: Maximum Mode of 8086.

MAXIMUM MODE:

1. Figure 1.2 shows the Maximum Mode Block Diagram of 8086.
2. In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
3. In this mode, the processor derives the status signal S2, S1, S0.
4. Another chip called bus controller derives the control signal using this status information.
5. In the maximum mode, there may be more than one microprocessor in the system configuration.
6. The components in the system are same as in the minimum mode system.

7. The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (fo memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the statu lines.
8. The bus controller chip has input lines S2, S1, S0 and CLK.
9. These inputs to 8288 are driven by CPU.
10. It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC.
11. The AEN, IOB and GEC pins are especially useful for multiprocessor systems.
12. AEN and IOB are generally grounded.
13. CEN pin is usually tied to +5V.
14. The significance of the MCE/PDEN output depends upon the status of the IOB.
15. If IOB is grounded, it acts as master cascade enable to control cascade 8259A, else it acts as peripher data enable used in the multiple bus configurations.
16. INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to a interrupting device.
17. IORC, IOWC are I/O read command and I/O write command signals respectively.
18. These signals enable an iO interface to read or write the data from or to the address port.
19. The MRDC, MWTC are memory read command and memory write command signals respectively ar may be used as memory read or write signals.
20. All these command signals instructs the memory to accept or send data from or to the bus.
21. For both of these write command signals, the advanced signals namely AIOWC and AMWTC a available.
22. Here the only difference between in timing diagram between minimum mode and maximum mod is the status signals used and the available control and advanced command signals.
23. The various machine cycle are initiated by 8288 bus controller using status signal S_2, S_1 and S_0 inp from 8086.

S_2	S_1	S_0	Processor States	8288 Active Output
0	0	0	Int. Acknowledge	INTA
0	0	1	Read I/O Port	IORC
0	1	0	Write I/O Port	IOWC and AIOWC
0	1	1	Halt	None
1	0	0	Instruction Fetch	MRDC
1	0	1	Memory Read	MRDC
1	1	0	Memory Write	MWTC and AMWTC
1	1	1	Inactive	None

Q3. Explain minimum mode configuration of 8086 microprocessor

Ans: **[10M | May19] [P | Medium]**

BLOCK DIAGRAM:

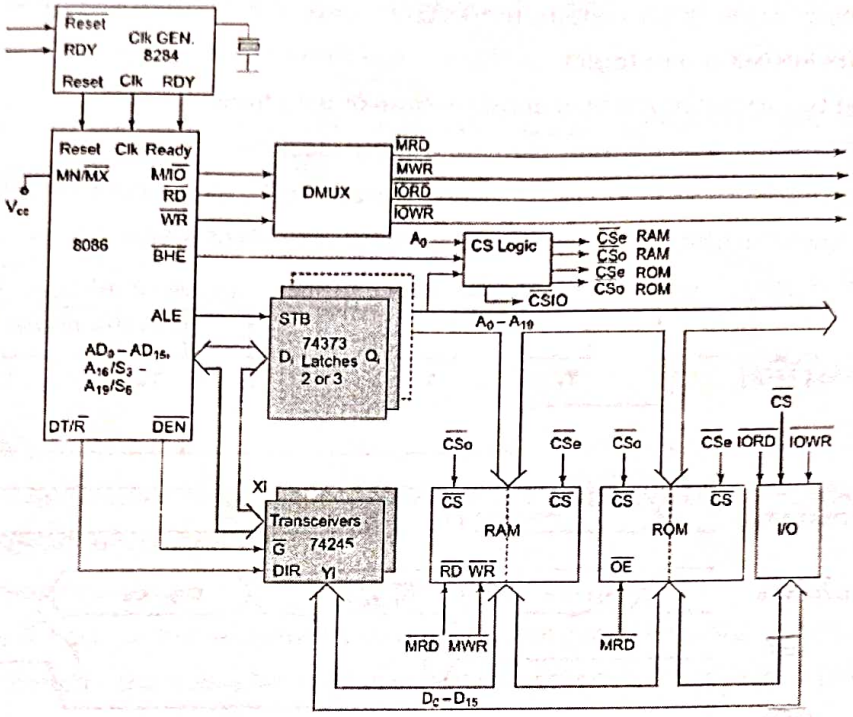


Figure 1.3: Maximum Mode of 8086.

MINIMUM MODE:

1. Figure 1.3 shows the Minimum Mode Block Diagram of 8086.
2. The microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic1.
3. In this mode, all the control signals are given out by the microprocessor itself.
4. There is a single microprocessor in the minimum mode system.
5. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices.
6. Latches are generally buffered output D-type flip-flops like 74LS373 or 8282.
7. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.
8. Transreceivers are the bidirectional buffers and they are called as data amplifiers.
9. They are required to separate the valid data from the time multiplexed address/data signals.
10. They are controlled by two signals namely, DEN and DT/R.
11. The DEN signal indicates the direction of data, i.e. from or to the processor.
12. The Opcode fetch and read cycles are similar.
13. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.

Q4. Draw and explain memory read machine cycle timing diagram in minimum mode of 8086

Ans: **[5M | Dec18] [P | Mediu**

TIMING DIAGRAM:

1. In a minimum mode 8086 system, the microprocessor 8086 is operated in **minimum mode** by **strapping its MN/$\overline{MX}$ pin to logic1**.
2. It is initiated by BIU of 8086 to read program code or data from memory.
3. The normal time taken by memory read cycle is four clock period.
4. In this mode, all the control signals are given out by the microprocessor chip itself.
5. Figure 1.4 shows the timing diagram for Read Machine Cycle of 8086.

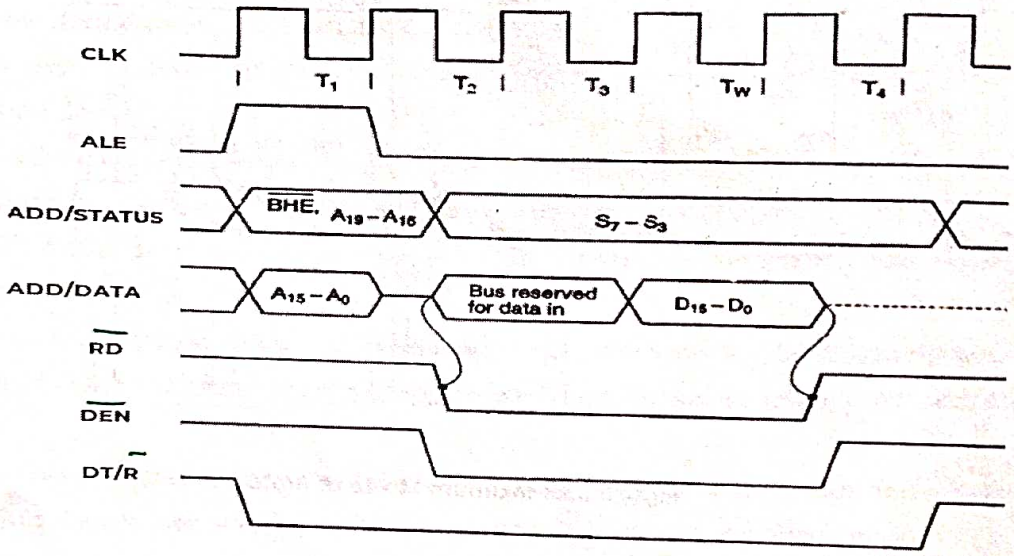


Figure 1.4: Timing Diagram for Read Machine Cycle in Minimum Mode of 8086.

SEQUENCES OF OPERATIONS DURING READ MACHINE CYCLE ARE AS FOLLOWS:

During T_1:

1. 8086 outputs 20 bit address on AD_0 - AD_{15} lines and ADDR/STATUS lines.
2. ALE is asserted high so as to latch the address and keep it on the output lines.
3. DT/$\overline{R}$ is LOW to inform external bidirectional data buffer that the processor has to receive data.
4. M/$\overline{IO}$ is high to indicate memory access.
5. $\overline{BHE}$ is low to enable upper (odd) memory bank.

During T_2:

1. AD_0 - AD_{15} lines become **inactive**.
2. The address is withdrawn from ADDR/STATUS lines and S_7 - S_3 signals are issued on these lines.
3. At the end of T_2, $\overline{RD}$ is asserted low to enable output buffer of memory.
4. The time during which it remains low is the time allowed for memory to load data into data bus.
5. The $\overline{DEN}$ signal is asserted low to enable external bidirectional data buffers.
6. The 8086 samples READY signal during T_2. (If READY is high then T_3 and T_4 are executed otherwise W states are introduced).

During T_3:

No activities are performed during T_3. Status of signals at end of T_2 are maintained same in T_3.

During T_4:

1. $\overline{RD}$ is asserted high and the data is latched into 8086.
2. $\overline{DEN}$ is made high to disable data memory buffer.

I/O Read Cycle: For reading data from an I/O device, M/$\overline{IO}$ is asserted low and data obtained is states T_3 and T_4 is from I/O device.

Q5. Draw and explain memory read and memory write machine cycle timing diagrams in maximum mode of 8086

Ans: **[10M | May19] [P | Medium]**

TIMING DIAGRAM:

1. In a maximum mode 8086 system, the microprocessor 8086 is operated in **maximum mode by strapping its MN/$\overline{MX}$ pin to Ground**.
2. In this mode, the processor derives the status signal S2, S1, S0.
3. Another chip called bus controller derives the control signal using this status information.
4. In the maximum mode, there may be more than one microprocessor in the system configuration.

READ CYCLE:

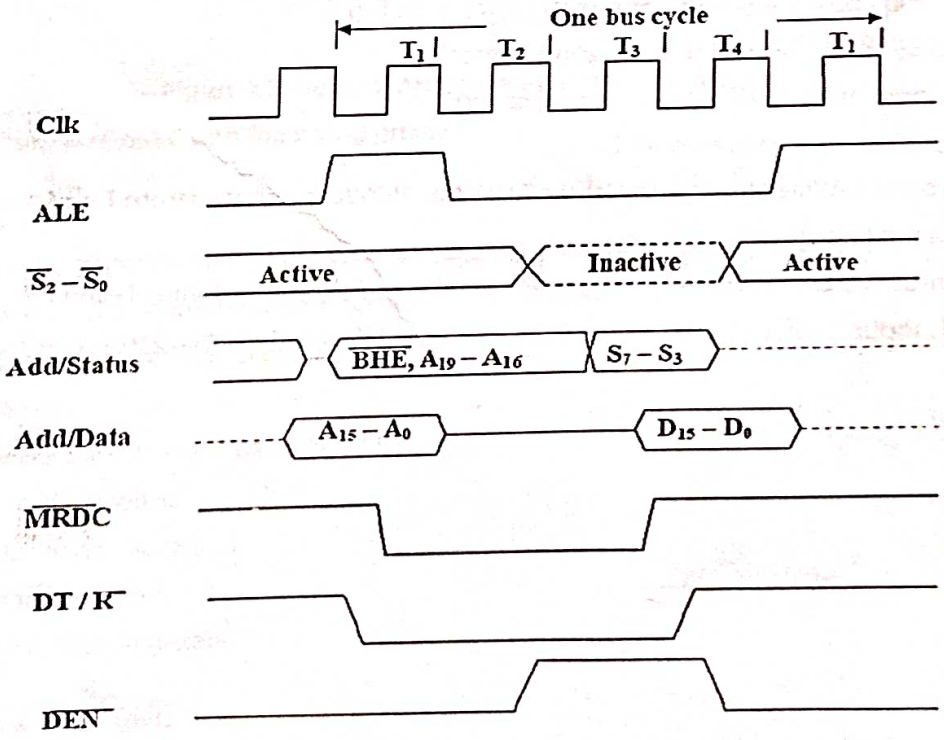


Figure 1.5: Timing Diagram for Read Machine Cycle in Maximum Mode of 8086.

WRITE CYCLE:

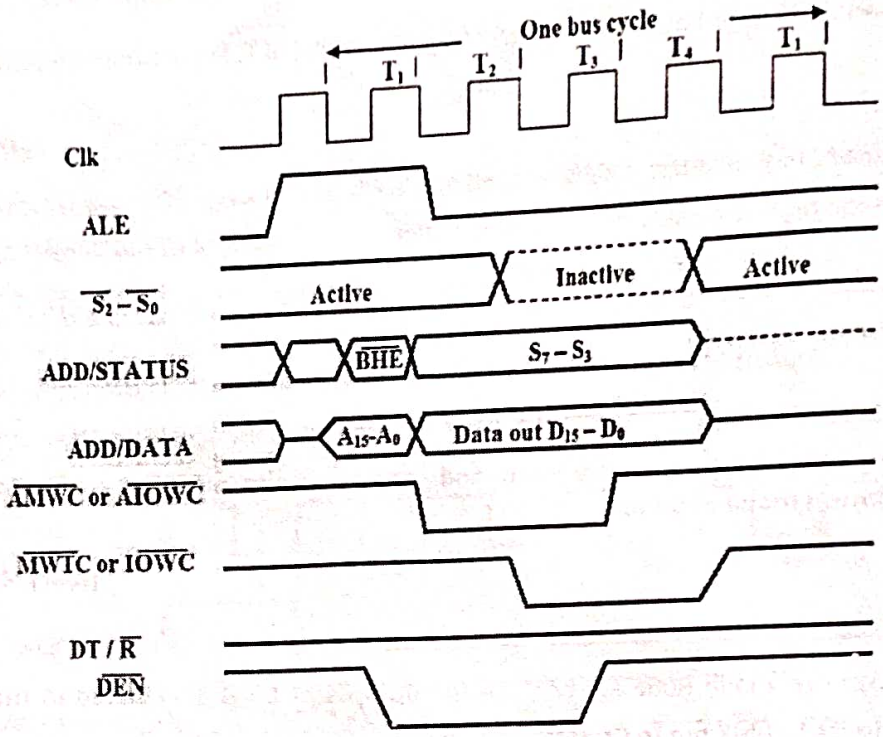


Figure 1.6: Timing Diagram for Write Machine Cycle in Maximum Mode of 8086.

EXPLANATION:

1. S_0, S_1, S_2 are set at the beginning of bus cycle.
2. On detecting the change on passive state $S_0 = S_1 = S_2 = 1$, the 8288 bus controller will output a pulse its ALE and apply a required signal to its DT/R pin during T_1.
3. In T_2, 8288 will set DEN = 1 thus enabling transceiver.
4. For an input, 8288 it will activates MRDC or IORC.
5. These signals are activated until T_4.
6. For an output (Write Machine Cycle), the AMWC or AIOWC is activated from T_2 to T_4 and MWTC or IOV is activated from T_3 to T_4.
7. The status bits S_0 to S_2 remain active until T_3, and become passive during T_3 and T_4.
8. If ready input is not activated before T_3, wait state will be inserted between T_3 and T_4.

-- EXTRA QUESTIONS --

Q1. Explain basic microprocessor architecture [P | Low]

Ans:

MICROPROCESSOR:

1. Microprocessor is a small IC which can **process data**.
2. That is it can perform **Arithmetic & Logical Operations**.
3. But this can also be done from ALU.
4. Figure 1.7 shows the general architecture of microprocessor.

BLOCK DIAGRAM:

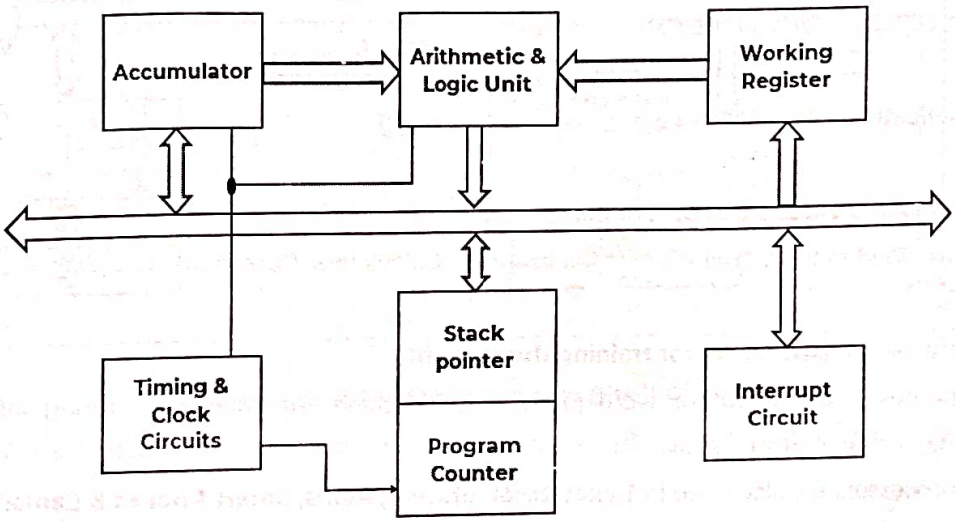


Figure 1.7: General Architecture of Microprocessor.

This architecture is divided into following groups:

I) Registers:

1. Microprocessor generally consists of **PIPO (Parallel In Parallel Out) Registers**.
2. Registers are used to store the data & address of the memory.
3. The architecture of microprocessor depends upon the number & type of the registers used in microprocessor.
4. Microprocessor can consist 8-bit or 16-bit registers.
5. Registers are classified as:
   a. General Purpose Registers.
   b. Temporary Registers.
   c. Special Purpose Registers.

II) Arithmetic & Logic Unit:

1. ALU is used to perform the **Arithmetic & Logical operations**.
2. It performs Arithmetic Operations like Addition, Subtraction & Logical Operations like AND, OR, EX-OR etc.
3. ALU is controlled by **Timing & Control Circuits**.

4. ALU is used to provide the Status of result to the flag register.
5. ALU looks after the branching decisions.

III) Interrupt Control:

1. Interrupt Control accepts different interrupt request inputs.
2. When a valid interrupt request is present it informs control logic to take action in response to ea signal.

IV) Timing & Control Unit:

1. Timing & Control Unit controls all internal and external circuits.
2. It operates with the reference to clock signal.
3. It accepts information from instruction decoder & generates micro steps to perform it.
4. This unit synchronizes all the data transfers.

Q2. Applications of Microprocessor [P | Low

Ans:

1. Microprocessor are used as **CPU of Computers**.
2. They are sued in industrial control Applications, Calculators, Commercial Appliances, Video Gam Toys Etc.
3. They are used in **laboratory for training the students**.
4. Microprocessor are used for word processing, database management, storing information a scientific & engineering calculations.
5. Microprocessors are also used in **Ticket Reservation Systems, Smart Phones & Cameras**.
6. They are also used to measure & control the temperature of a furnace.

Q3. Explain 8086 microprocessor architecture [P | Mediu

Ans:

FEATURES:

1. Intel 8086 was launched in 1978.
2. 8086 is **16-bit microprocessor.**
3. This microprocessor had major improvement over the execution speed of 8085.
4. It is available as **40-pin Dual-Inline-Package (DIP)**.
5. It is available in three versions:
   a. 8086 (5 MHz)
   b. 8086-2 (8 MHz)
   c. 8086-1 (10 MHz)
6. It includes 16 bit address bus & 20 bit data bus.
7. It consists of **29,000 transistors**.
8. 8086 can operate in 2 modes i.e. maximum and minimum modes.

4. ALU is used to provide the Status of result to the flag register.
5. ALU looks after the branching decisions.

III) Interrupt Control:

1. Interrupt Control accepts different interrupt request inputs.
2. When a valid interrupt request is present it informs control logic to take action in response to ea signal.

IV) Timing & Control Unit:

1. Timing & Control Unit controls all internal and external circuits.
2. It operates with the reference to clock signal.
3. It accepts information from instruction decoder & generates micro steps to perform it.
4. This unit synchronizes all the data transfers.

Q2. Applications of Microprocessor [P | Low

Ans:

1. Microprocessor are used as **CPU of Computers**.
2. They are sued in industrial control Applications, Calculators, Commercial Appliances, Video Gam Toys Etc.
3. They are used in **laboratory for training the students**.
4. Microprocessor are used for word processing, database management, storing information a scientific & engineering calculations.
5. Microprocessors are also used in **Ticket Reservation Systems, Smart Phones & Cameras**.
6. They are also used to measure & control the temperature of a furnace.

Q3. Explain 8086 microprocessor architecture [P | Mediu

Ans:

FEATURES:

1. Intel 8086 was launched in 1978.
2. 8086 is **16-bit microprocessor**.
3. This microprocessor had major improvement over the execution speed of 8085.
4. It is available as **40-pin Dual-Inline-Package (DIP)**.
5. It is available in three versions:
   a. 8086 (5 MHz)
   b. 8086-2 (8 MHz)
   c. 8086-1 (10 MHz)
6. It includes 16 bit address bus & 20 bit data bus.
7. It consists of **29,000 transistors**.
8. 8086 can operate in 2 modes i.e. maximum and minimum modes.

BLOCK DIAGRAM:

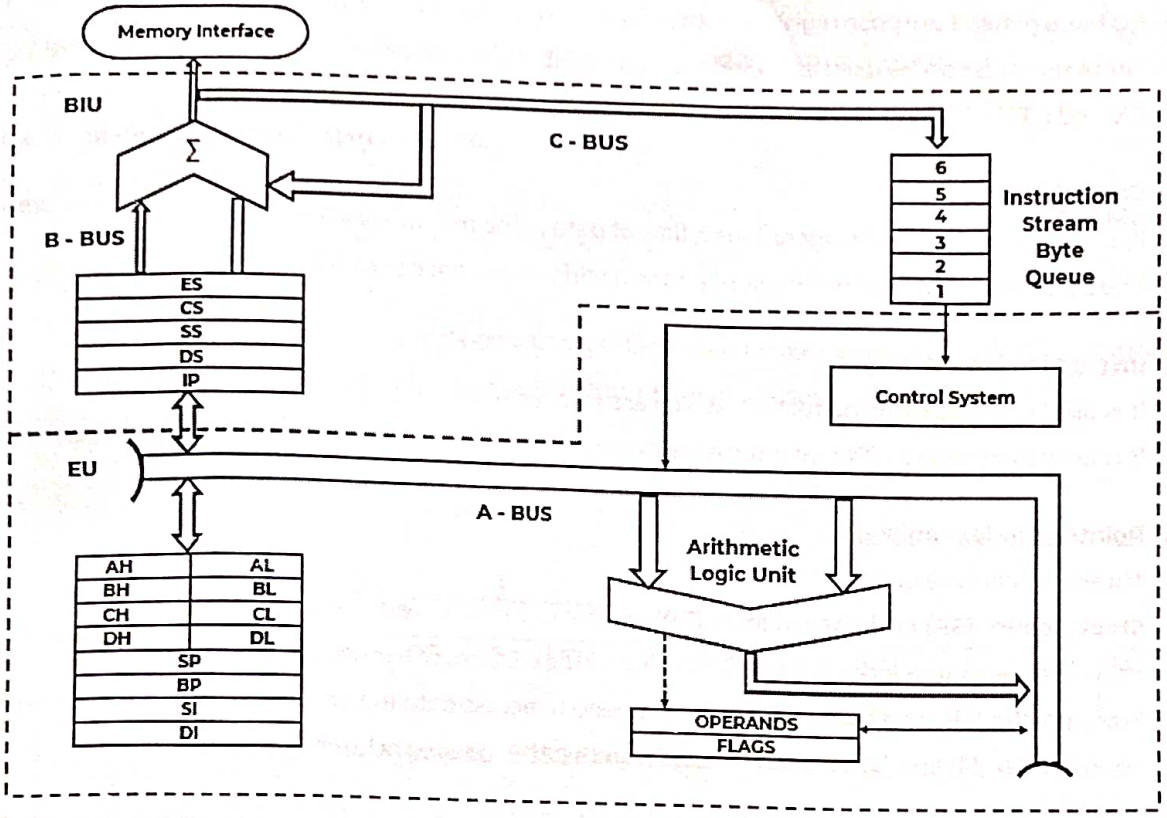


Figure 1.8: 8086 Microprocessor Block Diagram.

The 8086 CPU is divided into two independent functional units: Bus Interface Unit (BIU) & Execution Unit (EU)

EXECUTION UNIT (EU):

Execution unit performs the following functions:

1. The main function is decoding and execution of the instructions.
2. It performs the logic and arithmetic operation on memory or register.
3. It receives the instruction from Prefetch Queue and decodes it.
4. It stores the information in the register array.

Execution Unit consists of following parts:

I) ALU:

1. EU has 16-bit ALU so it can perform 16-bit operations simultaneously.
2. ALU is used to perform arithmetic and logical operations on 8-bit as well as 16-bit. (Addition, Subtraction, AND, OR, Increment, Decrement, Shift)

II) Flag Register:

1. EU has 16-bit Flag register.
2. Flag registers are used to control the certain operations of EU.
3. Flag registers includes carry flag, parity flag, auxiliary flag, zero flag, sign flag, trap flag, interrupt flag, direction flag and overflow flag.

III) General Purpose Registers:

1. EU has 8 general purpose registers named as AL, AH, BL, BH, CL, CH, DL & DH.
2. These registers can be used as 8-bit registers in which AH, AL, BH, BL, CH, CL, DH, DL or can be used as 16-bit registers in pairs of two registers AX, BX, CX, and DX.

IV) Control Circuit:

1. It is used to control all the operations & flow of data in the microprocessor.
2. The EU contains the control circuit to perform various internal operations.

V) Instruction Decoder:

1. It is used to decode the instructions which are fetched from memory.
2. It is used to generate different internal or external control signals required to perform the ope

VI) Pointer & Index register:

1. These registers are of 16 bit.
2. **Stack pointer (SP) and Base pointer (BP)** are the two pointer registers.
3. Whereas the Source Index (SI) and Destination Index (DI) are the index registers.
4. Stack pointer (SP) and base pointer (BP) are used to access data in the stack segment.
5. Source Index (SI) and Destination Index (DI) are used in indexed addressing.

BUS INTERFACE UNIT (BIU):

The function of BIU is to:

1. Fetch the instruction or data from primary memory.
2. Read / Write of data from / to primary memory.
3. I/O of data from / to peripheral ports.
4. Address generation for memory reference.

BIU consists of following parts:

I) Instruction Queue:

1. It is 6 bit long instruction register.
2. To increase the execution speed, BIU fetches six instruction bytes ahead to time from memo
3. All six bytes are then held in first in- first out (FIFO) queue.
4. Then all bytes have to be given to EU one by one.

II) Segment Registers:

1. **Code Segment (CS):** The CS register is used for addressing a memory location in the Code Se the memory, where the executable program is stored.
2. **Data Segment (DS):** The DS contains most data used by program. Data are accessed in Segment by an offset address or the content of other register that holds the offset address.
3. **Stack Segment (SS):** SS defined the area of memory used for the stack.
4. **Extra Segment (ES):** ES is additional data segment that is used by some of the string to destination data.

III) General Purpose Registers:

1. EU has 8 general purpose registers named as AL, AH, BL, BH, CL, CH, DL & DH.
2. These registers can be used as 8-bit registers individually or can be used as 16-bit in pair to have AX, B CX, and DX.

IV) Control Circuit:

1. It is used to control all the operations & flow of data in the microprocessor.
2. The EU contains the control circuit to perform various internal operations.

V) Instruction Decoder:

1. It is used to decode the instructions which are fetched from memory.
2. It is used to generate different internal or external control signals required to perform the operation

VI) Pointer & Index register:

1. These registers are of 16 bit.
2. **Stack pointer (SP) and Base pointer (BP)** are the two pointer registers.
3. Whereas the Source index (SI) and Destination index (DI) are the index registers.
4. Stack pointer (SP) and base pointer (BP) are used to access data in the stack segment.
5. Source Index (SI) and Destination Index (DI) are used in indexed addressing.

BUS INTERFACE UNIT (BIU):

The function of BIU is to:

1. Fetch the instruction or data from primary memory.
2. Read / Write of data from / to primary memory.
3. I/O of data from / to peripheral ports.
4. Address generation for memory reference.

BIU consists of following parts:

I) Instruction Queue:

1. It is 6 bit long instruction register.
2. To increase the execution speed, BIU fetches six instruction bytes ahead to time from memory.
3. All six bytes are then held in first in- first out (FIFO) queue.
4. Then all bytes have to be given to EU one by one.

II) Segment Registers:

1. **Code Segment (CS):** The CS register is used for addressing a memory location in the Code Segment the memory, where the executable program is stored.
2. **Data Segment (DS):** The DS contains most data used by program. Data are accessed in the D Segment by an offset address or the content of other register that holds the offset address.
3. **Stack Segment (SS):** SS defined the area of memory used for the stack.
4. **Extra Segment (ES):** ES is additional data segment that is used by some of the string to hold destination data.

III) Instruction Pointer:

1. Instruction pointer is of one bit.
2. It is used to point the particular instruction which is fetched by instruction register.

Q4. State use of control flags of 8086.

Ans: **[P | Medium]**

CONTROL FLAGS OF 8086:

1. Out of the 9 Active Flags, 6 are conditional (status) flags and the remaining 3 are called as control flags.
2. Control flags are used to control certain operations of the processors.
3. Figure 1.9 shows the 8086 flag register format.

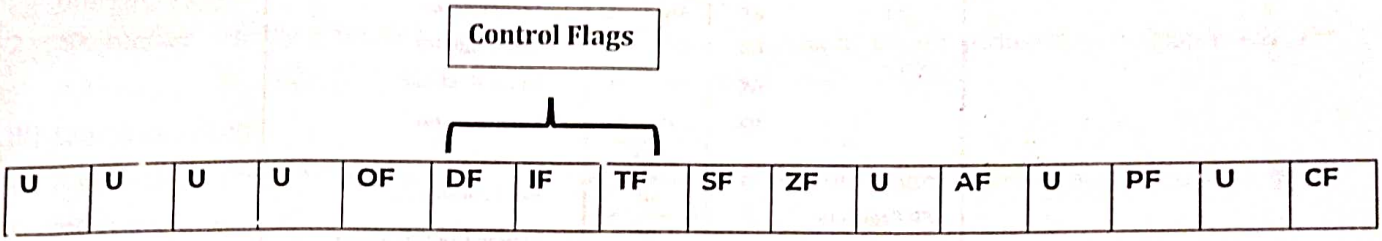


Figure 1.9: 8086 Flag Register Format.

I) Trap Flag (TF):

1. Setting TF puts the processor into **single step mode for debugging**.
2. In single stepping, microprocessor executes an instruction and enters into single step ISR.
3. After that the user can check register.
4. This utility is, to debug the program.
5. If TP = 1, the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it executes instruction by instruction.
6. This flag is used by debuggers for single step operations.
7. TF = 1 then Trap On, TF = 0 then Trap Off.

II) Interrupt Flag (IF):

1. If user sets IF flag, the CPU will recognize **external interrupt requests**.
2. Clearing IF disables these interrupts.
3. IF Flag has no effect on either non-maskable external or internally generated interrupt.
4. The IF Flag is used for allowing or prohibiting the interruption of a program.
5. IF = 1 then Interrupt Enabled, IF = 0 then Interrupt Disabled.

III) Direction Flag (DF):

1. DF Flag is used for **string instructions**.
2. In string instructions we use SI (Source Index) & DI (Destination Index) registers as offset registers to point source & destination area respectively.
3. DF Flags controls the direction of SI & DI Pointers.
4. If DF = 1, the string instruction will automatically decrement the pointers.

5. If DF = 0, the string instruction will automatically increment the pointers.
6. DF = 1 then Up, DF = 0 then Down.

Q5. Explain programming model of 8086. [P | Lo

Ans:

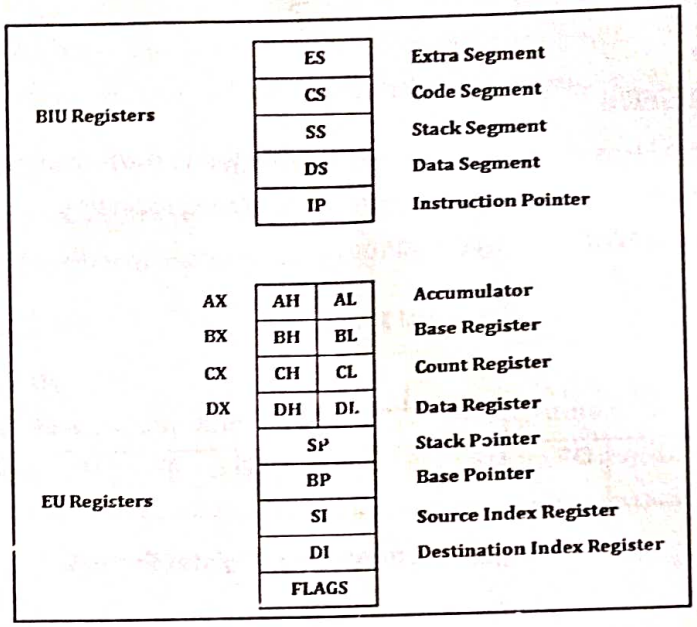


Figure 1.10: 8086 Programming Model.

1. Programming Model of 8086 is similar to the picture of the processor as available to programmer.
2. This registers are **used to hold number & addresses**.
3. It is also used to **indicate status and acts as controls**.
4. Figure 1.10 shows the 8086 Programming Model.

SEGMENT REGISTERS:

I) Extra Segment (ES):

1. It is a 16-bit register containing address of 64KB segment, usually with program data.
2. ES register can be changed directly using POP and LES instructions.

II) Code Segment (CS):

1. It is a 16-bit register containing address of 64 KB segment with processor instructions.
2. The CS register is automatically updated during far jump, far call and far return instructions.

III) Stack Segment (SS):

1. It is a 16-bit register containing address of 64KB segment with program stack.
2. SS register can be changed directly using POP instruction.

IV) Data Segment (DS):

1. It is a 16-bit register containing address of 64KB segment with program data.
2. DS register can be changed directly using POP and LDS instructions.

INSTRUCTION POINTER (IP):

1. It is a 16-bit register.
2. It is used to point the instructions.

GENERAL DATA REGISTERS:

I) Accumulator Register:

1. It consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
2. Accumulator can be used for I/O operations and string manipulation.

II) Base Register:

1. It consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
2. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

III) Count Register:

1. It consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX.
2. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation.

IV) Data Register:

1. It consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
2. Data register can be used as a port number in I/O operations.

POINTER & INDEX REGISTERS:

I) Stack Pointer (SP):

1. It is a 16-bit register pointing to program stack.

II) Base Pointer (BP):

1. It is a 16-bit register pointing to data in stack segment.
2. BP register is usually used for based, based indexed or register indirect addressing.

III) Source Index (SI):

1. It is a 16-bit register.
2. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

IV) Destination Index (DI):

1. It is a 16-bit register.
2. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

FLAGS:

1. Flag Register are **16 bits registers**.
2. Out of 16 bits only 9 bits are used as flags and the rest 7 bits are not used.
3. CF, AF, PF, ZF, SF and OF Flags are status indicators.

4. While TF, IF, and DF are Control Flags.
5. Figure 1.11 shows Flag registers.

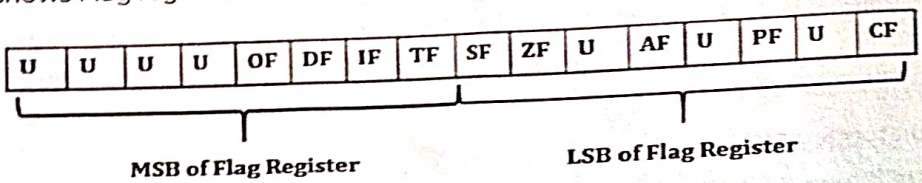


Figure 1.11: Flag Register.

Q6. Draw and explain timing diagram for write operation in minimum mode of 8086 **[P | Mediu**

Ans:

1. In a minimum mode 8086 system, the microprocessor 8086 is operated in **minimum mode strapping its MN/$\overline{MX}$ pin to logic1**.
2. In this mode, all the control signals are given out by the microprocessor chip itself.
3. The 8086 microprocessor uses memory and I/O in periods called 'bus cycles'.
4. 1 bus cycle is 4 system clocking periods T_1 to T_4 or T-states (assuming that there are no wait states) as shown in the figure 1.12 below.

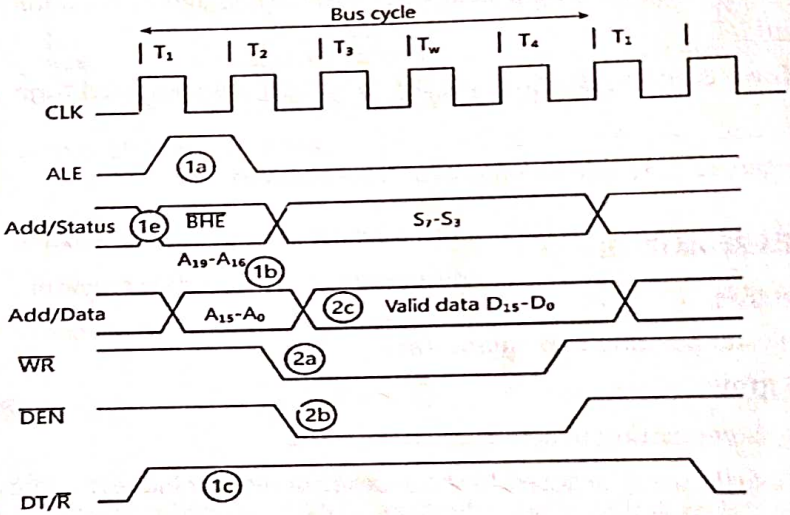


Figure 1.12: Timing Diagram for Write Machine Cycle of 8086.

SEQUENCES OF OPERATIONS DURING WRITE MACHINE CYCLE ARE AS FOLLOWS:

During T1:

1. ALE line is asserted.
2. Address of the memory or I/O location is put on the Address bus.
3. DT/$\overline{R}$ is made HIGH as it is a WRITE operation.
4. M/IO signal indicates whether the address bus contains a memory address or an I/O device addr
5. This line is HIGH for Memory WRITE and LOW for IO WRITE.
6. From T_1 to T_4 the M/IO signal is asserted to indicate a memory or I/O operation.
7. The $\overline{BHE}$ and AO signals are used to select the proper byte or bytes of memory or I/O word to be or written.
8. The M/IO, RD and $\overline{WR}$ signals indicate the types of data transfer.

During T2 state and until T4:

1. At the beginning of T_2, the WR signal goes LOW.
2. $\overline{DEN}$ become active.
3. The $\overline{DEN}$ signal turns ON the data bus buffers, so the memory or I/O can receive data sent out by the 8086 to the memory or I/O through the data bus.
4. Data to be written appears on the Data bus.
5. The data remains on the bus until middle of T_4 state.
6. These events cause the memory or I/O device to begin to perform a write.

READY is sampled at the end of T_2. If READY is low at this time T_3 becomes a WAIT state.

Q7. Write Short Note on 8288 Bus Controller.

Ans: **[P | Medium]**

8288 BUS CONTROLLER:

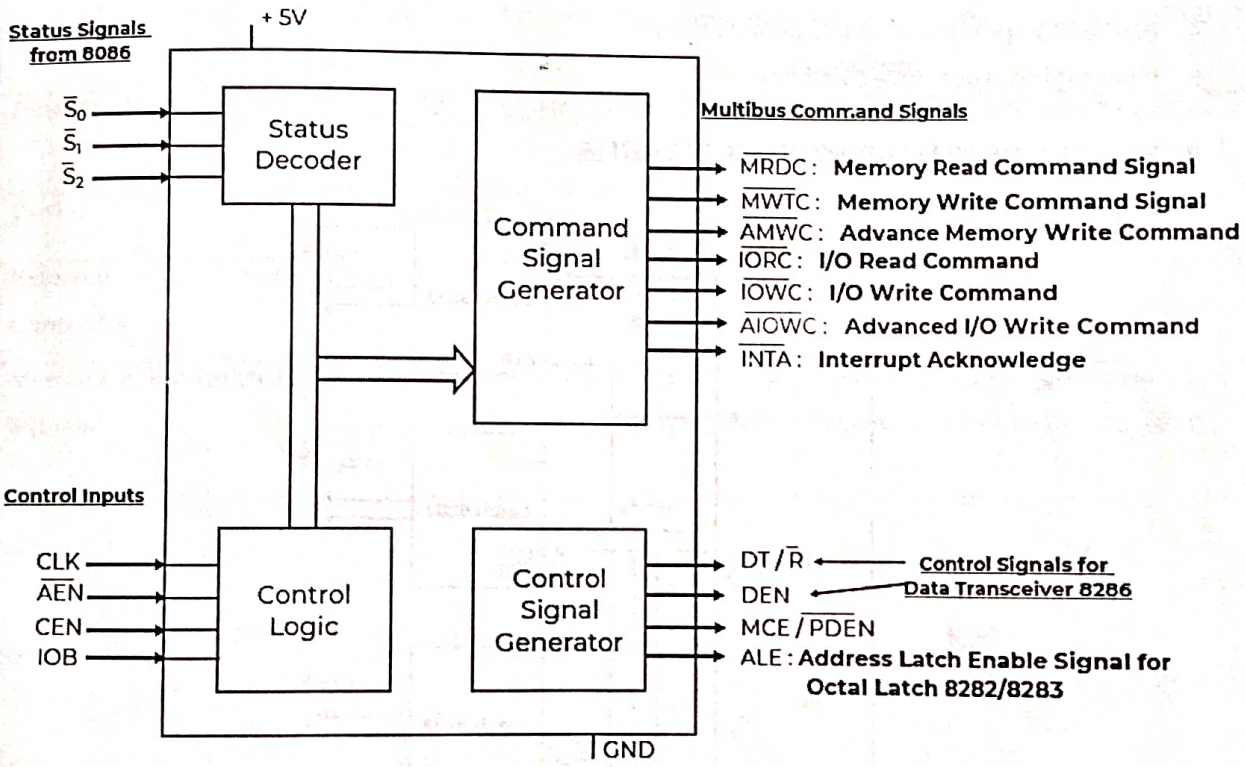


Figure 1.13 Internal Block Diagram of 8288 Bus Controller.

1. The Intel 8288 is a bus controller designed for **Intel 8086/8087/8088/8089.**
2. 8288 Bus Controller is a **20 Pin Bipolar IC.**
3. In maximum mode, 8086 doesn't provide various control signals like ALE, DT/$\overline{R}$, $\overline{DEN}$ etc. & command signals like $\overline{WR}$, M/$\overline{IO}$ etc.
4. Hence 8288 Bus Controller is used to provide all these signals.
5. 8288 Bus Controller accepts the CLK signal along with $\overline{S_0}$, $\overline{S_1}$ & $\overline{S_2}$ outputs of 8086 and generates command, control & timing signals at its output.
6. 8288 Bus Controller is used to **optimize the system performance.**
7. It also provides the bipolar bus drive capability.

8. Figure 1.13 shows the internal block diagram of 8288 Bus Controller.
9. Block Diagram of 8288 Bus Controller includes Status Decoder, Control Logic, Command Signal Generator and Control Signal Generator.
10. 8288 Bus Controller accepts the CLK signal along with S_0, S_1 & S_2 outputs of 8086 and generates command, control & timing signals at its output.

Q8. Design interfacing of 8282 latches to 8086 system. [P | Medium]

Ans:

8282 LATCH:

1. A latch is a **register with a specific purpose.**
2. 8282/8283 are 8 bit bipolar latches with tristate output buffer.
3. 8282 is a non-inverting latch, while 8283 is an inverting octal latch.
4. 8282/8283 operates on single polarity supply.
5. It has high output drive capability.

INTERFACING OF 8282 LATCHES TO 8086 SYSTEM:

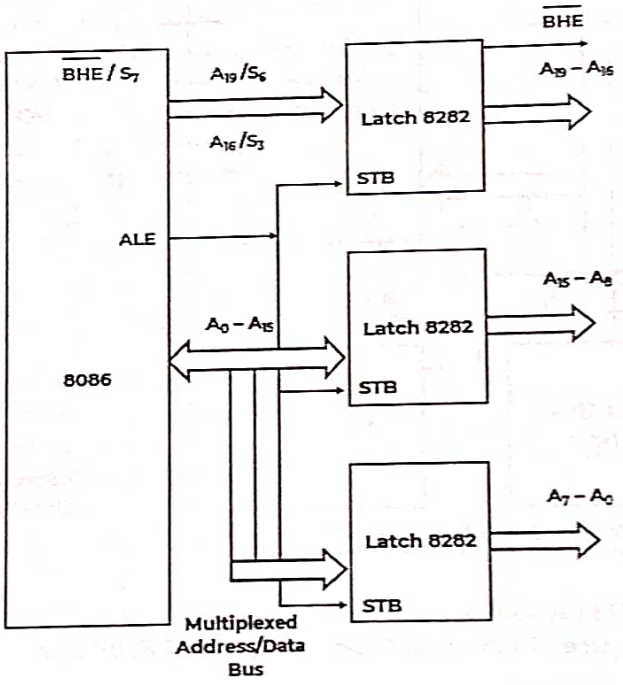


Figure 1.14: Interfacing of the latch 8282 with the 8086 processor.

1. Interfacing of the latch 8282 with the 8086 processor is shown in figure 1.11.
2. The ALE signal is being used as the strobe (STB) input for the latches, which enables the latches and loads the address into the latches.
3. Since each latch is an octal, we have to use 3 such latches in order to latch a 20 bit address.
4. On receiving the HIGH (1) ALE signal from the processor, the latches will be enabled and the A[illegible] address is latched.

5. Before the address disappears from the multiplexed bus, the ALE reduces to 0 and the latches are disabled.

Q9. Differentiate between minimum mode & maximum mode in 8086.

Ans: **[P | Medium]**

Parameter	Minimum Mode	Maximum Mode
Operating Mode	The microprocessor system operates in the minimum mode by default on power on.	If MN/$\overline{MX}$ pin is grounded, the system operates in maximum mode.
CPU	When only one CPU is to be used in a microcomputer system it is used in minimum mode operation.	Maximum mode uses Multiple CPUs to operate.
Control Signals	CPU issues the control signals that are required by the memory and I/O devices.	In Maximum mode the control signals are issued by the Intel 8288 bus controller.
Pins Used	M/$\overline{IO}$, $\overline{INTA}$, ALE, HOLD, HLDA, DT/$\overline{R}$, $\overline{DEN}$, WR	$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$, LOCK, QS_1, QS_0, $\overline{RQ_0}$ / GT_0, $\overline{RQ_1}$ / GT_1
Cost	Less Expensive.	More Expensive.
External Bus Controller	Not required.	Required like 8288
Memory & I/O Control Signals	Memory & I/O Control Signals are generated by the microprocessor itself.	Control Signals are generated by External Bus Controller like 8288.

JE = Jump equal
JC = Jump carry
JNC = Jump not carry

Comp al, 10
JE ew

CHAP - 2: INSTRUCTION SET & PROGRAMMING

Q1. Write a short note on mixed language programming

Ans: **[5-10M | Dec18 & May19] [P | Mediu**

MIXED LANGUAGE PROGRAMMING:

1. C generates an Object Code that is **Extremely Fast & Compact**.
2. But it is not as fast as the Object Code generated by a good programmer using Assembly Langua
3. Since time needed to write a program in Assembly Language is much more than time taken in Hig Level Languages like C.
4. There are some cases where certain instructions cannot be executed in Higher Level Language lik
5. For example: C does not have an instruction for performing bit-wise rotation operation.
6. Therefore combining multiple languages such as C & Assembly Language is known as Mu Language Programming.
7. There are two ways of combining C & Assembly Language.

I) Method 1:

1. In method 1 there is a Built-In-Inline assembler used to include assembly language routines in the program, without any need for a specific assembler.
2. Such assembly language routines are called as In-Line Assembly.
3. They are compiled along with C-routine & linked together using linked modules provided by th Compiler.
4. **Example:** Turbo C has inline Assembly.

```
#include<iostream.h>
void main()
{
    int a, b, c;
    cout << "Enter Two Numbers";
    cin >> a >> b;
    asm mov ax, a;
    asm mov bx, b;
    asm add ax, bx;
    asm mov c, ax;
    cout << "The Sum is" << c;

}
```

II) Method 2:

1. There are times when programs written in one language have to call modules written in o language.
2. Instead of compiling all source programs using the same compiler, different compilers or assem are used as per the language used in the program.
3. Microsoft C support **Mixed Language Programming**.
4. Therefore, it can combine assembly language routines in C as a separate language.

5. C program calls assembly language routines that are separately assembled by MASM (MASM Assembler) or TASM (Turbo Assembler).
6. These assembled modules are linked with the complied C modules to get the combine executable file.
7. Figure 2.1 shows Compile, Assemble & Link processes using C Compiler, MASM Assembler & TLINK.

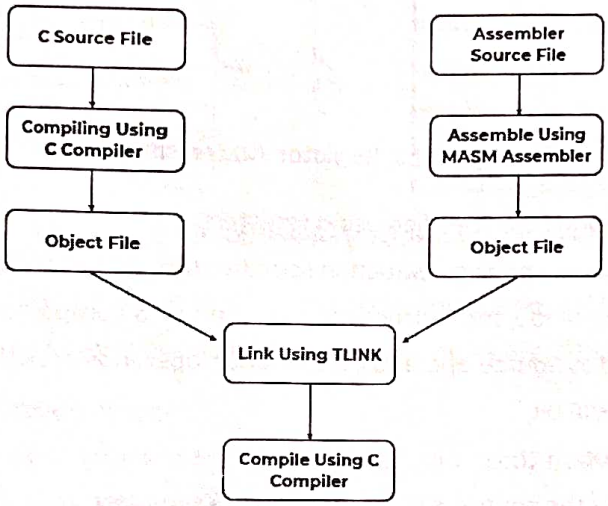


Figure 2.1: Compile, Assemble & Link processes in Mixed Language Programming.

Q2. Explain different addressing modes of 8086 microprocessor

Ans: **[10M | Dec18 & May19] [P | High]**

ADDRESSING MODE OF 8086:

1. Instruction consists of **Opcode followed by Operands**.
2. The Opcode specifies the operation to be performed.
3. Operand specifies where data is.
4. The data may be instruction or may be in the register or in the memory or in the I/O Port etc.
5. The different ways by which microprocessor generates operand address are called addressing mode.

TYPES:

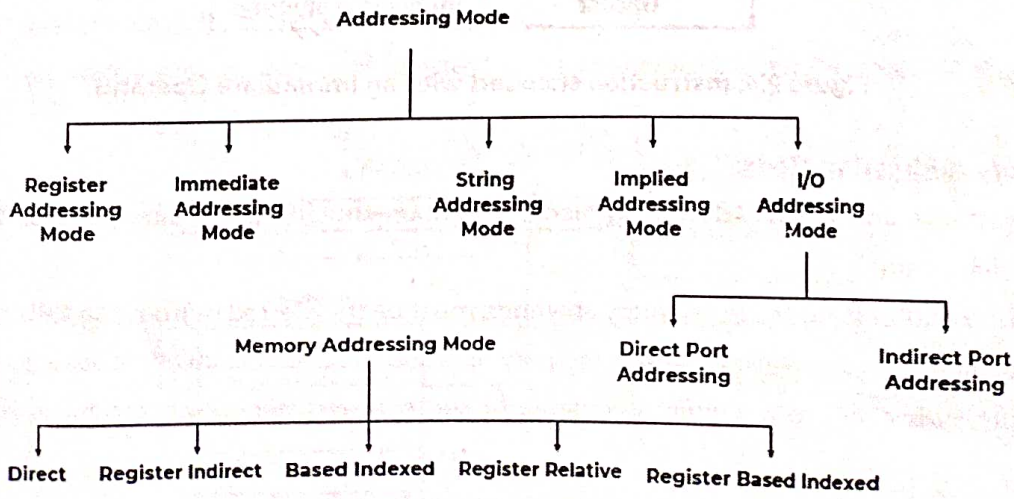


Figure 2.2: Addressing Modes of 8086.

I) Register Addressing Mode:

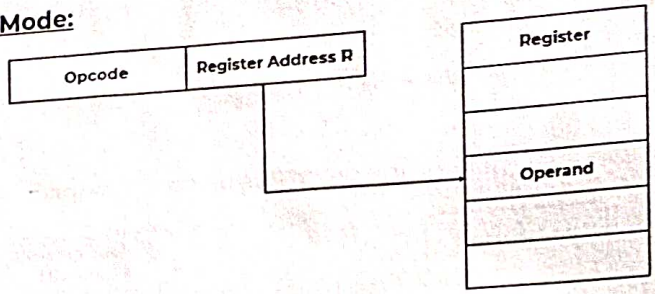


Figure 2.3: Register Addressing Mode.

1. In this mode the operands are specified using registers.
2. The data is in the register, and the instruction specifies the particular register as shown in figure
3. In Register Addressing Mode, the instructions are compact & comparatively faster for execution.
4. Register may be used as Source operands, Destination operands or both.
5. The register may be 8/16 bit.
6. **Example: MOV AX, BX**
7. This instruction copies the contents of BX register to AX register.

II) Immediate Addressing Mode:

1. In this mode the operand is specified by instruction itself.
2. Immediate operand is nothing but the constant data contained in an instruction.
3. Thus if the source operand is part of instruction instead of register or memory, it is referre Immediate Addressing Mode as shown in figure 2.4.
4. Immediate Data may be 8/16 bit.
5. Immediate operands are used as Source operands and they are constant values.
6. Immediate operands are accessed quickly like Register Addressing Mode.
7. **Example: MOV CL, 02H** **MOV CL, 2**
8. This instruction copies the immediate number 2H in the CL Register.

Opcode	Immediate operand

Figure 2.4: Instruction encoded with an Immediate Operand.

III) Memory Addressing Mode:

1. In Register & Immediate Addressing Mode, Execution Unit (EU) has direct access to Regis Immediate data.
2. In Memory Addressing Mode, memory operands must be transferred to/from the CPU over the
3. Whenever EU needs to read or write a memory operand, it must pass an offset value to the BIU.
4. The BIU adds offset to the shifted contents of segment register, which produces 20 bit ph address.
5. After that it executes the bus cycle needed to access the operand.
6. The offset for a memory operand is called the operand's Effective Address.

Types:

a. Direct Memory Addressing Mode:

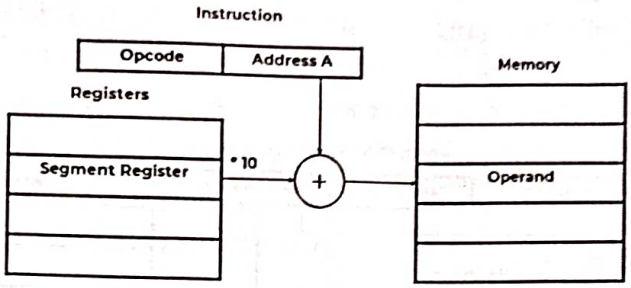


Figure 2.5: Direct Memory Addressing Mode.

- In this mode, the 16 bit Effective Address is taken from the displacement field of the instruction.
- The physical address is generated by adding this to segment register * 10 H as shown in figure 2.5.

b. Register Indirect Addressing Mode:

- In this mode, the Effective Address of the memory is taken directly from one of the base register or index register, specified by the instruction.
- The address is added with the segment register * 10 H to generate the physical address as shown in figure 2.6.

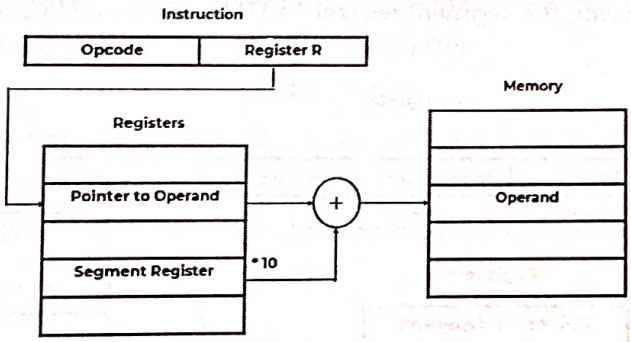


Figure 2.6: Register Indirect Addressing Mode.

c. Register Relative Addressing Mode:

- In this mode, the Effective Address is sum of an 8 or 16 bit displacement and the contents of base register or an index register are added.
- This sum is added with the segment register * 10 H to generate Effective Address as shown in figure 2.7.

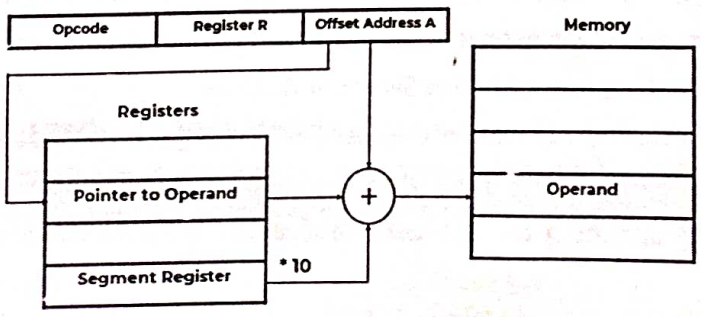


Figure 2.7: Register Relative Addressing Mode.

d. Based Indexed Addressing Mode:

- In this mode, the Effective Address is sum of a base register and an index register, both of which specified by the instruction.
- The sum is added with the segment register * 10 H to generate Effective Address as shown in fig 2.8

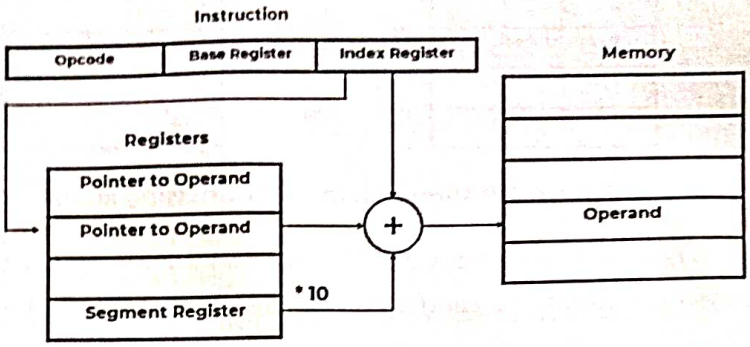


Figure 2.8: Based Indexed Addressing Mode.

e. Relative Based Indexed Addressing Mode:

- This Addressing Mode generates an Effective Address that is the sum of a base register, an index register and a displacement.
- This sum is added with the segment register * 10 H to generate Effective Address as shown in fig 2.9.

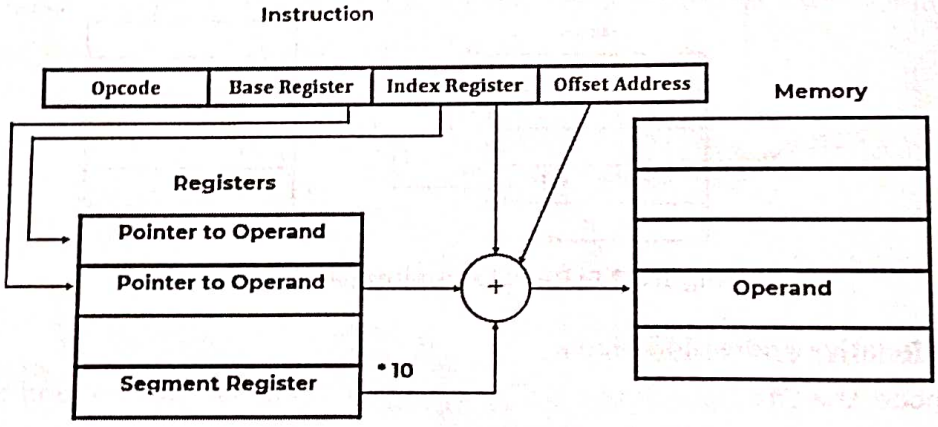


Figure 2.9: Relative Based Indexed Addressing Mode.

IV) String Addressing Mode:

1. String Addressing Mode does not use normal Memory Addressing Mode to access their operands
2. When a string instruction is executed, SI is assumed to point to the first byte or word of the source string and by default DS is assumed as Segment Register.
3. DI will point to the first byte or word of the destination string and by default ES is assumed as Segment Register.
4. In repeated string operation, the CPU automatically adjust the SI & DI to obtain subsequent byte/word.
5. This automatic adjustment is done with the help of Direct Flag (DF) in flag register.

V) I/O Addressing Mode:

1. I/O Addressing Mode is used for **Inputs & Outputs**.
2. I/O Addressing Mode consists of Memory Mapped I/O & I/O Mapped I/O.
3. **Memory Mapped I/O:**
   a. In this mode, the memory is mapped to I/O port.
   b. In this any memory operand addressing modes are used to access the port.
   c. For example, a group of terminals can be accessed as an array.
4. **I/O Mapped I/O:**
   a. In this mode, the I/O is mapped to I/O port.
   b. There are two types:
      - **Direct Port Addressing:**
        - In this 8 bit port address is specified in the instruction itself.
        - Therefore it consists of 256 I/O Address i.e. ports numbered 0-255.
        - For **Example:** In AL, 04 H means move 8 bit data from Input Device having address 04 H into AL register.
        - In AX, 04 H means move 16 bit data from Input Device having address 04 H and 05 H into AL and AH register respectively.
      - **Indirect Port addressing:**
        - It is similar to register indirect addressing of memory operands.
        - In this 16 bit I/O address is kept in DX register.
        - Therefore it consists of 65,536 I/O Address.
        - For **Example:** In AL, DX means move 8 bit data from the Input Device pointed by DX into AL register.
        - In AX, DX means move 16 bit data from the Input Device pointed by DX and DX + 1 into AX register.

VI) Implied Addressing Mode:

1. The instructions which do not have operands are considered as Implied Addressing Mode.
2. For **Example:** STC (Sets the Carry Flag), CLD (Clears the Direction Flag), STD (sets the Direction Flag) etc.

Q3. Differentiate procedure and macro. Write a program to find the factorial of a number using procedure

Q4. Differentiate Procedure and macro with example

Ans: **[5 - 10M | Dec18] [P | High]**

COMPARISON BETWEEN PROCEDURE & MACRO:

Macros	Procedure
Machine code is generated for instructions each time when macro is called.	Machine code for instructions is put only once in the memory.
With macros more memory is required.	With Procedure less memory is required.

Parameters passed as part of statement which calls macro.	Parameters can be passed in registers, mem locations or stack.
No transfer of program counter.	Transferring of program counter is required
Execution is fast.	Execution is comparatively slow.
Assembly time is more.	Assembly time is comparatively less.
There is no type checking.	Type checking is occurred.
Accessed during assembly with name given to macro when defined.	Accessed by CALL & RET instruction du program execution.
Macros are used for repetitive task, if the task is small.	Procedures are to be used for repetitive task, i task is very large.
Macros does not required any latency period.	Procedure requires latency period.

PROGRAM TO FIND THE FACTORIAL OF A NUMBER USING PROCEDURE:

Solution:

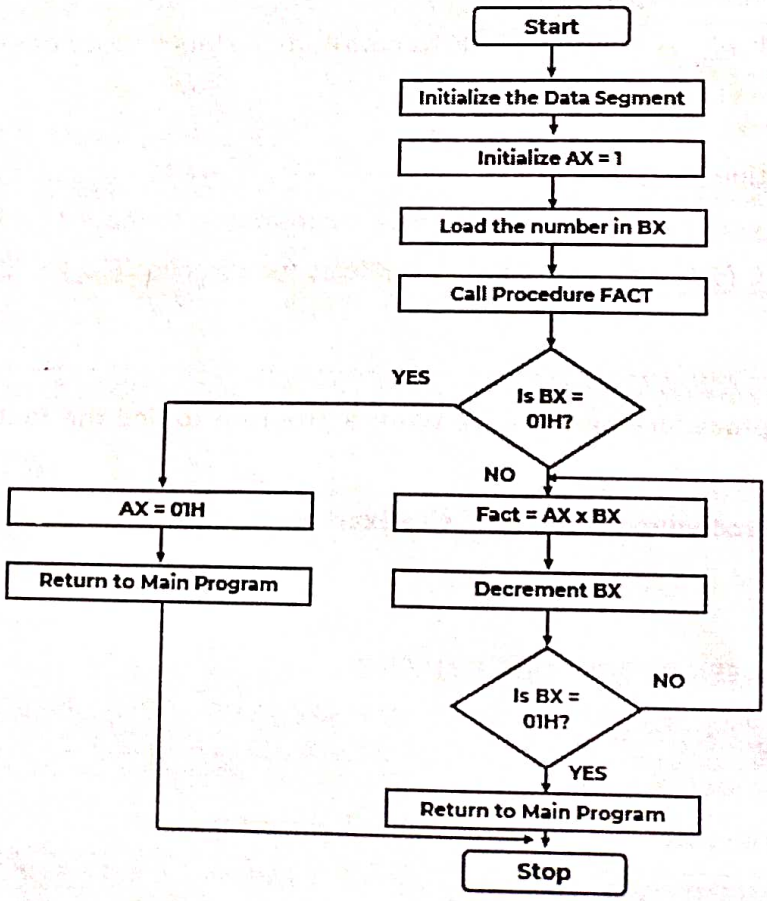
To compute the factorial of a number means to multiply the number 'n' with (n-1) (n-2).. x 2 x 1

Example: To Compute 5!

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

In our program, we will initialize AX = 1 and load the number whose factorial is to be computed in BX Procedure fact, which will calculate the factorial of the number.

Flowchart:



Algorithm:

1. Initialize the data segment.
2. Initialize AX = 1
3. Load the number in BX
4. Call Procedure Fact.
5. Compare BX with 1, if not goto step 7
6. AX = 1 and return back to calling program.
7. AX = AX x BX
8. Decrement BX
9. Compare BX with 1, if not goto step 7
10. Return back to calling program.
11. Stop

Program:

Label	Instruction	Comment
	.model small	
	.data	
	num dw 08h	
	.code	
	mov ax, @data	Initialize Data Segment
	mov ds, ax	
	mov ax, 01	Initialize ax = 1
	mov bx, num	
	call fact	Call Procedure
	mov di, ax	Store the LSB of result in di
	mov bp, 2	Initialize count for no of times display is called
	mov bx, dx	Store msb of result in reg bx
	mov bx, di	Store lsb of result in bx
	dec bp	Decrement bp
	mov ah, 4ch	
	int 21h	
	factproc near	Function for finding the factorial
	cmp bx, 01	Is bx = 1?
	jz L11	If yes, ax = 1
L12:	mul bx	Find factorial
	dec bx	Decrement bx
	cmp bx, 01	Multiply till bx = 1
	jne L12	
	ret	
L11:	mov ax, 01	Initialize ax = 1
	ret	Return to called program
	fact endp	End procedure
	end	End program

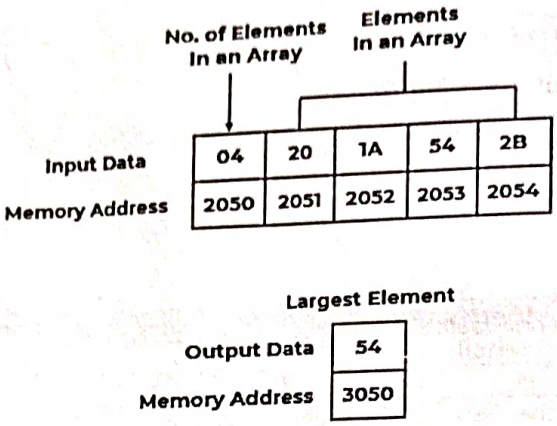
Q5. Write a program to find the largest number from an array **[10M | May19] [P | Me**

Ans:

PROBLEM:

Determine largest number in an array of n elements. Value of n is stored at address 2050 and array from address 2051. Result is stored at address 3050. Starting address of program is taken as 2000

EXAMPLE:



Algorithm:

1. We are taking first element of array in A.
2. Comparing A with other elements of array, if A is smaller than store that element in A othe compare with next element.
3. The value of A is the answer.

Program:

Label	Instruction	Comment
2000	LXI H 2050	H←20, L←50
2003	MOV C, M	C←M
2004	DCR C	C←C-01
2005	INX H	HL←HL+0001
2006	MOV A, M	A←M
2007	INX H	HL←HL+0001
2008	CMP M	A-M
2009	JNC 200D	If Carry Flag=0, goto 200D
200C	MOV A, M	A←M
200D	DCR C	C←C-1
200E	JNZ 2007	If Zero Flag=0, goto 2007
2011	STA 3050	A→3050
2014	HLT	

Explanation: Registers used: **A, H, L, C**

1. **LXI 2050** assigns 20 to H and 50 to L
2. **MOV C, M** copies content of memory (specified by HL register pair) to C (this is used as a counte
3. **DCR C** decrements value of C by 1
4. **INX H** increases value of HL by 1. This is done to visit next memory location
5. **MOV A, M** copies content of memory (specified by HL register pair) to A
6. **INX H** increases value of HL by 1. This is done to visit next memory location

7. **CMP M** compares A and M by subtracting M from A. **Carry flag and sign flag becomes set if A-M is negative**
8. **JNC 200D** jumps program counter to 200D if carry flag = 0
9. **MOV A, M** copies content of memory (specified by HL register pair) to A
10. **DCR C** decrements value of C by 1
11. **JNZ 2007** jumps program counter to 2007 if zero flag = 0
12. **STA 3050** stores value of A at 3050 memory location
13. **HLT** stops executing the program and halts any further execution.

-- EXTRA QUESTIONS --

Q1. Briefly explain string instructions of 8086

[P | Mediu

Ans:

STRING INSTRUCTION:

1. String is a group of bytes/words and their memory is always allocated in a sequential order.
2. **Operations** that can be performed with string instructions are:
   a. Copy a string into another string.
   b. Search a string for a particular byte or word.
   c. Store characters in a string.
   d. Compare strings of characters alphanumerically.

Following is the list of instructions under this group:

1. **REP:** Used to repeat the given instruction till CX ≠ 0.
2. **REPE/REPZ:** Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
3. **REPNE/REPNZ:** Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
4. **MOVS/MOVSB/MOVSW.** Used to move the byte/word from one string to another.
5. **COMS/COMPSB/COMPSW:** Used to compare two string bytes/words.
6. **INS/INSB/INSW:** Used as an input string/byte/word from the I/O port to the provided memory locat
7. **OUTS/OUTSB/OUTSW:** Used as an output string/byte/word from the provided memory location to I/O port.
8. **SCAS/SCASB/SCASW:** Used to scan a string and compare its byte with a byte in AL or string word w a word in AX.
9. **LODS/LODSB/LODSW:** Used to store the string byte into AL or string word into AX.

Q2. Write assembly language program for 8086 to exchange contents of two memory blocks.

Ans:

[P | Mediu

ASSEMBLY LANGUAGE:

1. Machine Language is the only language understood directly by CPU in computers.
2. So the instructions in the text form are called as Mnemonic.
3. Assembly Language is a Mnemonic representation of machine code.
4. There is a one-to-one correlation between assembly language instructions and the machine code

PROGRAM TO EXCHANGE CONTENTS OF TWO MEMORY BLOCK:

1. Let consider the Memory Locations as: **01000 H (First Memory Block) & 02000 H (Second Mem Block)**
2. Let the block size which is to be exchange between two memory blocks be 1 KB i.e. 1024 byte.
3. The source block is at address 01000 H and destination block is at address 02000 H.
4. We will first copy 1 KB block starting at location 01000 H to another place at 03000 H onwards.

5. Then the block from 02000 H onwards is transferred to the first block i.e. starting block 01000 H.
6. Finally the block copied in location 03000 H onwards is transferred to the location 02000 H onwards.

Instruction	Comments
MOV AX, 0000 H	Move 0000 H to AX Register.
MOV DS, AX	
MOV ES, AX	
MOV SI, 1000 H	Move 1000 H to Source Index.
MOV DI, 3000 H	Move 3000 H to Destination Index.
MOV CX, 0400 H	
CLD	
REP MOVSB	
MOV SI, 2000 H	Move 2000 H to Source Index.
MOV DI, 1000 H	Move 1000 H to Destination Index.
MOV CX, 0400 H	
CLD	Clear Direction Flag.
REP MOVSB	
MOV SI, 3000 H	Move 3000 H to Source Index.
MOV DI, 2000 H	Move 2000 H to Destination Index.
MOV CX, 0400 H	
CLD	Clear Direction Flag.
REP MOVSB	

Therefore the content of blocks is exchange between 01000 H (First Memory Block) & 02000 H (Second Memory Block) using above 8086 Assembly Program.

Q3. Write assembly language program for 8086 to reverse a string of 10 characters.

Ans: **[P | Low]**

ASSEMBLY LANGUAGE:

1. Machine Language is the only language understood directly by CPU in computers.
2. So the instructions in the text form are called as Mnemonic.
3. Assembly Language is a Mnemonic representation of machine code.
4. There is a one-to-one correlation between assembly language instructions and the machine code.

FLOW CHART:

Figure 2.10 shows the flow chart to reverse a string using 8086 assembly language program.

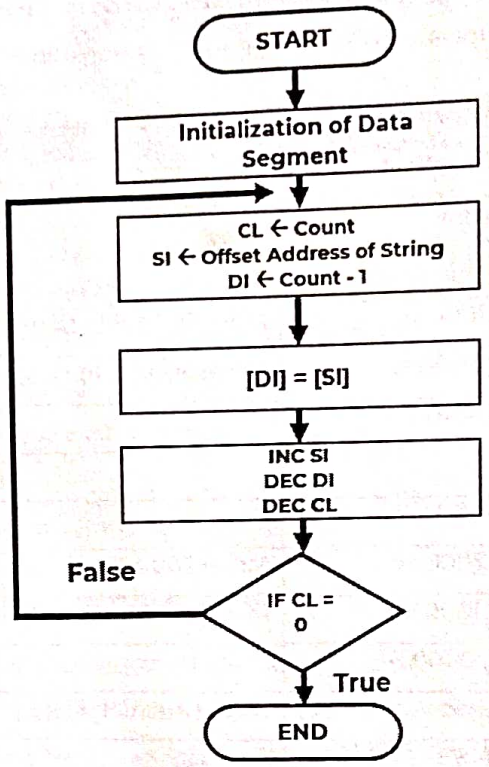


Figure 2.10: Flow chart to reverse a String.

PROGRAM TO REVERSE A STRING:

Let the string STR = "TECHNOLOGY"

Instruction	Comments
MOV AX, DATA	Move DATA to AX Register.
MOV DS, AX	Initialize DS
MOV CL, 10	Count is 10 Characters
MOV SI, OFFSET STR	
MOV DI, 9	Count – 1
REVERSE: MOV AL, [SI]	
XCHG [DI], AL	Exchange Contents of DI & AL
MOV [SI], AL	
INC SI	Increment Source Index.
DEC DI	Decrement Destination Index
DEC CL	
JNZ REVERSE	
HLT	

CHAP - 3: 8086 INTERRUPTS

Q1. Explain the interrupt structure of 8086 microprocessor

Q2. Types of interrupts

Ans: **[5-10M | Dec18 & May19] [P | Medium]**

8086 INTERRUPT STRUCTURE:

1. Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.
2. 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.
3. Figure 3.1 shows the types of interrupts we have in 8086 microprocessor.

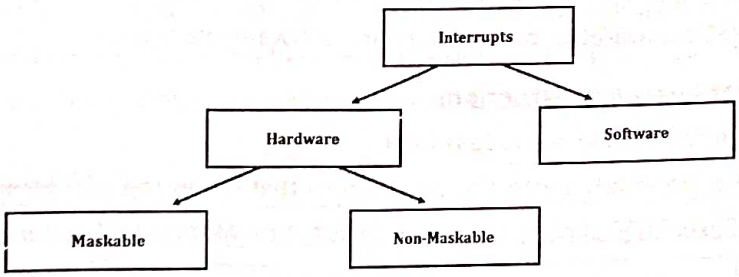


Figure 3.1: Types of Interrupts in 8086.

HARDWARE INTERRUPTS:

1. Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
2. The 8086 has two hardware interrupt pins, i.e. NMI and INTR.

I) NMI:

1. 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.

II) INTR:

1. The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction.
2. It should not be enabled using **clear interrupt Flag instruction**.
3. The INTR interrupt is activated by an I/O port.
4. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends '0' on INTA pin twice.
5. 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.

SOFTWARE INTERRUPTS:

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

It includes:

I) INT - Interrupt instruction with type number:

1. It is 2-byte instruction.
2. First byte provides the op-code and the second byte provides the interrupt type number.
3. There are 256 interrupt types under this group.
4. The first five pointers are dedicated interrupt pointers.
   - **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.
5. The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interr from 32 to Type 255 are available for hardware and software interrupts.

II) INT 3- Break Point Interrupt Instruction:

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

III) INTO - Interrupt on overflow instruction:

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

Q2. Explain the operation of three 8259 PIC in cascaded mode

[10M | May19] [P | Medi

Ans:

8259 PIC:

1. For the application where we require multiple interrupt sources, we need to use an external de called as **Programmable Interrupt Control (PIC)**.
2. By connecting a PIC to the microprocessor we can increase the interrupt handling capacity of microprocessor.
3. 8259 is commonly used PIC.

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN CASCADE MODE)

1. 8259 in cascade mode is preferred to increase number of interrupts more than 8 up to 64.
2. In this, three 8259 are used.
3. One 8259 master and other two 8259 are known as slave-0, slave-1.
4. For Master, $\overline{SP}/\overline{EN}$ → +5V, For Slave, $\overline{SP}/\overline{EN}$ → +0V.
5. First all 8259's are initialized by writing proper ICW's & OCW's.
6. INTR of 8086 is enabled by using the instruction STI (now IF is set).

It includes:

I) INT - Interrupt instruction with type number:

1. It is 2-byte instruction.
2. First byte provides the op-code and the second byte provides the interrupt type number.
3. There are 256 interrupt types under this group.
4. The first five pointers are dedicated interrupt pointers.
   - **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.
5. The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrup from 32 to Type 255 are available for hardware and software interrupts.

II) INT 3- Break Point Interrupt Instruction:

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

III) INTO - Interrupt on overflow instruction:

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

Q2. Explain the operation of three 8259 PIC in cascaded mode

[10M | May19] [P | Mediur

Ans:

8259 PIC:

1. For the application where we require multiple interrupt sources, we need to use an external dev called as **Programmable Interrupt Control (PIC)**.
2. By connecting a PIC to the microprocessor we can increase the interrupt handling capacity of t microprocessor.
3. 8259 is commonly used PIC.

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN CASCADE MODE)

1. 8259 in cascade mode is preferred to increase number of interrupts more than 8 up to 64.
2. In this, three 8259 are used.
3. One 8259 master and other two 8259 are known as slave-0, slave-1.
4. For Master, $\overline{SP}/\overline{EN}$ → +5V, For Slave, $\overline{SP}/\overline{EN}$ → +0V.
5. First all 8259's are initialized by writing proper ICW's & OCW's.
6. INTR of 8086 is enabled by using the instruction STI (now IF is set).

7. When request appears on any one of IR inputs of slave, then after resolving priority, slave makes INT high which is given at IR inputs of master.
8. After resolving priority, master makes INT high which is given to microprocessor.
9. Now microprocessor is interrupted.
10. Microprocessor gives 2 $\overline{INTA}$ pulses to 8259.
11. First $\overline{INTA}$ pulse is accepted only by master.
12. Now master gives slave identification number on CAS_2, CAS_1 & CAS_0.
13. The second $\overline{INTA}$ pulse is recognized only by that master for which its identification number matches with CAS_2, CAS_1 & CAS_0.
14. Now slave gives type number to 8086.
15. Now 8086 execute ISR.
16. At the end of ISR, 2 EOI command is executed.
17. One EOI command for master & another for slave to clear corresponding bit in ISR of master & slave.
18. Due to this, master & slave can respond to other low priority interrupts.
19. Figure 3.2 shows interfacing of 8259 with 8086 in minimum mode (8259 in cascade mode).

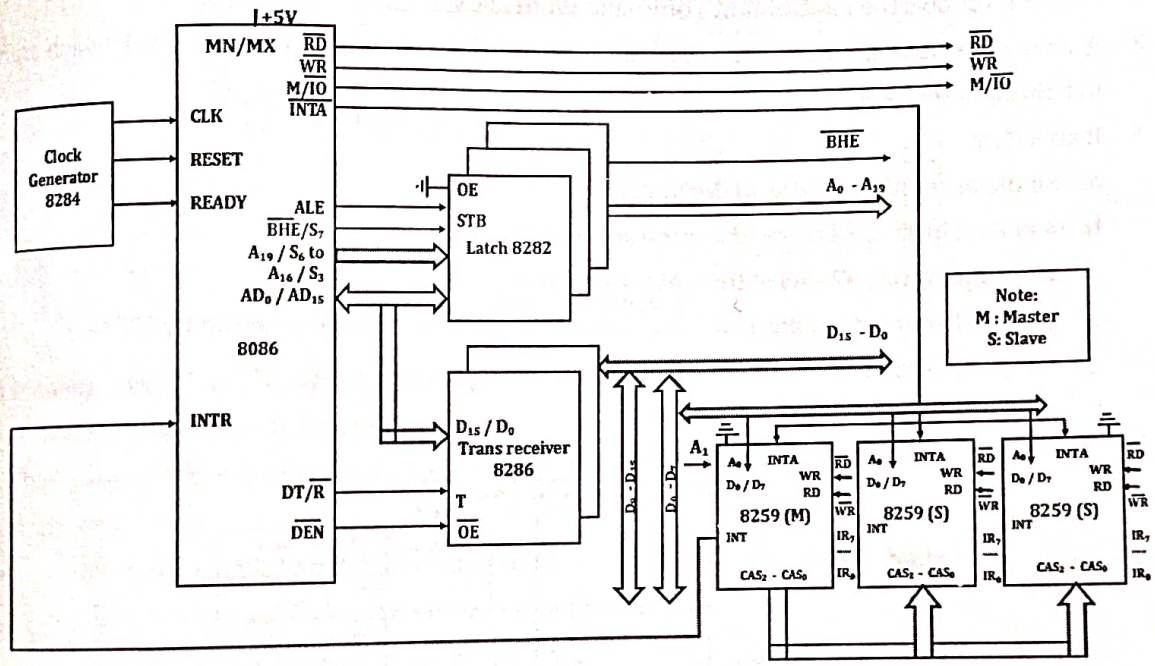


Figure 3.2: 8086 in Minimum Mode, 8259 in Cascade Mode.

Q3. Give formats of initialization command words (ICW's) of 8259 PIC

Ans: **[5M | Dec18] [P | Medium]**

8259 PIC INITIALIZATION COMMAND WORDS:

1. The Programming 8259 requires two types of command words. i.e. Initialization Command Words (ICWs) and Operational Command Words (OCWs).
2. The Programming 8259 can be initialized with four ICWs; the first two are compulsory, and the other two are optional based on the modes being used.
3. These words must be issued in a given sequence.

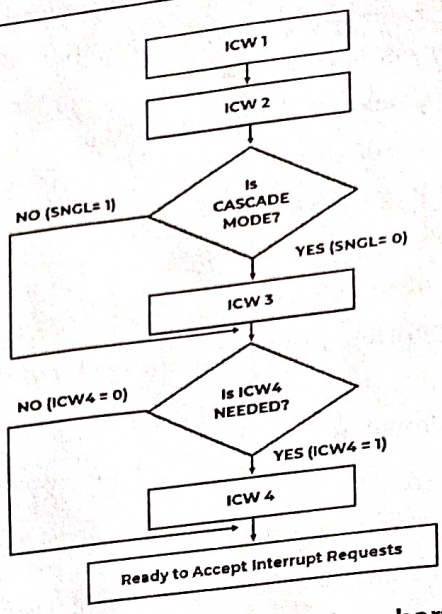


Figure 3.3: Initialization Flowchart.

I) Initialization Command Word 1 (ICW1):

1. Figure 3.4 shows the Initialization Command Word 1 (ICW1).
2. A write command issued to the 8259 with A0 = 0 and D4 = 1 is interpreted as ICW1, which starts initialization sequence.
3. It specifies:
   a. Single or multiple 8259As in the system.
   b. 4 or 8 bit interval between the interrupt vector locations.
   c. The address bits A7 – A5 of the CALL instruction.
   d. 3 bits of lower byte address of CALL are given by user, rest bits are inserted by 8259A
   e. Edge triggered or level triggered interrupts.
   f. ICW4 is needed or not.

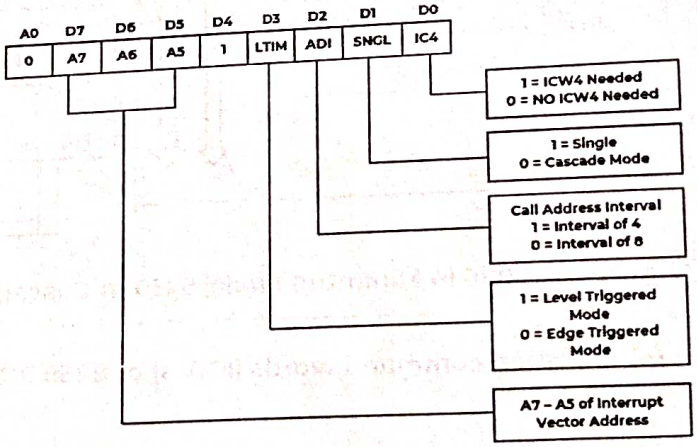


Figure 3.4: ICW1

II) Initialization Command Word 2 (ICW2):

1. Figure 3.5 shows the Initialization command Word 2 (ICW2).
2. A write command following ICW1, with A0 = 1 is interpreted as ICW2.
3. This is used to load the high order byte of the interrupt vector address of all the interrupts.

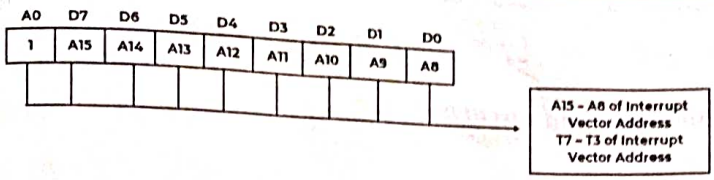


Figure 3.5: ICW2

III) Initialization Command Word 3 (ICW3):

1. ICW3 is required only if there is more than one 8259 in the system and if they are cascaded.
2. An ICW3 operation loads a slave register in the Programming 8259.
3. The format of the byte to be loaded as an ICW3 for a master 8259 or a slave is shown in the Figure 3.6.
4. For master, each bit in ICW3 is used to specify whether it has a slave 8259 attached to it on its corresponding IR (Interrupt Request) input.
5. For slave, bits D0 – D2 of ICW3 are used to assign a slave identification code (Slave ID) to the Programming 8259.

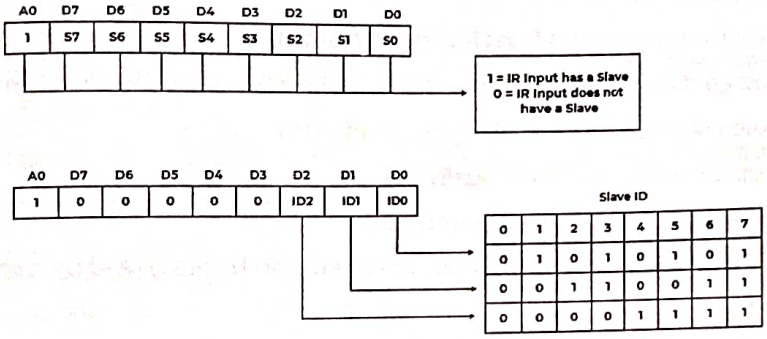


Figure 3.6: ICW3

IV) Initialization Command Word 4 (ICW4):

1. It is loaded only, if the D0 bit of ICW1 is set.
2. The format of ICW4 is shown in Figure 3.7.
3. It specifies:
   a. Whether to use special fully nested mode or non-special fully nested mode.
   b. Whether to use buffered mode or non-buffered mode.
   c. Whether to use Automatic EOI or Normal EOI.
   d. CPU used, 8086/8088 or 8085.

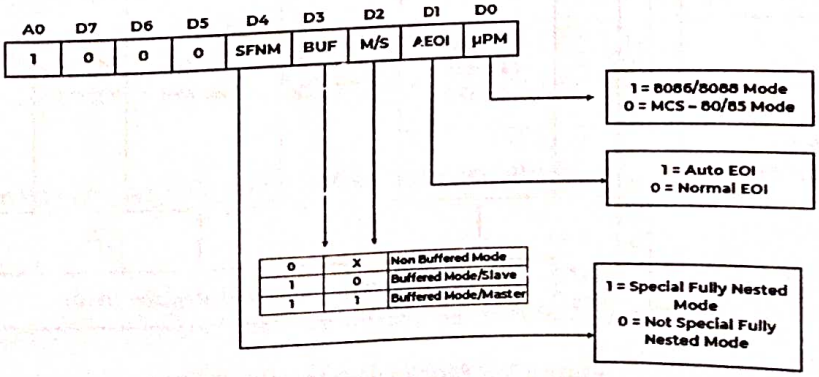


Figure 3.7: ICW4

-- EXTRA QUESTIONS --

Q1. Draw & Explain block diagram of 8259 PIC [P | High

Ans:

8259 PIC:

1. For the application where we require multiple interrupt sources, we need to use an external dev called as **Programmable Interrupt Control (PIC)**.
2. By connecting a PIC to the microprocessor we can increase the interrupt handling capacity of t microprocessor.
3. 8259 is commonly used PIC.

FEATURES OF 8259 PIC:

1. PIC 8259 is a **Programmable Interrupt Controller** that can work with 8085, 8086, etc.
2. 8259 has flexible priority structure.
3. 8259 PIC is used to implement **8 level interrupt system**.
4. While cascaded configuration of 1 master 8259 & 8 slave 8259s can handle up to 64 interrupt system
5. 8259 can handle edge as well as level triggered interrupts.
6. In 8259, interrupts can masked individually.
7. The vector address of interrupt is programmable.
8. Status of interrupts (pending, in-service, and masked) can be easily read by microprocessor.
9. 8259 does not requires clock signal.
10. It can also be **used in buffered mode**.

BLOCK DIAGRAM:

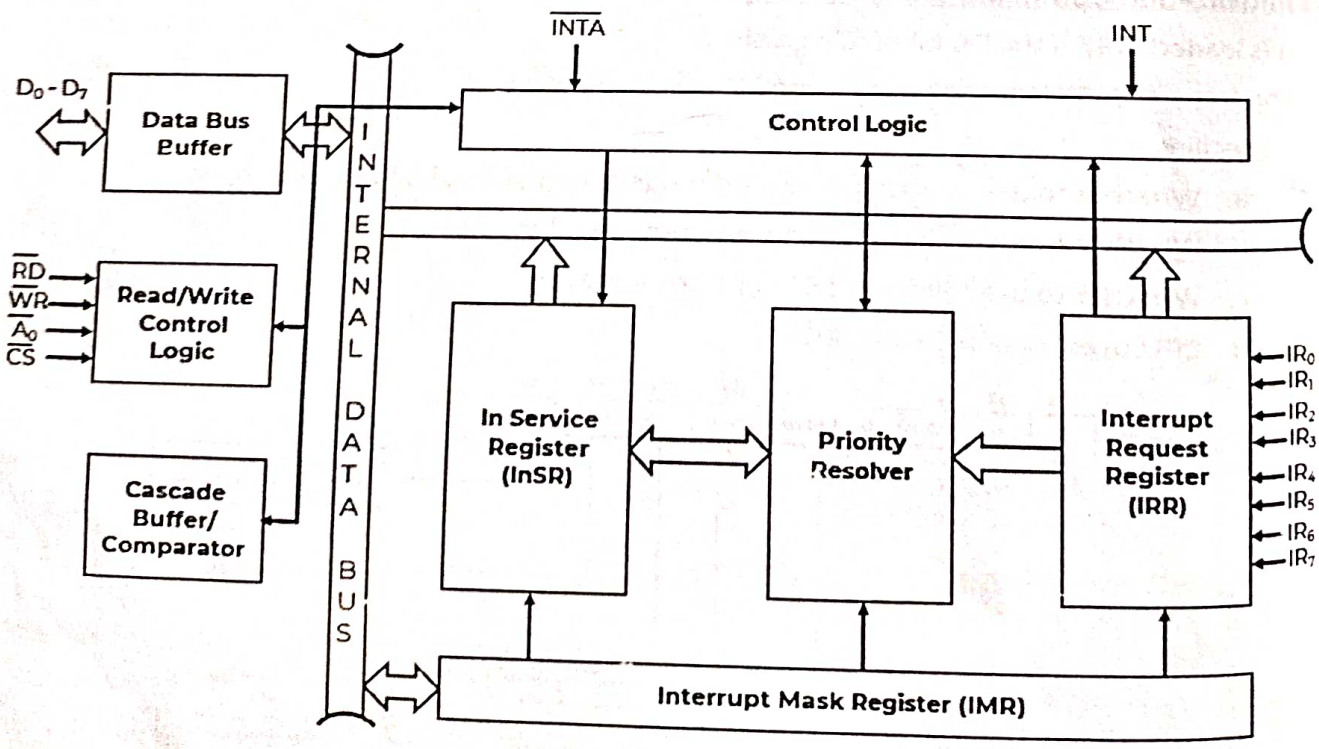


Figure 3.8: Block Diagram of 8259 PIC.

Block Diagram of 8259 PIC is shown in figure 3.8 & it contains following blocks:

II) **Data Bus Buffer:**

1. It is **8 bit Bidirectional Buffer.**
2. It is used to transfer data between microprocessor & internal bus.

III) **Read/Write Logic:**

1. It sets the direction of data bus buffer.
2. It controls all internal read/write operations.
3. It consists of Initialization Command Word Register (ICW) & Operation Command Word Register (OCW).

IIII) **Cascade Buffer & Comparator:**

1. In master mode, it functions as a cascaded buffer.
2. In slave mode, it functions as a comparator.
3. In Buffered mode it generates an $\overline{EN}$ Signal.

IV) **Control Logic:**

1. It has two signals: INT & $\overline{INTA}$
2. **INT:** It is an Output Signal. It is connected to INTR of microprocessor. Whenever this line goes high microprocessor is interrupted.
3. **$\overline{INTA}$:** It is an output signal to 8259. Whenever this line goes high microprocessor acknowledge the arrival of interrupt request to 8259.

V) **Interrupt Request Register (IRR):**

1. It has **8 Input Lines. ($IR_7 - IR_0$)**
2. Peripheral Devices are connected to these lines.
3. IRR is used to store all the pending interrupt requests.

VI) **In-Service Register (InSR):**

1. It is used to store all the Levels which are being serviced.
2. Microprocessor can read contents of this register by issuing appropriate command word.

VII) **Interrupt Mask Register (IMR):**

1. It is a **programmable register**.
2. It is used to mark some interrupt lines by selecting proper bits.
3. Microprocessor can read contents of this register without issuing any command word.

VIII) **Priority Resolver:**

1. Priority Register is used to examine **IRR, InSR & IMR**.
2. It determines which INT should be sent out of IRR (IR_7 to IR_0) to microprocessor.

Q2. Explain Interfacing of 8259 with 8086 in minimum mode.

Ans: **[P | Medium]**

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN SINGLE MODE):

1. Figure 3.9 shows interfacing of 8259 with 8086 in minimum mode (8259 in single mode).

2. 8259 is first initialized by writing proper **ICW's (Initialization Command Words) & OCW's (Operation Command Words)**.
3. INTR of 8086 is enabled by using the **STI Instruction.**
4. When request appears on any one of IR inputs, then after resolving priority, 8259 makes INT high.
5. Now microprocessor is interrupted.
6. Microprocessor gives 2 $\overline{INTA}$ pulses to 8259.
7. During 1st $\overline{INTA}$ pulse, **8259 calculates type number.**
8. At the same time the corresponding bit in ISR is set.
9. During 2nd $\overline{INTA}$ pulse, **8259 gives type number.**
10. Now 8086 execute particular type ISR.
11. At the end of ISR, EOI command is executed.
12. Due to this, corresponding bit in ISR is reset & so 8259 can response to other low priority interrupts.

DIAGRAM:

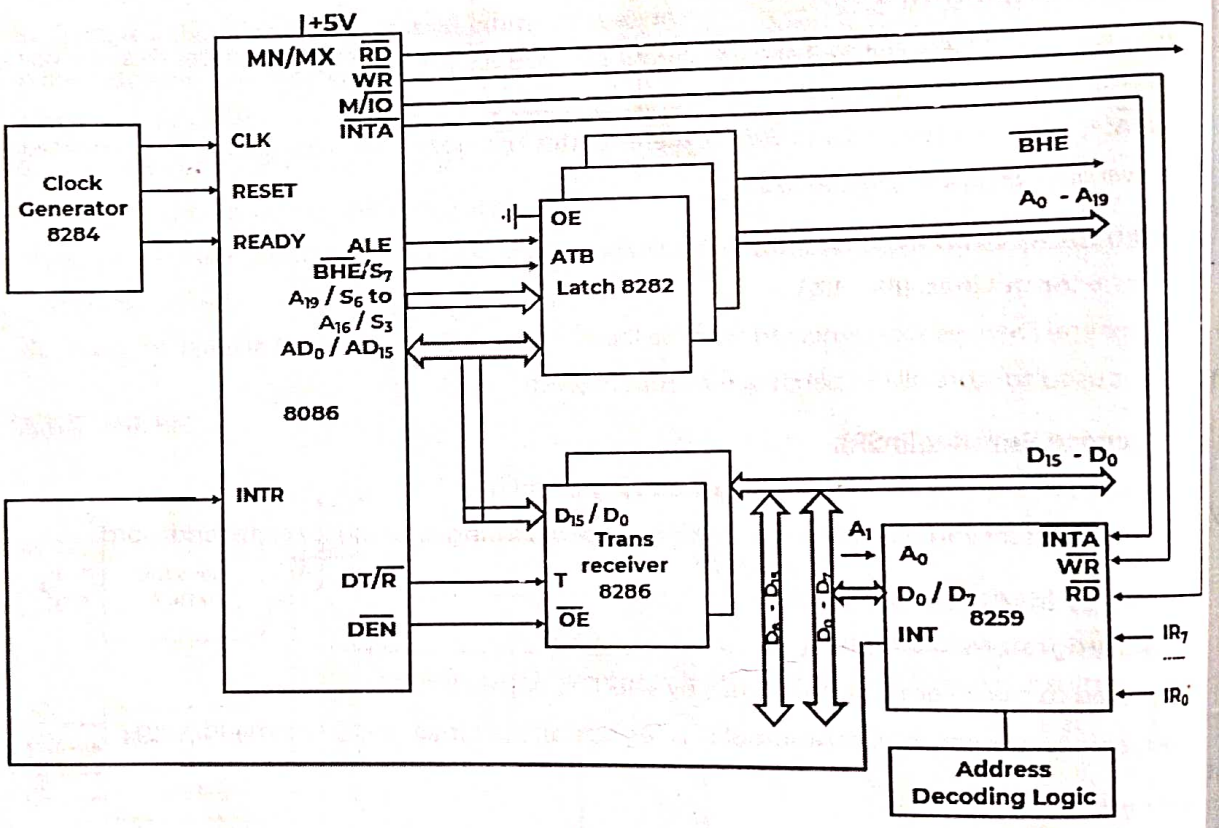


Figure 3.9: 8086 in Minimum Mode, 8259 in Single Mode.

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN CASCADE MODE)

Refer Q2 (University Question)

CHAP - 4: PERIPHERALS AND THEIR INTERFACING WITH 8086

Q1. Draw and explain the block diagram of 8257 DMA controller [10M | Dec18] [P | Medium]

Ans:

DMA CONTROLLER:

1. DMA stands for Direct Memory Access.
2. It is designed by Intel to transfer data at the fastest rate.
3. It allows the device to transfer the data directly to/from memory without any interference of the CPU.

FEATURES OF 8257:

1. It has four channels which can be used over four I/O devices.
2. Each channel has 16-bit address and 14-bit counter.
3. Each channel can transfer data up to 64kb.
4. Each channel can be programmed independently.
5. Each channel can perform read transfer, write transfer and verify transfer operations.
6. It generates MARK signal to the peripheral device that 128 bytes have been transferred.
7. It requires a single phase clock.
8. Its frequency ranges from 250Hz to 3MHz.
9. It operates in 2 modes, i.e., **Master mode** and **Slave mode**.

8257 BLOCK DIAGRAM:

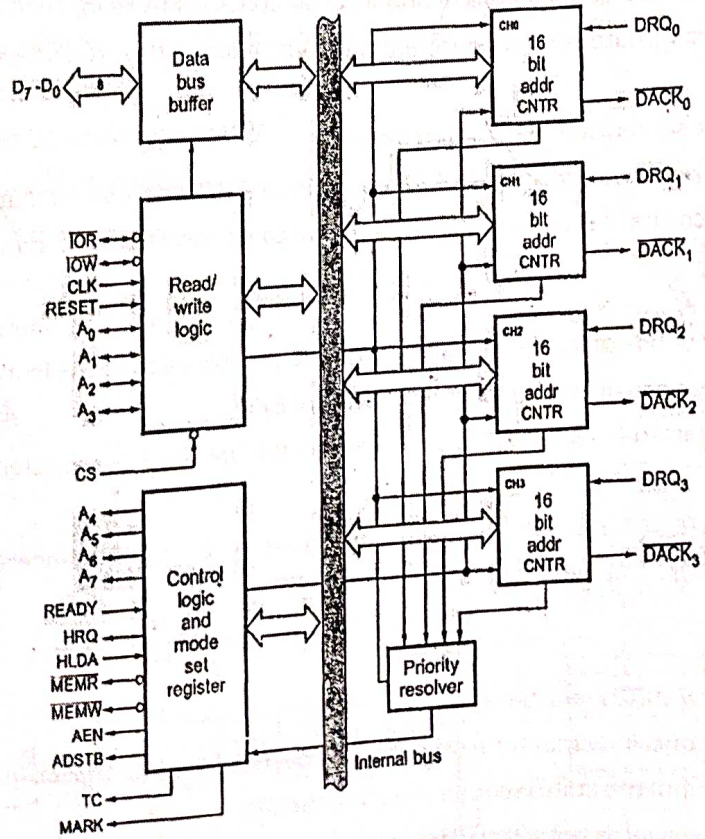


Figure 4.1: Block Diagram of 8257 DMA

I) Data Bus Buffer:

1. It is a tri-state, bi-directional, 8 bit buffer which interfaces the 8257 to the system data bus.
2. In the slave mode, it is used to transfer data between microprocessor and internal registers of 8257.
3. In master mode, it is used to send higher byte address (A8-A15) on the data bus.

II) Read/Write logic:

1. When the 8257 is in the slave mode, the Read/Write logic:
   a. Accepts the I/O Read (IOR) or I/O Write (IOW) signal.
   b. Decodes the least significant four address bits (A0 – A3)
   c. Writes the contents of the data bus into the addressed register (if IOW is low) or places the conte of the addressed register onto the data bus (if IOR is low).
2. During DMA cycles (i.e. when the 8257 is in the master mode) the Read/Write logic:
   a. Generates the I/O read and memory write (DMA write cycle) or I/O write and memory read (D read cycle) signals which control the data transfer between peripheral and memory device.

III) DMA Channels:

1. The Pin Diagram of 8257 provides four identical channels, labeled CH_0 to CH_3.
2. Each channel has two sixteen bit registers:
   a. A DMA address register.
   b. A terminal count register.

IV) Control Logic:

1. It controls the sequence of operations during all DMA cycles (DMA read, DMA write, DMA verify generating the appropriate control signals and the 16-bit address that specifies the memory locat to be accessed.
2. It consists of mode set register and status register.
3. Mode set register is programmed by the CPU to configure 8257 whereas the status register is read CPU to check which channels have reached a terminal count condition and status of update flag.

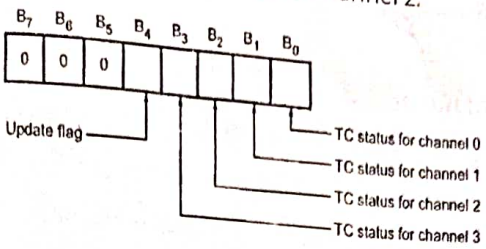
V) Mode Set Register:

1. Least significant four bits of mode set register, when set, enable each of the four DMA channels.
2. Most significant four bits allow four different options of 8257.
3. It is normally programmed by the CPU after initializing the DMA address registers and terminal co registers.
4. It is cleared by the RESET input, thus disabling all options, inhibiting all channels, and preventing conflicts on power-up.

VI) Status Register:

1. Figure 4.2 shows the status register format.
2. The TC status bit, if one, indicates terminal count has been reached for that channel.
3. TC bit remains set until the status register is read or the 8257 is reset.
4. The update flag, however, is not affected by a status read operation.
5. The update flag bit, if one, indicates CPU that 8257 is executing update cycle.

6. In update cycle 8257 loads parameters in channel 3 to channel 2.



VII) Priority Resolver:

1. It resolves the peripherals requests.
2. It can be programmed to work in two modes, either in fixed mode or rotating priority mode.

Q2. Draw and explain the block diagram of 8255 Programmable Peripheral Interface (PPI) with control word formats

[10M | May19] [P | Medium]

Ans:

8255 PPI:

1. 8255 is a **Programmable Peripheral Interface** i.e. PPI 8255.
2. It is general purpose programmable parallel I/O Device.
3. It has three 8-bit bidirectional I/O Ports: **Port A, Port B and Port C**.

FEATURES:

1. 8255 PPI contains 24 programmable I/O pins arranged as 2 8-bit ports & 2 4-bit ports.
2. 8255 PPI contains 3 ports and they are arranged in two groups of 12 pins.
3. It is fully compatible with Intel Microprocessor families.
4. It is also TTL compatible.
5. It has improved DC driving capability.
6. 8255 can operates in 3 modes:
   a. **Mode 0:** Simple I/O.
   b. **Mode 1:** Strobed I/O.
   c. **Mode 2:** Strobed bidirectional I/O.

BLOCK DIAGRAM:

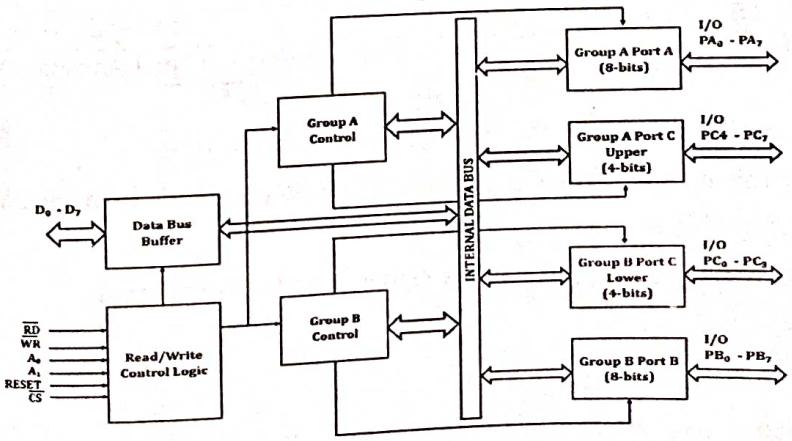


Figure 4.3: 8255 PPI Block Diagram.

It contains the following blocks:

I) Data Bus Buffer:

1. This is **8-bit bidirectional buffer.**
2. It is used to interface the internal data bus of 8255 with the external data bus.
3. When read is activated, it transmits data to the system data bus.
4. When write is activated, it receives data from the system data bus.

II) Read/Write Control Logic:

1. It accepts address and control signals from the microprocessor.
2. The control signals are read & write while address signals used are A_0 & A_1 and $\overline{CS}$.
3. The address bits (A_1, A_0) are used to select the ports or the control word register as shown:

For 8255		For 8086		Selection	Sample Address
A_1	A_0	A_2	A_1		
0	0	0	0	Port A	80 H (i.e. 1000 0000)
0	1	0	1	Port B	82 H (i.e. 1000 0010)
1	0	1	0	Port C	84 H (i.e. 1000 0100)
1	1	1	1	Control Word	86 H (i.e. 1000 1000)

4. $\overline{CS}$ is connected to address chip select decoder.
5. The 8255 operation/selection is enabled/disabled by $\overline{CS}$ signal.

III) Group A & Group B Control:

1. 8255 I/O ports are divided into 2 sections: **Group A & Group B**
2. Group A control block controls Port A & Port C upper i.e. $PC_7 - PC_4$.
3. Group B control block controls Port B & Port C lower i.e. $PC_3 - PC_0$.
4. Each group is programmed through software.
5. Group A & Group B controls accepts the control signals from the control word and forwards t the respective ports.

IV) Port A, Port B & Port C:

1. This are 8-bit bidirectional ports.
2. They can be programmed to work in the various modes as follows:

Port	Mode 0	Mode 1	Mode 2
Port A	Yes	Yes	Yes
Port B	Yes	Yes	No (Mode 0 or Mode 1)
Port C	Yes	No (Handshake signals)	No (Handshake signals)

3. Group A control block controls Port A & Port C upper i.e. $PC_7 - PC_4$.
4. Only Port C can also be programmed to work in Bit Set/Reset Mode to manipulate its individu
5. Port C function is dependent on mode of operation.
6. It can be used as **Simple I/O, Handshake Signals & Status Signal Input**.

Q3. Explain the I/O mode control word format of 8255 PPI **[5M | Dec18] [P | Medium]**

Ans:

Control Word Register defines the function of each I/O Port and in which mode they should operate.

CONTROL WORD OF 8255: (I/O MODE)

1. To perform 8-bit data transfer using Ports A, B & C, 8255 needs to be in the I/O Mode.
2. The bit pattern for the control word in the $\overline{I}/O$ Mode is as follows:
3. Figure 4.4 shows the bit pattern of control word in 8255 $\overline{I}/O$ Mode.

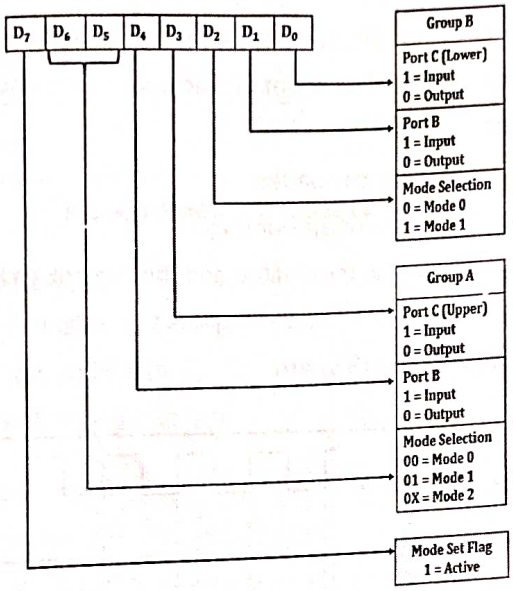


Figure 4.4: Bit pattern of control word in 8255 I/O Mode.

CONTROL WORD OF 8255: (BSR MODE)

1. The BSR Mode is used only for Port C.
2. In this mode the individual bits of Port C can be set or reset.
3. This is very useful as it provides 8 individually controllable lines which can be used while interfacing with devices like an A to D Converter or a 7-segment display etc.
4. The individual bit is selected and set/reset through the control word.
5. Since the D7 bit of the control word is 0, the BSR operation will not affect the I/O operations of 8255.
6. Figure 4.5 shows bit pattern for the control word in the BSR Mode.

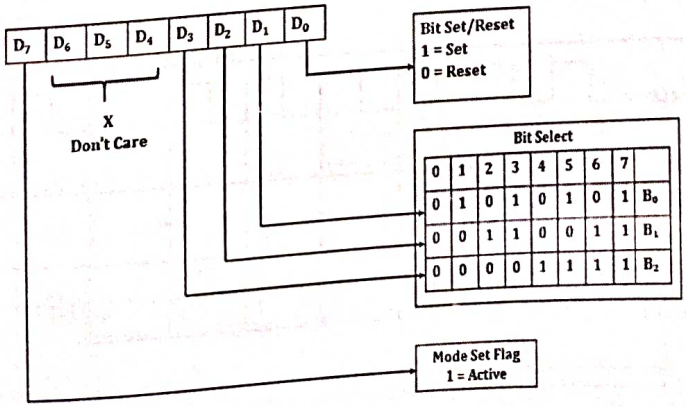


Figure 4.5: Bit pattern of control word in 8255 BSR Mode.

Q4. Modes of 8253 Programmable Interval timer [5M | May19] [P | Mediu

Ans:

OPERATING MODES OF 8253/8254:

I) Mode 0: Interrupt on Terminal Count.

1. It helps in generating an interrupt to the microprocessor after a specific interval.
2. Initially the output becomes low after the mode is set.
3. The output becomes LOW when the count value is loaded into the counter.
4. The output becomes low as the loaded count value decreases per every cycle.
5. This process continues till the terminal count is reached to zero and the output goes HIGH and considered as an interrupt.
6. It will remain high until a new count is reloaded.
7. The GATE signal would be high for normal counting.
8. When GATE goes low, counting gets terminated and the current count will be latched till the G goes high again.
9. Figure 4.6 shows the Mode 0 Timing Diagram.

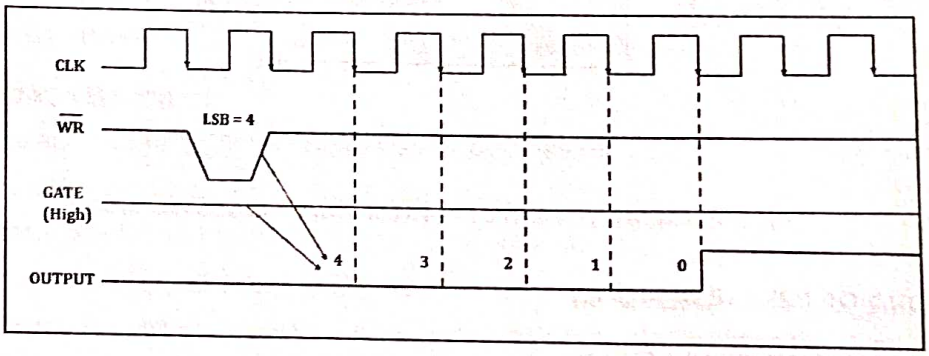


Figure 4.6: Mode 0 Timing Diagram.

II) Mode 1: Programmable One Slot.

1. It can be used as a **mono stable multi-vibrator**.
2. The gate input is used as a trigger input and the output becomes high in this mode.
3. The output remains high at the end of the count that is loaded and a trigger is applied.
4. Figure 4.7 shows the Mode 1 Timing Diagram.

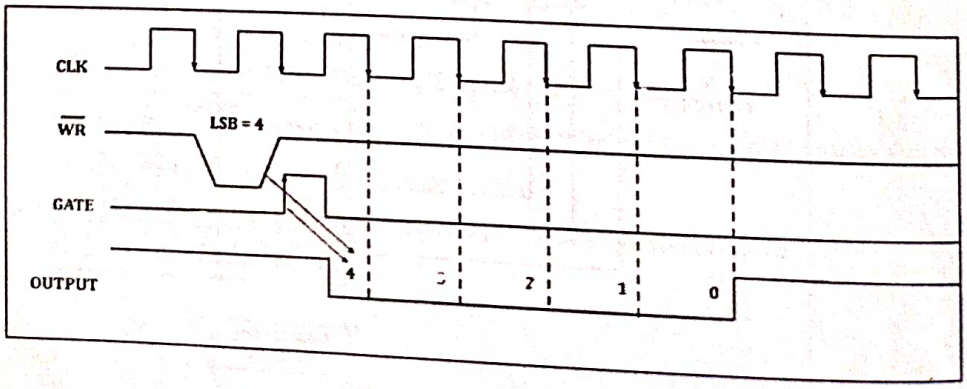


Figure 4.7: Mode 1 Timing Diagram.

iii) **Mode 2: Rate Generator.**

1. The output remains normally high after initialization.
2. Whenever the count comes to zero, generation of low pulse occurs at the output and reloading of the counter will take place.
3. Figure 4.8 shows the Mode 2 Timing Diagram.

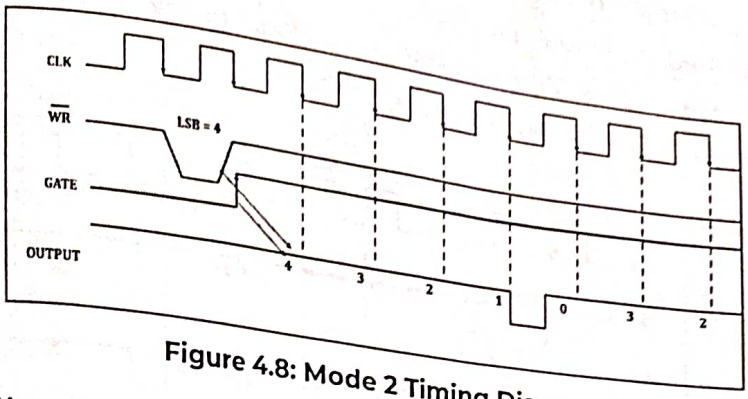


Figure 4.8: Mode 2 Timing Diagram.

iv) **Mode 3: Square Wave Generator.**

1. This mode acts similar to that of Mode 2 except the output remains low for half of the timer period and other half of the period it should be high.
2. Figure 4.9 shows the Mode 3 Timing Diagram.

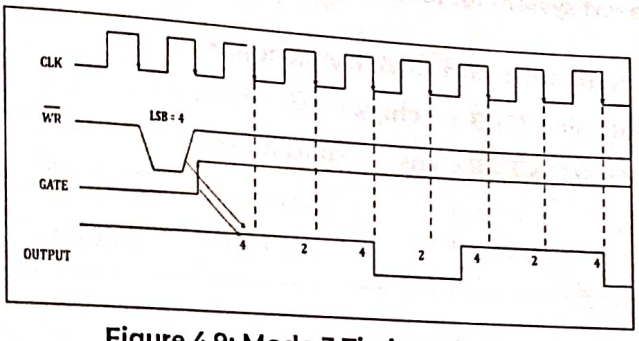


Figure 4.9: Mode 3 Timing Diagram.

v) **Mode 4: Software Triggered Strobe.**

1. In this mode, the output would remain high until the timer has to be counted to zero, at certain point the output would pulse low and then would go high again.
2. The count will become latched when the GATE signal goes LOW.
3. At the end of the count, the output will go low for one clock cycle then it would go HIGH. This low pulse can be used as a strobe.
4. Figure 4.10 shows the Mode 4 Timing Diagram.

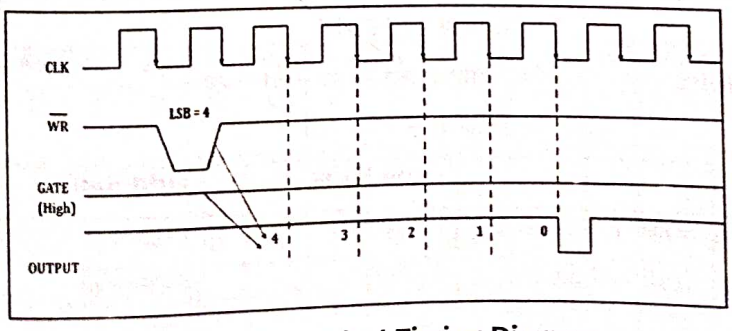


Figure 4.10: Mode 4 Timing Diagram.

VI) Mode 5: Hardware Triggered Strobe.

1. This mode produces a strobe with response to an externally generated signal.
2. This mode acts similar to that of mode 4 except that the counting was initiated by a signal at the ga input, which implies that it is hardware triggered instead of software triggered.
3. After it gets initialized, the output goes high.
4. When the count was reached at the end, the output would go low for one clock cycle.
5. Figure 4.11 shows the Mode 5 Timing Diagram.

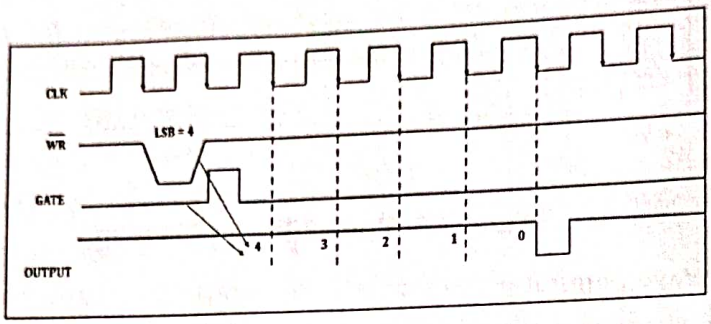


Figure 4.11: Mode 5 Timing Diagram.

Q5. Design 8086 based system for following specifications:

i) 8086 in minimum mode with clock frequency 5MHz.

ii) 128 KB EPROM using 32KB x 8 chips

iii) 32 KB RAM using 16KB x 8 chips

Ans: **[10M | May19] [P | Medi**

SOLUTION:

Step 1:

Total ROM Required	= 128 KB
Chip Size Available	= 32 KB
No. of chips required	= 128 KB/32 KB = 4
No. of sets required	= 4/2 = 2

Set 1:

Ending Address	= FFFFF H
Set Size	= Chip Size x 2
	= 32 x 2
	= 64 KB (0FFFF H)
Starting Address	= Ending Address – Set Size
	= F0000 H

	Even Bank	Odd Bank
Starting Address	F0000 H	F0001 H
Ending Address	FFFFE H	FFFFF H

Set 2:

Ending Address = Previous Starting – 1 = EFFFF H

Set Size = 64 KB (0FFFF H)

Starting Address = EFFFF H – 0FFFF H

= E0000 H

	Even Bank	Odd Bank
Starting Address	E0000 H	E0001 H
Ending Address	EFFFE H	EFFFF H

Step 2:

Total RAM Required = 32 KB.

Chip Size Available = 16 KB

No. of chips required = 32 KB/16 KB = 2

No. of sets required = 2/2 = 1

Set 1:

Starting Address = 00000 H

Set Size = Chip Size x 2

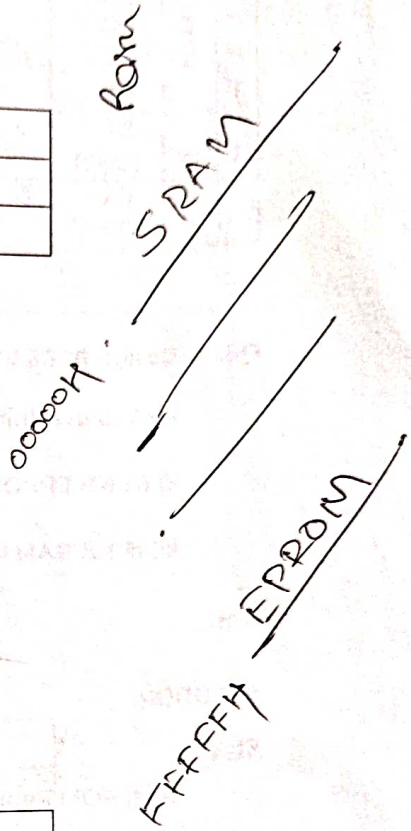
= 16 x 2

= 32 KB (07FFF H)

Ending Address = Starting Address + Set Size

= 00000 H + 07FFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	07FFE H	07FFF H



Step 3: Memory Map

EPROM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = F0000	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F0001	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-2	EB	SA = E0000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = EFFFE	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = E0001	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = EFFFF	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

RAM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 07FFE	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 07FFF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Q6. Design 8086 based system for following specifications:

i) 8086 in minimum mode with clock frequency 5MHz.

ii) 64 KB EPROM using 16KB x 8 chips.

iii) 16 KB RAM using 8KB x 8 chips

[10M | Dec18] [P | Medium]

Ans:

SOLUTION:

Step 1:

Total ROM Required = 64 KB

Chip Size Available = 16 KB

No. of chips required = 64 KB/16 KB = 4

No. of sets required = 4/2 = 2

Set 1:

Ending Address = FFFFF H

Set Size = Chip Size x 2

= 16 x 2

= 32 KB (07FFF H)

Starting Address = Ending Address – Set Size

= F8000 H

	Even Bank	Odd Bank
Starting Address	F8000 H	F8001 H
Ending Address	FFFFE H	FFFFF H

Set 2:

Ending Address = Previous Starting – 1

= F8000 H – 1

= F7FFF H

Set Size = Chip Size x 2

= 16 x 2

= 32 (07FFF H)

Starting Address = Ending Address – Set Size
= F7FFF H - 07FFF H
= F0000 H

	Even Bank	Odd Bank
Starting Address	F0000 H	F0001 H
Ending Address	F7FFE H	F7FFF H

Step 2:

Total ROM Required = 16 KB
Chip Size Available = 8 KB
No. of chips required = 16 KB/8 KB = 2
No. of sets required = 2/2 = 1

Set 1:

Starting Address = 00000 H
Set Size = Chip Size x 2
= 8 x 2
= 16 KB (03FFF H)
Ending Address = Starting Address + Set Size
= 00000 H + 03FFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	03FFE H	03FFF H

Step 3: Memory Map

EPROM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = F8000	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F8001	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-2	EB	SA = F0000	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = F7FFE	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F0001	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = F7FFF	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting Address = Ending Address – Set Size
= F7FFF H - 07FFF H
= F0000 H

	Even Bank	Odd Bank
Starting Address	F0000 H	F0001 H
Ending Address	F7FFE H	F7FFF H

Step 2:

Total ROM Required = 16 KB
Chip Size Available = 8 KB
No. of chips required = 16 KB/8 KB = 2
No. of sets required = 2/2 = 1

Set 1:

Starting Address = 00000 H
Set Size = Chip Size x 2
= 8 x 2
= 16 KB (03FFF H)
Ending Address = Starting Address + Set Size
= 00000 H + 03FFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	03FFE H	03FFF H

Step 3: Memory Map

EPROM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = F8000	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F8001	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-2	EB	SA = F0000	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = F7FFE	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F0001	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = F7FFF	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

RAM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 03FFE	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 03FFF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

RAM:

			A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
SET-1	EB	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 03FFE	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 03FFF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

-- EXTRA QUESTIONS --

Q1. 8087 Math Co-processor.

Ans: [P | Medium]

FEATURES:

1. 8087 is a **high performance numeric co-processor.**
2. It can work on integer, decimal and real type numbers.
3. It has an instruction set capable of doing complex arithmetic and trigonometric calculations.
4. It follows **IEEE Floating Point Standard.**

BLOCK DIAGRAM:

Figure 4.12 shows the block diagram of 8087 Math Co-processor. It consists of following blocks:

I) **Control Unit:**

1. It receives all instructions for the **Numeric Data Processor (NDP).**
2. Instructions involving the register stack are given to the **Numeric Execution Unit (NEU)**, while the control unit executes the remaining instructions.
3. Its main components are:
   a. Instruction Queue.
   b. Control Word.
   c. Status Word.
   d. Instruction Pointer.
   e. Data Pointer.

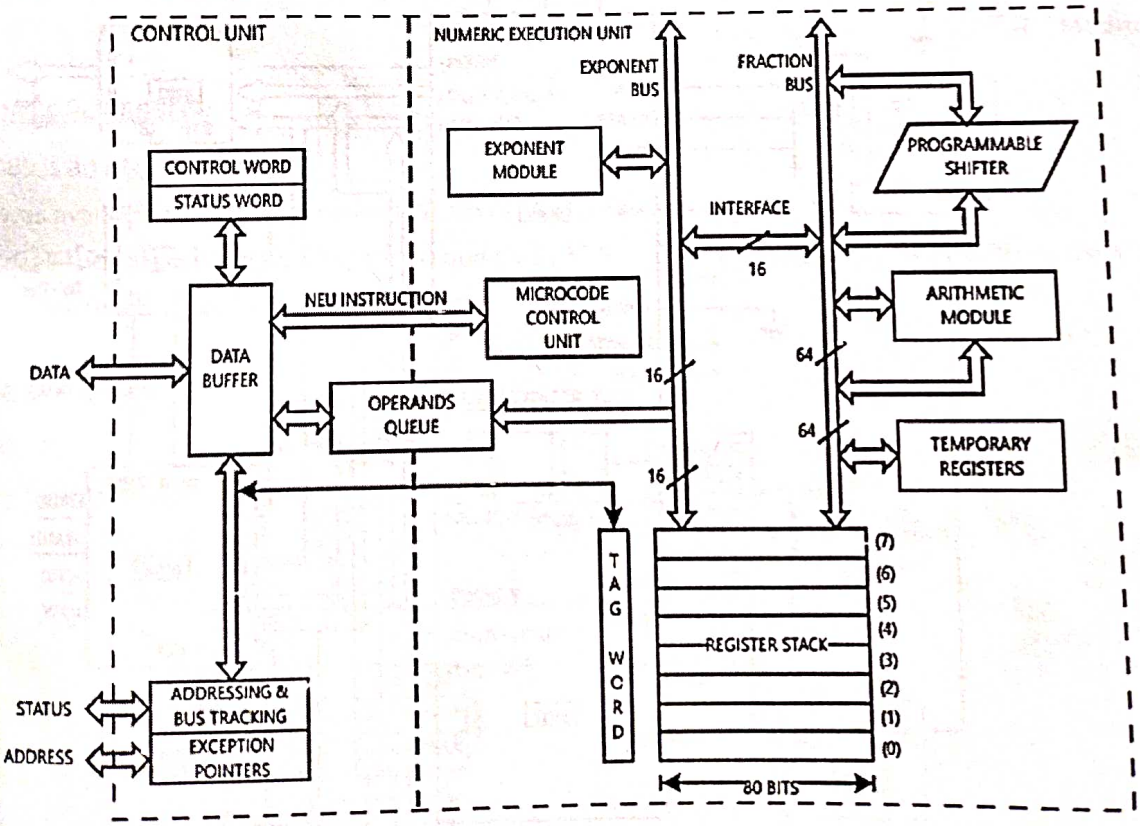


Figure 4.12: Block Diagram of 8087 Math Coprocessor.

-- EXTRA QUESTIONS --

Q1. 8087 Math Co-processor. [P | Medium]

Ans:

FEATURES:

1. 8087 is a **high performance numeric co-processor.**
2. It can work on integer, decimal and real type numbers.
3. It has an instruction set capable of doing complex arithmetic and trigonometric calculations.
4. It follows **IEEE Floating Point Standard.**

BLOCK DIAGRAM:

Figure 4.12 shows the block diagram of 8087 Math Co-processor. It consists of following blocks:

I) Control Unit:

1. It receives all instructions for the **Numeric Data Processor (NDP).**
2. Instructions involving the register stack are given to the **Numeric Execution Unit (NEU)**, while the control unit executes the remaining instructions.
3. Its main components are:
   a. Instruction Queue.
   b. Control Word.
   c. Status Word.
   d. Instruction Pointer.
   e. Data Pointer.

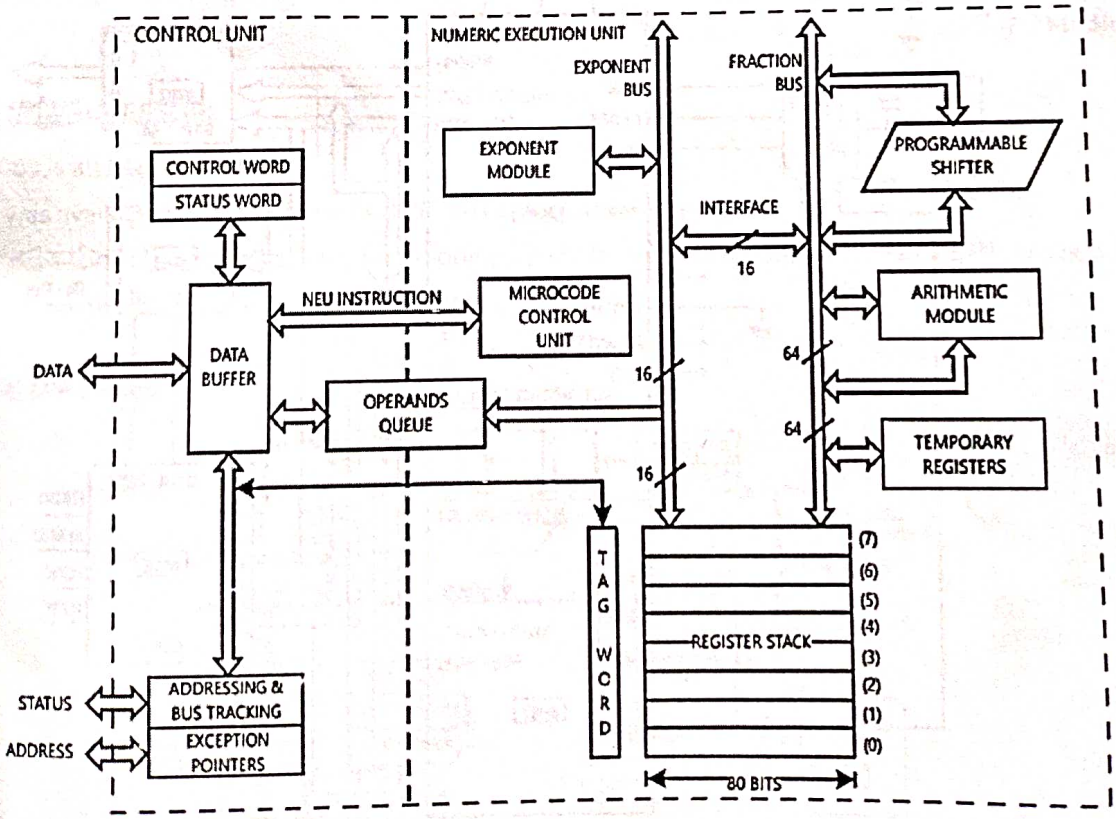


Figure 4.12: Block Diagram of 8087 Math Coprocessor.

II) Numeric Execution Unit:

1. Instructions involving the register stack are executed by the NEU.
2. It uses operands from register stack.
3. During arithmetic operations, 8087 handles exponents and mantissas separately.
4. **Example:** For addition, it will first compare and equalize the exponents by shifting, and only then add the mantissa.
5. Its main components are:
   a. Register Stack.
   b. Tag Word.

Q2] Draw and explain interfacing of math coprocessor (8087) with 8086 [P | Medium]

Ans:

8087 MATH CO-PROCESSOR:

1. 8087 is a **high performance numeric co-processor.**
2. It can work on integer, decimal and real type numbers.
3. It has an instruction set capable of doing complex arithmetic and trigonometric calculations.
4. It follows **IEEE Floating Point Standard**.

INTERFACING OF 8086 WITH 8087:

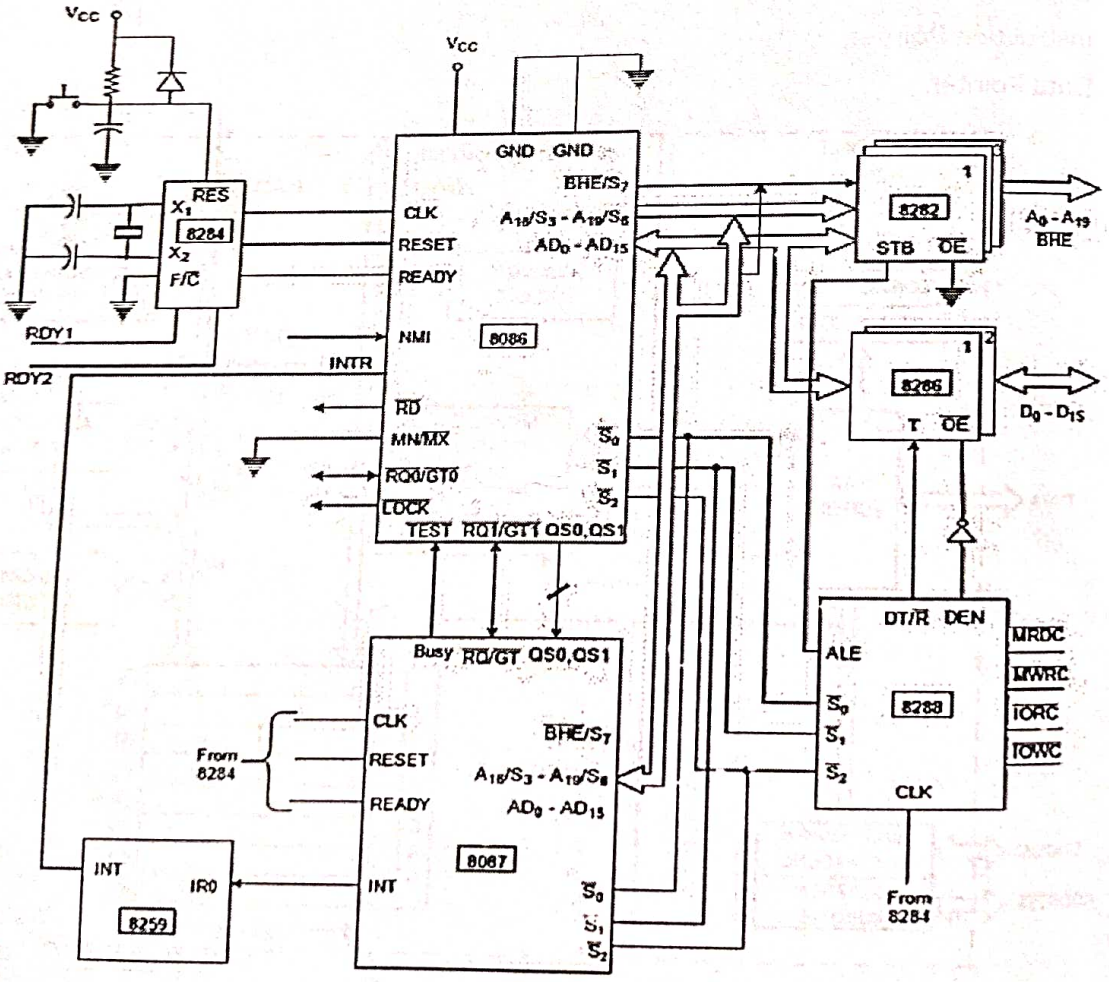


Figure 4.12: Interfacing diagram of 8086 with 8087.

1. Figure 4.12 shows interfacing diagram of 8086 with 8087.
2. 8087 can be connected to 8086 / 8088 only in their maximum mode of operation.
3. In the maximum mode, all the control signals are derived using a separate chip called as a **bus controller**.
4. The **8288** is a bus controller compatible with 8086 / 8088.
5. 8288 generates control signals using $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$ as input from the currently active processor.
6. The BUSY pin of 8087 is connected to the pin of the CPU.
7. The QS_0 and QS_1 lines may be directly connected to the corresponding pins in the case of 8086 /8088 based systems.
8. The clock pin of 8087 is connected to clock input of CPU.
9. The interrupt output of 8087 is connected to the CPU through a **Programmable Interrupt Controller 8259**.
10. 8259 PIC is used to accept the interrupt from 8087 and send it to the microprocessor.
11. 8284 provides the common CLK, RESET and READY signals.
12. 8282 are used to latch the address.
13. 8286 are used as **data trans-receivers**.
14. This interface is also called as **coprocessor configuration**.
15. Here 8086 is called as the host and 8087 as coprocessor as it cannot operate all by itself.
16. We write a homogeneous program which contains both 8086 as well as 8087 instructions.
17. Only 8086 can fetch instructions but these instructions also enter 8087. 8087 treats 8086 instructions as NOP.

Q3. 8089 I/O Processor **[P | Medium]**

Ans:

8089 I/O PROCESSOR:

1. 8089 is an **I/O processor**.
2. It was available for use with the 8086/8088 central processor.
3. It uses the same programming technique as 8087 for I/O Operations, such as transfer of data from memory to a peripheral device.

BLOCK DIAGRAM:

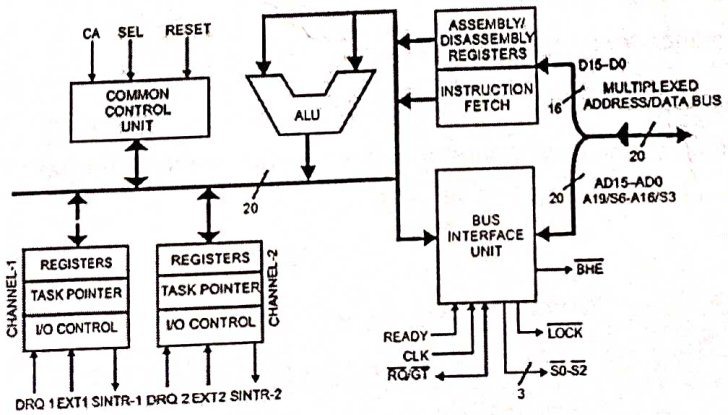


Figure 4.13: 8089 I/O Processor Block Diagram.

FEATURES:

1. 8089 has very **high speed DMA capability**.
2. It has 1 MB address capability.
3. It is compatible with iAPX 86, 88.
4. It supports local mode and remote mode I/O processing.
5. 8089 allows mixed interface of 8-and 16-bit peripherals, to 8-and 16-bit processor buses.
6. It supports **two I/O channels**.
7. Multibus compatible system interface.
8. Memory based communications with CPU.

COMPONENTS:

I) Common Control Unit (CCU):

1. 8089 I/O Processor has **two channels**.
2. The activities of these two channels are controlled by CCU.
3. CCU determines which channel—1 or 2 will execute the next cycle.
4. In a particular case where both the channels have equal priority, an interleave procedure is adopted in which each alternate cycle is assigned to channels 1 and 2.

II) Arithmetic & Logic Unit (ALU):

1. ALU is used to perform the **Arithmetic & Logical operations**.
2. It performs Arithmetic Operations like Addition, Subtraction & Logical Operations like AND, OR, EX-OR etc.
3. ALU looks after the branching decisions.

III) Assembly/Disassembly Registers.

1. This registers permits 8089 to deal with 8-or 16-bit data width devices or a mix of both.
2. In a particular case of an 8–bit width I/O device inputting data to a 16-bit memory interface, 8089 capture two bytes from the device and then write it into the assigned memory locations with the help of assembly/disassembly register.

IV) Bus Interface Unit (BIU):

1. Fetch the instruction or data from primary memory.
2. Read / Write of data from / to primary memory.
3. I/O of data from / to peripheral ports.
4. Address generation for memory reference.

V) Instruction Fetch:

1. It is used to fetches the instructions from the external memory and stores them in the Queue to be executed further.

Draw & Explain Block Diagram of 8253 PIT. [P | High]

ANS:

1. 8253/8254 is used as **Programmable Interval Timer** i.e. 8253/8254 PIT.
2. It is used as a device to produce Hardware delays.

FEATURES:

8253/8254 PIT has 3 independent 16 bit down counters.
Counters can be programmed in 6 different programmable counter modes.
This counters can take a count in BCD or Binary.
It is compatible with Intel & other Microprocessor.
It consists of single +5V supply.
It consists of 24 dual in-line package.
It is completely TTL Compatible.
It can be used to generate a real time clock, or a square wave generator etc.

BLOCK DIAGRAM:

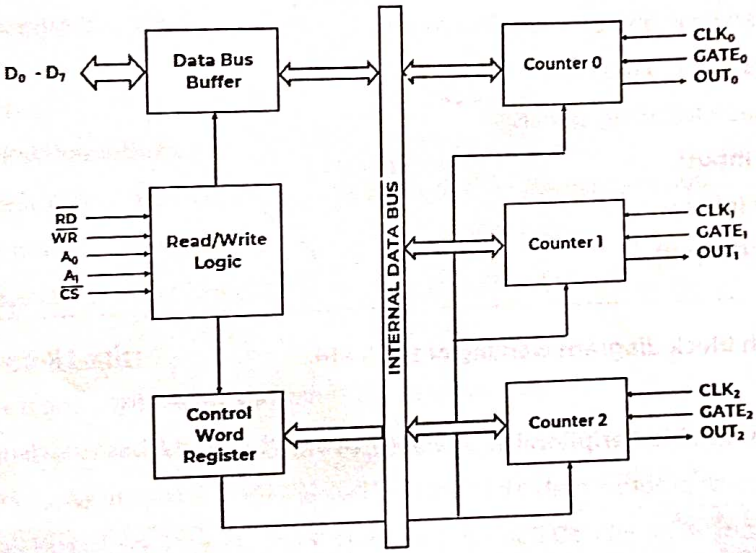


Figure 4.14: Block Diagram of 8253/8254 PIT.

It contains the following blocks:

I) Data Bus Buffer:

1. It is tristate, 8-bit bidirectional data bus buffer.
2. It is used to interface the internal data bus of 8253/8254 with the system data bus.
3. It is internally connected to Internal Data Bus & its outer pins $D_0 - D_7$ are connected to system data bus.
4. The direction of data buffer is decided by read and write control signals.

II) Read/Write Logic:

1. It accepts the read & write signals, which are used to control the flow of data through data bus.
2. It also accepts the $A_1 - A_0$ Address lines which are used to select one of the counters or the control word as shown below:

A_0	A_1	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word

3. It also accepts the CS signal to select the 8254 chip.

III) Control Word Register:

1. Control Word Register is 8 bit register that holds the Control Word.
2. It is selected when $A_1 - A_0$ contain 11.
3. It is used to specify the BCD or Binary Counter to be used, its mode of operation and the data transfer to be used.
4. The data can only be written into control word register, no read operation is allowed.

IV) Counters:

1. There are 3 independent, 16 bit down counters.
2. Each counter can operate have a binary or BCD count.
3. Each counter can be in one of six possible modes.
4. Each counter can have a max count of $2^{16} = 65535$.
5. Each counter has a following signals:
   a. **CLK (Clock Input).**
   b. **GATE (Gate Input).**
   c. **OUT (Clock Output).**

Q5. Explain with block diagram working of 8255 PPI. [P | High

Ans:

1. 8255 is a **Programmable Peripheral Interface** i.e. PPI 8255.
2. It is general purpose programmable parallel I/O Device.
3. It has three 8-bit bidirectional I/O Ports: **Port A, Port B and Port C.**
4. These I/O ports can be programmed in different modes.

FEATURES:

1. 8255 PPI contains 24 programmable I/O pins arranged as 2 8-bit ports & 2 4-bit ports.
2. 8255 PPI contains 3 ports and they are arranged in two groups of 12 pins.
3. It is fully compatible with Intel Microprocessor families.
4. It is also TTL compatible.
5. It has improved DC driving capability.
6. 8255 can operates in 3 modes:
   a. **Mode 0:** Simple I/O.
   b. **Mode 1:** Strobed I/O.
   c. **Mode 2:** Strobed bidirectional I/O.

BLOCK DIAGRAM:

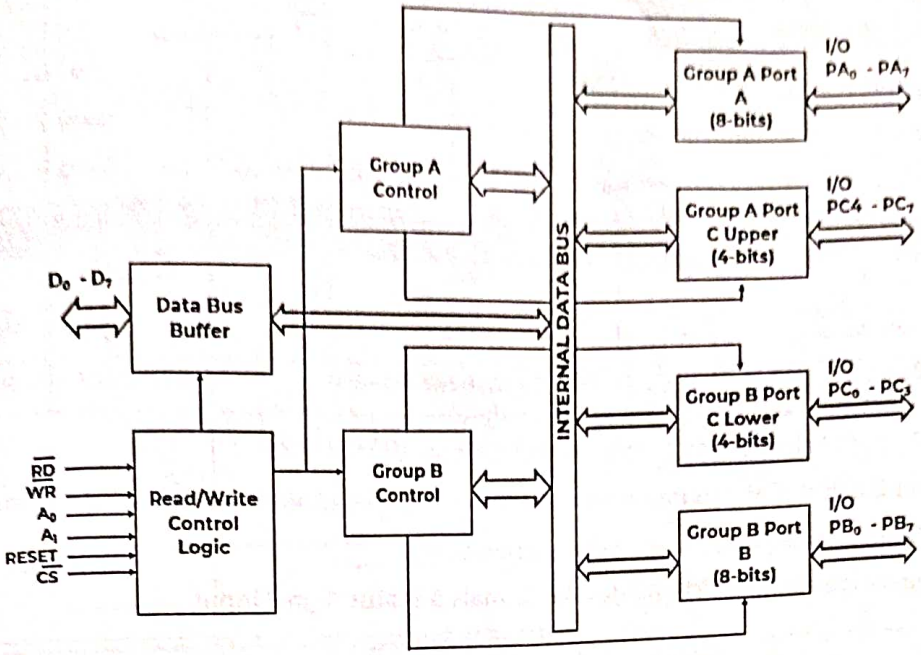


Figure 4.15: 8255 PPI Block Diagram.

It contains the following blocks:

I) Data Bus Buffer:

1. This is **8-bit bidirectional buffer**.
2. It is used to interface the internal data bus of 8255 with the external data bus.
3. When read is activated, it transmits data to the system data bus.
4. When write is activated, it receives data from the system data bus.

II) Read/Write Control Logic:

1. It accepts address and control signals from the microprocessor.
2. The control signals are read & write while address signals used are A_0 & A_1 and $\overline{CS}$.
3. The address bits (A_1, A_0) are used to select the ports or the control word register as shown:

For 8255		For 8086		Selection	Sample Address
A_1	A_0	A_2	A_1		
0	0	0	0	Port A	80 H (i.e. 1000 0000)
0	1	0	1	Port B	82 H (i.e. 1000 0010)
1	0	1	0	Port C	84 H (i.e. 1000 0100)
1	1	1	1	Control Word	86 H (i.e. 1000 1000)

4. $\overline{CS}$ is connected to address chip select decoder.
5. The 8255 operation/selection is enabled/disabled by CS signal.

III) Group A & Group B Control:

1. 8255 I/O ports are divided into 2 sections: **Group A & Group B**
2. Group A control block controls Port A & Port C upper i.e. $PC_7 - PC_4$.
3. Group B control block controls Port B & Port C lower i.e. $PC_3 - PC_0$.
4. Each group is programmed through software.

5. Group A & Group B controls accepts the control signals from the control word and forwards them to the respective ports.

IV) Port A, Port B & Port C:

1. This are 8-bit bidirectional ports.
2. They can be programmed to work in the various modes as follows:

Port	Mode 0	Mode 1	Mode 2
Port A	Yes	Yes	Yes
Port B	Yes	Yes	No (Mode 0 or Mode 1)
Port C	Yes	No (Handshake signals)	No (Handshake signals)

3. Group A control block controls Port A & Port C upper i.e. $PC_7 - PC_4$.
4. Only Port C can also be programmed to work in Bit Set/Reset Mode to manipulate its individual bits.
5. Port C function is dependent on mode of operation.
6. It can be used as **Simple I/O, Handshake Signals & Status Signal Input**.

Q6. Explain different data transfer modes of 8237 DMA Controller.

Ans: **[P | Medium]**

DMA:

1. Usually Data Transfer takes place between Microprocessor & Peripheral Device by using a program stored in a memory.
2. It is known as **Microprocessor Controlled Data Transfer**.
3. This method is used when the speed of Peripheral is less than or equal to the speed of Microprocessor.
4. If the speed of Peripheral is greater than the speed of microprocessor then microprocessor is disconnected from the system and **DMA Controller** is used to transfer the data between peripheral devices.
5. DMA Stands for **Direct Memory Access**.
6. Now it is known as **Device Controlled Data Transfer**.
7. This method does not requires software and so data transfer takes place at high speed.
8. **For example:** Data transfer between system memory and floppy disk.

INTERFACING DMA CONTROLLER 8237 WITH 8086:

1. Figure 4.16 shows Interfacing of 8237 with 8086.
2. Address generated by 8237 is only 16 bit.
3. $A_0 - A_3$ & $A_4 - A_7$ is directly connected.
4. While $A_8 - A_{15}$ is de-multiplexed from $DB_0 - DB_7$.
5. The upper 4 bit address i.e. $A_{16} - A_{19}$ is provided through a latch.
6. This latch is to be written on by the programmers.
7. When 8237 issues the address, corresponding latches are enabled and similarly for 8086 issuing the address.
8. This is controlled by AEN Pin of 8237.

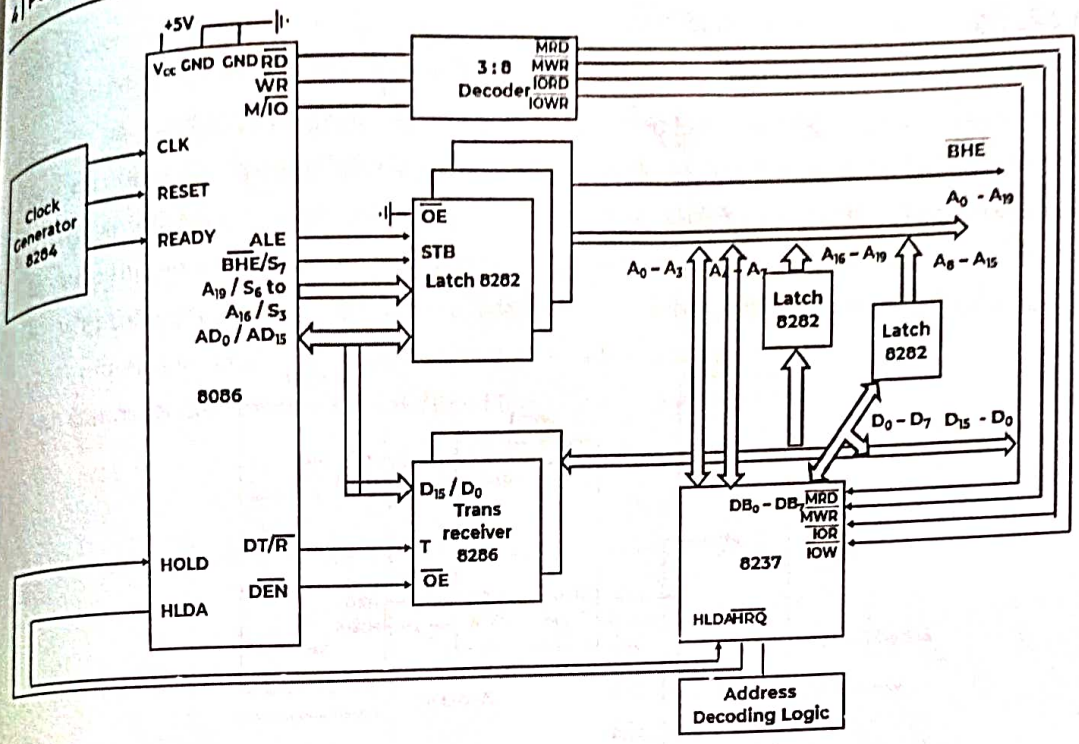


Figure 4.16: Interfacing of 8237 with 8086.

DATA TRANSFER MODES OF 8237:

I) Single Transfer Mode:

1. In this mode, 8237 is programmed to make only one data transfer.
2. After one data transfer, it decrements the word count register & increments or decrements the address register.
3. After transferring one byte the 8237 disables HRQ and enters into idle state or slave mode.
4. It is also called as **Cycle Steal Mode**.
5. Then buses are returned to microprocessor.
6. If we want other data transfer then DREQ has to go high again.

II) Block Transfer Mode:

1. In this mode, all the bytes are transferred continuously.
2. After each transfer, it decrements the word count register & increments or decrements the address register.
3. It maintains HRQ high during all DMA Cycles.
4. The action continuous until TC is reached or $\overline{EOP}$ is activated.
5. In this DREQ has to be kept high only at the beginning.

III) Demand Transfer Mode:

1. It is very similar to block transfer, except that the DREQ must active throughout the DMA operation.
2. In this mode, the number of bytes to be transferred is controlled by I/O device.
3. If during the operation DREQ gets low, the DMA operation is stopped and the busses are returned to the microprocessor.
4. In meantime, the microprocessor can continue with its own operations.
5. Once DREQ goes high again, the DMA operation continues from where it had stopped.

IV) Cascade Mode:

1. This mode more than one DMACs are cascaded together.
2. It is used to increase the number of devices interfaced to the microprocessor.
3. Here we have one master DMAC, to which one or more slave DMACs are connected.
4. The slave gives HRQ to the master on the DREQ of the master, and the master gives HRQ to the microprocessor on the HOLD of the microprocessor.
5. Figure 4.17 shows the cascaded 8237's.

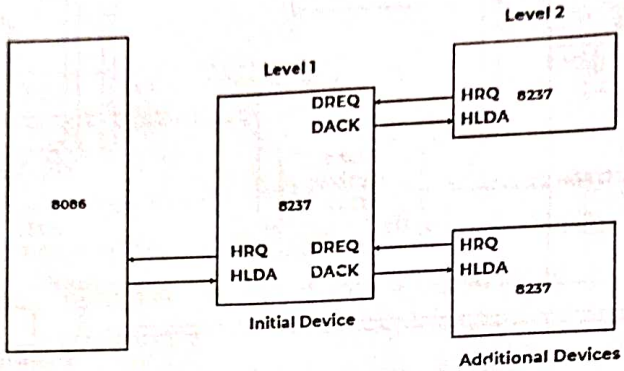


Figure 4.17: Cascaded 8237's.

Q7. Explain the need of DRAM controller for interfacing DRAM with 8086. Draw and explain interfacing of DRAM controller with 8086

[P | Medium]

Ans:

NEED OF DRAM CONTROLLER:

1. DRAM Stands for **Dynamic RAM**.
2. **IC 8203** is used as DRAM Controller.
3. The basic DRAM cell uses a capacitor to store the charge as a representation of data.
4. This capacitor is manufactured as a diode that is reverse-biased so that the storage capacitance comes into the picture.
5. To avoid this possible data loss, the data stored in a dynamic RAM cell must be refreshed after a fixed time interval regularly.
6. The process of refreshing the data in the RAM is known as refresh cycle.
7. During this refresh period all other operations related to the memory subsystem are suspended.
8. The advantages of dynamic RAM are:
   a. Low power consumption.
   b. Higher packaging density.
   c. Low cost.

INTERFACING OF DRAM CONTROLLER WITH 8086:

1. In 8203 DRAM, X_1 and X_0 are clock pins.
2. They are provided by 8284.
3. 16K / 64 K pin is used as mode select pin.
4. The pin is used to select mode of operation i.e. 16 KB Mode or 64 KB Mode.

5. The lower order address lines AL_0 – AL_6 are available in 16K Mode while AL_0 – AL_7 in 64K Mode.
6. The higher order address lines AH_0 – AH_6 are available in 16K Mode while AH_0 – AH_7 in 64K Mode.
7. AL_0 – AL_6/AL_7 and AH_0 – AH_6/AH_7 are interfaced with 8086.
8. Control signals RD, WR and WE are used in DRAM 8203.
   a. **RD:** This signal is given from 8086 to indicate the processor wants to read.
   b. **WR:** This signal is also given by 8086 to indicate it wants to write.
   c. **WE:** This signal is given from 8203 to indicate the processor wants to write to DRAM.
9. OUT Address pins OUT_0 – OUT_6/OUT_7 carry the row address or column address.
10. CAS stands for Column Address Select and RAS stands for Row Address Select.

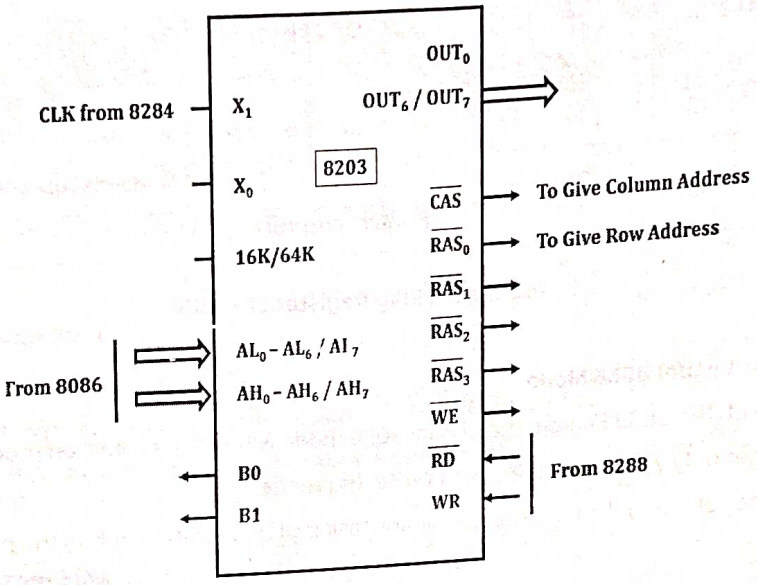


Figure 4.18: Interfacing of DRAM controller with 8086.

CHAP - 5: INTEL 80386DX PROCESSOR

Q1. Explain flag register of 80386 microprocessor

Q2. Explain VM, RF, IOPL and NT flags of 80386 microprocessor **[5M | Dec18 & May19] [P | High]**

Ans:

FLAG REGISTER:

1. The flags register is a 32-bit register named EFLAGS.
2. The flags control certain operations and indicate the status of the microprocessor.
3. Figure 5.1 shows the Flag Register for 80386.

Reserved by Intel	VM	RF	0	NT	IOPL	OF	DF	IF	TF	SF	ZF	0	AC	0	PF	1	CF
31 – 18	17	16	15	14	13 -12	11	10	9	8	7	6	5	4	3	2	1	0
										8085 Microprocessor Flags							
						8086 Microprocessor Flags											

Figure 5.1: Flag Register of 80386.

I) VM:

1. VM stands for **Virtual 8086 Mode**.
2. If this flag is set, the 80386 enters the virtual 8086 mode within the protection mode.
3. This is to be set only when the 80386 is in protected mode.
4. This bit can be set using IRET instruction or any task switch operation only in the protected mode.

II) RF:

1. RF stands for **Resume Flag**.
2. This flag is used with the debug register breakpoints.
3. It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle.
4. The RF is automatically reset after successful execution of every instruction.

III) NT:

1. NT stands for **Nested Task**.
2. When NT = 1, it indicates that the currently executing task is nested within another task and it has valid link to caller task.
3. The processor uses the nested task flag to control chaining of interrupted and called tasks.

IV) IOPL:

1. IOPL stands for **I/O Privilege Level**.
2. The IOPL field in the FLAGS register defines the right to use I/O-related instructions.

V) OF:

1. OF Stands for **Overflow Flag**.
2. It is used to indicate an overflow during signed operation.

vii) DF:

1. DF Stands for **Direction Flag**.
2. It is used to give the direction during string instruction.
3. If DF = 1, then decrement. If DF = 0 then auto increment.

viii) IF:

1. IF Stands for **Interrupt Enable Flag**.
2. It is used to enable the hardware interrupt INTR.
3. When IF = 1, INTR is enabled, else it is disabled.

ix) **TF:**

1. TF stands for **Trap Flag**.
2. Setting TF puts the processor into single-step mode for debugging.
3. In this mode, the CPU automatically generates an exception after each instruction, allowing a program to be inspected as it executes each instruction.
4. Single-stepping is just one of several debugging features of the 80386.

x) **SF:**

1. SF Stands for **Sign Flag**.
2. It is used to indicate MSB of the result.
3. If SF = 1 then MSB of the result is 1, the number is treated as negative.
4. If SF = 0 then MSB of the result is 1, the number is treated as positive.

xi) **ZF:**

1. ZF Stands for **Zero Flag**.
2. It is used to indicate if the result has become Zero.
3. If ZF = 1 then result is zero, else result is Non-Zero.

xii) **AC:**

1. AC Stands for **Auxiliary Carry Flag**.
2. It indicates a carry from the lower nibble to the higher nibble.

xiii) **PF:**

1. PF Stands for **Parity Flag**.
2. It indicates whether the result has even or odd parity.
3. If PF = 1 then result has even parity, else result has odd parity.

xiv) **CF:**

1. CF Stands for **Carry Flag**.
2. It indicates if a carry was produced beyond the MSB of the result.
3. If CF = 1 then carry produced beyond the MSB of the result.
4. If CF = 0 then carry is not produced beyond the MSB of the result.

Q3. Explain the modes of operation of 80386 microprocessor **[10M | Dec18] [P | High]**

Ans:

PROCESSING MODES OF 80386:

1. The processing modes of the 80386 determine the features that are accessible.
2. There are three processing modes:
    a. Real Address Mode.
    b. Protected Mode.
    c. Virtual 8086 Mode.

I) Real Address Mode:

1. It is also known as **Real Mode**.
2. It is the mode of processor immediately after RESET.
3. In the real mode, 80386 works as a fast 8086 with 32-bit registers and data types.
4. In real mode, the default operand size is 16 bit but 32- bit operands and addressing modes may be used with the help of override prefixes.
5. The segment size in real mode is 64k, hence the 32-bit effective addressing must be less than 0000FFFFFH.
6. The real mode initializes the 80386 and prepares it for protected mode.

Memory Addressing in Real Mode:

1. In the real mode, the 80386 can address at the most 1Mbytes of physical memory using address lines A0-A19.
2. Paging unit is disabled in real addressing mode, and hence the real addresses are the same as the physical addresses.
3. To form a physical memory address, appropriate segment registers contents (16-bits) are shifted left by four positions and then added to the 16-bit offset address formed using one of the addressing modes, in the same way as in the 80386 real address mode.
4. The segment in 80386 real mode can be read, write or executed, i.e. no protection is available.
5. Any fetch or access past the end of the segment limit generate exception 13 in real address mode.
6. The segments in 80386 real mode may be overlapped or non-overlapped.
7. The interrupt vector table of 80386 has been allocated 1Kbyte space starting from 00000H to 003FFH.
8. Figure 5.2 shows Real Mode Addressing.

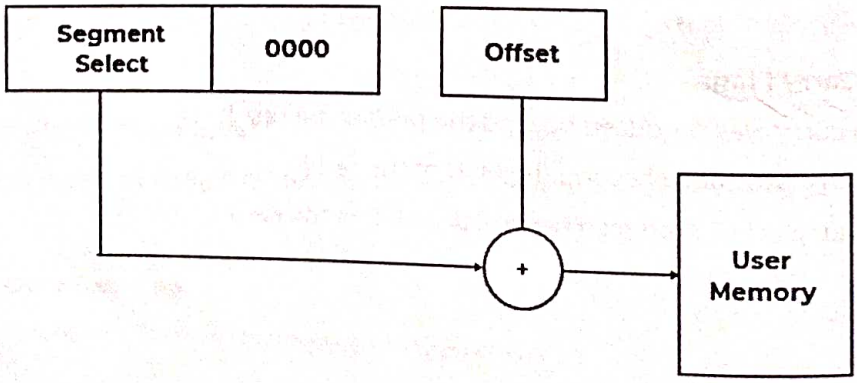


Figure 5.2: Real Mode Addressing.

II) Protected Mode:

1. All the capabilities of 80386 are available for utilization in its protected mode of operation.
2. The 80386 in protected mode support all the software written for 80286 and 8086 to be executed under the control of memory management and protection abilities of 80386.
3. The protected mode allows the use of additional instruction, addressing modes and capabilities of 80386.

Memory Addressing in Protected Mode:

1. In this mode, the contents of segment registers are used as selectors to address descriptors which contain the segment limit, base address and access rights byte of the segment.
2. The effective address (offset) is added with segment base address to calculate linear address.
3. This linear address is further used as physical address, if the paging unit is disabled, otherwise the paging unit converts the linear address into physical address.
4. The paging unit is a memory management unit enabled only in protected mode.
5. The paging mechanism allows handling of large segments of memory in terms of pages of 4Kbyte size.
6. The paging unit operates under the control of segmentation unit.
7. The paging unit if enabled converts linear addresses into physical address, in protected mode.
8. Figure 5.3 shows Protected Mode Addressing.

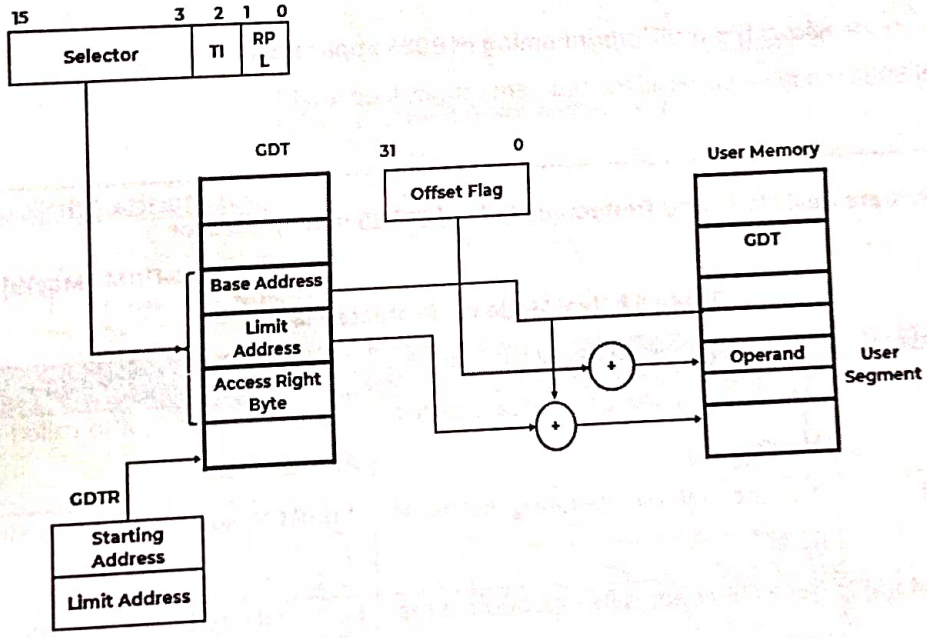


Figure 5.3: Protected Mode Addressing.

GDT: Global Descriptor Table.

GDTR: Global Descriptor Table Register.

TI: Table Identity.

RPL: Requested Privilege Level.

III) Virtual 8086 Mode:

1. In protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.

2. The real mode can also use to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions.
3. Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.
4. Thus, the virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode.
5. V86 Mode is also known as **Virtual Mode of 80386**.
6. V86 Mode is a **Dynamic Mode**.
7. It can switch repeatedly & rapidly between V86 Mode & Protected Mode.
8. To execute an 8086 program, the CPU enters in V86 Mode from Protected Mode.
9. CPU Leaves V86 Mode and enters protected mode to continue executing a native 80386 program.

Addressing in Virtual Mode:

1. The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode.
2. In virtual mode, 8086 can address 1MB of physical memory that may be anywhere in the 4GB address space of the protected mode of 80386.
3. Like 80386 real mode, the addresses in virtual 8086 mode lie within 1MB of memory.
4. In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers.
5. The virtual mode **allows the multiprogramming of 8086 applications.**
6. The virtual 8086 mode executes all the programs at privilege level 3.

Q4. Differentiate Real Mode and Protected Mode of 80386 microprocessor

[10M | May19] [P | High]

Ans:

Table 5.1: Real Mode v/s Protected Mode

Parameter	Real Mode	Protected Mode
General	Real Mode is also called **Real Address Mode.**	Protected Mode is also called **Protected Address Mode.**
Default Mode	It is the default operating mode on Reset.	It is not default operating mode on reset.
Entering the Mode	The 80386 begins its execution in real mode on power up or reset.	For protected mode operation the Protection Enable (PE) bit of Control Register 0 must be set.
Leaving the Mode	To leave the Real Mode & Enter Protected Mode the PE bit of Control Register 0 must be set.	To leave the Protected Mode the User can clear the PE bit in Control Register 0.
Use	It is use to initialize 80386 for Protected Mode operation.	It is use for Segmentation, Paging, Multi-tasking, Virtual Memory and Protection.

Access	In the Real Mode 80386 can access all the registers.	Protected Mode can access all general purpose registers, control registers, debug registers, test registers, segment selectors and segment descriptors.
Memory Addressing	It can address up to 1 MB of memory.	It can address up to 4 GB of memory with 32 bit addressing.
Segment Base Address	20 Bit.	32 bit, from descriptor for 32 bit Protected Mode.
Segment protection	No	Yes

REAL MODE ADDRESSING:

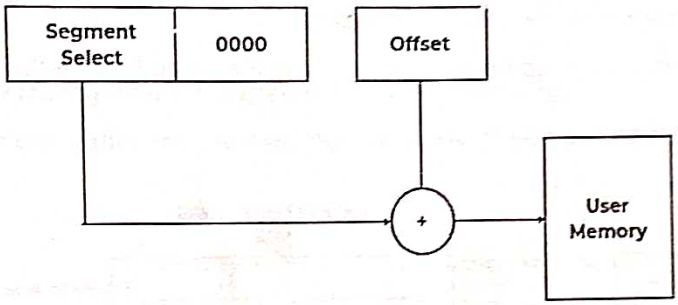


Figure 5.4: Real Mode Addressing.

PROTECTED MODE ADDRESSING:

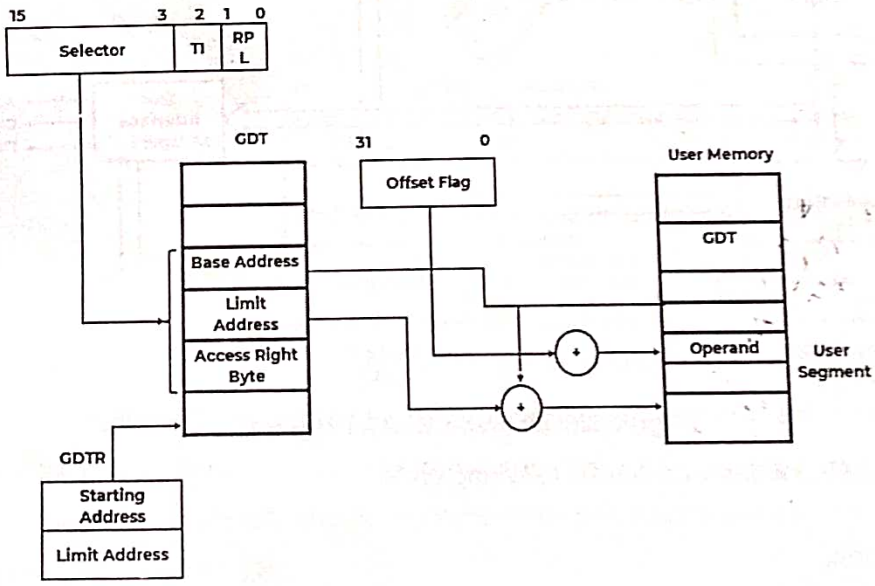


Figure 5.5: Protected Mode Addressing.

GDT: Global Descriptor Table.

GDTR: Global Descriptor Table Register.

TI: Table Identity.

RPL: Requested Privilege Level.

Access	In the Real Mode 80386 can access all the registers.	Protected Mode can access all general purpose registers, control registers, debug registers, test registers, segment selectors and segment descriptors.
Memory Addressing	It can address up to 1 MB of memory.	It can address up to 4 GB of memory with 32 bit addressing.
Segment Base Address	20 Bit.	32 bit, from descriptor for 32 bit Protected Mode.
Segment Protection	No	Yes

REAL MODE ADDRESSING:

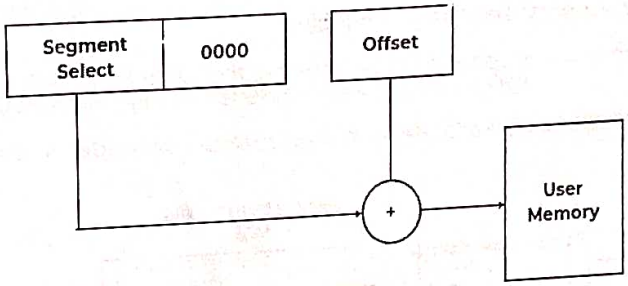


Figure 5.4: Real Mode Addressing.

PROTECTED MODE ADDRESSING:

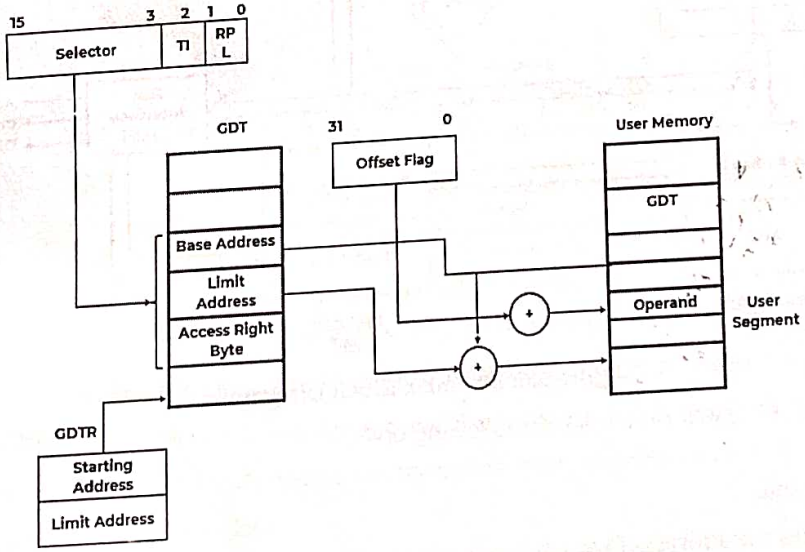


Figure 5.5: Protected Mode Addressing.

GDT: Global Descriptor Table.

GDTR: Global Descriptor Table Register.

TI: Table Identity.

RPL: Requested Privilege Level.

-- EXTRA QUESTIONS --

Q1. Draw & Explain block diagram of 80386 DX [P | Medium]

Ans:

FEATURES:

1. Microprocessor 80386 DX is a 32-bit processor.
2. It has 32-bit address bus and a 32-bit data bus.
3. It has 8 General Purpose 32 bit registers supports 8, 16, 32 bit data types.
4. 80386 DX can access up to 4 GB physical memory or 64 TB of virtual memory.
5. It runs with speed up to 20 MHz instructions per second.
6. It has 4 levels of protection. i.e. $PL_0 - PL_3$.
7. The 80386DX can operate in real mode, protected mode & virtual 8086 mode.
8. It has built-in virtual memory management circuitry and protection circuitry.
9. 80386 Microprocessor supports hardware debugging.
10. The 80386DX microprocessor is compatible with their earlier 8086, 8088, 80186, 80188, 80286 chips.

BLOCK DIAGRAM:

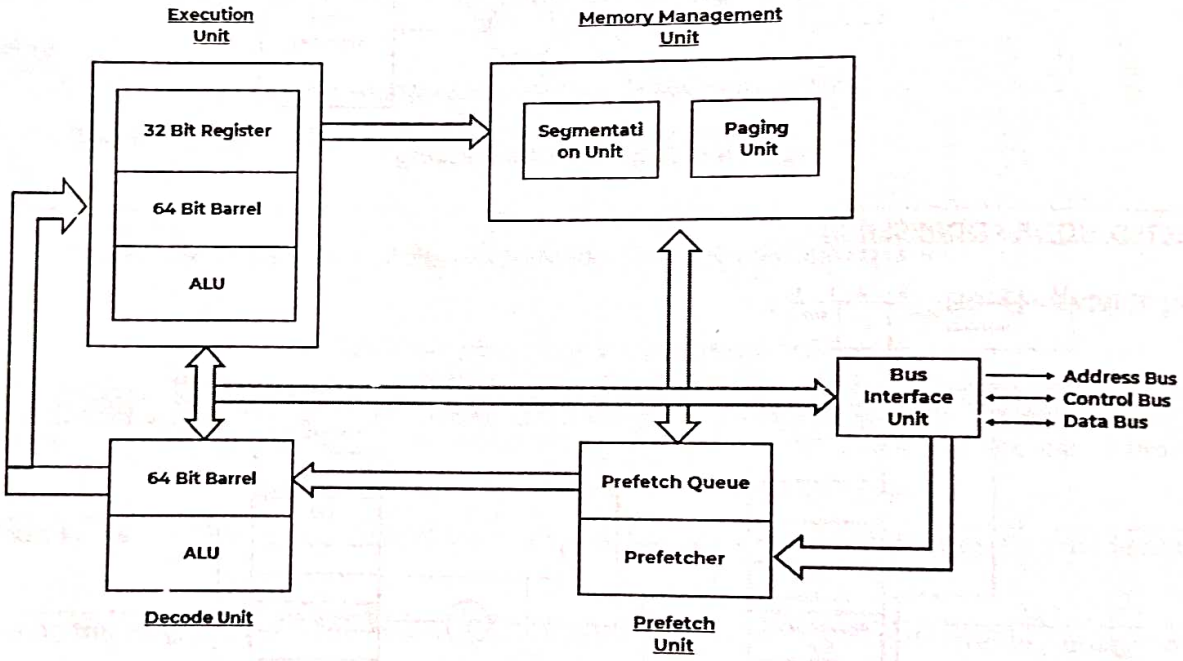


Figure 5.6: 80386DX Block Diagram.

The block diagram of 80386 DX includes the following units:

I) Bus Interface Unit:

1. This unit includes the address drivers, transceivers for data bus & bus control signals.
2. It handles the communication with devices external to the microprocessor chip.

II) . Prefetcher & Prefetch Queue:

1. The Prefetcher fetches the instructions from the external memory and stores them in the Prefetch queue to be executed further.
2. Prefetch queue is 16 byte in size.

III) Instruction Decoder and Decoded Instruction Queue:

1. The instruction decoder takes the instruction from the Prefetch queue and after decoding it, stores them in the decoded instruction queue.
2. The decoded instruction queue can store up to three decoded instructions.

IV) Execution Unit:

1. It consists of 8 genera! purpose 32 bits register, 64 bit barrel shifter & ALU.
2. The execution unit executes each instruction received from the decoded instruction queue.

V) Segmentation Unit:

1. This unit is responsible for segmentation mechanism.
2. Its support multitasking in protected mode.

VI) Paging Unit:

1. This unit converts the linear address to physical address.
2. It is the part of memory management unit.

Q2. Explain Memory Management in details in 80386DX Processor.

Q3. Explain address translation mechanism used in protocol mode of 80386.

Ans: **[P | High]**

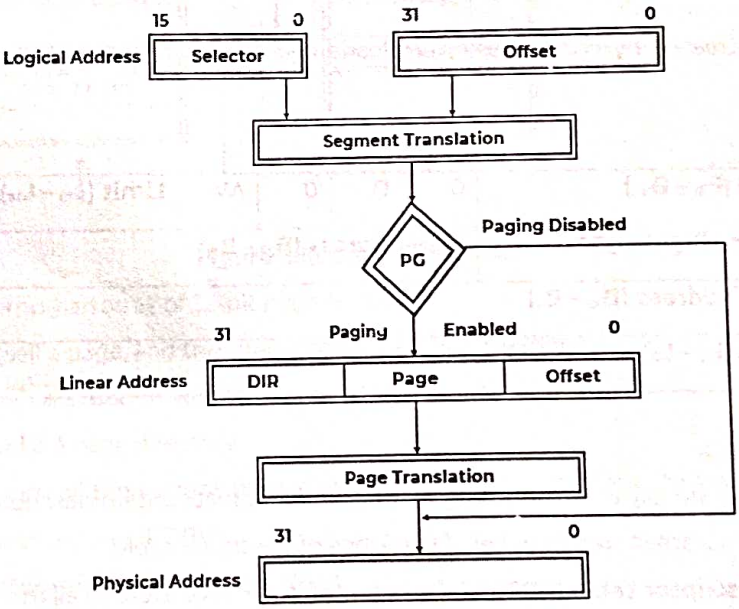


Figure 5.7: Logical to Physical Address Translation in 80386.

1. Memory management is the act of managing computer memory at the system level.
2. Memory Management dynamically allocate portions of memory to programs at their request, and free it for reuse when no longer needed.
3. In 80386 DX, the logical (virtual) address is converted to the linear address.
4. This is done with the help of segmentation mechanism.
5. While the linear address is converted to physical address with the help of paging mechanism.
6. Thus, memory management in 80386 DX is done by converting logical address to physical address with the help of segmentation & paging.
7. Figure 5.7 shows logical to physical address translation in 80386.

SEGMENT TRANSLATION:

1. Segment Translation is used to **convert Logical Address to Linear Address.**
2. Figure 5.8 shows the segment translation.

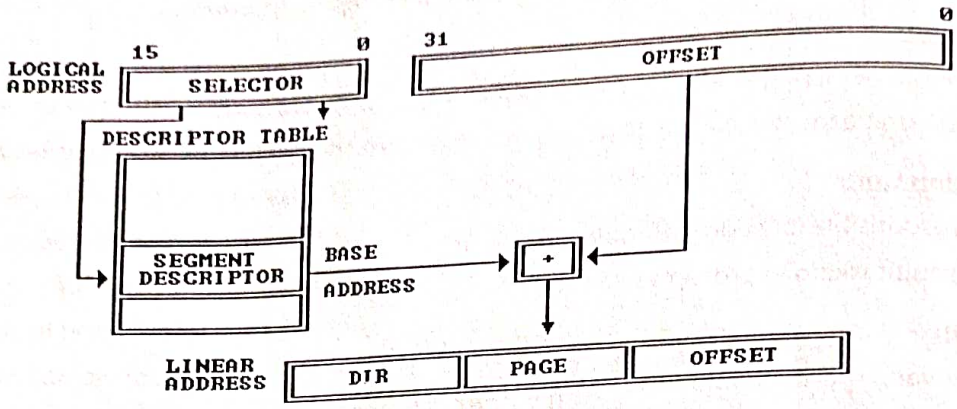


Figure 5.8 Segment Translation.

3. To perform this translation, the processor uses the following data structures:

DESCRIPTORS:

1. The segment descriptor is used to provide the data which is required to map a logical address into linear address.
2. Descriptors are created by compilers, linkers, loaders, or the operating system, not by application programmers.

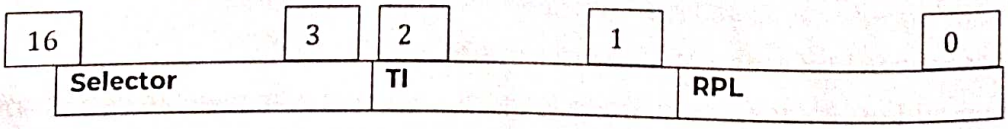
Byte No.							Byte No.
7	Base (B_{31} – B_{24})	G	D	0	AV	Limit (L_{19} – L_{16})	6
5	Access Right Byte	Base Address (B_{23} – B_{16})					4
3	Base Address (B_{15} – B_0)						2
1	Limit (L_{15} – L_0)						0

DESCRIPTOR TABLE:

1. A descriptor table is simply a memory array of 8-byte entries that contain descriptors.
2. Segment descriptors are stored in either of two kinds of descriptor table:
   a. **The global descriptor table (GDT):** GDT is common table accessible to all the tasks.
   b. **A local descriptor table (LDT):** LDT is separate for each task.

SELECTOR:

1. Selector is used to select one of the 8192 descriptors from one of the tables Viz. GDT & LDT.
2. Structure of selector:



3. When TI = 0 then GTD is used, when TI = 1 then LDT is used.

SEGMENT REGISTER:

1. 80386 stores information from descriptors in segment registers.
2. Every segment register has a "visible" portion and an "invisible" portion.
3. The visible portions of segment address registers are manipulated by programs as if they were simply 16-bit registers.
4. The invisible portions are manipulated by the processor.

PAGE TRANSLATION:

1. Page Translation is used to **convert Linear Address to Physical Address**.
2. The page-translation step is optional.
3. Page translation is in effect only when the PG bit of CR0 is set.
4. This bit is typically set by the operating system during software initialization.
5. The PG bit must be set if the operating system is to implement multiple virtual 8086 tasks, page-oriented protection, or page-oriented virtual memory.
6. Figure 5.10 shows page translation.

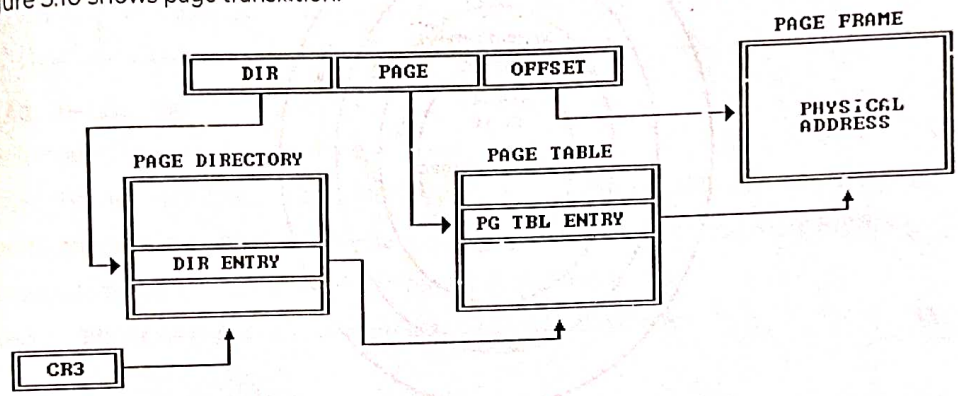


Figure 5.10: Page Translation.

7. A page table is simply an array of 32-bit page specifiers.
8. A page table is itself a page, and therefore contains 4 KB of memory.
9. Two levels of tables are used to address a page of memory.
10. At the higher level is a page directory.
11. The physical address of the current page directory is stored in the CPU register CR3, also called the page directory base register (PDBR).
12. Page Table/Directory Entry is shown as follows:

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Frame Table Address	OS Reserved	0	0	D	A	PCT	PWT	U/S	R/W	P

13. D stands for Dirty, A stands for Access, P for Present & R/W for Read & write.
14. PCD stands for Page Cache Disabled & PWT Stands for Page Write Through.
15. U/S bit differentiates between User & Supervisor Privileges.

Q4. Explain in brief Protection Mechanism in 80386DX Processor. [P | Medium]

Ans:

PROTECTION MECHANISM:

1. Protection Mechanism in 80386 is **used to detect & identify the bugs.**
2. Protection Mechanism operates in Protection Mode.
3. It is meant for **Multi-tasking operations.**
4. Most processor have only 2 protection levels i.e. User & Supervisor.
5. But x86 architecture features 4 levels of protection, called Privilege Levels (PL).
6. They are designed to support the needs of multi-masking OS to isolate and protect user programs from each other.
7. Figure 5.11 shows Privilege levels of the tasks in x86 Architecture.

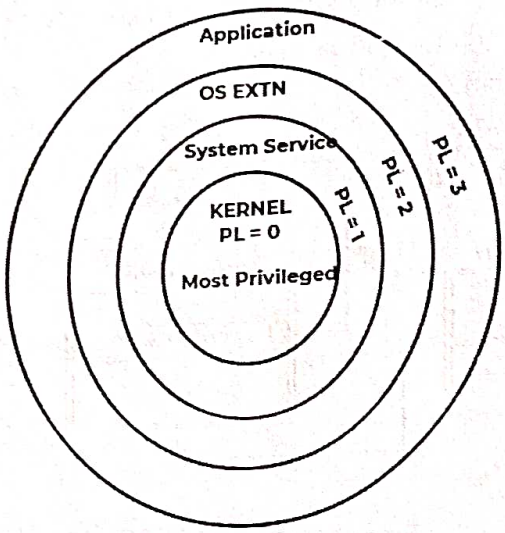


Figure 5.11: Privilege levels of the tasks in x86 Architecture.

8. The privilege levels (PL) are numbered 0, 1, 2, and 3.
9. Level 0 is the most privileged level.
10. Level 3 is the least privileged.
11. The privilege levels control the use of privileged instructions, I/O instructions, access to segments & segment descriptors.
12. The x86 architecture offers an additional type of protection on a page basis, when paging is enabled.
    a. RPL → Requested PL.
    b. DPL → Descriptor PL.
    c. CPL → Current PL.
    d. EPL → Effective PL.
13. The x86 architecture control access to both data & code between levels of task, according to the following rules of privilege:
    a. Data stored in a segment with PL = a can be accessed only by code executing at PL, numerically, at least as privilege as p.
    b. A code segment with PL = p can be called only by a task executing at, numerically, the same or lower PL than p.
    c. A stack segment with PL = p can be used only by a task executing at the same PL.

Q5. **Draw a segment descriptor format & explain different fields.**

Q6. **Explain data segment descriptor with neat diagram.**

[P | Medium]

Ans:

1. The segment descriptor is used to provide the data which is required to map a logical address into a linear address.
2. Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers.
3. Descriptor is used to give the different details of the segment.
4. The structure of Segment Descriptors is shown below:

Byte No.							Byte No.
7	Base ($B_{31} - B_{24}$)	G	D	0	AV	Limit ($L_{19} - L_{16}$)	6
5	Access Right Byte	Base Address ($B_{23} - B_{16}$)					4
3	Base Address ($B_{15} - B_0$)						2
1	Limit ($L_{15} - L_0$)						0

Limit Address ($L_0 - L_9$):

1. Limit Address defines the size of the segment.
2. When the processor concatenates the two parts of the limit field, a 20-bit value results.
3. Limit Address when added with the base address gives the last address of the segment.
4. In case of 286 there is a 24 bit base address and 16 bit limit.
5. While in 386 we have 32 bits base address and 20 bit limit address.

Base Address ($B_0 - B_{31}$):

1. Base Address indicates the starting address of the memory segment.
2. It defines the location of the segment within the 4 gigabyte linear address space.
3. The processor concatenates the three fragments of the base address to form a single 32-bit value.

Granularity Bit (G):

1. Granularity Bit specifies the units with which the LIMIT field is interpreted.
2. When G = 0, 20 bit Limit specifies the segment size from 1 byte to 1 MB.
3. When G = 1, 20 bit Limit is to be multiplied by 4K hence segment size varies from 4 KB to 4 GB.

Availability (AV):

1. It indicates the availability of segment.
2. When AV = 1, it indicates the corresponding segment is not used by any other task and hence is available.
3. When AV = 0, it indicates the corresponding segment is used by some other task and hence is not available.

Data Size (D):

1. It indicates the Data Size.
2. When D = 0, it indicates 16 bit OS instructions.

3. When D = 1, it indicates 32 bit OS instructions.

Access Right Byte (ARB):

7	6 5	4	3	2	1	0
P	DPL	S	E	ED/C	RW	A

a. Present (P):

1. When P = 1, the entry of the descriptor is initialized.
2. When P = 0, the entry of the descriptor is not initialized.

b. Descriptor Privilege Level (DPL):

1. Segment Privilege Level, used in Privilege tests.

c. Segment Descriptor (S):

1. When S = 1, Code or Data Segment Descriptor.
2. When S = 0, System Segment Descriptor or Gate Descriptor.

d. Executable (E):

1. When E = 0, Descriptor type is data (or stack) segment.
2. When E = 1, Descriptor type is code segment.

e. Expansion Direction (ED):

1. When ED = 0, Expand up segment.
2. When ED = 1, Expand down Segment.

f. Writeable (W):

1. When W = 0, Data Segment cannot be written into, i.e. Read only.
2. When W = 1, Data Segment may be written into.

g. Readable (R):

1. When R = 0, Code Segment cannot be read.
2. When R = 1, Code Segment may be read.

h. Accessed (A):

1. When A = 0, Segment has not been accessed.
2. When A = 1, Segment has been accessed earlier.

Q7. Control register of 80386DX

Ans:

[P | Medium]

CONTROL REGISTER OF 80386DX:

1. There are 4 control registers in 80386DX (CR_0, CR_1, CR_2 and CR_3).
2. They are **32 bit control registers**.
3. Register CR_1 is reserved for future use.
4. Control registers CR_0, CR_2 and CR_3 are used to hold global machine status.
5. The road and store instructions are available to access these control registers.
6. Figure 5.12 shows the control registers of 80386DX.

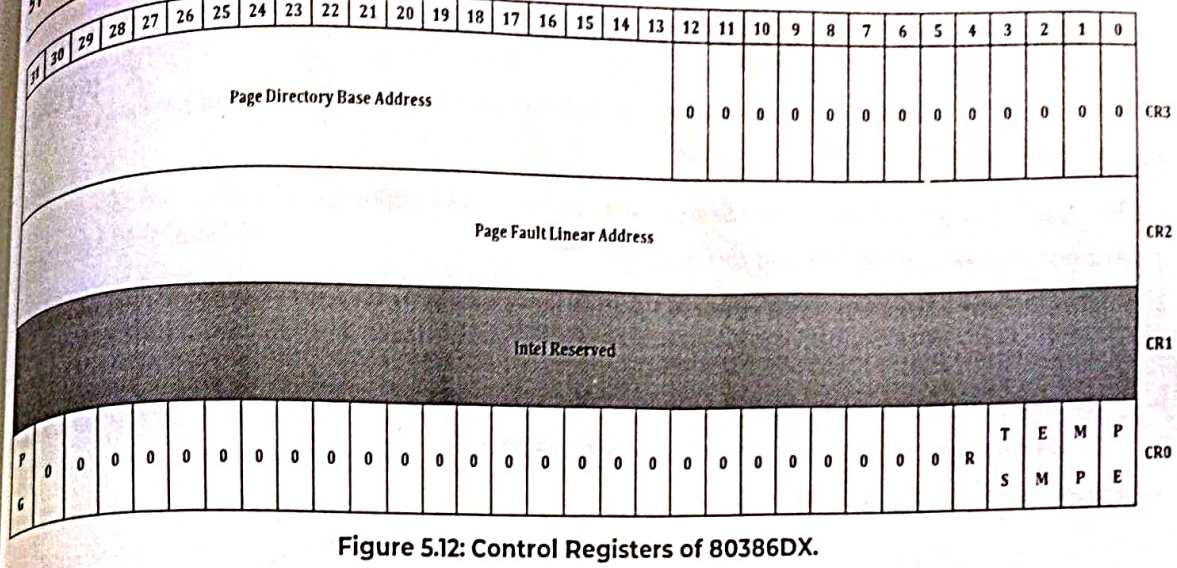


Figure 5.12: Control Registers of 80386DX.

CR_0:

1. The lower 16-bits of CR_0 are also called as **Machine Status Word (MSW)**.
2. It contains system control flags, which control modes of operation or indicates state of processor.
3. It contains:
   a. **Paging Enable (PE):** When PG = 1, paging is enabled. When PG = 0, paging is disabled.
   b. **Reserved (R):** It is reserved by Intel.
   c. **Task Switched (TS):** The processor sets this bit whenever a task switch operation is performed.
   d. **Emulation (EM):** The 2nd bit in CR_0 is set when coprocessor software is present & when it is absent then this flag is set too.
   e. **Monitor coprocessor (MP):** The 1st bit in CR_0 is set when 80387 coprocessor is present & when it is reset i.e. 0 then coprocessor is absent.
   f. **Protection Enable (PE):** When the processor enters into the protected mode it makes PE = 1, and the protection mechanism is enabled. When PE = 0, the processor operates in real mode.

CR_1:

1. It is a 32 bit control register used for Intel higher processor.
2. It is reserved for future use.

CR_2:

1. It is a 32 bit register which stores the linear address of last page fault.
2. The error codes are pushed into the stack of the page cache.

CR_3:

1. This is a 32 bit control register which stores page directory base address.
2. Only upper 20 bits are utilized for generation of page directory base address, because lower 12 bits will be common, because of 4KB page size.

Q8. What is GDT? Explain structure of GDT

Ans: [P | Low]

GDT:

1. Two types of descriptor tables are used by the processor when working in protected-mode.
2. The first is known as the GDT and second one is LDT.
3. GDT Stands for Global Descriptor Table.
4. GDT is used mainly for holding descriptor entries of operating system segments.
5. Local Descriptor Table (LDT) contains entries of normal application segments
6. GDT is a data structure used by Intel x86 family processors.
7. Figure 5.13 shows the structure of GDT.

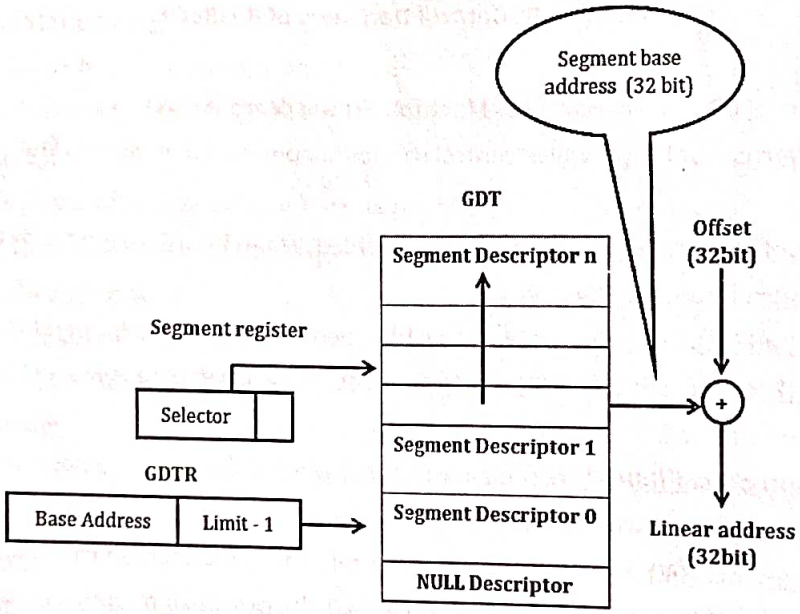


Figure 5.13: GDT Structure.

8. The structure resides in memory, consists of multiple 8-byte descriptors, and is pointed to by the GDT register (GDTR).
9. GDT provides a mechanism for defining the characteristics of the 80386's global memory address space.
10. Global memory is a general system resource that is shared by many or all software tasks.
11. GDT is used to maintain a list of most segment.
12. It is usually used by all programmers to refer to the segment of memory.
13. 80386 processor in protected mode can have many LDT's but only one GDT.
14. GDT entries can be segment descriptors, call gates, task state segments, or LDT descriptors.
15. The GDT table contains a number of entries called **Segment Descriptors**.
16. Each is 8 bytes long and contains information on the starting point of the segment, the length of the segment, and the access rights of the segment

CHAP - 6: PENTIUM PROCESSOR

Explain an instruction issue algorithm of Pentium processor

[5M | Dec18 & May19] [P | High]

ns:

ENTIUM PROCESSOR:

Pentium Processor is produced by Intel since 1993.

It is the brand used for a series of P5 x86 compatible microprocessor.

LGORITHM:

Decode the two consecutive instructions I_1 and I_2.

If all the following conditions are true, the two instructions are pairable and issue I_1 to 'U' Pipeline and I_2 to 'V' Pipeline.

Else they are not pairable, and only the instruction I_1 is to be given to 'U' Pipeline.

The Conditions are:

a. I_1 and I_2 are simple instructions.
b. I_1 is not a jump instruction.
c. Destination of I_1 is not a source of I_2.
d. Destination of I_1 is not a destination of I_2.

Explain the branch prediction logic used in Pentium Processor

Explain how the flushing of pipeline problem is minimized in Pentium Arch

[10M | Dec18] [P | High]

Ans:

BRANCH PREDICTION LOGIC:

Program Transfer Instructions such as JMP, CALL, RET and Conditional Jumps reduces the performance gain through pipelining.

2. This is because they change the sequence of all the instructions that entered the pipeline after Program Transfer Instruction, thus assuming the previous instructions invalid.
3. Suppose instruction I_3 is a conditional jump to I_{50} at some other address (target address), then the instructions that entered after I_3 is invalid and new sequence beginning with I_{50} need to be loaded in.
4. This causes bubbles in pipeline, where no work is done as the pipeline stages are reloaded.
5. To avoid this problem, the Pentium uses a scheme called **Dynamic Branch Prediction**.
6. In this scheme, a prediction is made concerning the branch instruction currently in pipeline.
7. Prediction will be either taken or not taken.
8. If the prediction turns out to be true, the pipeline will not be flushed and no clock cycles will be lost.
9. If the prediction turns out to be false, the pipeline is flushed and started over with the correct instruction.
10. It results in a 3 cycle penalty if the branch is executed in the u-pipeline and 4 cycle penalty in v-pipeline.
11. It is implemented using a 4-way set associative cache with 256 entries.
12. This is referred to as the **Branch Target Buffer (BTB)**.

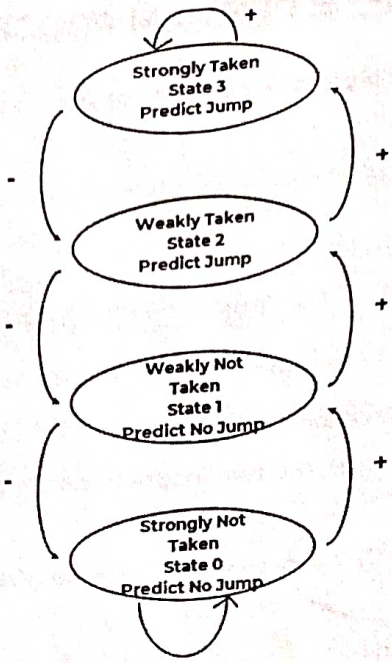


Figure 6.1: State Transition Diagram of BTB Entry.

13. The directory entry for each line contains the following information:
    a. **Valid Bit:** Indicates whether or not the entry is in use.
    b. **History Bits:** Track how often the branch has been taken.
    c. Source memory address that the branch instruction was fetched from (Address of I_3).
14. The history bits indicates one of four possible states.

History Bits	Resulting Description	Prediction Made	If branch is taken	If branch is not taken
11	**Strongly Taken**	Branch Taken	Remains Strongly Taken	Downgrades to Weakly Taken
10	**Weakly Taken**	Branch Taken	Upgrades to Strongly Taken	Downgrades to Weakly Not Taken
01	**Weakly Not Taken**	Branch Not Taken	Upgrades to Weakly Taken	Downgrades to Strongly Not Taken
00	**Strongly Not Taken**	Branch Not Taken	Upgrades to Weakly Not Taken	Remains Strongly Not Taken

15. Thus, if the branch was correctly predicted to be taken, the history bits are upgraded and no further action necessary i.e. correct instructions are already in the pipeline.
16. While, if branch was incorrectly predicted to be taken, the history bits are downgraded and pipeline needs to be flushed and switching of pre-fetcher queue takes place.
17. And if the branch was correctly predicted not to be taken, history bits are downgraded and no action required.
18. While, if incorrectly predicted not to be taken then history bits are upgraded and the queue is flushed and instructions fetched from previous Prefetch queue that contains sequential instructions.
19. Hence time is saved in this case because there are two Prefetch queues.

Q. Explain cache organization of Pentium processor

[10M | May19] [P | High]

Ans:

CODE CACHE ORGANIZATION:

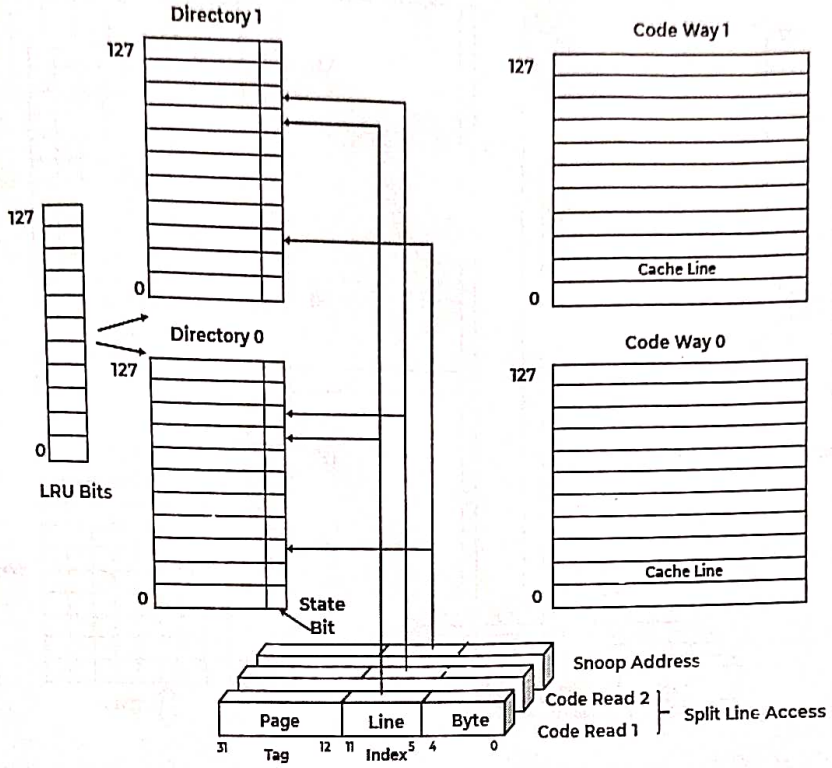


Figure 6.2: Pentium Code Cache Architecture.

1. Code Cache is designed to permit two simultaneous Prefetch accesses.
2. This helps in accessing an instruction that resides in two adjacent cache lines in a single cycle.
3. Code Cache in Pentium is 8 KB in Size.
4. It is organized as 2 way set-associative mapping configuration.
5. There are 2 cache ways i.e. Cache Way 0 & Cache Way 1.
6. Each cache line is (256 bits) 32 bytes wide.
7. The bus connected from the cache to the Prefetcher is also (256 bits) 32 bytes, allowing 32 bytes to be delivered to Prefetcher Queue during a single Prefetch.
8. Each Cache Way contains 128 cache lines with an associated 128 entry directory with each of the cache ways.
9. The cache directories are tripled ported, to support split line access and snooping.
10. Figure 6.2 shows Pentium Code Cache Architecture.
11. The directory entry consists Tag Field, State Bit & Parity Bit.
    a. **Tag Field:** It is of 20 bit, used to identify the page in the memory.
    b. **State Bit:** It indicates whether the line in cache contains valid or invalid information.
    c. **Parity Bit:** It is used to detect error when reading each entry.
12. The directories are accessed by the address issued by the Prefetcher.
13. When the Prefetcher initiates a split-line access, the two line address are submitted to the code cache.
14. Address bits $A_{11} - A_{15}$ from the Prefetcher identify the set where the target line may reside in cache, and are used as index into the cache directories.

15. The lower portion of the Prefetcher address $A_4 - A_0$ identifies a byte within the line.

DATA CACHE ORGANIZATION:

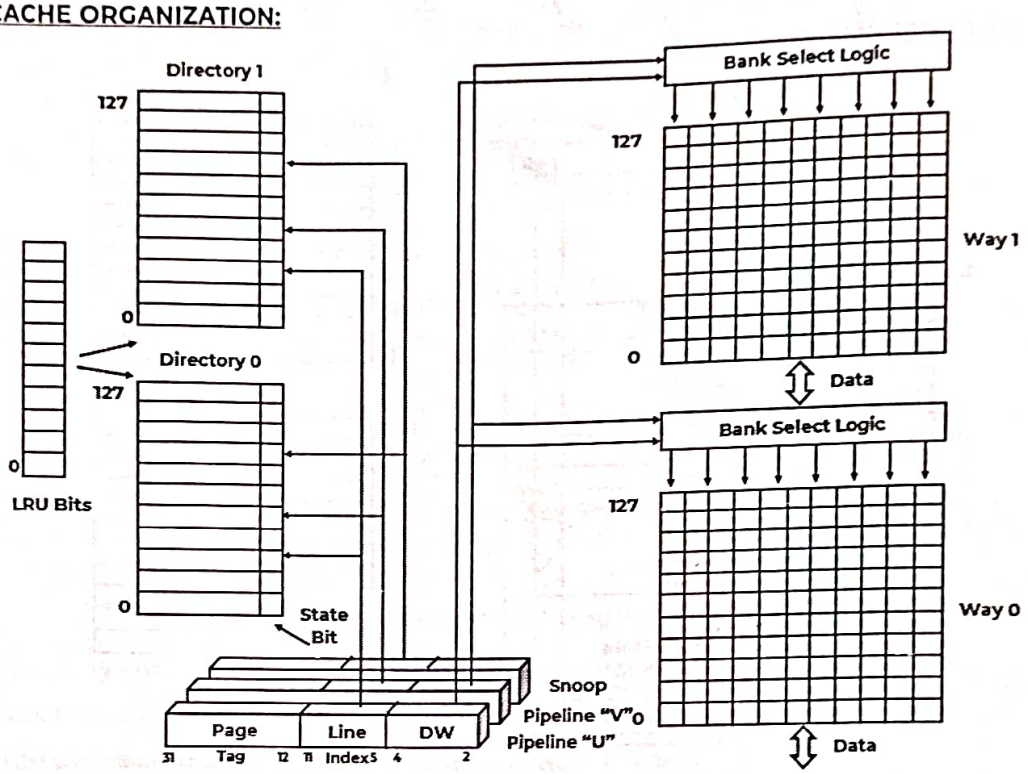


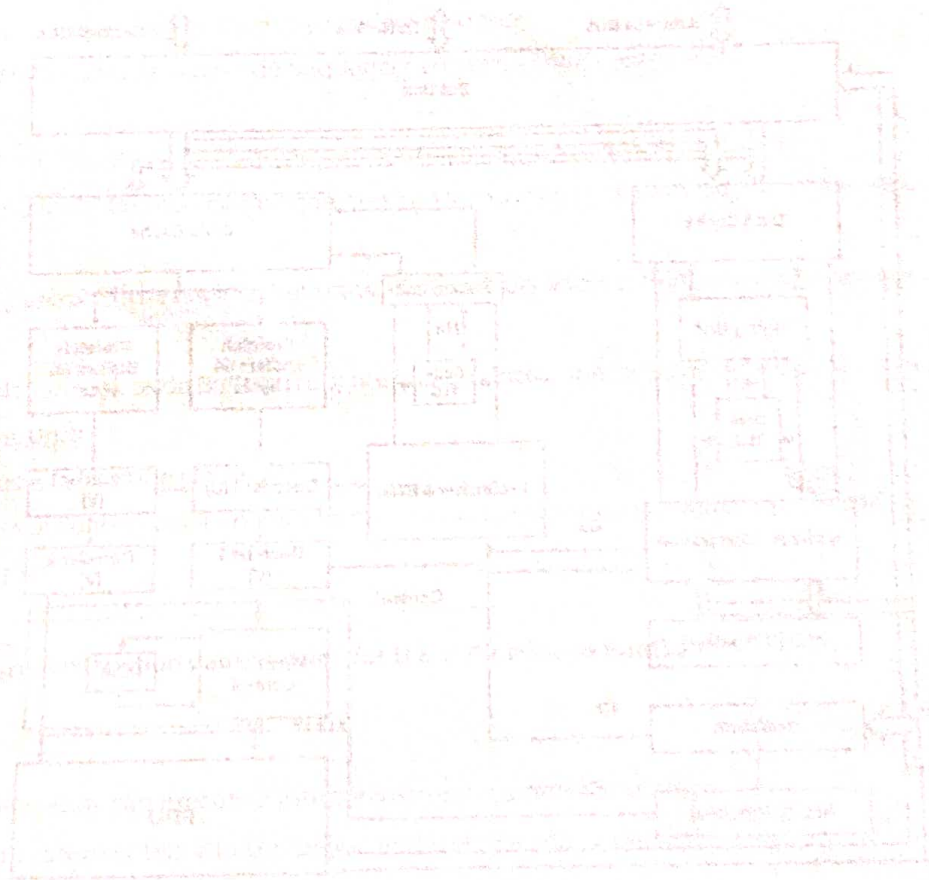
Figure 6.3: Pentium Data Cache Architecture.

1. All accesses by the Execution Units for data are routed through the data cache.
2. Data Cache is used to stores operand information.
3. Figure 6.3 shows Pentium Data Cache Architecture.
4. Data Cache in Pentium is 8 KB in Size.
5. It is organized into two 4 KB ways referred to as Way 0 & Way 1.
6. It contains triple-ported high speed SRAM.
7. Each way consists of 128 lines numbered 0 through 127 with a line size of 32 bytes.
8. Each data cache line consists of 8 double words and the cache ways are banked on double word boundaries.
9. A parity is generated for each byte within a line that is placed in the internal cache.
10. When a byte of information is read from cache, parity is checked.
11. And in case of parity error being detected, an internal parity error is signaled to external logic through the Internal Error Output.
12. Also the processor generates a special shutdown bus cycle and stops execution.
13. Each Directory entry has a tag field used to record the page number of the memory page where the line of information came from.
14. Figure 6.4 shows the Data Cache Directory Entry Structure.

P	TAG (PAGE) $A_{31} : A_{12}$	State Bits

Figure 6.4: Data Cache Directory Entry Structure.

15. **State Bits:**
    00 → Invalid.
    11 → Exclusive.
    22 → Modified.
    33 → Shared.



-- EXTRA QUESTIONS --

Q1. Draw and explain architecture of Pentium Processor. [P | High]

Ans:

1. Pentium Processor is produced by Intel since 1993.
2. It is the brand used for a series of P5 x86 compatible microprocessor.

FEATURES:

1. Pentium Processor has a **superscalar architecture**.
2. Pentium Processor is **32 bit microprocessor**.
3. Physical memory is of 64 GB.
4. Virtual memory is of 64 TB.
5. It has 36 bit addressing lines.
6. It has 64 bit data path.
7. It has Dedicated Instruction Cache & Dedicated Data Cache.
8. Pentium Processor supports **Parallel Integer Execution**.

BLOCK DIAGRAM:

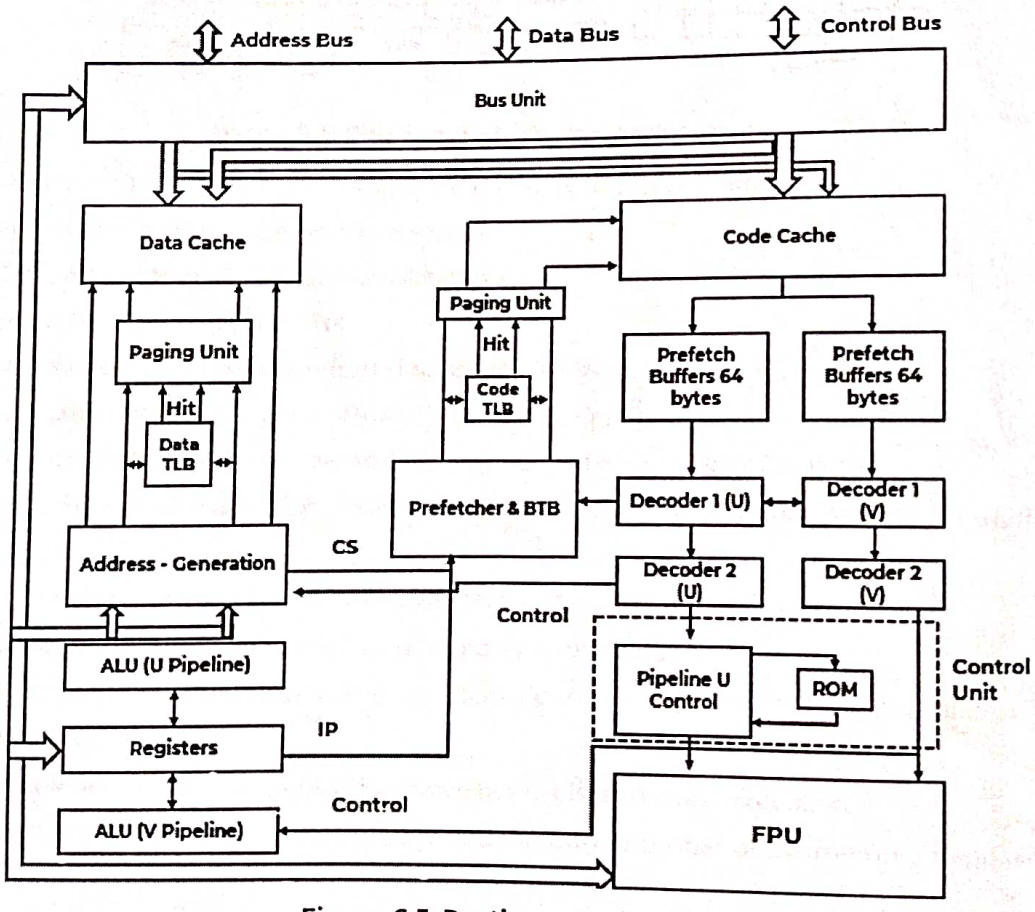


Figure 6.5: Pentium Architecture.

Pentium Architecture includes following blocks:

Bus Unit:

1. It provides the physical interface between the Pentium processor & rest of the system.
2. It consists of Address Drivers & Receivers, Data Bus Transceiver & Bus Control Logic.

Data Cache:

1. Data Cache in Pentium is 8 KB write back cache.
2. It is triple ported to allow simultaneous access from each of the pipelines and snooping.
3. Data cache keeps copy of most frequently used data by 2 integer pipelines and FPU.

Code Cache:

1. Data Cache in Pentium is 8 KB write back cache.
2. It is triple ported to allow simultaneous access from each of the pipelines and snooping.
3. Data cache keeps copy of most frequently used data by 2 integer pipelines and FPU.

Prefetcher:

1. Instructions are requested from code cache by the Prefetcher.
2. If the requested line is not in cache, a burst cycle is run to external memory to perform a cache line fill.

Prefetch Buffers:

1. Pentium processor consists of four Prefetch buffers as 2 independent pairs.
2. When instruction is prefetched it is placed into one set of Prefetch buffers, while the other pair is idle.

Instruction Decode Unit:

1. It occurs in two stages – Decode1 (D1) and Decode2 (D2).
2. D1 checks whether instructions can be paired.
3. D2 calculates the address of memory resident operands.

Control Unit: It Consists of Microcode Sequencer and Microcode Control ROM.

ALU:

1. The ALU for 'U' Pipeline can complete an instruction prior to the ALU in 'V' Pipeline.
2. But the ALU for 'V' Pipeline cannot complete an instruction prior to the ALU in 'U' Pipeline.

Paging Unit:

1. If Paging is enabled, the Paging Unit translates the linear address from address generator to a physical address.
2. Two translation look aside Buffers (TLB) are implemented, one for each code & data cache.

Floating Point Unit:

1. The FPU uses an 8 stage pipeline.
2. Three types of floating point operations can operate simultaneously within FPU: Addition, Division and Multiplication.

Q2. Enlist the instruction pairing rules for U & V Pipeline in Pentium.

[P | Medium]

Ans:

1. Pentium processor can execute 2 integer instructions simultaneously.
2. The first instruction enters into U pipeline and the next into V pipeline.
3. U pipeline has barrel shifter and V pipeline has no barrel shifter.
4. Due to this some instructions can be executed only in U pipeline.
5. The instruction are pairable only if following conditions are satisfied.
   a. Instruction should be simple such as:
      - MOV register, immediate.

- MOV register, register.
- MOV register, memory.
- MOV memory, register.
- MOV memory, immediate.
- INC register.
- INC memory.
- DEC register.
- DEC memory.

b. Instruction should not have register contention:
   - Register contention occurs if two instruction access the same register at a time.

     MOV AX, 5238H

     MOV [SI], AX
   - Old value of AX is stored in the memory instead of new value 5238. So they cannot be paired.

     ADD AX, BX

     ADD AX, CX
   - Old value to AX is added with CX instead of new value in AX. So they cannot be paired.

     DEC CL

     JNZ DOWN
   - Both access flag register at a time. So they cannot be paired.

c. Some instruction are only pairable only if they are first instruction in the pair. i.e.: these instruction needs compulsorily U pipeline. The reason is V pipeline has no barrel shifter.

d. Some instruction are only pairable if they are second instruction pair.

e. **For example:** Unconditional branch instruction.

Q3. Explain integer pipeline of Pentium Processor?

Ans:

[P | Medium]

INTEGER PIPELINE OF PENTIUM PROCESSOR:

1. To execute instruction at high speed, pipelining is used.
2. In Pentium, there are two pipeline known as U pipeline and V pipeline.
3. There are 5 stages in Integer Pipelining.
4. They are:

I) Prefetch Stage:

1. The 2 Prefetch queue A and B are available in integer pipelining stage.
2. Each has 64 bytes.
3. At a time only one queue is active.
4. Each queue have 32 bytes buffer.
5. It is used to fetch instructions from code cache and align the code to the initial byte of next instruction.
6. Then place it in the active Prefetch Buffer.
7. Two instructions from active Prefetch buffer reaches decode one stage of pipeline.

II) **Decode One:**

1. In this stage partial decoding is performed.
2. It is responsible to identify whether 2 instructions or functions can be perform in parallel.
3. The basic function of Decode 1 stage is:
   a. Check pair ability.
   b. Barrier shifter.
   c. Branch prediction.
4. The 2 instructions are checked whether they can form a pair.
5. If they can form a pair, then both the instruction move in together.
6. If they cannot, then instruction in the V pipeline of decode one stage is transferred to decode one stage of U pipeline.
7. At the same time, first instruction in decode one stage of U pipeline is moved to decode two stage of U pipeline.

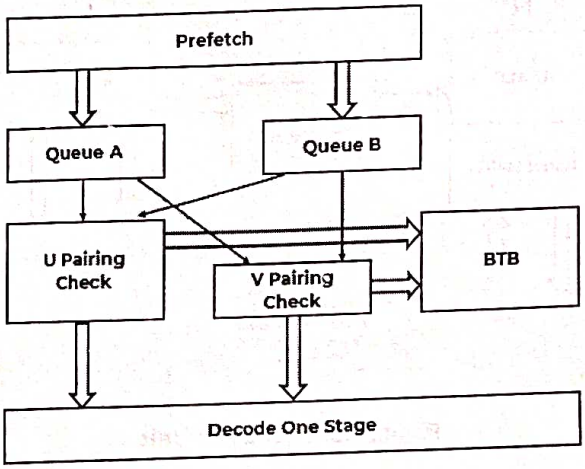


Figure 6.6: Decode One Stage.

III) **Decode Two:**

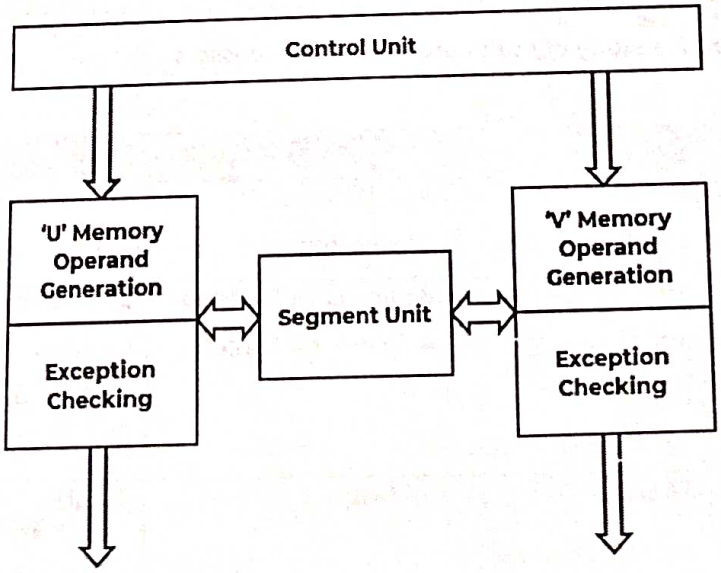


Figure 6.7: Decode Two Stage.

1. In this stage, complete decoding is performed.
2. It is responsible for address calculation.
3. If the instruction try to access the address which is not allowed to be access by the instruction till the interrupt is generated.
4. The responsibility of decode two is to serve interrupt.
5. In this stage protection change is performed.

IV) Execution Unit:

1. It is used to access all the data from main memory and perform all the operation on that data.
2. In this U pipeline has ALU and barrel shifter.
3. V pipeline only have ALU.
4. Due to this V pipeline cannot handle all instructions.

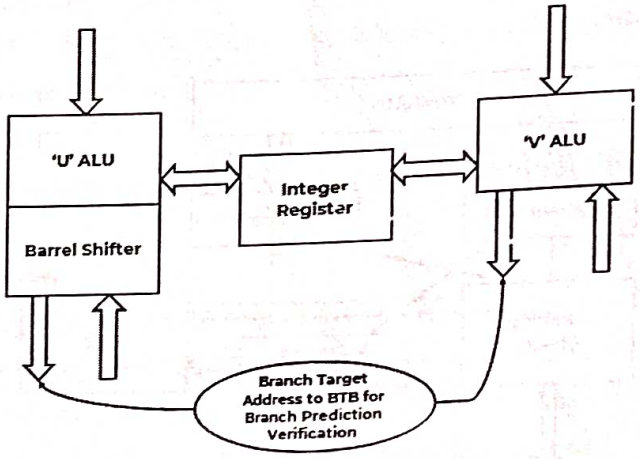


Figure 6.8: Execution Unit.

V) Write Back Stage:

1. Memory location are updated during this stage.
2. In this stage, final result is written on integer register.

Q4. Comparative Study of multicore i3, i5 & i7 processors. **[P | High]**

Ans:

Features	i3	i5	i7	i7 Extreme
Number of Cores	2 for Desktop as well as for Laptop.	4 for Desktop. 2 for Laptop.	4 or 6 for Desktop. 2 or 4 for Laptop.	6 for Desktop. 4 for Mobile.
Processing Threads	4 for Desktop as well as Laptop.	8 threads for Desktop. 4 threads for Laptop.	8 or 12 threads for Desktop. 4 or 8 threads for Laptop.	12 threads for Desktop. 8 threads for Laptop.
Maximum Base Clock Frequency	3.4 GHz	3.4 GHz	3.2 GHz	3.3 GHz

Maximum Turbo Boost Frequency	Not Applicable.	3.8 GHz	3.8 GHz	3.9 GHz
Maximum Smart Cache Size	3 MB	6 MB	12 MB	15 MB
Intel Turbo Boost 2.0	Not present.	Present.	Present.	Present.
Intel Hyper Threading	Present.	Present only in Laptop Processor.	Present.	Present.
K Model	Not present.	Present.	Present.	Present.
Best Desktop Processor	Intel Core i3-2130 (3.4 GHz, 3 MB)	Intel Core i5-2550 K (3.4 GHz, 6 MB)	Intel Core i7-3930 (3.2 GHz, 12 MB)	Intel Core i7-3960 (3.3 GHz, 15 MB)
Best Mobile (Laptop) Processor	Intel Core i3-2370 (2.4 GHz, 3 MB)	Intel Core i5-2540 M (2.6 GHz, 3 MB)	Intel Core i7-2860 (2.5 GHz, 8 MB)	Intel Core i7-2960 XM (2.7 GHz, 8 MB)

Q5. Compare Pentium 2, Pentium 3 & Pentium 4 Processor. **[P | High]**

Ans:

Features	Pentium	Pentium 2	Pentium 3	Pentium 4
Processor Size.	32 Bits	32 Bits.	32 Bits.	32 Bits.
Speed.	60 to 66 MHz.	233 to 300 MHz.	450 to 500 MHz.	400 MHz.
Address Bus Size.	32 bit.	32 bit.	32 bit.	32 bit.
Data Bus Size.	64 bit.	64 bit.	64 bit.	64 bit.
No. of Transistors.	3.1 Million.	7.5 Million.	9.5 To 28 Million.	77 Million.
Addressable Memory.	4 GB.	4 GB.	4 GB.	4GB.
Virtual Memory.	64 TB.	64 TB.	64 TB.	64 TB.
Superscalar.	Yes.	Yes.	Yes.	Yes.
MMX Instruction Set.	-	SSE1.	SSE2.	SSE2.
Hyper Threading Support.	No.	No.	No.	Yes.
Generation.	P5.	P6.	P6.	Net Burst.
Multiprocessor Support.	No.	No.	No.	Yes.
Integer Pipeline Stages.	5.	14.	14.	20.
No. of Integer Pipelines.	2.	2.	2.	4.
Floating Point Pipeline Stages.	8.	8.	11.	20.
No. of Floating Point Pipelines.	1.	1.	1.	2.

On Chip FPU.	Yes.	Yes.	Yes.	Yes.
No. of Cores.	1.	1.	1.	1.
Overclocking Feature.	No.	No.	No.	Yes.
Fabrication Processor.	0.6 µm.	0.35 µm.	0.18 µm.	0.13 µm.

Q6. Compare 8086, 80386 & Pentium.

Ans: **[P | Medium]**

Features	8086	80386	Pentium
Processor Size.	16 Bits.	32 Bits.	32 Bits.
Speed.	5 to 10 MHz.	16 to 33 MHz.	60 To 66 MHz.
Address Bus Size.	20 bit.	32 bit.	32 bit.
Data Bus Size.	16 bit.	32 bit.	64 bit.
No. of Transistors.	29000.	275000.	3.1 Million.
Addressable Memory.	1 MB.	4 GB.	4GB.
Virtual Memory.	-	64 TB.	64 TB.
Superscalar.	No	No	Yes.
MIPS	0.33 to 0.75.	5 to 11.4.	100 to 112.
L1 Cache.	Not Present.	Not Present.	16 KB Split.
Generation.	P1.	P3.	P5.
Multiprocessor Support.	No.	No.	No.
Integer Pipeline Stages.	2.	3.	5.
No. of Integer Pipelines.	1	1.	2.
Floating Point Pipeline Stages.	Not Present.	Not Present.	8.
No. of Floating Point Pipelines.	Not Present.	Not Present.	1.
On Chip FPU.	No.	No.	Yes.
No. of Cores.	1.	1.	1.
Overclocking Feature.	No.	No.	No.
Fabrication Processor.	03 µm.	1 µm.	0.6 µm.

DEC-2018

[3 Hours] – [80 Marks]

Q1 a] Draw and explain memory read machine cycle timing diagram in minimum mode of 8086. [05]

Ans: [Chap 1 | Page No. 10]

b] Write a short note on mixed language programming. [05]

Ans: [Chap 2 | Page No. 24]

c] Explain flag register of 80386 microprocessor. [05]

Ans: [Chap 5 | Page No. 68]

d] Give formats of initialization command words (ICW's) of 8259 PIC. [05]

Ans: [Chap 3 | Page No. 39]

Q2 a] Explain the maximum mode configuration of 8086 microprocessor. [10]

Ans: [Chap 1 | Page No. 7]

b] Design 8086 based system for following specifications: [10]

i) 8086 in minimum mode with clock frequency 5MHz.

ii) 64 KB EPROM using 16KB x 8 chips

iii)16 KB RAM using 8KB x 8 chips

Ans: [Chap 4 | Page No. 54]

Q3 a] Explain the branch prediction logic used in Pentium Processor. [10]

Ans: [Chap 6 | Page No. 83]

b] Draw and explain the block diagram of 8257 DMA controller. [10]

Ans: [Chap 4 | Page No. 45]

Q4 a] Explain the modes of operation of 80386 microprocessor. [10]

Ans: [Chap 5 | Page No. 70]

b) i) Explain the I/O mode control word format of 8255 PPI. [05]

Ans: [Chap 4 | Page No. 49]

ii) Explain an instruction issue algorithm of Pentium processor. [05]

Ans: [Chap 6 | Page No. 83]

Q5 a] Differentiate procedure and macro. Write a program to find the factorial of a number using procedure. [10]

Ans: [Chap 2 | Page No. 29]

b] Explain the interrupt structure of 8086 microprocessor [10]

Ans: [Chap 3 | Page No. 37]

Q6 a] Explain segmentation of 8086 microprocessor. Give its advantages. [10]

Ans: [Chap 1 | Page No. 1]

b] Explain different addressing modes of 8086 microprocessor. [10]

Ans: [Chap 2 | Page No. 25]

MAY-2019 **[3 Hours] – [80 Marks]**

Q1 a] Give the advantages of memory segmentation of 8086 microprocessor. [05]

Ans: [Chap 1 | Page No. 1]

b] Differentiate Procedure and macro with example. [05]

Ans: [Chap 2 | Page No. 29]

c] Explain VM, RF, IOPL and NT flags of 80386 microprocessor. [05]

Ans: [Chap 5 | Page No. 68]

d] Explain an instruction issue algorithm of Pentium processor. [05]

Ans: [Chap 6 | Page No. 83]

Q2 a] Explain minimum mode configuration of 8086 microprocessor. [10]

Ans: [Chap 1 | Page No. 9]

b] Explain cache organization of Pentium processor [10]

Ans: [Chap 6 | Page No. 85]

Q3 a] i) Write a short note on mixed language programming. [05]

Ans: [Chap 2 | Page No. 23]

ii) Write a program to find the largest number from an array. [05]

Ans: [Chap 2 | Page No. 32]

b] Draw and explain the block diagram of 8255 Programmable Peripheral Interface (PPI) with control word formats. [10]

Ans: [Chap 4 | Page No. 47]

Q4 a] Differentiate Real Mode, Protected Mode and virtual 8086 mode of 80386 microprocessor. [10]

Ans: [Chap 5 | Page No. 72]

b] Design 8086 based system for following specifications:

i) 8086 in minimum mode with clock frequency 5MHz.

ii) 128 KB EPROM using 32KB*8 chips

iii) 32 KB RAM using 16KB*8 chips

Ans: [Chap 4 | Page No. 52]

Q5 a] Explain different addressing modes of 8086 microprocessor [10]

Ans: [Chap 2 | Page No. 25]

b] Explain the operation of three 8259 PIC in cascaded mode [10]

Ans: [Chap 3 | Page No. 38]

Q6 a] Draw and explain memory read and memory write machine cycle timing diagrams in maximum mode of 8086 [10]

Ans: [Chap 1 | Page No. 11]

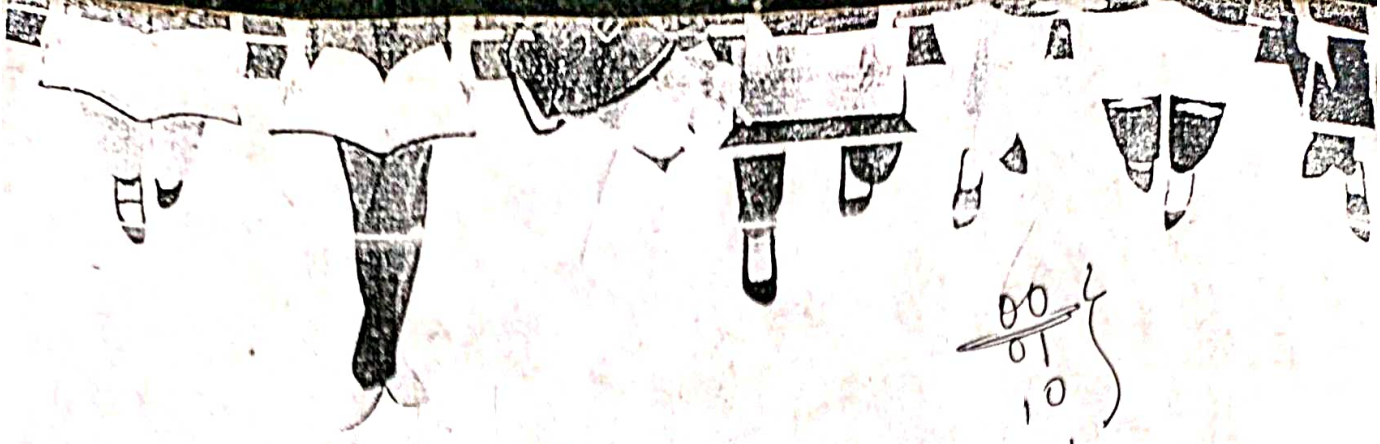
b] Explain the following: [10]

i) Types of interrupts

Ans: [Chap 3 | Page No. 37]

ii) Modes of 8253 Programmable Interval timer

Ans: [Chap 4 | Page No. 50]



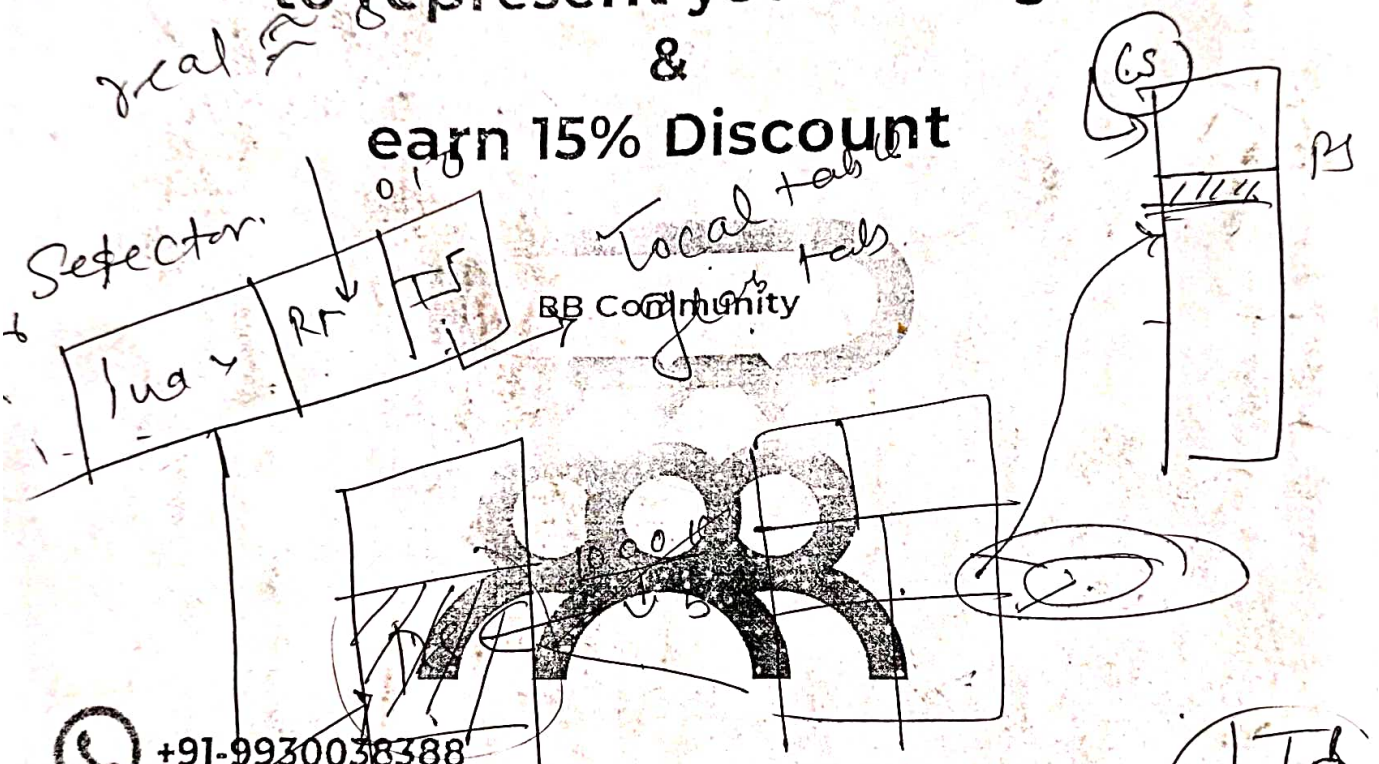
Join BackkBenchers Community

&

become the Student Ambassador to represent your college

&

earn 15% Discount



+91-9930038388

Support@BackkBenchers.com

BackkBenchersCommunity

BackkBenchersCommunity

www.BackkBenchers.com

Price ₹ 100/-