TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Bachelor Level/ First Year/ Second Semester/ Science Full Marks: 60
Microprocessor (CSC 162) Pass Marks: 24
Time: 3 hours.

TU QUESTIONS-ANSWERS 2075

Candidates are required to give their answers in their own words as for as practicable.

The figures in the margin indicate full marks.

Group A (Long Answer Question Section)

Attempt any TWO questions. **(2x10=20)**

1. **Draw the block diagram of 80286 microprocessor and explain its functional units.**

Ans: The functional block diagram of 80286 microprocessor is as given below:

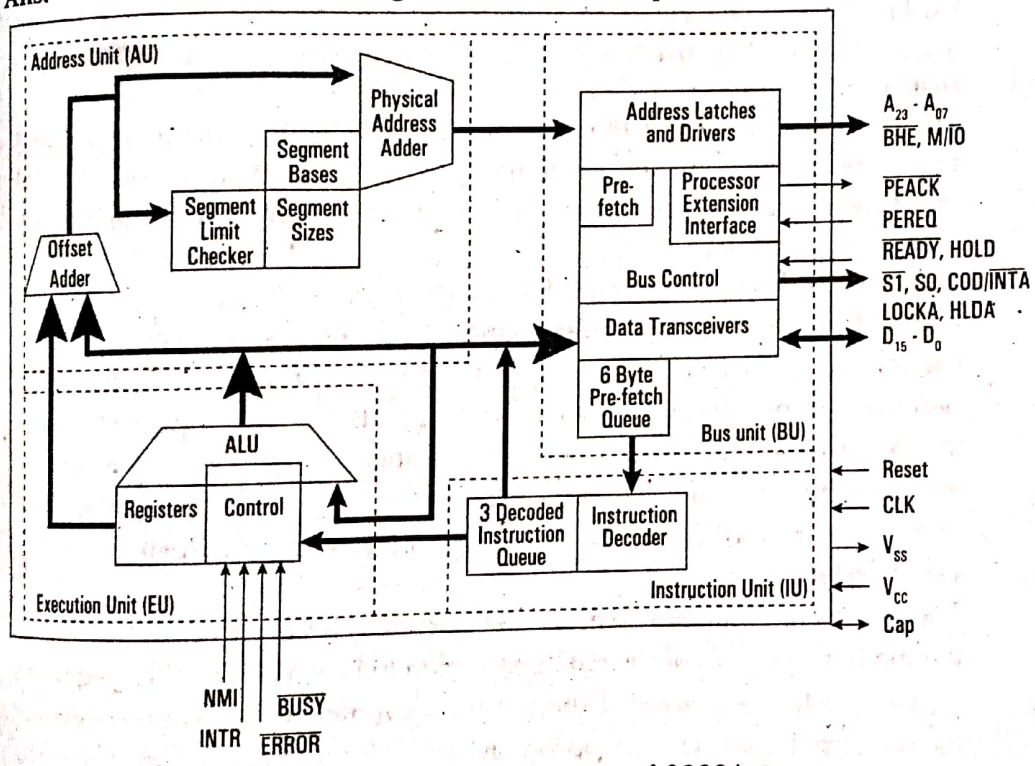


Figure: Internal Block Diagram of 80286

The CPU contain four functional blocks

1. Address Unit (AU)
2. Bus Unit (BU)
3. Instruction Unit (IU)
4. Execution Unit(EU)

The address unit is responsible for calculating the physical address of instructions and data that the CPU wants to access. Also the address lines derived by this unit may be used to address different peripherals. The physical address computed by the address unit is handed over to the bus

unit (BU) of the CPU. Major function of the bus unit is to fetch instruction bytes from the memory. Instructions are fetched in advance and stored in a queue to enable faster execution of the instructions. The bus unit also contains a bus control module that controls the prefetcher module. These prefetched instructions are arranged in a 6-byte instructions queue. The 6-byte prefetch queue forwards the instructions arranged in it to the instruction unit (IU). The instruction unit accepts instructions from the prefetch queue and an instruction decoder decodes them one by one. The decoded instructions are latched onto a decoded instruction queue. The output of the decoding circuit drives a control circuit in the execution unit, which is responsible for executing the instructions received from decoded instruction queue. The decoded instruction queue sends the data part of the instruction over the data bus. The EU contains the register bank used for storing the data as scratch pad, or used as special purpose registers. The ALU, the heart of the EU, carries out all the arithmetic and logical operations and sends the results over the data bus or back to the register bank.

2. Explain instruction cycle, machine cycle and T-states? Draw a timing diagram for STA instruction. Make necessary assumptions.

Ans: Instruction cycle

The necessary steps that the CPU carries out to fetch an instruction and necessary data from the memory and to execute it constitute and instruction cycle it is defined as the time required to complete the execution of an instruction.

An instruction cycle consists of fetch cycle and execute cycle. In fetch cycle CPU fetches opcode from the memory. The necessary steps which are carried out to fetch an opcode from memory constitute a fetch cycle. The necessary steps which are carried out to get data if any from the memory and to perform the specific operation specified in an instruction constitute and execute cycle. The total time required to execute an instruction given by IC = Fc+ Ec the 8085 consists of 1-6 machine cycles or operations.

Fetch cycle:

The first byte of an instruction is its opcode. The program counter keeps the memory address of the next instruction to be executed in the beginning of fetch cycle the content of the program counter, which is the address of the memory location where opcode is available, is send to the memory. The memory places the opcode on the data bus so as to transfer it to CPU. The entire process takes 3 clock cycle.

Execute cycle/Operation:

The opcode fetched from the memory goes to IR from the IR it goes to the decoder which decodes instruction. After the instruction is decoded execution begins.

- If the operand is in general purpose register, execution is performed immediately. I,e in one clock cycle.
- If an instruction is contains data or operand address, then CPU has to perform some read operations to get the desired data.

- In some instruction write operation is performed. In write cycle data are sent from the CPU to the memory of an o/p device.
- In some cases execute cycle may involve one or more read or write cycle or both.

Instruction Cycle (IC) = Fetch cycle (FC) +

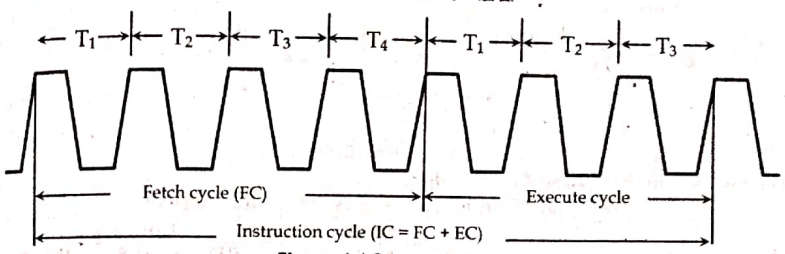


Figure: (a) Processor cycle

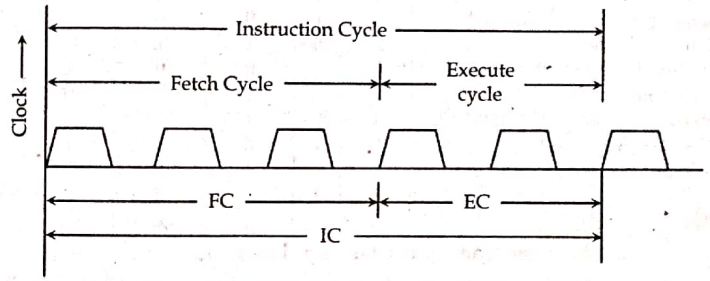


Figure: (b) Instruction cycle showing FC, EC and IC

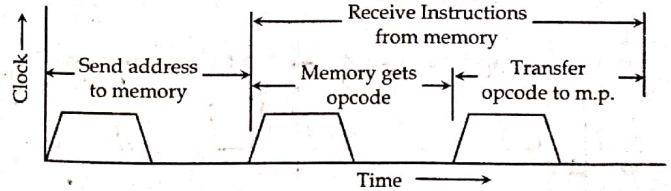


Figure: (c) A Typical Fetch cycle

Machine cycle

It is defined as the time required to complete one operation of accessing memory i/p, o/p or acknowledging and external request. This cycle may consists of 3 to 6 T states. T-states: It is defined as one sub division of the operation performed in one clock period. These sub division are internal states synchronized with system clock and each T states precisely equal to one clock period.

Machine cycles of 8085

The 8085 microprocessor has 5 (seven) basic machine cycles. They are

- ✓ Opcode fetch cycle (4T)
- ✓ Memory read cycle (3 T)

- ✓ Memory write cycle (3 T)
- ✓ I/O read cycle (3 T)
- ✓ I/O write cycle (3 T)
- ✓ Interrupt

Time period, T = 1/f; where f = Internal clock frequency

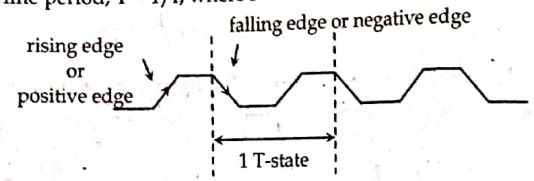


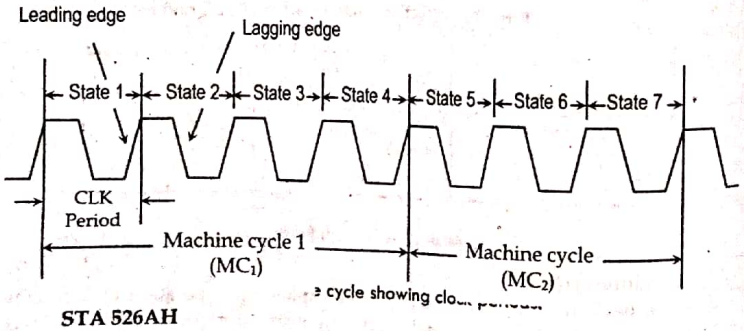
Figure: Clock Signal.

Machine Cycle Status and Control Signals

Table 5.1 (a) Machine cycle status and control signals

	Status			Controls		
Machine cycle	**IO/**	**S1**	**S0**	**RD**	**WR**	**INTA**
Opcode Fetch (OF)	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
I/O Read (I/OR)	1	1	0	0	1	1
I/O Write (I/OW)	1	1	1	1	0	1
Acknowledge of INTR (INTA)	1	0	1	1	1	0
BUS Idle (BI) : DAP	0	1	0	1	1	1
ACK of RST, TRAP	1	1	1	1	1	1
HALT	Z	1	0	Z	Z	1
HOLD	Z	0	X	Z	Z	1

X ⇒ Unspecified, and Z ⇒ High impedance state



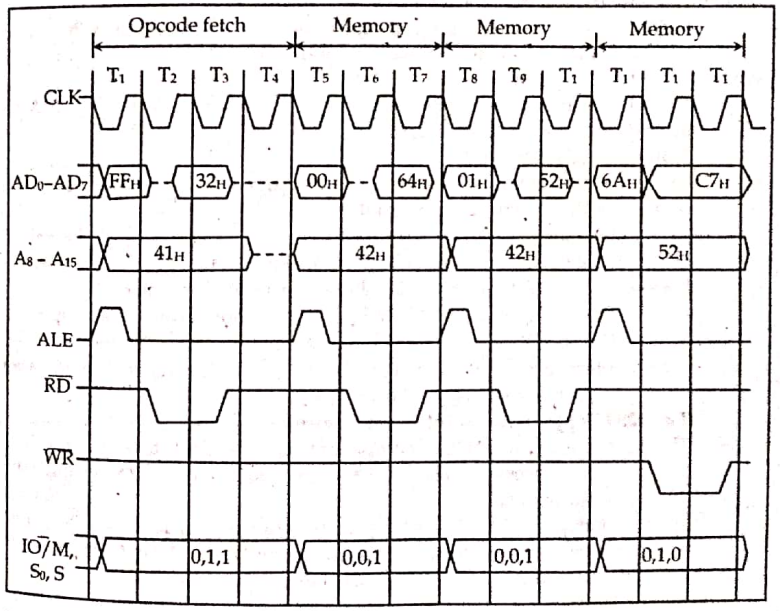
e cycle showing clo... p...

STA 526AH

- STA means Store Accumulator -The contents of the accumulator is stored in the specified address(526A).
- The opcode of the STA instruction is said to be 32H. It is fetched from the memory 41FFH (see fig). - *OF machine cycle.*
- Then the lower order memory address is read(6A). - *Memory Read Machine Cycle*

- Read the higher order memory address (52).- *Memory Read Machine Cycle*
- The combination of both the addresses are considered and the content from accumulator is written in 526A. - *Memory Write Machine Cycle*
- Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

Address	Memonics	Op code
41FF	STA 526 AH	32_H
4200		$6A_H$
4201		52_H



3. Write an assembly language program to find the smallest number in an array using 8 bit microprocessor. (Assume appropriate array data and address where minimum array size of 15 should be considered.)

Ans:

```
LXI H 8D01      ; LOAD COUNTER TO MEMORY ADDRESS 8D01
MOV B M         ; ASSIGNS B AS COUNTER
INX H           ; POINTS WHERE DATA ARE STORED
```

```
MOV A M          ; CONTENT TRANSFERRED TO ACCUMULATOR
DCR B            ; COUNTER VALUE DECREASED
LOOP: INX H      ; POINTS TO ANOTHER MEMORY ADDRESS
CMP M            ; ADDRESSED MEMORY AND
                   ACCUMULATOR ARE COMPARED
JNC/JC AHEAD     ; JUMPS IF ACCUMULATOR IS GREATER
MOV A M          ; ACCUMULATOR CARRIES GREATER VALUE
AHEAD: DCR B     ; DECREASE OF COUNTER
JNZ LOOP         ; IF COUNTER IS NOT ZERO REPEAT THE
                   PROCESS
STA 8D00 H       ; STORES IN MEMROY ADDRESS 8D00 HLT
```

Group B (Short Answer Question Section)

Attempt any EIGHT questions. (8x5=40)

4. Differentiate between vectored and non-vectored interrupt. Where and how 8259 PIC can be used to handle interrupts.

Ans: Vectored Interrupt

In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt.

Non-Vectored Interrupt

But in non-vectored interrupts the interrupted device should give the address of the interrupt service routine (ISR).

In vectored interrupts, the manufacturer fixes the address of the ISR to which the program control is to be transferred. The vector addresses of hardware interrupts are given in table above in previous page.

- The TRAP, RST 7.5, RST 6.5 and RST 5.5 are vectored interrupts.
- The INTR is a non-vectored interrupt. Hence when a device interrupts through INTR, it has to supply the address of ISR after receiving interrupt acknowledge signal.

The 8259 Programmable Interrupt Controller

The 8259A programmable interrupt controller designed to work with Intel microprocessors 8085, 8086 and 8088. The 8259A interrupt controller can

1. Manage eight interrupts according to the instructions written into its control registers. This is equivalent to proving eight interrupt pins on the processor in place of one INTR (8085) pin.
2. Vector can interrupt request anywhere in the memory map. However, all eight interrupts are spaced at the interval of either four or eight locations. This eliminates all the major drawback of the 8085 interrupts in which all interrupts are vectored to memory locations on page 00H.
3. Resolve eight levels of interrupt priorities in a variety of modes, such as fully nested mode, automatic rotation mode, and specific rotation mode.
4. Mask each interrupt request individually.

5. Read the status of pending interrupts, in-service interrupts, and masked interrupts.
6. Be set up to accept either the level-triggered or the edge-triggered interrupt request
7. Be expanded to 64 priority levels by cascading additional 8259As.
8. Be set up to work with either the 8085 microprocessor mode or the 886/8088 microprocessor mode.

The 8259A is upward-compatible with its predecessor, the 8259. The main difference between the two is that the 8259A can be used with Intel's 8086/88 16-bit microprocessor. It also includes additional features such as the level-triggered mode, buffered mode, and automatic-end-of interrupt mode. To simplify the explanation of the 8259A, illustrative examples will not include the cascade mode or the 8086/88 mode and will be limited to modes continuously used with the 8085.

5. Explain addressing modes of 8085 microprocessor with examples.

Ans: Addressing modes:

Instructions are command to perform a certain task in microprocessor. The instruction consists of op-code and data called operand. The operand may be the source only, destination only or both of them. In these instructions, the source can be a register, a memory or an input port. Similarly, destination can be a register, a memory location, or an output port. The various format (way) of specifying the operands are called addressing mode. So addressing mode specifies where the operands are located rather than their nature. The 8085 has 5 addressing mode:

1) Direct addressing mode:

The instruction using this mode specifies the effective address as part of instruction. The instruction size either 2-bytes or 3-bytes with first byte op-code followed by 1 or 2 bytes of address of data.

E. g. LDA 9500H A ← [9500]

IN 80H A ← [80]

This type of addressing is called absolute addressing.

2) Register Direct addressing mode:

This mode specifies the register or register pair that contains the data.

E g. MOV A, B

Here register B contains data rather than address of the data.

Other examples are: ADD, XCHG etc.

3) Register Indirect addressing mode:

In this mode the address part of the instruction specifies the memory whose contents are the address of the operand. So in this type of addressing mode, it is the address of the address rather than address itself. (One operand is register)

e. g. MOV R, M

MOV M, R

STAX, LDAX etc.

STAX B B= 95 C =00 [9500] ← A

4) **Immediate addressing mode:**

In this mode, the operand position is the immediate data. For 8-bit data, instruction size is 2 bytes and for 16 bit data, instruction size is 3 bytes.

E.g. MVI A, 32H
LXI B, 4567H

5) **Implied or Inherent addressing mode:**

The instructions of this mode do not have operands.

E.g. NOP: No operation
HLT: Halt
EI: Enable interrupt
DI: Disable interrupt

6. **Write an ALP to read a string and display the string in uppercase.**

```
;Program To Change The String Into Toggle Case
;-----------------------------------------------------
.MODEL SMALL
.STACK
.DATA
    STRING DB 'Welcome'
    CASE DB 7 DUP(' ')
.CODE
;-----------------------------------------------------
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    LEA SI,STRING ;Load Effective Address of STRING into SI
    LEA DI,CASE  ;Load Effective Address of CASE
    MOV CX,7     ;Load Counter with 7
    TOP:
        CMP CX,0000H ;Compare CX with 0
        JE EXIT ;If CX=0 then Goto Exit
        MOV AH,[SI] ;Load Content of SI into AH
CMP AH,60H  ;Compare AH with 60H i.6 96
        JA ISSMALL
        CMP AH,5AH  ;Compare AH with 5AH i.e 90
        JB ISCAP
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;TO UPPERCASE
    ISSMALL:
            AND AH,11011111B ;Mask With 11011111B
            MOV [DI],AH
            INC SI
            INC DI
DEC CX
            JMP TOP
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;TO LOWERCASE
ISCAP:
            OR AH,00100000B ;Mask With 00100000B
MOV [DI],AH
            INC SI
            INC DI
            DEC CX
            JMP TOP
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        EXIT:
        MOV AL,'$'     ;Add String Terminator.
         MOV [DI],AL ;Pass the Content of AL to DI.
         MOV DX,OFFSET CASE
         MOV AH,09H
         INT 21H
         MOV AH,4CH
         INT 21H
MAIN ENDP
END MAIN
;-----------------------------------------------------
```


7. **What is system bus? Explain different types of system bus in detail.**

Ans: **System Bus:**

It is a communication path between the microprocessor and peripherals; it is nothing but a group of wires to carry bits.

Data Bus: A collection of wires through which data is transmitted from one part of a computer to another is called Data Bus. Data Bus can be thought of as a highway on which data travels within a computer. This bus connects all the computer components to the CPU and main memory. The size (width) of bus determines how much data can be transmitted at one time.

E.g.:

- A 16-bit bus can transmit 16 bits of data at a time.
- 32-bit bus can transmit 32 bits at a time.

Address Bus:

A collection of wires used to identify particular location in main memory is called Address Bus. In other words, the information used to describe the memory locations travels along the address bus. The size of address bus determines how many unique memory locations can be addressed.

E.g.:

- A system with 4-bit address bus can address 24 = 16 Bytes of memory.

- A system with 16-bit address bus can address 216 = 64 KB of memory.
- A system with 20-bit address bus can address 220 = 1 MB of memory.

Control Bus

The connections that carry control information between the CPU and other devices within the computer is called Control Bus. The control bus carries signals that report the status of various devices.

E.g.: This bus is used to indicate whether the CPU is reading from memory or writing to memory.

8. **How DTE and DCE are wired using RS 232 cable? Explain the process of double handshaking I/o.**

Ans: **RS 232C interface with DTE and DCE:**

The figure below shows a interfacing with minimum lines.

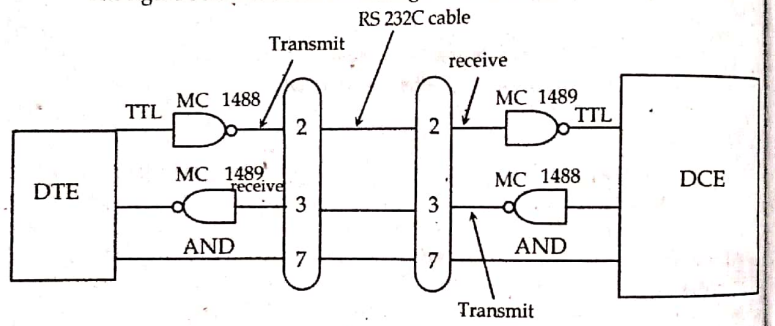


Figure: RS 232C interface

The signaling in RS-232C is not compatible with the TTL logic level. For TTL 0 v to 0.2V is considered a logic 0 and 3.4 v to 5v as logic 1. But RS-232C works in a negative logic -3 to -15v considered as logic 1 and +3 to +15v as logic 0. Because of this incompatibility of the data lines with the TTL logic, voltage translators called line drivers and line receivers are required to interface TTL logic with RS-232C signals. The line driver MC 1488 converts logic 1 into approx. -9V and logic 0 into +9v. Before it is received by the DCE it is again converted by the line receiver MC 1489 into TTL-Compatible logic.

The minimum interface required both a computer and a peripheral device requires three lines; pin 2, 3 and 7. These lines are defined in relation to the DTE; the terminal transmits on pin 2 and receives as pin 3. On the other hand the DCE transmits on pin 3 and receives on pin 1. Pin 7 is ground pin.

DOUBLE HANDSHAKE I/O DATA TRANSFER

For data transfers where coordination is required between sending system and the receiving system, a double handshake is used.

- The sending device asserts its STB low to ask ,**Are you ready?"**
- The receiving system raises its ACK line high to say ,**I'm ready".**
- The peripheral device then sends the data byte and raises its STB signal high to say **"Here is valid data for you".**

- When the receiving system finishes to read the data, receiving system drop its ACK line low to say **"I have data than you and I await your request to send the next byte of data".**

9. **What is instruction set? Explain different kinds of instruction set used in 8085 microprocessor.**

Ans: **THE 8085 INSTRUCTION SETS**

An instruction is a binary pattern designed inside a microprocessor to perform a specific function (task). The entire group of instructions called the instruction set. The 8085 instruction set can be classified into 5-different groups:

- Data Transfer Instructions
- Arithmetic Instructions
- Branching Instructions
- Logical Instructions
- Control Instructions

DATA TRANSFER INSTRUCTIONS

It is the longest group of instructions in 8085. This group of instruction copy data from a source location to destination location without modifying the contents of the source. The transfer of data may be between the registers or between register and memory or between an I/O device and accumulator. None of these instructions changes the flag.

ARITHMETIC INSTRUCTIONS

The 8085 microprocessor performs various arithmetic operations such as addition, subtraction, increment and decrement.

The arithmetic operation add and subtract are performed in relation to the contents of accumulator. The features of these instructions are

- They assume implicitly that the accumulator is one of the operands.
- They modify all the flags according to the data conditions of the result. They place the result in the accumulator.
- They do not affect the contents of operand register or memory. But the INR and DCR operations can be performed in any register or memory. These instructions

✓ Affect the contents of specified register or memory.

✓ Affect the flag except carry flag.

BRANCHINGINSTRUCTIONS

The microprocessor is a sequential machine; it executes machine codes from one memory location to the next. The branching instructions instruct the microprocessor to go to a different memory location and the microprocessor continues executing machine codes from that new location.

LOGICAL INSTRUCTIONS

A microprocessor is basically a programmable logic chip. It can perform all the logic functions of the hardwired logic through its instruction set. The 8085 instruction set includes such logic functions as AND, OR, XOR and NOT (Complement).

10. **What is mean by memory interfacing? Explain the address decoding process in 8085 microprocessor with 3 X 8 decoder.**

Ans: Interfacing a microprocessor is to connect it with various peripherals to perform various operations to obtain a desired output.

Memory Interfacing and I/O Interfacing are the two main types of interfacing.

Memory Interfacing is used when the microprocessor needs to access memory frequently for reading and writing data stored in the memory. It is used when reading/writing to a specific register of a memory chip.

I/O Interfacing is achieved by connecting keyboard (input) and display monitors (output) with the microprocessor.

The method use by the system to select the correct location on the correct chip is called addressing decoding. In another way, it is the process of generating a chip select (CS) or enable signals from the address bus for each device in the system. The address bus lines are split into two sections:

- The N, MSB are used to generate a chip select (CS) signal for different device.
- The M, LSB are passed in the device as address to the different memory cells or internal registers.

ADDRESS BUS

N- bits to decoder(MSB)	M- bits to memory (LSB)

Address decoding can be done using 3 to 8 decoder.

- Assume a microprocessor with 10 address lines that give total of mapping 1 KB memory.
- Let us consider that we want to implement all its memory space and we use 128*8 memory chips.
- So we have 8 memory chips with 128 bytes storage, so we need 3 address lines connected with the decoder to generate 8 different chip select signals and select one of the 8 chips.
- And 7 address lines to select a particular memory word of the selected chip.

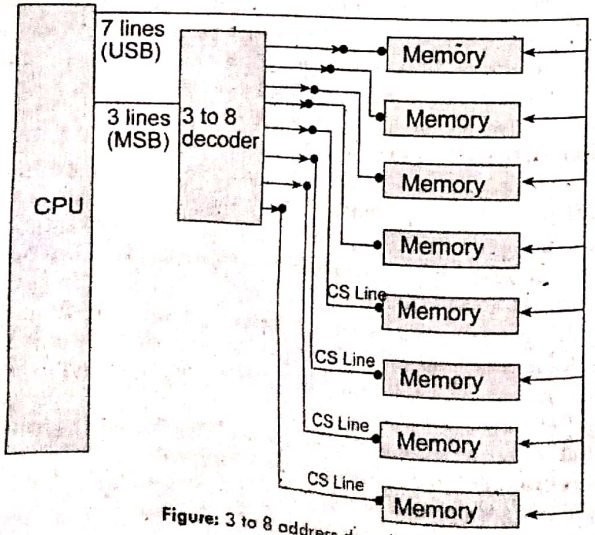


Figure: 3 to 8 address decoding.

11. **Explain how pipelining is achieved in 8086 microprocessor.**

Ans: **Pipelining:**

Pipelining is an implementation technique where multiple instructions are overlapped in execution. The computer pipeline is divided in **stages**. Each stage completes a part of an instruction in parallel. The stages are connected one to the next to form a pipe - instructions enter at one end, progress through the stages, and exit at the other end.

Pipelining does not decrease the time for individual instruction execution. Instead, it increases instruction throughput. The **throughput** of the instruction pipeline is determined by how often an instruction exits the pipeline.

Because the pipe stages are hooked together, all the stages must be ready to proceed at the same time. We call the time required to move an instruction one step further in the pipeline ***a machine cycle***. The ***length*** of the machine cycle is determined by the time required for the slowest pipe stage.

The pipeline designer's *goal is to balance the length of each pipeline stage*. If the stages are perfectly balanced, then the time per instruction on the pipelined machine is equal to

Time per instruction on nonpipelined machine

Number of pipe stages

Under these conditions, the speedup from pipelining equals the number of pipe stages. Usually, however, the stages will not be perfectly balanced; besides, the pipelining itself involves some overhead.

Instruction Pipeline:

Any architecture can be pipelined by making each clock cycle into a pipe stage.

Clock#	1	2	3	4	5	6	7
Instruction i	Fetch	Decode	Execute				
Instr. i+1		Fetch	Decode	Execute			
Instr. i+2			Fetch	Decode	Execute		
Instr. i+3				Fetch	Decode	Execute	
Instr. i+4					Fetch	Decode	Execute

Here instruction i is fetched in clock #1. After it has been fetched it is decoded in clock #2 and at the same time next instruction i.e. instr. i+1 is fetched. At Clock#3, instruction i is executed, instr. i+1 is decoded and instr. i+2 is fetched and so on.

Hence at the end of clock #3, instruction i is executed. Sometime later at the end of clock #4 instruction i+1 is executed.

If we assume that unpipelined architecture took 3ns then this pipelined architecture will take 1ns to finish a stage (fetch, decode and execute).

Advantages of pipelining:

- The execution unit always reads the next instruction byte from the queue in BIU. This is faster than sending out an address to the memory and waiting for the next instruction byte to come.
- In short pipelining eliminates the waiting time of EU and speeds up the processing. The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in queue. 8086 BIU normally obtains two instruction bytes per fetch.

12. Write short notes on: (any two)

a. Von Neumann architecture

Ans: The simplest way to organize a computer is to have one processor, register and instruction code format with two parts op-code and address/operand. The memory address tells the control where to find an operand in memory. This operand is read from memory and used as data to be operated on together with the data stored in the processor register. Instructions are stored in one section of same memory. It is called stored program concept. The task of entering and altering the programs for ENIAC was tedious. It could be facilitated if the program could be represented in a form suitable for storing in memory alongside the data. So the computer could get its instructions by reading from the memory and program could be set or altered by setting the values of a portion of memory. This approach is known as 'stored- program concept' was first adopted by John Von Neumann and such architecture is named as von-Neumann architecture and shown in figure below.

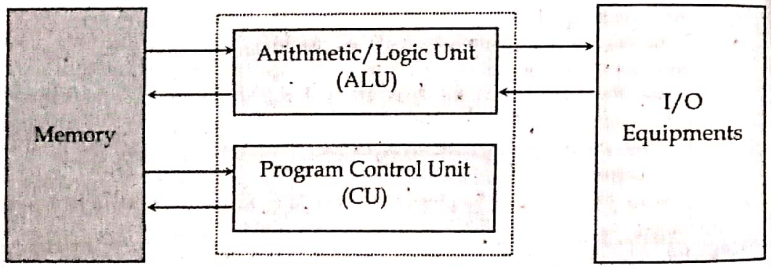


Figure: Von–Neumann Architecture.

The main memory is used to stare both data and instructions. The arithmetic and logic unit is capable of performing arithmetic and logical operation on binary data. The program control unit interprets the instruction in memory and causes them to be executed. The I/O unit gets operated from the control unit.

The Von-Neumann architecture is the fundamental basis for the architecture of modern digital computers. It consisted of 1000 storage locations which can hold words of 40 binary digits and both instructions as well as data are stored in it. The storage location of control unit and ALU are called registers and the various models of registers are:

MAR - memory address register - contains the address in memory of the word to be written into or read from MBR.

MBR - memory buffer register - consists of a word to be stored in or received from memory.

IR - instruction register - contains the 8-bit op-code instruction to be executed.

IBR - instruction buffer register - used to temporarily hold the instruction from a word in memory.

PC - program counter - contains the address of the next instruction to be fetched from memory.

AC & MQ (Accumulator and Multiplier Quotient) - holds the operands and results of ALU after processing.

Macro assembler

Ans: A macro assembler is able to generate a program segment, which is defined by a macro, when the name of the macro appears as an opcode in a program. The macro assembler is still capable of regular assembler functioning, generating a machine instruction for each line of assembly language code; but like a compiler, it can generate many machine instructions from one line of source code. Its instruction set can be expanded to include new mnemonics, which generate these program segments of machine code. The following discussion of how a macro works will show how this can be done.

A frequently used program segment can be written just once, in the macro definition at the beginning of a program. For example, the macro

```
LFEED MACRO
MOV AH, 06H
MOV DL, 0AH
INT 21H
MOV DL, 0DH
INT 21H
ENDM
```

-Is used to put down the cursor in next line.

TU QUESTIONS-ANSWERS 2076

Candidates are required to give their answers in their own words as for as practicable.

The figures in the margin indicate full marks.

Group A (Long Answer Question Section)

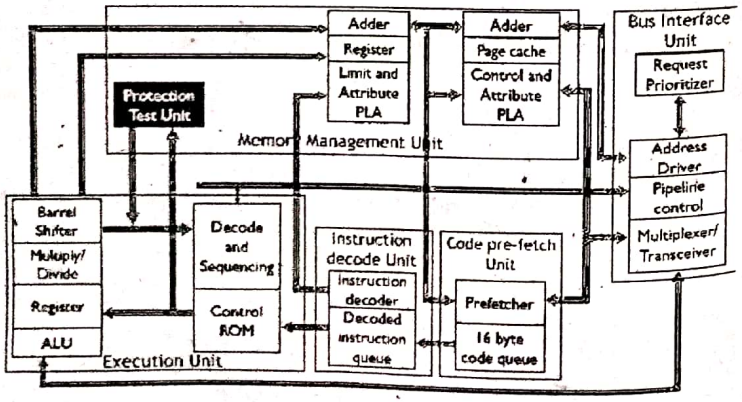
Attempt any TWO questions. (2x10=20)

1. Draw block diagram of 80386 and explain its functional units.

Ans: The internal architecture of the 80386 inch1des six functional units that operate in parallel. The parallel operation is called as pipeline processing. Moreover, Fetching, decoding execution, memory management, and bus access, for several instructions are performed simultaneously.

Also, the six functional units of the 80386 Architecture are

a. Bus Interface Unit
b. Code Pre-fetch Unit
c. Instruction Decoder Unit
d. Execution Unit
e. Segmentation Unit
f. Paging Unit



- The Bus Interface Unit connects the 80386 with memory and I/O. Based on internal requests for fetching instructions and transferring data from the code pre-fetch unit, the 80386 Architecture generates the address, data and control signals for the current bus cycles.
- Also, The code pre-fetch unit pre-fetches instructions when the bus interface unit is not executing the bus cycles. It then stores them in a 16-byte instruction queue for decoding by the instruction decode unit.

- Moreover, The instruction decode unit translates instructions from the pre-fetch queue into microcode. The decoded instructions then stored in an instruction queue (FIFO) for processing by the execution unit.
- The execution unit processes the instructions from die instruction queue. It contains a control unit a data unit and a protection test unit.
- The control unit contains microcode and parallel hardware for fast multiply, divide, and effective address calculation. The unit includes a 32-bit ALU, 8 general purpose registers and a 64-bit barrel shifter for performing multiple bit shifts in one clock. The data unit carries out data operations requested by the control unit.
- Moreover, The protection test unit checks for segmentation violations under the control of microcode.
- Also, The segmentation unit calculates and translates the logical address into linear addresses at the request of the execution unit.
- The translated linear address sent to the paging unit. Upon enabling the paging mechanism, the 80386 translates these linear addresses into Physical addresses.
- Also if paging not enabled, the physical address is identical to the linear address and no translation is necessary.

2. Describe the working mechanism of DMA. Draw the internal architecture of the 8237 DMAC along with a timing diagram illustrating the process of DMA transfers.

Ans: Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.

A computer's system resource tools are used for communication between hardware and software. The four types of system resources are:

- I/O addresses.
- Memory addresses.
- Interrupt request numbers (IRQ).
- Direct memory access (DMA) channels.

DMA channels are used to communicate data between the peripheral device and the system memory. All four system resources rely on certain lines on a bus. Some lines on the bus are used for IRQs, some for addresses (the I/O addresses and the memory address) and some for DMA channels.

A DMA channel enables a device to transfer data without exposing the CPU to a work overload. Without the DMA channels, the CPU copies every piece of data using a peripheral bus from the I/O device. Using a peripheral bus occupies the CPU during the read/write process and does not allow other work to be performed until the operation is completed.

With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. The data block can be transferred to and from memory by the DMAC in three ways.

In burst mode, the system bus is released only after the data transfer is completed. In cycle stealing mode, during the transfer of data between the DMA channel and I/O device, the system bus is relinquished for a few clock cycles so that the CPU can perform other tasks. When the data transfer is complete, the CPU receives an interrupt request from the DMA controller. In transparent mode, the DMAC can take charge of the system bus only when it is not required by the processor.

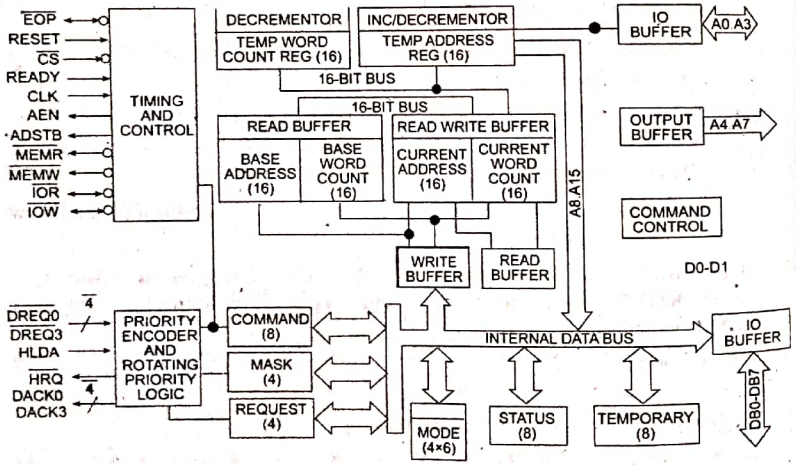


Figure: Architecture of the 8237 DMAC

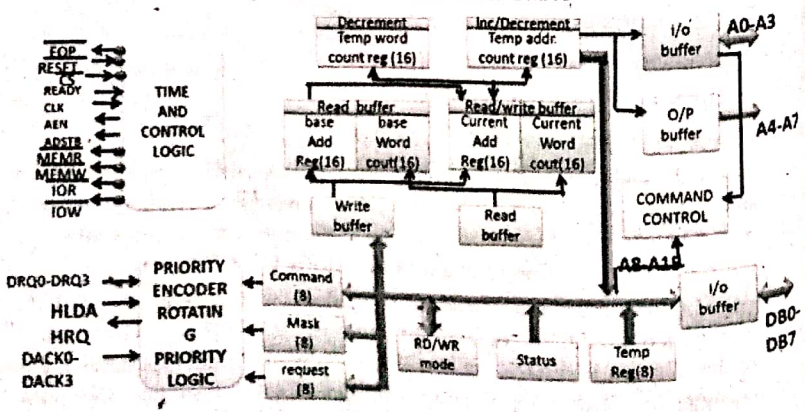


Figure: Internal architecture of the 8237 DMAC

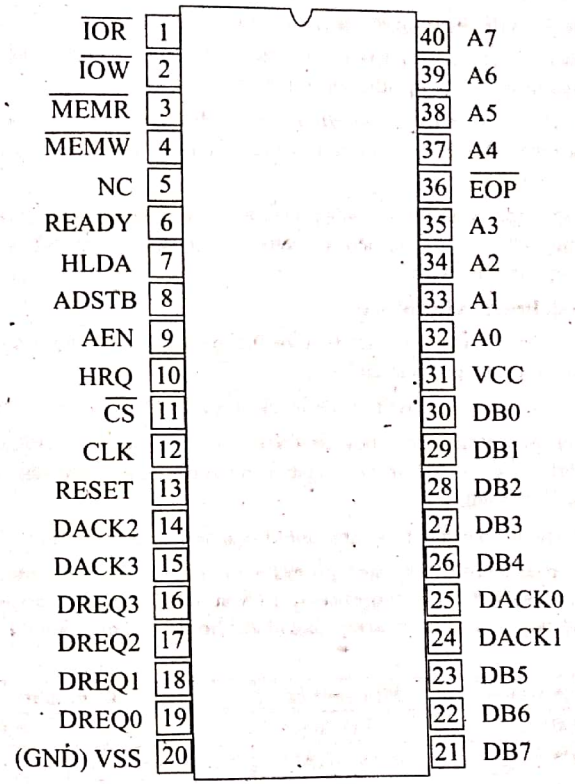


Figure: Pin Diagram of DMA

DMA and its working it's the time to analyze Modes of DMA Transfer.

- During the DMA Transfer CPU can perform only those operation in which it doesn't require the access of System Bus which means mostly CPU will be in blocked state.
- For how much time CPU remains in the blocked state or we can say for how much time CPU will give the control of DMAC of system buses will actually depend upon the following modes of DMA Transfer and after that CPU will take back control of system buses from DMAC.

Mode-1: Burst Mode

- In this mode Burst of data (entire data or burst of block containing data) is transferred before CPU takes control of the buses back from DMAC.
- This is the quickest mode of DMA Transfer since at once a huge amount of data is being transferred.
- Since at once only the huge amount of data is being transferred so time will be saved in huge amount.

Mode-2: Cycle Stealing Mode

- Slow IO device will take some time to prepare data (or word) and within that time CPU keeps the control of the buses.
- Once the data or the word is ready CPU give back control of system buses to DMAC for 1-cycle in which the prepared word is transferred to memory.
- As compared to Burst mode this mode is little bit slowest since it requires little bit of time which is actually consumed by IO device while preparing the data.

Mode-3: Interleaving Mode

- Whenever CPU does not require the system buses then only control of buses will be given to DMAC.
- In this mode, CPU will not be blocked due to DMA at all.
- This is the slowest mode of DMA Transfer since DMAC has to wait might be for so long time to just even get the access of system buses from the CPU itself.
- Hence due to which less amount of data will be transferred

3. Write an assembly language program to find the greatest number in an array in using 8 bit microprocessor. (Assume appropriate array data and address where minimum array size of 20 should be considered.)

Ans:

Memory Address	Mnemonics	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	

Group B

Short answer questions:

Attempt any Eight questions: **(8 x 5 = 40)**

4. Explain the addressing modes of 8086 microprocessor with examples.

Ans: Types of addressing modes of 8086 microprocessor are listed below:

Register mode– In this type of addressing mode both the operands are registers.

Example:

MOV AX, BX

XQR AX, DX

ADD AL, BL

Immediate mode – In this type of addressing mode the source operand is a 8 bit or 16 bit data. Destination operand can never be immediate data.

Example:

MOV AX, 2000

MOV CL, 0A

ADD AL, 45

AND AX, 0000

Note that to initialize the value of segment register an register is required.

MOV AX, 2000

MOV CS, AX

Displacement or direct mode – In this type of addressing mode the effective address is directly given in the instruction as displacement.

Example:

MOV AX, [DISP]

MOV AX, [0500]

Register indirect mode – In this addressing mode the effective address is in SI, DI or BX.

Example: Physical Address = Segment Address + Effective Address

MOV AX, [DI]

ADD AL, [BX]

MOV AX, [SI]

Based indexed mode: In this the effective address is sum of base register and index register.

- Base register: BX, BP
- Index register: SI, DI

The physical memory address is calculated according to the base register.

Example:

MOV AL, [BP+SI]

MOV AX, [BX+DI]

Indexed mode – In this type of addressing mode the effective address is sum of index register and displacement.

Example:

MOV AX, [SI+2000]

MOV AL, [DI+3000]

Based mode – In this the effective address is the sum of base register and displacement.

Example:

MOV AL, [BP+ 0100]

Based indexed displacement mode – In this type of addressing mode the effective address is the sum of index register, base register and displacement.

Example:

MOV AL, [SI+BP+2000]

String mode – This addressing mode is related to string instructions. In this the value of SI and DI are auto incremented and decremented depending upon the value of directional flag.

Example:

MOVS B

MOVS W

Input/Output mode – This addressing mode is related with input output operations.

Example:

IN A, 45

OUT A, 50

Relative mode – In this the effective address is calculated with reference to instruction pointer.

Example:

JNZ 8 bit address

IP=IP+8 bit address

5. Write an ALP for 8086 to read string and print it in the reverse order.

Ans: Reversing a string in 8086 ALP

```
;macro for printing a string
print macro m
mov ah,09h
mov dx,offset m
int 21h
endm
.model small
;****** Data Segment ******
.data
empty db 10,13, "  $"
str1 db 25,?,25 dup('$')
str2 db 25,?,25 dup('$')
mstring db 10,13, "Enter the string: $"
mstring2 db 10,13, "Enter second string: $"
mreverse db 10,13, "Reversed string: $"
;********** Code Segment ***********
.code
start:
mov ax,@data
mov ds,ax
print mstring
call accept_string
```



```
mov si,offset str1     ;point si to start of string1
mov di,offset str2     ;point di to start of string2
mov al,[si]        ;copy first two locations of string1 to string2
mov [di],al        ;since these contain the size and length of the string
inc si             ;which are same in reverse string also
inc di
mov al,[si]
mov [di],al
inc si
inc di

mov cl,str1+1 ;      copy length in cl
mov ch,00
add si,cx          ;add length of string1 to si to move it to last location
dec si             ;si at last location of string1
move_more: mov al,[si]      ;copying character one by one from string1
    pointed by si
mov [di],al      ; to string2 pointed by "di" in reverse order as si moves
dec si         ; from last character to first character
inc di
dec cl
jnz move_more

print mreverse
print str2+2    ;  printing the reversed string
print empty

exit:
mov ah,4ch      ;exit the program
int 21h

;accept procedure
accept proc near
mov ah,01
int 21h
ret
accept endp
display1 proc near
mov al,bl
mov bl,al
and al,0f0h
mov cl,04
rol al,cl
cmp al,09
jbe number
add al,07
number:  add al,30h
mov dl,al
mov ah,02
int 21h
```

```
mov al,bl
and al,00fh
cmp al,09
jbe number2
add al,07
number2: add al,30h
mov dl,al
mov ah,02
int 21h
ret
display1 endp

accept_string proc near
mov ah,0ah      ;accept string from user function
mov dx,offset str1  ; store the string in memory pointed by "DX"
int 21h
ret
accept_string endp
end start
end
```

6. Differentiate between PUSH and POP instruction with example illustrating the use of these instruction.

Ans: The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data when the microprocessor branches to a subroutine. Then the return address used to get pushed on this stack. Also to swap values of two registers and register pairs we use the stack as well.

On a stack, we can perform two operations. PUSH and POP. In case of PUSH operation, the SP register gets decreased by 2 and new data item used to insert on to the top of the stack. On the other hand, in case of POP operation, the data item will have to be deleted from the top of the stack and the SP register will get increased by the value of 2.

Thus, the contents of SP specify the top most useful location in the stack. In other words, it indicates the memory location with the smallest address having useful information. This is pictorially represented in the following figure:

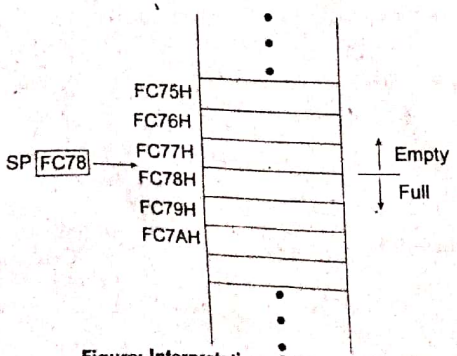


Figure: Interpretation of SP contents

In 8085 Instruction set, PUSH rp instruction stores contents of register pair rp by pushing it into two locations above the top of the stack. rp stands for one of the following register pairs. So 4 register pairs can be mentioned with POP. As mentioned earlier, they are BC, DE, HL and AF or PSW.

Write the process of address and data separation in DE-multiplexed address/data bus in 8085 microprocessor.

Ans: Intel 8085 is an 8-bit microprocessor which has 16 address line for 16-bit address of a memory location. 8 higher order address bits are transferred through 8 bit lines out of this 16 address line while remaining lower order 8 bits of the address are sent through another 8 lines multiplexed with the 8-bit data lines. ALE (Address Enable Latch) is the control signal which is nothing but a positive going pulse generated when a new operation is started by microprocessor. So when pulse goes high means ALE=1, it makes address bus enable and when ALE=0, means low pulse makes data bus enable.

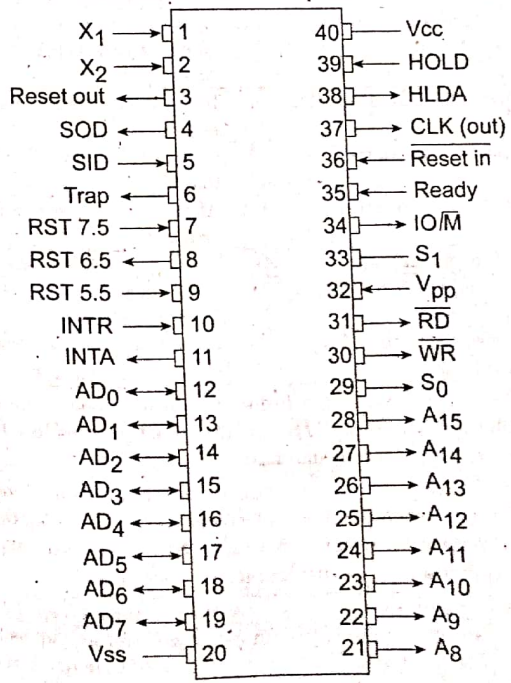


Figure: Pin diagram of 8085

AD0-AD7 lines are multiplexed and the lower half of address A0-A7 is available only during T1 of the machine cycle. This lower half of address is also essential during T2 and T3 of the machine cycle lower half of address is also essential during T2 &T3 of machine cycle to access particular location in memory or I/O part .this means that the lower half of an address bus must be latched in T1 of the machine cycle. The latching of lower half of an address is completed by using external latch and ALE signal from 8085.

8. What is CALL operation? How does it differ with JUMP operation?

Ans: CALL instruction is used to call a subroutine. Subroutines are often used t perform tasks that need to be performed frequently.

The call instruction transfers the program sequence to the memory addres given in the operand. Before transferring, the address of the next instructio after CALL is pushed onto the stack. Call instructions are 2 type Unconditional Call Instructions and Conditional Call Instructions.

The JMP instruction is used to cause the PLC (Programmable Logic Contro to skip over rungs. The differences Between CALL and JUMP instructio are:

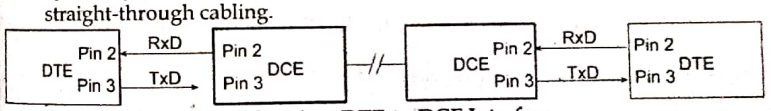
JUMP	CALL
1. Program control is transferred to a memory location which is in the main program	1. Program Control is transferred to a memory location which is not a part of main program
2. Immediate Addressing Mode	2. Immediate Addressing Mode + Register Indirect Addressing Mode
3. Initialization of SP(Stack Pointer) is not mandatory	3. Initialization of SP(Stack Pointer) is mandatory
4. Value of Program Counter(PC) is not transferred to stack	4. Value of Program Counter(PC) is transferred to stack
5. After JUMP, there is no return instruction	5. After CALL, there is a return instruction
6. Value of SP does not changes	6. Value of SP is decremented by 2
7. 10 T states are required to execute this instruction	7. 18 T states are required to execute this instruction
8. 3 Machine cycles are required to execute this instruction	8. 5 Machine cycles are required to execute this instruction

9. Differentiate between synchronous and asynchronous seria communication. Show DTE-DTE and DTE-DCE connection according t RS-232 serial communication standard.

Ans: Transmission is the action of transferring or moving something from on position or person to another. It is a mechanism of transferring data betwee two devices connected using a network. Now, let's see the differen between Synchronous and Asynchronous Transmission:

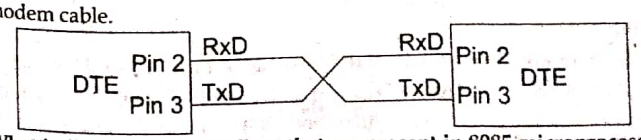
Synchronous Transmission	Asynchronous Transmission
1. In Synchronous transmission, Data is sent in form of blocks or frames.	1. In asynchronous transmission, Data i sent in form of byte or character.
2. Synchronous transmission is fast.	2. Asynchronous transmission is slow.
3. Synchronous transmission is costly.	3. Asynchronous transmission economical.
4. In Synchronous transmission, time interval of transmission is constant.	4. In asynchronous transmission, tim interval of transmission is n constant, it is random.
5. In Synchronous transmission, There is no gap present between data.	5. In asynchronous transmission, There is present gap between data.
6. Efficient use of transmission line is done in synchronous transmission.	6. While in asynchronous transmission, transmission line remains empty during gap in character transmission.
7. Synchronous transmission needs precisely synchronized clocks for the information of new bytes.	7. Asynchronous transmission have no need of synchronized clocks as parity bit is used in this transmission for information of new bytes.

In the RS232 specification, DTE (Data Terminal Equipment) and DCE (Data Communications Equipment) refer to the types of equipment on either end of a serial connection. In general, DTE and DCE refer to computer equipment and modems, respectively. Because the RS232 specification mainly involves connecting a DTE directly to a DCE and vice versa, the pinouts are defined so that cabling is simple. That is, a cable connected a computer to a modem by wiring pin 1 to pin 1, pin 2 to pin 2, and so on. This method is known as straight-through cabling.



Straight-Through Cabling in a DTE-to-DCE Interface

Straight-through cabling is still the standard method to connect a modem to your PC. However, because many applications use serial communication to connect two or more DTEs without modems, the cabling becomes more complicated. If two DTEs are wired together using a straight-through cable, one transmitter is connected to the other transmitter, and one receiver is connected to the other receiver. In this setup, no transmissions can occur. Thus, these applications must use a cabling scheme that connects the transmitter on one device to the receiver on the other device and vice versa. This method is known as null-modem cabling, because it replaces the two modems that traditional RS232 applications would require between the two DTEs. To communicate from one DTE serial port to another, use a null-modem cable.



10. What is flag? Explain the flags that are present in 8085 microprocessor.

Ans: Flags are a modified kind of register that record the condition of a microprocessor's calculation. For instance, a "zero status" flag is activated only when the microprocessor's calculation concludes with a "zero" status. The status of each flag determines the microprocessor's next action, thus enabling it to make decisions.

The Flag register is a Special Purpose Register. Depending upon the value of result after any arithmetic and logical operation the flag bits become set (1) or reset (0). In 8085 microprocessor, flag register consists of 8 bits and only 5 of them are useful.

The 5 flags are:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Sign Flag (S) – After any operation if the MSB (B(7)) of the result is 1, it indicates the number is negative and the sign flag becomes set, i.e. 1. If the MSB is 0, it indicates the number is positive and the sign flag becomes reset i.e. 0.

from 00H to 7F, sign flag is 0
from 80H to FF, sign flag is 1
1- MSB is 1 (negative)
0- MSB is 0 (positive)
Example:
MVI A 30 (load 30H in register A)
MVI B 40 (load 40H in register B)
SUB B (A = A – B)
These set of instructions will set the sign flag to 1 as 30 – 40 is a negative number.
MVI A 40 (load 40H in register A)
MVI B 30 (load 30H in register B)
SUB B (A = A – B)
These set of instructions will reset the sign flag to 0 as 40 – 30 is a positive number.

Zero Flag (Z) – After any arithmetical or logical operation if the result is 0 (00)H, the zero flag becomes set i.e. 1, otherwise it becomes reset i.e. 0.

00H zero flag is 1.
from 01H to FFH zero flag is 0
1- zero result
0- non-zero result
Example:
MVI A 10 (load 10H in register A)
SUB A (A = A – A)
These set of instructions will set the zero flag to 1 as 10H – 10H is 00H

Auxiliary Carry Flag (AC) – This flag is used in BCD number system(0-9). If after any arithmetic or logical operation D(3) generates any carry and passes on to B(4) this flag becomes set i.e. 1, otherwise it becomes reset i.e. 0. This is the only flag register which is not accessible by the programmer

1-carry out from bit 3 on addition or borrow into bit 3 on subtraction
0-otherwise

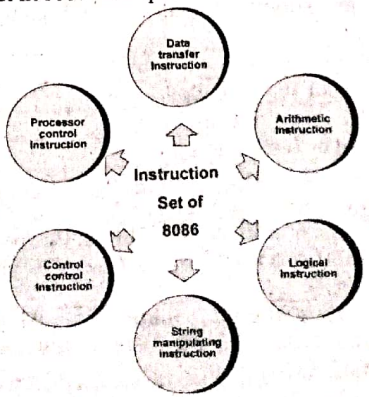
Example:
MOV A 2B (load 2BH in register A)
MOV B 39 (load 39H in register B)
ADD B (A = A + B)
These set of instructions will set the auxiliary carry flag to 1, as on adding 2B and 39, addition of lower order nibbles B and 9 will generate a carry.

Parity Flag (P) – If after any arithmetic or logical operation the result has even parity, an even number of 1 bits, the parity register becomes set i.e. 1, otherwise it becomes reset i.e. 0.
1-accumulator has even number of 1 bits
0-accumulator has odd parity
Example:
MVI A 05 (load 05H in register A)
This instruction will set the parity flag to 1 as the BCD code of 05H is 00000101, which contains even number of ones i.e. 2.

Carry Flag (CY) – Carry is generated when performing n bit operations and the result is more than n bits, then this flag becomes set i.e. 1, otherwise it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset and if (A<B) it becomes set.
Carry flag is also called borrow flag.
1-carry out from MSB bit on addition or borrow into MSB bit on subtraction
0-no carry out or borrow into MSB bit
Example:
MVI A 30 (load 30H in register A)
MVI B 40 (load 40H in register B)
SUB B (A = A – B)
These set of instructions will set the carry flag to 1 as 30 – 40 generates a carry/borrow.
MVI A 40 (load 40H in register A)
MVI B 30 (load 30H in register B)
SUB B (A = A – B)
These set of instructions will reset the sign flag to 0 as 40 – 30 does not generate any carry/borrow.

11. What is instruction set? Explain various Kinds of instructions of 8086 microprocessor.

Ans: The instruction set, also called ISA (instruction set architecture), is part of a computer that pertains to programming, which is more or less machine language. The instruction set provides commands to the processor, to tell it what it needs to do. The instruction set consists of addressing modes, instructions, native data types, registers, memory architecture, interrupt, and exception handling, and external I/O.
The instruction set in 8086 microprocessor are classified as follows:



Data Transfer instruction

Instruction	Description
MOV	Moves data from register to register, register to memory, memory to register, memory to accumulator, accumulator to memory, etc.
LDS	Loads a word from the specified memory locations into specified register. It also loads a word from the next two memory locations into DS register.
LES	Loads a word from the specified memory locations into the specified register. It also loads a word from next two memory locations into ES register.
LEA	Loads offset address into the specified register.
LAHF	Loads low order 8-bits of the flag register into AH register.
SAHF	Stores the content of AH register into low order bits of the flags register.
XLAT/XLATB	Reads a byte from the lookup table.
XCHG	Exchanges the contents of the 16-bit or 8-bit specified register with the contents of AX register, specified register or memory locations.
PUSH	Pushes (sends, writes or moves) the content of a specified register or memory location(s) onto the top of the stack.
POP	Pops (reads) two bytes from the top of the stack and keeps them in a specified register, or memory location(s).
POPF	Pops (reads) two bytes from the top of the stack and keeps them in the flag register.
IN	Transfers data from a port to the accumulator or AX, DX or AL register.
OUT	Transfers data from accumulator or AL or AX register to an I/O port identified by the second byte of the instruction.

Arithmetic Instructions

Instruction	Description
ADD	Adds data to the accumulator i.e. AL or AX register or memory locations.
ADC	Adds specified operands and the carry status (i.e. carry of the previous stage).
SUB	Subtract immediate data from accumulator, memory or register.
SBB	Subtract immediate data with borrow from accumulator, memory or register.
MUL	Unsigned 8-bit or 16-bit multiplication.
IMUL	Signed 8-bit or 16-bit multiplication.
DIV	Unsigned 8-bit or 16-bit division.
IDIV	Signed 8-bit or 16-bit division.
INC	Increment Register or memory by 1.
DEC	Decrement register or memory by 1.
DAA	**Decimal Adjust after BCD Addition:** When two BCD numbers are added, the DAA is used after ADD or ADC instruction to get correct answer in BCD.
DAS	**Decimal Adjust after BCD Subtraction:** When two BCD numbers are added, the DAS is used after SUB or SBB instruction to get correct answer in BCD.

AAA	**ASCII Adjust for Addition:** When ASCII codes of two decimal digits are added, the AAA is used after addition to get correct answer in unpacked BCD.
AAD	**Adjust AX Register for Division:** It converts two unpacked BCD digits in AX to the equivalent binary number. This adjustment is done before dividing two unpacked BCD digits in AX by an unpacked BCD byte.
AAM	**Adjust result of BCD Multiplication:** This instruction is used after the multiplication of two unpacked BCD.
AAS	**ASCII Adjust for Subtraction:** This instruction is used to get the correct result in unpacked BCD after the subtraction of the ASCII code of a number from ASCII code another number.
CBW	Convert signed Byte to signed Word.
CWD	Convert signed Word to signed Doubleword.
NEG	Obtains 2's complement (i.e. negative) of the content of an 8-bit or 16-bit specified register or memory location(s).
CMP	Compare Immediate data, register or memory with accumulator, register or memory location(s).

Logical Instructions

Instruction of this group perform logical AND, OR, XOR, NOT and TEST operations. The following instructions come under this category:

Instruction	Description
AND	Performs bit by bit logical AND operation of two operands and places the result in the specified destination.
OR	Performs bit by bit logical OR operation of two operands and places the result in the specified destination.
XOR	Performs bit by bit logical XOR operation of two operands and places the result in the specified destination.
NOT	Takes one's complement of the content of a specified register or memory location(s).
TEST	Perform logical AND operation of a specified operand with another specified operand.

Rotate Instructions

Instruction	Description
RCL	Rotate all bits of the operand left by specified number of bits through carry flag.
RCR	Rotate all bits of the operand right by specified number of bits through carry flag.
ROL	Rotate all bits of the operand left by specified number of bits.
ROR	Rotate all bits of the operand right by specified number of bits.

Shift Instructions

Instruction	Description
SAL or SHL	Shifts each bit of operand left by specified number of bits and put zero in LSB position.
SAR	Shift each bit of any operand right by specified number of bits. Copy old MSB into new MSB.
SHR	Shift each bit of operand right by specified number of bits and put zero in MSB position.

Branch Instructions

Instruction	Description
JA or JNBE	Jump if above, not below, or equal i.e. when CF and ZF = 0
JAE/JNB/ JNC	Jump if above, not below, equal or no carry i.e. when CF = 0
JB/JNAE/ JC	Jump if below, not above, equal or carry i.e. when CF = 0
JBE/JNA	Jump if below, not above, or equal i.e. when CF and ZF = 1
JCXZ	Jump if CX register = 0
JE/JZ	Jump if zero or equal i.e. when ZF = 1
JG/JNLE	Jump if greater, not less or equal i.e. when ZF = 0 and CF = OF
JGE/JNL	Jump if greater, not less or equal i.e. when SF = OF
JL/JNGE	Jump if less, not greater than or equal i.e. when SF ≠ OF
JLE/JNG	Jump if less, equal or not greater i.e. when ZF = 1 and SF ≠ OF
JMP	Causes the program execution to jump unconditionally to the memory address or label given in the instruction.
CALL	Calls a procedure whose address is given in the instruction and saves their return address to the stack.
RET	Returns program execution from a procedure (subroutine) to the next instruction or main program.
IRET	Returns program execution from an interrupt service procedure (subroutine) to the main program.
INT	Used to generate software interrupt at the desired point in a program.
INTO	Software interrupts to indicate overflow after arithmetic operation.
LOOP	Jump to defined label until CX = 0.
LOOPZ/ LOOPE	Decrement CX register and jump if CX ≠ 0 and ZF = 1.
LOOPNZ/LOOPNE	Decrement CX register and jump if CX ≠ 0 and ZF = 0.

Flag Manipulation and Processor Control Instructions

Instruction	Description
CLC	**Clear Carry Flag:** This instruction resets the carry flag CF to 0.
CLD	**Clear Direction Flag:** This instruction resets the direction flag DF to 0.
CLI	**Clear Interrupt Flag:** This instruction resets the interrupt flag IF to 0.
CMC	This instruction take complement of carry flag CF.
STC	Set carry flag CF to 1.
STD	Set direction flag to 1.
STI	Set interrupt flag IF to 1.
HLT	Halt processing. It stops program execution.
NOP	Performs no operation.
ESC	**Escape:** makes bus free for external master like a coprocessor or peripheral device.
WAIT	When WAIT instruction is executed, the processor enters an idle state in which the processor does no processing.
LOCK	It is a prefix instruction. It makes the LOCK pin low till the execution of the next instruction.

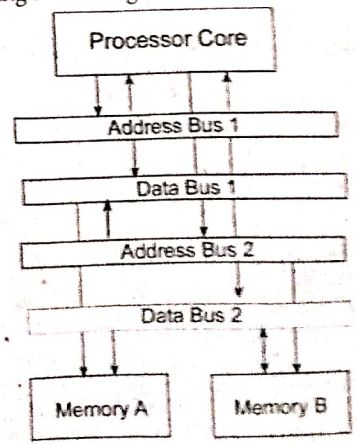
String Instructions

Instruction	Description
MOVS/MOVSB/MOVSW	Moves 8-bit or 16-bit data from the memory location(s) addressed by SI register to the memory location addressed by DI register.
CMPS/CMPSB/ CMPSW	Compares the content of memory location addressed by DI register with the content of memory location addressed by SI register.
SCAS/SCASB/ SCASW	Compares the content of accumulator with the content of memory location addressed by DI register in the extra segment ES.
LODS/LODSB/ LODSW	Loads 8-bit or 16-bit data from memory location addressed by SI register into AL or AX register.
STOS/STOSB/ STOSW	Stores 8-bit or 16-bit data from AL or AX register in the memory location addressed by DI register.
REP	Repeats the given instruction until CX ≠ 0
REPE/ REPZ	Repeats the given instruction till CX ≠ 0 and ZF = 1
REPNE/REPNZ	Repeats the given instruction till CX ≠ 0 and ZF = 0

12. Write short notes on:

a) Harvard architecture

Ans: Harvard architecture refers to a memory structure in which the processor is connected to two independent memory banks via two independent sets of buses. In the original Harvard architecture, one memory bank holds program instructions and the other holds data. Commonly, this concept is extended slightly to allow one bank to hold program instructions and data, while the other bank holds data only. This "modified" Harvard architecture is shown in Figure below. The key advantage of the Harvard architecture is that two memory accesses can be made during any one instruction cycle. Thus, the four memory accesses required for the example FIR filter can be completed in two instruction cycles. This type of memory architecture is used in many DSP families including the Analog Devices ADSP21xx.



b) GDT and LDT

Ans: **GDT, Global Descriptor Table,** is used to define the characteristics of the various memory areas used during program execution, including the base address, the size and access privileges like executability and writability. These memory areas are called segments in Intel terminology. (From Wikipedia) Segment is a term for memory management in Intel x86 architecture, which is also used collaboratively with paging mechanism.

LDT, Local Descriptor Table, acts similar to GDT, which also saves segments descriptor. The main differences between GDT and LDT is: 1) GDT have only one copy in system while LDT can have many, 2) GDT may not change during execution which LDT often changes when task switches, 3) entry of LDT is save in GDT. Entries in GDT and LDT have the same structure.

TU QUESTIONS-ANSWERS 2078

Bachelor Level/ First Year/ Second Semester/ Science — Full Marks: 60
Microprocessor (CSC 162) — Pass Marks: 24
Time: 3 hours.

Group A (Long Answer Question Section)

Attempt any TWO questions. **(2x10=20)**

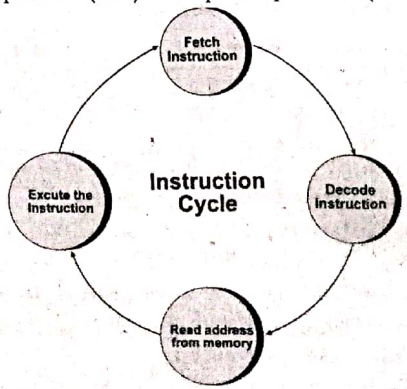
1. **Explain instruction cycle, machine cycle and T-states. Draw timing diagram of IN instruction with brief description.**

Ans: **Instruction Cycle**

The instruction cycle (also known as the fetch-decode-execute cycle, or simply the fetch-execute cycle) is the cycle that the central processing unit (CPU) follows from boot-up until the computer has shut down in order to process instructions.

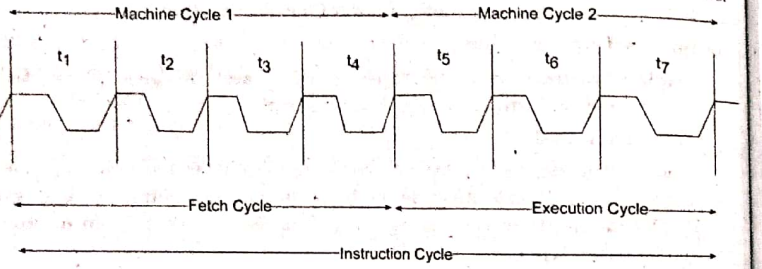
A program residing in the memory unit of a computer consists of a sequence of instructions. These instructions are executed by the processor by going through a cycle for each instruction. In a basic computer, each instruction cycle consists of the following phases:

- Fetch instruction from memory.
- Decode the instruction.
- Read the effective address from memory.
- Execute the instruction.
- **Fetch cycle:** The next instruction is fetched by the address stored in program counter (PC) and then stored in the instruction register.
- **Decode instruction:** Decoder interprets the encoded instruction from instruction register.
- **Reading effective address:** The address given in instruction is read from main memory and required data is fetched. The effective address depends on direct addressing mode or indirect addressing mode.
- **Execution cycle:** consists memory read (MR), memory write (MW), input output read (IOR) and input output write (IOW)



Machine Cycle

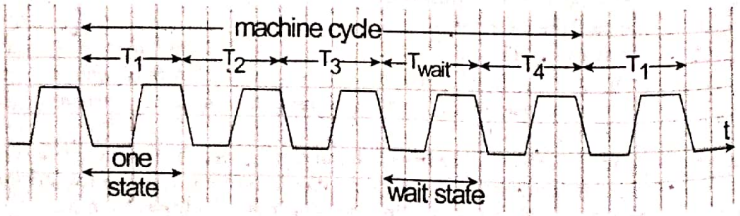
The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called machine cycle. One time period of frequency of microprocessor is called t-state. A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse. Fetch cycle takes four t-states and execution cycle takes three t-states.



T-state

One complete cycle of clock is called as T-state as shown in the above figure. The time intervals T1 orT2 are the examples of T-state.

A T-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse. Various versions of 8086 have maximum clock frequency from 5MHz to 10MHz. Hence the minimum time for one T-state is between 100 to 200 n sec.



Timing diagram of IN instruction

IN structuration are tabulated below;

Address	Mnemonics	Opcode
800F	IN 80H	DB
8010		80

During the first machine cycle, the opcode DB is fetched from the memory, placed in the instruction register and decoded.

During second machine cycle, the port address 80H is read from the next memory location.

During the third machine cycle, the address 80H is placed in the address bus and the data read from that port address is placed in the accumulator.

The timing diagram is shown in Fig below,

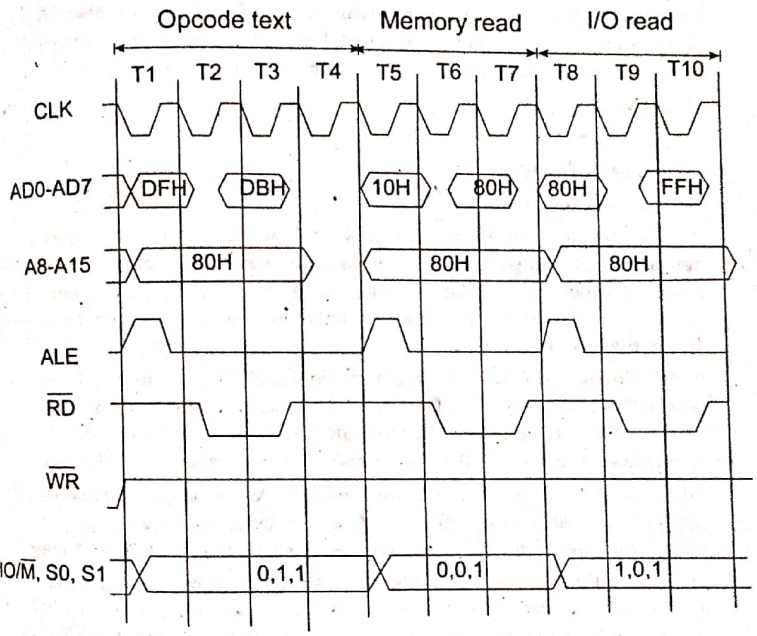
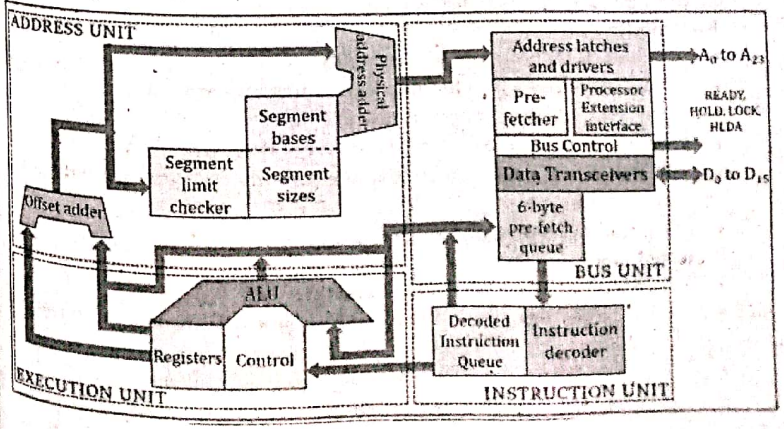


Figure: Timing diagram of IN instruction

2. **Draw block diagram of 80286 microprocessor and explain its main four functional sub-units. Differentiate between real address mode and Protected virtual address mode.**

Ans: The figure below shows the architectural representation of 80286 microprocessor:



Fugure: Block diagram of 80286 microprocessor

80286 is composed of nearly around 125K transistors and the pin configuration has a total of 68 pins. The CPU, central processing unit of 80286 microprocessor, consists of 4 functional block:

- Address Unit
- Bus Unit
- Instruction Unit
- Execution Unit

Firstly, the physical address from where the data or instruction is to be fetched is calculated, by the **address unit.** Once the physical address is calculated then the calculated address is handed over to the bus unit. More specifically we can say, that the calculated address is loaded on the address bus of the bus unit.

This address specifies the memory location from where the data or instruction is to be fetched. The fetching of data through the memory is done through the data bus. For faster execution of instruction, the BU fetches the instructions in advanced from the memory and stores them in the queue.

This is done through the bus control module. As we have discussed that the pre-fetched instructions are stored in a 6-byte instruction queue. This instruction queue then further sends the instruction to the instruction unit.

The instruction unit on receiving the instructions now starts decoding the instruction. As instructions are stored in pre-fetched queue thus the decoder continuously decodes the fetched instructions and stores them into decoded instruction queue.

Now after the instructions gets decoded then further these are needed to be executed. So, the instructions from decoded instruction queue are fed to the execution unit. The main component of EU is ALU i.e., arithmetic and logic unit that performs the arithmetic and logic operations over the operand according to the decoded instruction.

Real mode:

- In this mode processor works as 8088 / 8086.
- This mode has only 1 MB memory addressing capability.
- This mode handles only one task at a time.
- In this memory address translation is not required.
- In this mode processor or computer directly communicate with ports and devices.
- This mode is not supporting memory management.

Protected mode:

- In this processor works in full capacity.
- This mode has more than 1MB to few GB memory addressing capability.
- This mode handles multiple task at time.
- In this memory address translation is required.
- In this processor communicates with ports and devices through OS.
- This mode is supporting memory management

3. **Explain LXI and CMP instruction. Write an assembly language program, for 8-bit microprocessor to divide 8-bit data stored in memory location 8050 by 8 bit data stored in 8051 and store the quotient in 8052 and remainder in 8053.**

Ans: LXI(Load register pair immediate): - In the 8085 Instruction set there are four instructions, which belong to the type LXI rp, d16. These instructions are used to load the 16-bit address into the register pair. We can use this instruction to load data from memory location using the memory address, which is stored in the register pair rp. For an example, if the instruction is LXI H, FE50. It means that the FE50 is loaded into the HL register pair. E.g.: - LXI H, 2034H (2034H is stored in HL pair so that it act as memory pointer) LXI H, XYZ (address of level XYZ is copied in HL pair).

CMP Instruction

The CMP instruction compares two operands. It is generally used in conditional execution. This instruction basically subtracts one operand from the other for comparing whether the operands are equal or not. It does not disturb the destination or source operands. It is used along with the conditional jump instruction for decision making.

Syntax

CMP destination, source

CMP compares two numeric data fields. The destination operand could be either in register or in memory. The source operand could be a constant (immediate) data, register or memory.

Example

CMP DX, 00 ; Compare the DX value with zero

Program part,

ADDRESS	MNEMONICS	COMMENT
2000	LXI H, 8050	
2003	MOV B, M	B<-M
2004	MVI C, 00	C<-00H
2006	INX H	
2007	MOV A, M	A<-M
2008	CMP B	
2009	JC 2011	check for carry
200C	SUB B	A<-A-B
200D	INR C	C<-C+1
200E	JMP 2008	
2011	STA 8053	8053<-A
2014	MOV A, C	A<-C
2015	STA 8052	8052<-A
2018	HLT	terminate the program

Group B

Attempt any 8 questions 8×5=40

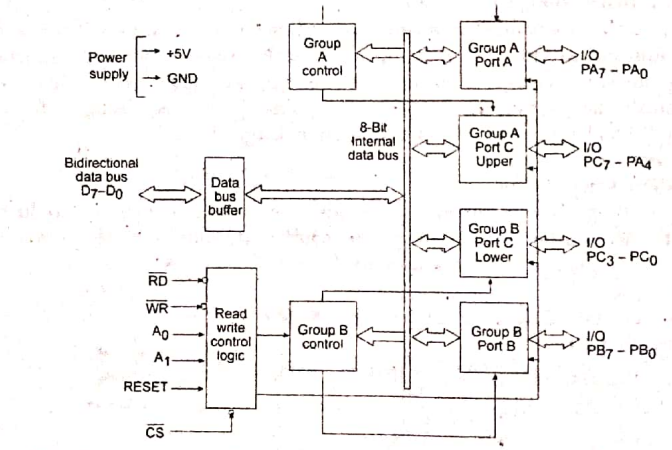
4. **What are the different modes of parallel communication? Construct a control word for 8255 PPI for following configuration:**

Port A and Port C_{upper} -mode 0
Port B and Port C_{lower} -mode 0
Port A and Port C_{upper} -as input port
Port B and Port C_{lower} -as putput port

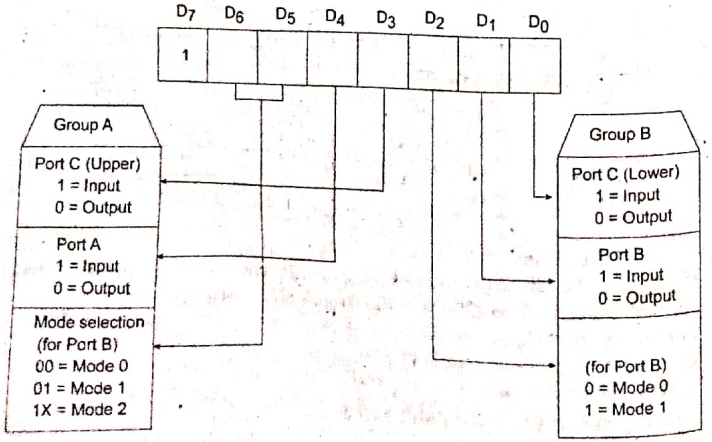
Ans: The 8255 is a general purpose programmable, Parallel I/O device designed for use with Intel microprocessors. It consists of three 8-bit bidirectional I/O ports (24 I/O lines) that can be programmed to transfer data under various conditions from simple I/O to interrupt I/O.

The three ports are PORT A, PORT B & PORT C. Port A contains one 8-bit output latch/buffer and one 8-bit input buffer. Port B is same as PORT A. However, PORT C can be split into two parts PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word. The three ports are divided in two groups Group A (PORT A and upper PORT C) Group B (PORT B and lower PORT C).

Block Diagram



The three ports are divided into two groups—Groups A and B. Group A consists of Port A and CU (PC_4–PC_7) . Port A can be operated in any of the modes—0, 1 or 2. Group B consists of Port B and CL(PC_0–PC_3). Here Port B can be operated in either mode 0 or 1.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	0
I/O mode BSR	Port A in mode 0		Port A as input	Port Cupper as input	Port B in mode 0	Port B as output	Port Clower as output

=98H

5. Differentiate between interrupt base I/O and DMA based I/O. Explain based DMA operation in brief.

Ans: The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system.

Interrupt

The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work. For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory.

Below are the basic operations of Interrupt:

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Direct Memory Access (DMA)

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred

bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus.

DMA Operation

Direct Memory Access involves transfer of data between I/O devices and memory by an external circuitry system called DMA controller without involving the microprocessor. However, microprocessor itself initiates the DMA control process by providing starting address, size of data block and direction of data flow. DMA contains a control unit to deals with the control functions during DMA operations such as read, write and interrupt. The address register of DMA controller is used to generate address and select I/O device to transfer the data block. The Count register counts and hold no. of data block transferred. It also specifies direction of data transfer.

Steps of DMA Operation:

- For DMA operation to occur, the DMA controller first make a bus request (BR) by sending a control signal HOLD to the control line.
- On receiving the BR through HOLD pin high, the microprocessor completes the current instruction execution and afterward it generates HLDA control signal and sends it to the DMA-Controller. This event switches over the control from microprocessor to DMA Controller. The microprocessor gets idle.
- As soon as DMA controller receives HLDA (Hold Acknowledged) through Bus Grant (BG) line, it takes the control of system bus and start transferring the data blocks between memory and Input / Output devices, without involving the microprocessor.
- On completion of data transfer, the DMA controller sends a low signal to the HOLD pin and hence microprocessor makes the HLDA pin low and takes the control over system bus.

6. Differentiate between PUSH and POP instruction with example Illustrating the use of these instruction.

Ans: The stack is a LIFO (last in, first out) data structure implemented in the RAM area and is used to store addresses and data when the microprocessor branches to a subroutine. Then the return address used to get pushed on the stack. Also to swap values of two registers and register pairs we use the stack as well.

On a stack, we can perform two operations. PUSH and POP. In case of PUSH operation, the SP register gets decreased by 2 and new data item used to insert on to the top of the stack. On the other hand, in case of POP operation the data item will have to be deleted from the top of the stack and the SP register will get increased by the value of 2.

Thus, the contents of SP specify the top most useful location in the stack. In other words, it indicates the memory location with the smallest address having useful information. This is pictorially represented in the following figure:

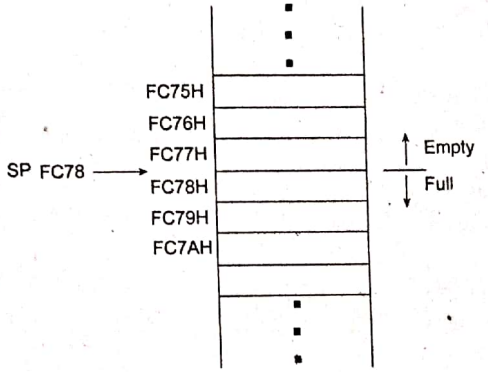


Figure: Interpretation of SP contents

In 8085 Instruction set, PUSH rp instruction stores contents of register pair rp by pushing it into two locations above the top of the stack. rp stands for one of the following register pairs. So 4 register pairs can be mentioned with POP. As mentioned earlier, they are BC, DE, HL and AF or PSW.

7. Write an assembly language program for 16 bit microprocessor to reverse the string "This is microprocessor"

Ans: **Data Segment**

```
str1 db 'This is microprocessor ','$'
strlen1 dw $-str1
strrev db 20 dup(' ')
Data Ends
```

Code Segment

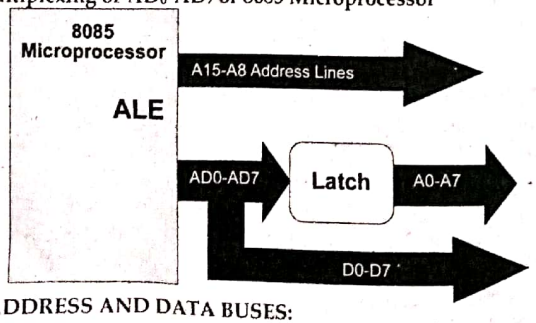
```
Assume cs:code, ds:data
Begin:
 mov ax, data
 mov ds, ax
 mov es, ax
 mov cx, strlen1
 add cx, -2
 lea si, str1
 lea di, strrev
 add si, strlen1
 add si, -2
```

```
L1:
  mov al, [si]
  mov [di], al
  dec si
  inc di
  loop L1
  mov al, [si]
  mov [di], al
  inc di
  mov dl, '$'
  mov [di], dl
Print:
  mov ah, 09h
  lea dx, strrev
  int 21h
Exit:
  mov ax, 4c00h
  int 21h
Code Ends
End Begin
```

8. What is the use of AD7-AD0 in 8085 microprocessor? Explain address de-multiplexing process in 8085 microprocessor with suitable diagram.

Ans: The dual-purpose of the AD_0-AD_7 pins is achieved through multiplexing. In simple words, multiplexing allows us to use the pins of a microprocessor for more than one function.

De-multiplexing of AD_0-AD_7 of 8085 Microprocessor



THE ADDRESS AND DATA BUSES:

- 8085 Microprocessor have total 16 address lines and 8 data lines. 8 signal lines A8 - A15 which are unidirectional.
- The other 8 address bits are multiplexed with the 8 data bits. So, the bits AD_0 - AD_7 are bi-directional and serve as A_0 - A_7 (address bus) and D_0 - D_7 (data bus) at the same time.

- During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits. In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

THE CONTROL AND STATUS SIGNALS:

There are 4 main control and status signals.

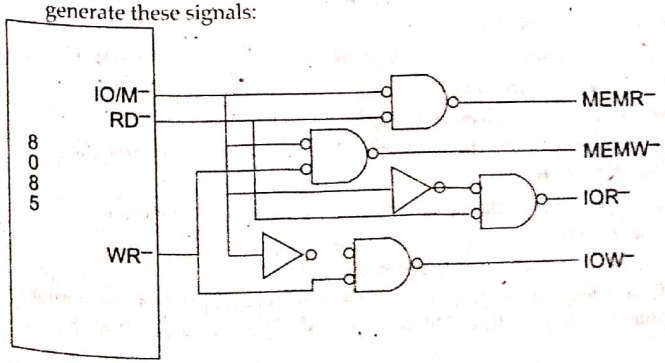
1. **ALE (Address Latch Enable):**
   This signal is a pulse that become 1 when the AD_0 -AD_7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
2. **RD (Read):**
   When it is active low then microprocessor reads the instructions.
3. **WR (Write):**
   When it is active low then microprocessor writes the instructions.
4. **IO/M:**
   This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M = 1).
5. **S1 and S0:** Status signals to specify the kind of operation being performed.

DE-MULTIPLEXING OF AD_0-AD_7 OF 8085 MICROPROCESSOR:

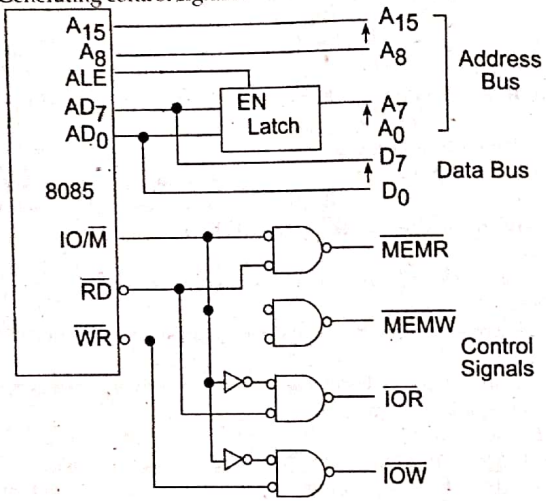
- The AD7- AD0 lines are serving a dual purpose and that they need to be de-multiplexed to get all the information.
- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7- AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.
- ALE operates as a pulse during T1, we will be able to latch the address.
- Then when ALE goes low, the address is saved and the AD7- AD0 lines can be used as the bi-directional data lines.

GENERATING CONTROL SIGNALS:

- The 8085 generates a RD & WR signals. However, the signal needs to be used with both memory and I/O device. So, it must be combined with the IO/M signal to generate different control signals for the memory and I/O device. The following circuitry can be used to generate these signals:



- The following circuitry will show De-multiplexing of AD_0 - AD_7 and Generating control signals:



9. What is mean by addressing mode? Explain all the addressing mode available in 8085 microprocessor.

Ans: Addressing modes:

Instructions are command to perform a certain task in microprocessor. The instruction consists of op-code and data called operand. The operand may be the source only, destination only or both of them. In these instructions, the source can be a register, a memory or an input port. Similarly, destination can be a register, a memory location, or an output port. The various format (way) of specifying the operands are called addressing mode. So addressing mode specifies where the operands are located rather than their nature. The 8085 has 5 addressing mode:

1) Direct addressing mode:

The instruction using this mode specifies the effective address as part of instruction. The instruction size either 2-bytes or 3-bytes with first byte op-code followed by 1 or 2 bytes of address of data.

E. g. LDA 9500H A ← [9500].
IN 80H A ← [80]

This type of addressing is called absolute addressing.

2) Register Direct addressing mode:

This mode specifies the register or register pair that contains the data.

E g. MOV A, B

Here register B contains data rather than address of the data.

Other examples are: ADD, XCHG etc.

3) Register Indirect addressing mode:

In this mode the address part of the instruction specifies the memory whose contents are the address of the operand. So in this type of addressing mode,

it is the address of the address rather than address itself. (One operand is register)

e. g. MOV R, M
MOV M, R
STAX, LDAX etc.
STAX B B= 95 C =00 [9500] ← A

4) Immediate addressing mode:

In this mode, the operand position is the immediate data. For 8-bit data, instruction size is 2 bytes and for 16 bit data, instruction size is 3 bytes.

E.g. MVI A, 32H
LXI B, 4567H

5) Implied or Inherent addressing mode:

The instructions of this mode do not have operands.

E.g. NOP: No operation
HLT: Halt
EI: Enable interrupt
DI: Disable interrupt

10. Explain Register Organization in 80386 microprocessor.

Ans:

A. **General Purpose Registers:** The 80386 has eight 32-bit general purpose registers such as EAX, EBX, ECX, EDX, EBP, ESP, ESI and EDI which may be also used as either 8-bit or 16-bit registers. A 32-bit register, known as an extended register, is represented by the register name with prefix E.

B. **Segment Registers and Instruction Pointer:** The six segment registers available in 80386 are CS, SS, DS, ES, FS and GS. The CS and SS are the code and the stack segment registers respectively, while DS, ES, FS and GS are the four data segment registers. A 16-bit instruction pointer IP is available along with its 32-bit counterpart EIP.

C. **Flag Register:** The flag register of 80386 is a 32-bit register. Out of the 32-bits, Intel has reserved bits D18 to D31, D15, D5 and D3, while D1 is always set at 1. The lower 15 bits (D0- D14) of this flag register are exactly the same as the 80286 flag registers. Only two extra new flags are added in to the flag register of 80386. These are the VM and RF flags.VM-Virtual Mode Flag: If this flag is set, the 80386 enters the virtual 8086 mode within the protected mode. This is to be set only when the 80386 is in protected mode. In this mode, if any privileged instruction is executed an exception-13 is generated. This bit can be set using the IRET instruction or any task switch operation only in the protected mode. RF-Resume Flag: This flag is used with the debug register breakpoints. It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle. The RF is automatically reset after successful execution of every instruction, except for the IRET and POPF instructions. Also, it is not automatically cleared after the successful execution of JMP, CALL and INT instructions causing a task switch. These instructions are used to set the RF to the value specified by the memory data available at the stack.

D. Segment Descriptor Registers: The segment descriptor registers of 80386 are not available for programmers; they are internally used to store the descriptor information, like attributes, limit and base addresses of segments. The six segment registers have corresponding six 73-bit descriptor registers. Each of them contains 32-bit base address, 32-bit base limit and 9-bit attributes.

E. Control Registers: The 80386 has three 32-bit control registers CR0, CR2 and CR3 to hold global machine status independent of the executed task. The load and store instructions are available to access these registers. The control register CR1 is reserved for use in future Intel processors.

F. System Address and Segment Registers: Four special registers are used to refer the descriptor tables supported by 80386. The 80386 supports four types of descriptor tables such as global descriptor table (GDT), interrupt descriptor table (IDT), local descriptor table (LDT) and task state segment descriptor (TSS). The system address registers and system segment registers hold the addresses of these descriptor tables and the corresponding segments. These segments are named as GDTR, IDTR, LDTR and TR respectively. The GDTR and IDTR are called as system address and LDTR and TR are called as system segment registers.

G. Debug and Test Registers: Intel has provided a set of eight debug registers for hardware debugging. Out of these eight registers DR0 to DR7, two registers DR4 and DR5 are Intel reserved. The initial four registers DR0 to DR3 store four program controllable break point addresses, while DR6 and DR7 respectively hold breakpoint status and break point control information. Two more test registers are provided by 80386 for page caching namely test control and test status registers.

11. Draw a logic diagram showing generation of memory and I/O read/write control signals in 8085 microprocessor.

Ans: The 8085 Microprocessor provides RD and WR signals to initiate read or write cycle. Because these Control Signals of 8085 are used both for reading/writing memory and for reading/writing an input device, it is necessary to generate separate read and write signals for memory and I/O devices.

The 8085 provides IO/M signal to indicate whether the initiated cycle is for I/O device or for memory device. Using IO/M signal along with RD and WR, it is possible to generate separate four Control Signals of 8085:

$\overline{MEMR}$	(Memory Read)	: To read data from memory.
$\overline{MEMW}$	(Memory Write)	: To read data in memory.
$\overline{IOR}$	(I/O)	: To read data from I/O device.
$\overline{IOW}$	(I/O Write)	: To write data from I/O device.

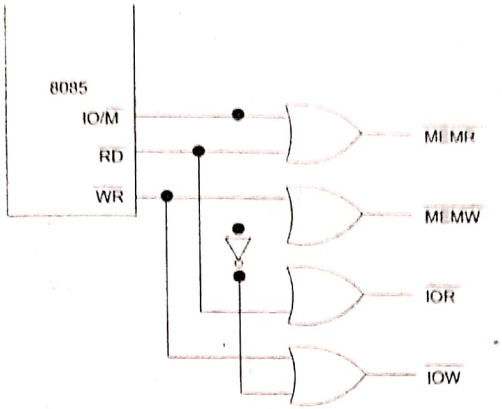


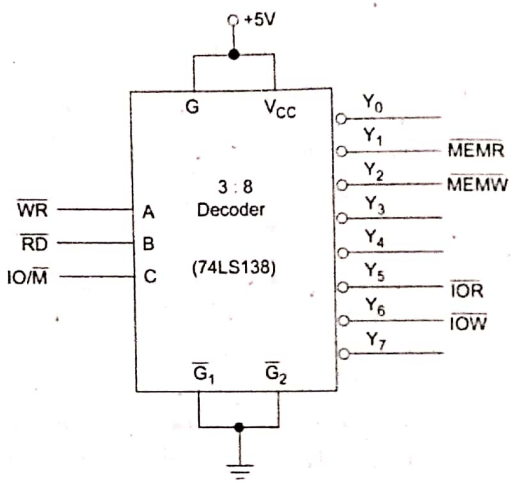
Figure: circuit which generates MEMR, MEMW, IOR and IOW signals

We know that for OR gate, when both the inputs are low then only output is low.

$IO/\overline{M}$	$\overline{RD}$	$\overline{WR}$	$\overline{MEMR}$ $\overline{WR}+IO/\overline{M}$	$\overline{MEMW}\,\overline{WR}+$ $IO/\overline{M}$	$\overline{IOR}$ $\overline{RD}+\overline{IOM}$	$\overline{IOW}$ $\overline{WR}+\overline{IOM}$
0	0	0	Condition never exists, because $\overline{RD}$ and $\overline{WR}$ signals does not go low simultaneously			
0	0	1	0	1	1	1
0	1	0	1	0	1	1
0	1	1	1	1	1	1
1	0	0	Condition never exists, because $\overline{RD}$ and $\overline{WR}$ signals does not go low simultaneously			
1	0	1	1	1	0	1
1	0	1	1	1	1	0
1	1	1	1	1	1	1

Table: truth table used to generate MEMR, MEMW, IOR and IOW signals

The signal IO/M goes low for memory operation. When both RD and IO/M signals go low, MEMR signal goes low. Similarly, when both WR and IO/M Signals go low, MEMW signal goes low. To generate IOR and IOW signals for I/O operation, IO/M signal is first inverted and then logically ORed with RD and WR signals. Same truth table can be implemented using 3:8 decoder as shown in Fig below,



12. Write short notes on (Any two):

a. Program Counter

Ans: A program counter is basically a special purpose register in a computer. It contains the memory address or location of the instruction being executed by a CPU in the current time. As soon as the CPU finishes the execution of the current instruction, the program counter increases its value by 1 and points to the next instruction to be executed by the OS.

A **program counter** contains the memory location of the next instruction. We can view a program counter as a modern digital counter. It facilitates faster execution of the instructions. Furthermore, it provides tracking of the execution points while the CPU executes the instructions.

b. Von-Neumann Architecture

Ans: On Neumann Architecture also known as the Von Neumann model, the computer consisted of a CPU, memory and I/O devices. The program is stored in the memory. The CPU fetches an instruction from the memory at a time and executes it.

Thus, the instructions are executed sequentially which is a slow process. Neumann microprocessor are called control flow computer because instruction are executed sequentially as controlled by a program counter. To increase the speed, parallel processing of computer have been developed in which serial CPU's are connected in parallel to solve a problem. Even in parallel computers, the basic building blocks are Neumann processors.

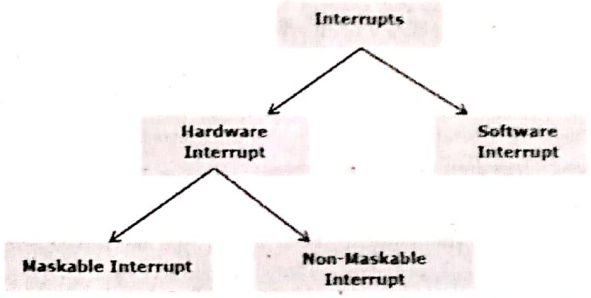
The von Neumann architecture is a design model for a stored-program digital computer that uses a processing unit and a single separate storage structure to hold both instructions and data. It is named after mathematician and early computer scientist John von Neumann. Such a computer implements a universal Turing machine, and the common "referential model" of specifying sequential architectures, in contrast with parallel architectures.

c. Interrupt Masking

Ans: An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task. Interrupt is an event or signal that request to attention of CPU. This halt allows peripheral devices to access the microprocessor.

Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled. INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupts in 8085 microprocessor.

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



The maskable interrupts by default masked by the Reset signal. So no interrupt recognized by the hardware reset.

❑❑❑