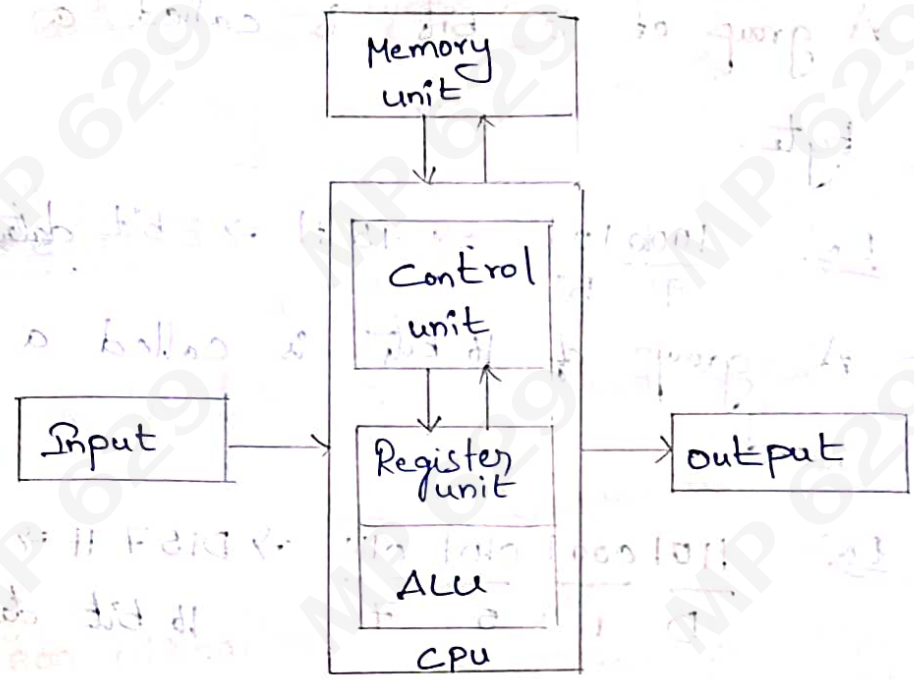
Block Diagram of Computer.



Block Diagram of Micro computer.

Computer Languages:-

=> There are 4 types of computer languages

* High level language
* Low level language
* Machine level language
* Assembly level language.

* In micro processor we use assembly level language.

=> Assembly language is written by using mnemonics (or) opcodes. Ex:- Addition => ADD

Substraction => SUB

Multiplication => MUL

Division => DIV

Computer Terminologies :-

Bit :- A binary that represents 0's and 1's.

Byte :- A group of 8 bits is called a byte.

Ex :- 1000 1011 (9 5) ⇒ 95 H ⇒ 8 bit data

Word :- A group of 16 bits is called a word.

Ex :- 1101 0001 0101 0111 (D 1 5 7) ⇒ D157 H ⇒ 16 bit data

Nibble :- A group of 4 bits is called nibble.

Ex :- 1111 = F H

Instruction :- Instruction is a command given to a computer to perform particular / specific task (operation)

Program :- A set of instructions written in a sequence to perform a particular operation.

Software :- A set of programs written for computer.

Types of Buses :-

1. Address Bus
2. Data Bus
3. Control Bus.

Memory :- The device which stores the data in electric Magnetic tape.

Data :- The quantity of information operated by an instruction in a program.

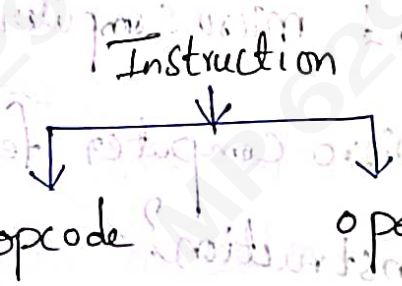
Address :- Address is a memory location. It is an identification number.

pheripherals :- The input and output devices are called pheripherals.

Read operation :- Read means to getting the information from memory.

Write operation :- Write means to store the information in to the memory.

Instruction :- Instruction contain opcode & operand



Mnemonics :- Instruction is also called mnemonic.

=> It contains a english letters (or) alphabets.

CHAPTER-1

INTRODUCTION AND ARCHITECTURE:

8086 MICRO PROCESSOR.

⇒ Cpu is the brain of the computer.

1. Define Microprocessor?

⇒ In general the cpu of a micro computer is called micro processor.

⇒ Microprocessor is denoted by MP (or) μP

⇒ Microprocessor is a single I.C. chip.

2. Define micro Computer?

A digital computer having a microprocessor as its cpu combines with memory and I/o devices is called micro computer.

3. Describe how a Micro computer fetches and Executes and Instruction?

A. ⇒ Any micro computer perform four operation on Instruction.

⇒ The processing of Instruction Involves the following operations.

* Fetch / Read
* Decode
* Execute
* Store / write

=> The instructions are stored in memory.
=> Cpu is fetch the instruction byte from memory.
=> Cpu sends out the address of the first instruction to memory.
=> Cpu sends out the memory read signal.
=> Decodes the fetched instruction.
=> The cpu executes the Decode Instruction.
=> ~~After execution~~ execution
=> After execution the result is stored in a memory.
=> Then the cpu is fetched the next instruction & executed. This process is repeated.

Features of 8086 MICROPROCESSOR

=> Intel 8086 MP (or) MP is a 16-bit microprocessor. Introduced in the year 1978.
=> It has 16-bit data lines (or) databus [D_0 - D_{15}]
=> It has 20 bit address lines (or) address bus [A_0 - A_{19}]
=> It has 20-bit address lines. Hence it is addressing up to $\boxed{2^{20} = 1MB}$
=> It is a 40-PIN IC package.

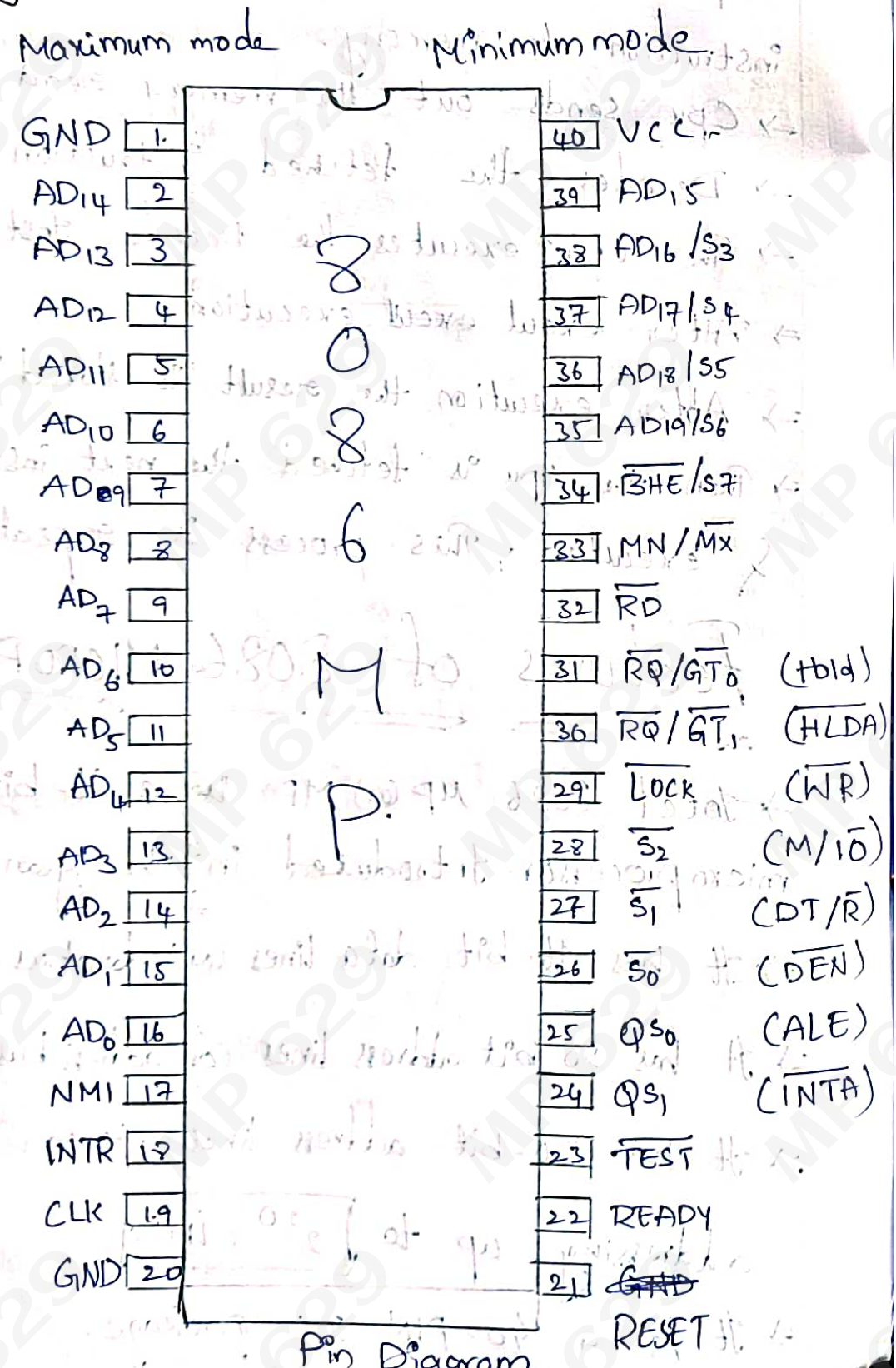
=> The clock frequency is 5MHz.

=> It require +5V DC supply for its operation.

=> It has multiplexed Address/data lines.

Draw the PIN DIAGRAM of 8086 MP?

Diagram:-



Pin Diagram.

=> It is a 40 pin IC package.

=> It has 16 data lines ($D_0, D_1, D_2, \cdots D_{15}$) [$D_0 - D_{15}$]

=> It has 20 address lines [$A_0 - \cdots - A_{19}$] => [$A_0 - A_{19}$]

=> It has multiplexed address/data lines. That is lower order address lines are multiplexed with lower order data lines.

=> Higher order address lines are multiplexed with status lines.

=> [$AD_0 - AD_{15}$] these are called multiplexed address data lines.

=> 8086 operate in two modes. They are

* minimum mode
* Maximum mode.

Description of pins :-

Pin 1 and 20 :-

=> Pin 1 and 20 are GND pins.

=> These pins are used for ground.

Pin 2-16 and 39 :- [$AD_0 - AD_{14}$ and AD_{15}]

=> These pins are multiplexed address data lines. These are bidirectional.

Pin 17 :- NMI [Non-maskable Interrupt]

⇒ The standard of NMI is Non maskable Interrupt.

⇒ It is an edge triggered input signal.
⇒ It is vectored interrupt.
⇒ It is a non-maskable interrupt signal.
⇒ It is highest priority interrupt.

Pin 18 :- INTR [Interrupt request]

⇒ It is lowest priority interrupt.
⇒ Intr standard for Interrupt request.
⇒ It is non-vectored interrupt.
⇒ It is an level triggered input signal.
⇒ It is an maskable interrupt signal.

Pin 19 :- CLK [clock]

⇒ CLK standard for clock.
⇒ 8086 requires 5 MH_2 clock frequency for its operation.

Pin 21 :- RESET

⇒ It resets the system. (ie) this input causes processor to terminate the current activity.

Pin 22 :- READY.

⇒ It is acknowledgement signal sent by the peripherals.
⇒ This is a active high signal.

Pin 23 :- $\overline{Test}$

=> This is an active low signal.

=> If $\overline{Test} = 0$, execution will continue the processor remains in idle state.

=> This pin is input signal used for Wait operation.

Maximum Mode Signals

Pin 24 & 25 :- (QS_1 and QS_0)

(Queue / status)

=> These signals are used for give information about the status of the Queue.

Pin 26, 27 & 28 ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$)

(Status signal)

=> These are status signals connected to the Bus controller.

=> The status signals represent the type of operation being carried out by the processor.

Pin 29 :- $\overline{LOCK}$

=> This is an active low signal.

=> This signal is connected to an external bus controller.

Pin 30 and 31 :- ($RQ/\overline{GT_1}$, and $RQ/\overline{GT_0}$).

=> $RQ/\overline{GT}$ means Request / Grant.

=> These pins are input as well as output pins of 8086.

Pin 32:- $\overline{RD}$

=> RD means Read.

=> It is an active low signal.

=> When $\overline{RD} = 0$, it performs read operation

Pin 33:- [MN / $\overline{MX}$]

=> MN / $\overline{MX}$ means minimum / $\overline{maximum}$.

=> If mn/$\overline{mx}$ = 0, the 8086 operates in maximum mode.

=> If mn/$\overline{mx}$ = 1, the 8086 operates in minimum modes.

Pin 34 :- [$\overline{BHE}$ / S_7] => [Bus high Enable / status]

=> BHE/S means Bus high Enable/status.

=> The BHE signal is used to indicate the transfer of data over the higher order data lines.

=> ~~These~~ These signal is active low signal.

Pin 35, 36, 37 & 38 :- [$\overline{AD_{19}}/S_6$ - AD_{16}/S_3]

=> These are multiplexed address data / status lines.

Pin 40:- VCC

=> This pin provides +5VDC supply for internal operation of up.

Minimum Mode Signals:-

Pin 24 :- ALE => Address Latch Enable.

[$\overline{INTA}$] => Interrupt Acknowledge.

=> INTA stands for Interrupt Acknowledge.

=> It is when this signal is low, the processor receiving the interrupt Acknowledge signal.

PIN 25 :- ALE => Address LATCH Enable.

=> This is active high signal.

=> This pin is used to seperate the address data lines.

PIN 26 :- $\overline{DEN}$ => Data Enable.

=> DEN means Data Enable.

=> This is an active low signal.

=> It is used to enable the data from multiplexed address data lines.

PIN 27 :- [DT/$\overline{R}$] => Data Transfer / $\overline{receive}$

=> If DT/$\overline{R}$ = 0, it perform receive operation.

=> If DT/$\overline{R}$ = 1, it perform data Tranfer operation.

Pin 28 :- [M/$\overline{IO}$] => [memory / $\overline{input\ output}$]

When M/$\overline{IO}$ = 0, It performs input output operation

When M/$\overline{IO}$ = 1, It performs memory operation.

Pin 29 :- [$\overline{WR}$] = [Write]

when this signal is zero, it performs Write operation.

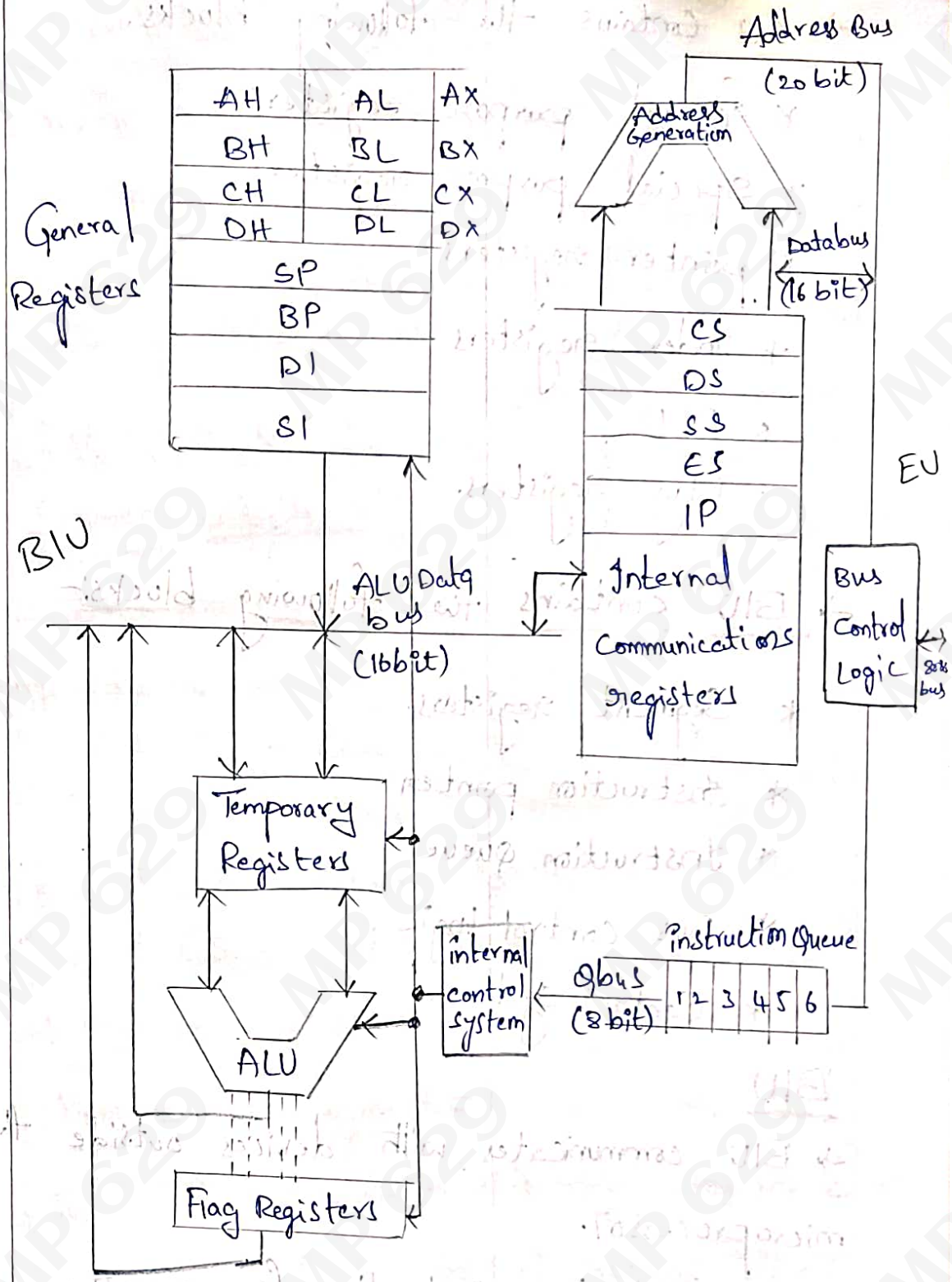
Pin 30 & 31 :- HLDA & HOLD

Hold Acknowledge and hold signals are used for DMA data trasfer. This two signals are high for requesting the bus access.

Draw the Internal architecture of INTEL 8086 Micro processor?

(or)

Draw the functional block Diagram of Intel 8086 MP?



=> Internal architecture of 8086 is divided in to 2 units.

1* BIU [Bus interface unit]

2* EU [Execution unit]

⇒ EU contains the following blocks.

* General purpose registers
* Special purpose registers
* pointer registers
* Index registers
* ALU
* Flag registers.

⇒ BIU contains the following blocks:-

* Segment registers.
* Instruction pointer
* Instruction Queue
* Bus control logic
* Control system.

BIU

⇒ BIU communicates with devices outside the microprocessor.

⇒ It fetch the instruction from memory and stores them in a instruction queue.

⇒ The BIU unit generates 20-bit physical address by using Segment registers.

Instruction Queue:-

=> The length of the instruction queue is 6 bytes.

=> Queue works on the principle of FIFO.

=> The prefetched instructions are stored in instruction Queue.

Segment Registers:-

=> There are 4 types of segment register in 8086.

* CS - Code Segment
* DS - Data segment
* SS - Stack segment
* ES - Extra Segment.

=> Segment registers are 16 bit registers.

i) CS - Code Segment:-

=> The code segment holds the program code.

=> It is used to store instruction of a program.

ii) DS - Data Segment:-

=> The data segment holds the data needed by the program.

iii) SS - Stack Segment:-

=> Stack segment holds the stack information.

=> The stack works on the principle of LIFO [Last-in-First-Out].

iv) ES - Extra Segment:-

=> It holds the string information which can be operated using special instructions.

=> Stack segment is used in a program where microprocessor branches to a subroutine.

Instruction pointer [IP]:-

=> IP is a 16 bit register.

=> The value contained in IP register is knows as offset.

=> The IP is used to hold the address of next instruction to be executed.

Bus Control Logic:-

=> The control unit generates all the bus controlled signals such as memory read, memory write, I/O read and I/O write.

EU → EXECUTION UNIT

⇒ Execution unit works in ~~path~~ parallel with BIU.

⇒ The BIU fetches the information while the EU executing the instructions.

⇒ The EU perform 4 operations.

* Fetch
* Decode
* Execute
* Write

General Purpose Registers :-

8086 has 8 general purpose registers.

AH
BH
CH
DH
AL
BL
CL
DL

H ⇒ Higher order
L ⇒ Lower order.

All the registers are 8-bit registers.

It stores 8-bit data.

Ex:- 95 ⇒ 10010101

Special Purpose Registers :-

All the special purpose registers are 16-bit registers.

It is used to store 16-bit data.

Ax is a accumalator register.

Pointer Register:-

8086 has two types of pointers.

SP and BP

SP - stack pointer

BP - Base pointer.

The pointer registers are 16-bit registers.

The pointer register is used for string the offset address.

* pointer register are used for computing the memory address.

* Stack works on the principle of LIFO.

Index Registers:-

8086 has 2 types of Index Registers.

1. DI - Destination Index

2. SI - Source Index

Index registers are 16-bit registers.

Index registers are used in array processing and string operation.

Index registers are used for computing the memory address.

ALU = Arithmetic Logic Unit.

It performs 8 bit and 16 bit arithmetic operations such as addition, substraction, multiplication, division etc..... and logical operations such as logical and, OR, NOT, Ex-NOR

=> It also perform register Increment, decrement and shift operations.

Flag Registers:-

Flag format

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	0
X	X	X	X	O	D	I	T	S	Z	X	AC	X	P	X	C

O - overflow flag
D - Direction flag
I - Interrupt flag
T - Trap flag
S - Sign flag
Z - Zero flag
AC - Auxiliary carry flag
P - Parity flag
CY - Carry flag
X - NOT used

=> flag is a 16 bit register

=> 8086 has 9 flags remaining are not used

=> They are 6 status flags.

=>

Z - Zero flag
S - Sign flag
C - carry flag
AC - Auxiliary flag
P - parity flag
O - Overflow flag.

⇒ 8086 has 3 control flag.

T - Trap flag
I - Interrupt flag
D - Direction flag.

⇒ flag is nothing but a flipflop. It indicates some condition produced by the execution of instruction is set (1) (or) reset (0)

* Describe how 8086 calculates memory address? (or) calculation of physical address (PA)

* Segment registers are used to generate 20 bit physical address.
* The 20-bit physical address is generated by BIU.

⇒ The value Contain in IP is called offset (or) effective.

⇒ The segment register always start with memory location 0H (zero H) as LSB bit.

=> The 20bit physical address and segment is calculated by combining the offset address and segment value.

=> physical address = effective address + Segment value

Ex:- The CS register has 7234 H and IP has 5432H. find the physical address.

Ans

PA = EA + SV

EA = 5432H

Segment value = 7234 => 72340 H

7234 H => is segment base value

72340 H => is segment starting Address.

$$\begin{array}{r} PA = 72340\ H \\ 5432\ H \\ \hline 77772\ H \end{array}$$

Define Bus Cycle (or) Machine Cycle?

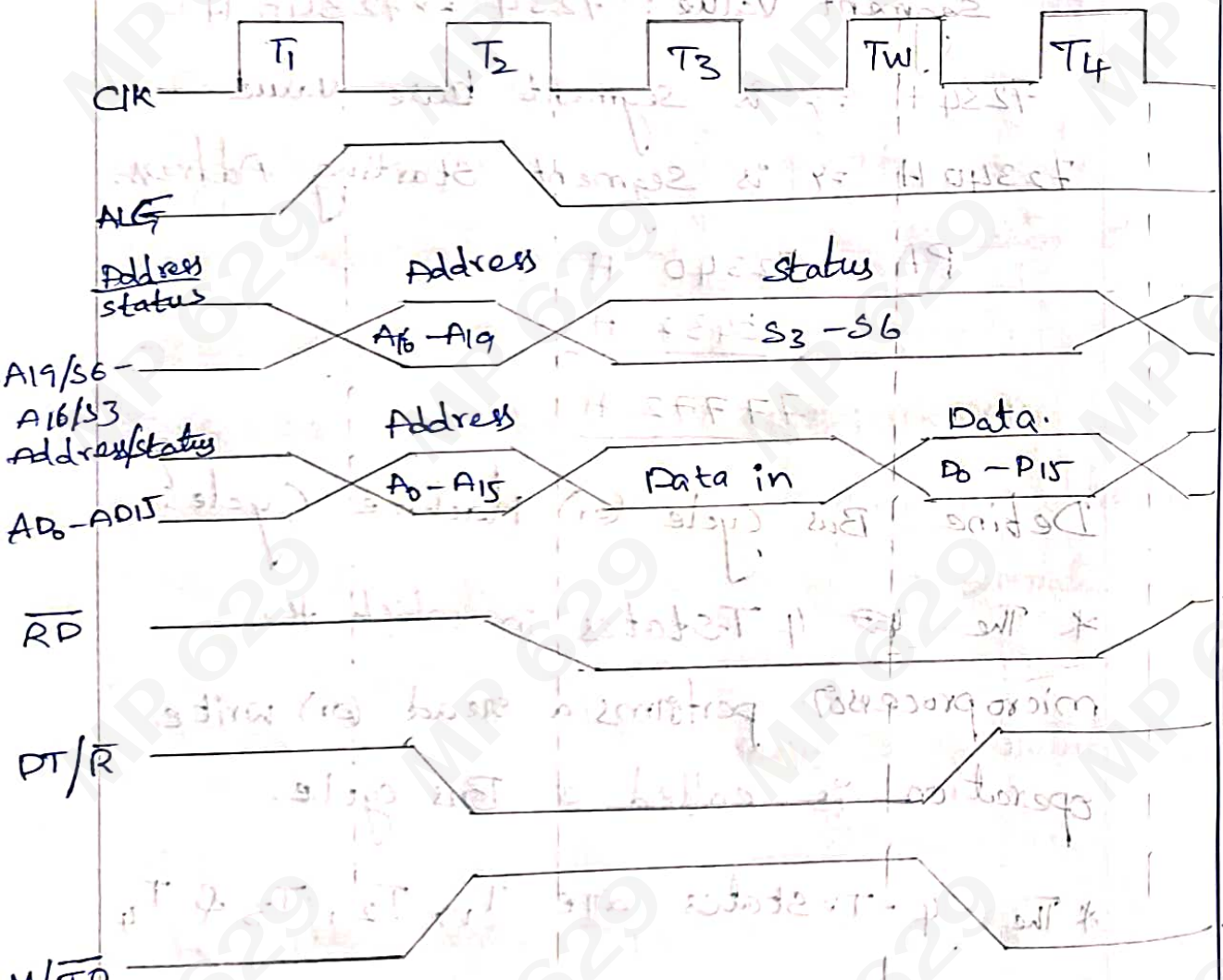
* The 4 T-states in which the microprocessor performs a read (or) write operation is called a Bus cycle.

* The 4-T-states are T_1, T_2, T_3 & T_4

* The 8086 generally needs 4-T-states to perform read / write operation.

* The Bus cycle can be classified as
  * memory read cycle.
  * Memory write cycle
  * I/o write cycle
  * I/o read cycle.
* 8086 also needs wait state (TW)

Draw the timing Daigram of Memory read cycle?



What is segment memory.

The 8086 microprocessor uses memory segmentation. The 1MB memory is divided in to 4 segments. Each segment is 64kB. Each segment registers has 16-bit.

If 'CS' register has CS = 7234 and IP register has IP = 541C? Calculate the PA (Physical Address)?

Sol PA = EA + SV

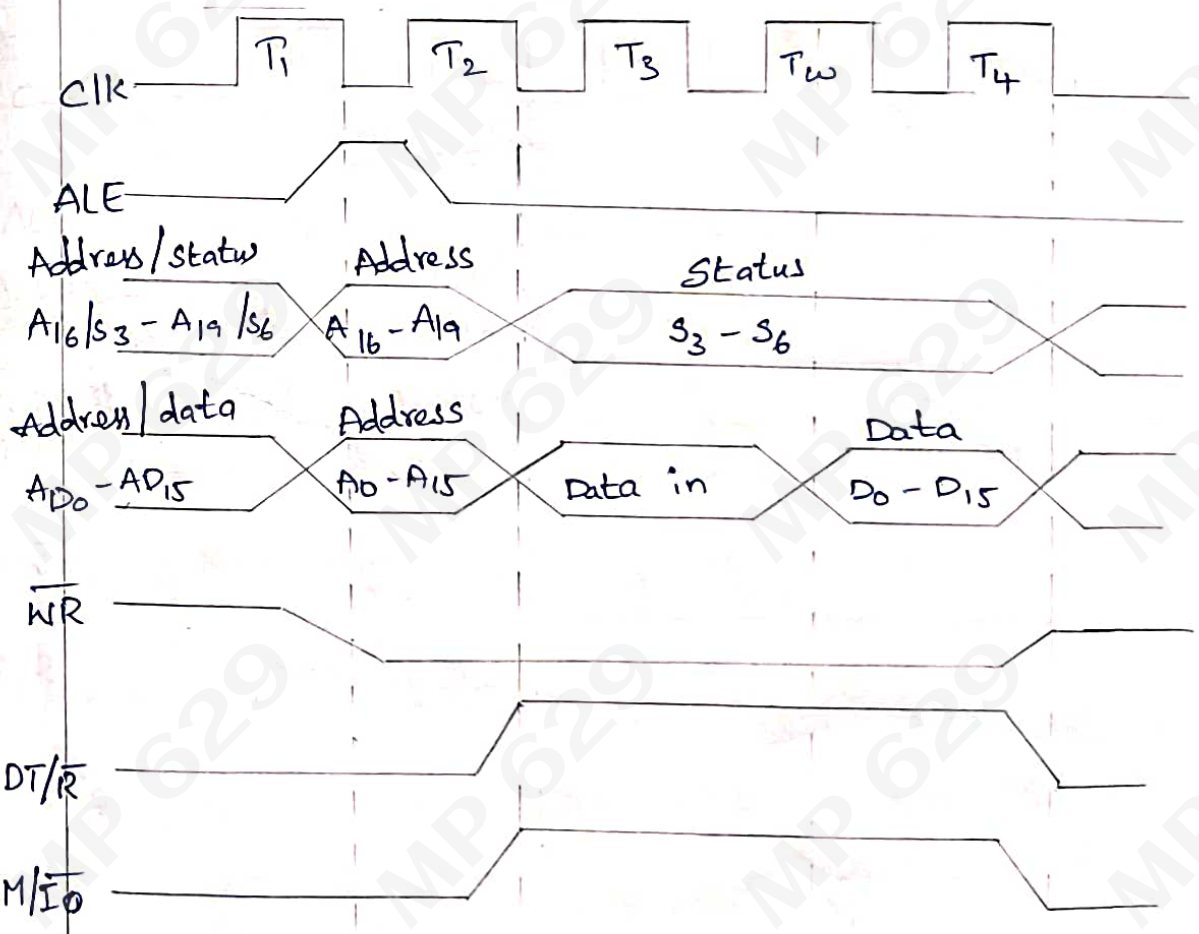
EA = 541C H

SV = 7234 H

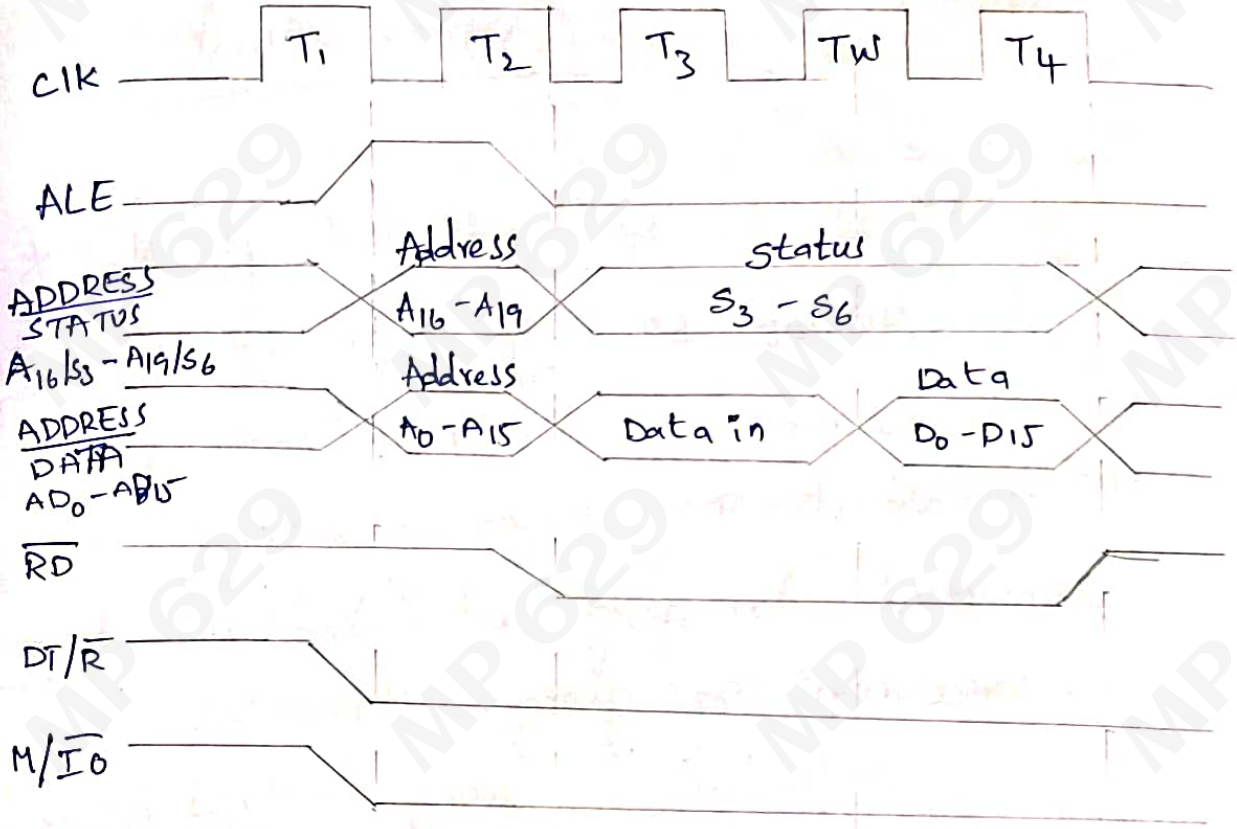
Starting address of SV = 72340 H

$$\begin{array}{r} 72340\ H \\ 541C\ H \\ \hline 7775C\ H \end{array}$$

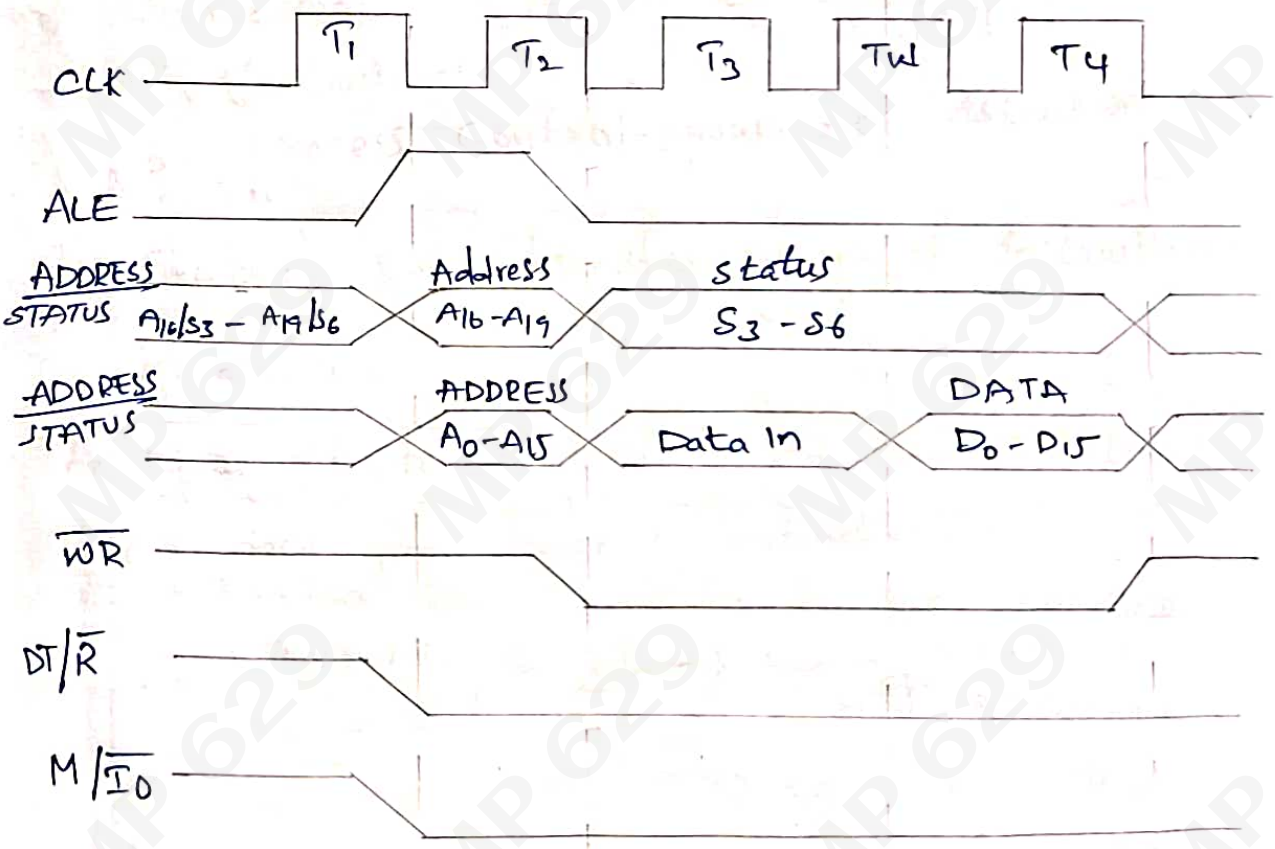
Draw the timing Diagram of memory write cycle?



Draw the timing Diagram of I/o read cycle.



Draw the timing Diagram of I/o write cycle?



CHAPTER 2

Instruction Set of 8086

* Classify instruction set of 8086? (or) List the major group of instruction in 8086?

The 8086 instruction set can be classified into the following major group of instructions.

1. Data transfer group of instructions
2. Arithmetic group of instructions.
3. Logical group of instructions.
4. String Manipulation group of instructions.
5. Control transfer (or) Branching group of instructions.
6. Process control group of instructions.

2. Explain Data transfer group of instructions in 8086?

A. Data transfer instructions are used to transfer the data from source to destination.

i.e. R → ML ML → Memory location
ML → R R → Register
R → R

transfer immediate data into register.

Byte or Word transfer Instructions :-

1 **Move a byte or Word From Source to destination**

The type of Instruction is MOV.

Ex :-

```
MOV   AX, 1234 H      [Immediate Data transfer]
MOV   AL, 12 H
MOV   AH, 34H
```

The 16-bit data 1234 H is moved in to Ax Register

```
MOV   CX, DX      [Register - Register Data transfer]
MOV   AX, BX
```

The data in Dx Register is moved in cx Register.

```
MOV   AX, [SI]      [] → Memory location
```

The content (data) of SI will be moved in to Ax register.

2. PUSH [Push to stack]

Push is specified byte (or) word to top of the stack. When push instruction is executed the SP (Stack Pointer) is decremented by 2

Ex :- PUSH Ax = The data of Ax register is moved in to stack

PUSH AX

PUSH [3000H] :- The content of location 3000H are pushed in to the stack.

3. POP :- [POP From Stack]

POP a specified byte or) word from top of the stack. When pop instruction is executed the SP is incremented by 2.

Ex :- POP AX
POP [3000H]

4 XCHG :- [Exchange]

This instruction exchanges the content of specified source and destination operands (Data).

Ex :- XCHG CX

Exchanges data between AX and CX registers.

5. XLAT (XLATB) ⟶ [Translate a byte]

Translate a byte in AC Register using a label in memory.

6. IN :- [Input the port]

This instruction is used for reading an Input port.

Ex :- IN AX, Port Address

Reads the data from a port and store the result in AX Register.

7 OUT :- [output to port]
This instruction is used for writing from a output port:
Ex:- OUT Portaddress, AX
The data presented in AX register is transferd to specified port address.

8. LEA [Load Effective Address]
The instruction loads the offset of an operand in the specified address.
Ex:-
LEA BX, 1234 H
The BX register is loaded with the effective address of memory location specified by the instruction.

9. LDS/LES [Load Register and DS Register/ Load Register and ES Register].
This instruction copies the content of memory locations to a specified 16 bit register and the next consective two memory locations in to DS. (when LDS area) or ES (when LES) registers.

10. LAHF [Load AH from flags].
This instruction is useful while we want to know the status of flag registers. This instruction loads the AL registers with the

lower byte of the flag registers.

Ex:- LAHF

11. SAHF [Store AH in Flag register].

The instruction copies the content of AH into the lower byte to 0 the flag register depending upon the corresponding bit positions in AH the flags states will be modified. But over flow flag is not altered.

Ex:- SAHF

12. PUSHF [Push flag Register].

The instruction decrements the stack pointers by two and copies the contents of flag register in to the stack.

Ex:- PUSHF

13. POPF [POP register or Memory]

This instruction will copy the word from stack pointed to by the stack pointer to the flag register.

Ex:- POPF.

Explain arithmetic group of instructions?

The arithmetic operations such as addition, substraction, multiplication, division, increment, Decrement, compare, complement etc.....

Addition Instruction:-

For addition operation the instruction used is ADD

Ex:- ADD AX, BX

AX and BX register contents are added and the result is stored in AX.

2. ADC (Addition With Carry)

Ex:- ADC AX, BX

AX and BX register contents are added and AX and BX carrys also included in result.

3. INC Increment

This instruction adds one to the specified register.

Ex:- INC CX

4. AAA → [ASCII Adjust After Addition]

5. DAA → [Decimal Adjust After Addition]

SUB

Substraction operation is used the instruction used is SUB.

Ex:- SUB AX, BX

This instruction substracts source from destination and the places the result in destination.

Ex:- SUB AX, BX
↓ D, ↓ S

AX ← (BX - AX)

2. SBB (Substract with Barrow)

Ex:- SBB AX, BX

3. CMP → compare

This instruction compares two specified bytes. Comparing is nothing but substraction. After compare instruction, the flags will be affected.

4. DEC → Decrement

This instruction substract one from the specified byte / register.

Ex:- Dec cx.

5. AAS → [Ascii Adjust After Substractions]

6. DAS → Decimal Adjust After Substractions]

Multiplication Instructions:-

1. MUL:- Multiplication operation is instruction used is MUL. [unsigned multiplication]

Ex:- MUL BX

multiply the contents AX and BX and the result is stored in AX.

2. IMUL:- [signed Multiplication]

Ex:- IMUL CX

3. AAM :- Ascii adjust for multiplication.

Division Instruction.

1. DIV :- For division operation the instruction used is DIV [unsigned Division].

Ex:- DIV BX

This instruction performs divisions of Accumalator by source operand.

Quotient stored in BL

Remainder stored in BH.

2. IDIV :- [signed division]

Ex:- IDIV BX

3. AAD → Ascii Adjust for Division

4. CBW → Convert byte to word.

5 CWB → Convert word to byte

Compliment Instructions :-

NEG → Negation

This instruction is used to perform direct 2's complement.

Ex:-

NEG CL

Explain logical group of Instructions?

A. The logical operation are and, OR, NOT, X-OR, Shift, Rotate, etc.....

Logical Instructions:-

1. AND (Logical AND)

This instruction performs the logical AND of two operands and places the result in the Destination.

Ex:- AND AX, BX

2. OR (Logical OR)

This instruction performs the logical OR of two operands and places the result in the destination.

Ex:- OR AX, BX

3. NOT (Logical NOT)

This instruction performs the complement (or) Invert of a byte.

Ex:- NOT BX, NOT AX

4. X-OR (Logical XOR)

This instruction performs the logical X-OR of two operands and places the result in destination.

Ex:- XOR AX, BX

5. Test:- (Logical compare and set flags)

This instruction performs logical compare & the update flags.

Ex:- Test AL, AL, CL.

Shift Instruction:-

1. SHL → Shift Logical Left
2. SHR → Shift Logical Right

1. **SHL:-**
This instruction shift the operand towards left.

Ex:- SHL AL

2. **SHR:-**
This instruction shift the operand toward Right.

Ex:- SHR AL

Rotate Instruction:-

1. **ROL** (Rotate Left without carry)
This instructions rotates the bytes towards left without carry.

Ex:- ROL AL, CL.

2. **ROR** → (Rotate Right without carry)
This instruction rotates the byte towards right without carry.

Ex:- ROR AL, CL

3. **RCL** (Rotate left with carry).
This instruction rotates the byte towards left with carry.

Ex:- RCL AL, CL

4. **RCR** → Rotate Right with carry
This instruction rotates the byte towards Right with carry. Ex:- RCR AL, CL.

Explain String Manipulation?

A string is a sequence of character (or) series of bytes / words in a sequential memory location.

The 8086 microprocessor supports string instructions such as string moment, comparision, Loading, scanning

The letter S (or) SB (or) SW is used at every end of instruction.

1. Moving A String:-

MOV S / MOV SB / MOV SW from source string to destination string.

2. Comparing a String:-

Cmp S / CMPSB / CMPSW

This instructions compares a source string to destination string.

3. Load a string:-

LODS / LODSB / LODSW

This instruction loads the some ~~thing~~ string to Accumalator.

4. Scan a string:-

SCAS / SCASB / SCASW

This instruction scans a string byte with a byte in the accumalator.

5. Repeat (REP)

This instruction is used to repeat the following instructions until count CX=0.

6. Input String:-

IN s / InsB / InsW

Input string byte from specified port to Accumulator.

7. Out s / Out SB / Out SW

Input string byte from specified port.

* Explain control transfer (or) Branching instructions of 8086?

A. There are two types of branching instructions.

1. Conditional
2. Unconditional

Conditional Branch Instruction:-

It depends on particular condition. Based on condition the control is transfer to another operation / instruction / condition.

1. JC (Jump if carry)

CF = 1

2. JNC (Jump if no Carry)

CF = 0.

3. JZ (Jump if zero).

4. JNZ :- (Jump if No zero.)

5. JO → (Jump if overflow.) OF = 1

6. JNO → (Jump if No overflow.) OF = 0

7. JP → (Jump if parity) PF = 1

8. JNP → (Jump if No parity) PF = 0

9. JS → (Jump if sign) SP = 1

10. JNS → (Jump if no sign) SP = 0

11. JCXZ → Jump if CX = 0.

UNconditional Control transfer :-

unconditional control transfer does not depend on the particular condition.

1. CALL

This instruction is used to call a subroutine or subprogram / procedure. 2 types of CALL.

1. Near CALL
2. Far CALL

2. RET [Return]

This instruction is used to return from Subroutine. 2 types of RETURN.

1. Near RET
2. Far RET

3. JMP [Jump]

Jumped to specified address to get next instruction.

4. LOOP

Loop through a sequence of instructions until cx=0.

5. LOOP z

Loop through a sequence of instructions until cx=1 , cx≠0.

Interrupt Instructions :-

1. INT [Interrupt program execution]

End of program.

2. INTO [Interrupt on overflow]

Interrupt program execution if OF=1

3. IS IRET :- [Return from ISS / ISR]

ISS → Interrupt service subrotine.

ISR → Interrupt service Return.

* Explain processor control instruction of 8086?

A. 2 types of processor control instructions.

1. Flag Manipulation Instructions.
2. External Hardware Instructions (OR) Machine control instructions.

Processor control instructions are used to control the functioning of available hardware inside the ~~following~~ processor chip.

1. Flag Manipulation Instructions:-

1. CLC → Clear carry Flag , CF = 0
2. STC → set carry flag , CF = 1
3. CLD → clear Direction flag , DF = 0
4. STD → set Direction flag , DF = 1
5. CLI → clear Interrupt Flag , IF = 0
6. STI → set Interrupt Flag , IF = 1
7. CMC → Complement carry flag

2. External Hardware Instructions:-

The machine control instructions control the bus usage and execution. They do not require operand.

1. WAIT :- Wait until Signal on the $\overline{TEST}$ pin is slow.
2. HLT :- [Halt the processor] end of program.
3. NOP :- Nooperation.
4. ESC :- [Escape] to external devices.
5. LOCK :- Bus Lock instruction prefix.

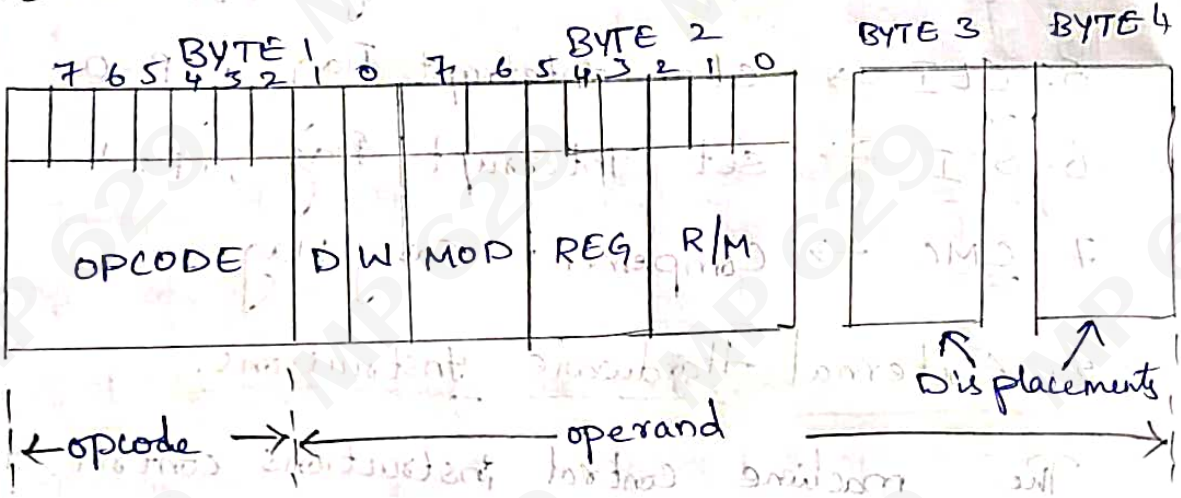
4. Define instruction format?

The way of representing the instruction in a program is known as instruction format. Ex:- ADD AX, BX
MOV AX, 1234H

Instruction

opcode	operand
MOV	AX, 1234
ADD	AX, BX

* Draw the instruction format of 8086?



Instruction format of 8086.

* The instruction format is used for generating opcodes (machine codes) for the instruction.

* opcode :-

It is the main part of the instruction that identifies the specific operation.

ADD
SUB
MOV
} mnemonics

* D :- It is a 1-bit field. It identifies direction in data transfer between two registers.

If D=0, it specifies Source register.
If D=1, it specifies Destination register.

*W :- It is a 1-bit field. If w=1, it specifies word (16-bit) operation. If w=0, it specifies byte (8-bit) operation.

*MOD :- It is a 2-bit field. It indicates the mode of operation. If MOD = 00, 01, 10, 11 specifies the type of addressing mode.

*REG :- It is a 3-bit field. It ~~indicates the mode of a~~ specifies an 8-bit or 16-bit register. Depending on W-bit.

*R/M :- It is a 3-bit field. 000, 001, 010, 011, 100, 101, 110, 111. It specifies a register (or) Memory location.

Displacement :- If the instruction contains any 8-bit or 16-bit displacements this may be placed in byte-3 and byte-4.

*. List the types of addressing modes are used in 8086?

A. *1. Immediate addressing mode
*2. Register addressing mode.
*3. Register Indirect addressing mode.
*4. Direct addressing mode
*5. Based addressing with displacement
*6. Indexed addressing with displacement
*7. Based Indexed addressing mode.

* 8. Based indexed addressing with displacement.
* 9. I/O port addressing mode.
* 10. Implicit addressing mode.

* Define addressing mode?

The way of representing the "operand" in the instruction is known as addressing mode. The way of representing the operand in the instruction.

* Explain the different types of addressing mode of 8086?

A. 1. Immediate addressing mode:-

The immediate data 8-bit (or) 16-bit is moved in to register.

Ex:- MOV AX, 23H
MOV AL, 1234H

The data 23H is source operand.
The Ax, is the destination operand.
The data 1234H is source operand
The AL, is the destination operand.

2. Register Addressing Mode:-

In this mode both source and destination operands are specified in a register.

Ex:- MOV BX, CX.
ADD AX, BX
Bx is the destination operand
CX is the source operand.

3. Register Indirect Addressing Mode:-

Ex:- MOV AX, [BX]

MOV BX, [SI]

In this mode the whose address specified in Bx register of data moved into ax register.

* The data of a memory location where the 16-bit effective address is provided in the register Bx is tranferred to ax register.

4. Direct Addressing Mode:-

Ex:- MOV AX, [1200H]

In this mode the 16-bit effective address [EA] 1200H is moved to AX Register.

* The 16-bit Address 1200H is directly specified in a instruction.

5. Based addressing with displacement:-

Ex:- MOV AX, 1234H [BX] BX → Base Register

MOV AX, 1234H [BP] BP → Base pointer.

1234H is a 16-bit displacement. In this mode the effective address is provided by adding the contents of Bx and 16-bit displacement 1234H, is moved to AX register.

6. Indexed Addressing with Displacement

Ex:- MOV AX, 1234H [SI]

MOV AX, 1234H [DI]

1234H is a 16-bit displacement. In this mode the effective address is provided by adding the

contents of SI and 16-bit displacement 1234H, is moved to Ax register.

7. Based Indexed Addressing:-

Ex:- MOV Ax, [Bx] [SI]

MOV AX, [BP] [DI]

In this mode the 16-bit effective address is provided by adding the contents of Bx and SI are moved to AX register.

8. Based indexed addressing with Displacement:-

Ex:- MOV AX, 1234H [Bx] [SI]

MOV AX, 1234H [BP] [DI]

In this mode the 16-bit effective address is provided by adding the contents of Bx and SI and 16-bit displacement 1234H are moved to Ax register.

9 I/O port Addressing mode:-

IN AL, 13H

OUT 13H, AL

The content of Input port whose address 13H is moved to AL. The contents of AL are moved to output port whose address is BH.

10. Implicit Addressing Mode:-

In this mode the instruction contains only the opcode but not the operand.

Ex:- INT, HLT, NOP, CMC, STC.

NOTE:- ALP can be implemented by the following methods.

1. By using system:-

By using s/w

MASM
TASM } → softwares

ASM → Assembler

MASM → Microsoft Assembler

TASM → Turbo Assembler.

2. By using Debug Method:- (without softwares)

3. ~~Microsoft~~

3 Microprocessor kit

1. By using coding
2. Tracing Method.

* List the Assembly language development tools?
(or)
List the softwares tools?

A. The following are the various types assembly language development tools:-

1. Editor
2. Assembler
3. Debugger
4. Locator
5. Linker
6. Library builder
7. compiler.

* Explain the different Assembly language development tools?

A. ① Editor :- Editor is a program which allows to create a file called source file and it is saved with file extension .ASM.

* The main function of the editor is to help the user to construct ALP in the right format.
* Editor is used to allow the user to type (or) enter and modify the program.

② Assembler :-

The following are the different types of Assemblers.

1. One pass Assembler
2. Two pass Assembler.
3. Macro Assembler
4. Cross Assembler
5. Resident Assembler.

* Assembler is a program which is used to translate Assembly language instructions into the corresponding machine codes.
* Assembler creates two files.
  1. object file [.obj]
  2. Assembler listfile [.LST]

Ex :- 1. MASM
2. TASM.

3. Locator :- Locator is a program used to assign the specific address of where the segments of object code is to be loaded in to the memory.

4. Linker :- Linker is a program which is used to combine many object files into an object file.

* The linker generates a link file and it is saved with file exetension. .EXE.

5. Debugger :- Bug = Error

Debugger is a program which allows to execute and debug the program.

* The process of locating and correcting the errors in a program is known as debugging.

* The errors are eliminated by using single step method (or) break point method in a program.

Assembler Directives :-

1. Directives are used to specify the start and end of the program.

2. Directives are used to intialize variable, constants values etc....

3. Only the instruction are converted in to machine codes. but not the directives. Hence the directives are called Pseudo instructions.

4. The Assembler directives are classified into the following :-
   1. Data definition directives.
   2. Program organisation directives
   3. Alignment Directives.
   4. Programs termination Directives.

Data Definition Directives:-

Assembler directive	Function:
DB	Define Byte used to define byte type variable.
DW	Define word used to define word type variable
DD	Define Double word (32-bit) used to define Double word type variable.
DQ	Define Quad word (64-bit) used to define Quad word type variable.
DT	Define Ten bytes used to define ten byte type of variable.

2. Program Organisation Directives:-

Assembler Directive	Function:
ASSUME	This directive specifies the name of each segment to the Assembler.
SEGMENT	This directive defines the start of a memory segment.

3 Alignment Directives:-

Assembler Directive:	function.
EVEN	This directive informs the assembler to align the data array starting from even address.
ORG origin	This directive is used to assign the starting address for a program module.
MACRO	This directive defines the name, Parameters and start of the macro.
PROC	This directive defines the start of the procedure.

4. Program termination directives

Assembler Directive	function.
ENDM	End of Macro
ENDP	End of procedure.
ENDS	End of memory segment.
END	End of program. This directive informs to the assembler that it is end of program module.

5. Operators.

Assembler Directive	function.
Stack	Indicates, that is a stack segment. It is written in a program .stack.
DATA	Indicates, that is a DATA segment. It is written in a program .DATA
CODE	Indicates, that is a CODE segment. It is written in a program .CODE
MODEL	It is available in model version Ex:- 4.0, 4.1, 4.2, 4.9 etc---
PUBLIC	used to declare variables as common to various program modules.

3. Interrupts And Assembly Language Programming

1. Classify the different types of interrupts in 8086 MP?
   1. External and internal interrupts
   2. Hardware and Software interrupts
   3. Maskable and Non maskable interrupts.
   4. vectored and Non vectored interrupts
   5. Intel predefined interrupts
2. Define the term Interrupt?
   Interrupt is a signal generated with in the MP or external I/O devices.
   => The process of interrupting the normal sequence of program to carry out a specific task are referred to as interrupt.
3. State the need for interrupt?
   Interrupt is a signal that breaks the normal sequence of execution of program.
   * Sometimes it is necessary to have the computer automatically execute one of a collection of subroutine, whenever certain conditions exists with in a program.

* After executing ISR the control is transfer back to the main program.

Note:- subroutine
(or)
sub program
(or)
small program
(or)
procedure.

4. Explain the Interrupts of 8086 MP?

External and internal Interrupts:-

* The interrupts caused by an external phesripherals are called external interrupt
* The interrupt caused by with in the MP are called internal interrupt.

Hardware & Software Interrupt:-

The interrupt caused by external hardware is refered to as hardware interrupt.

* The hardware interrupts of 8086 are NMI, INTR
* During the execution of the program the MP, is interrupted by inserting a special instruction is called a software interrupt.

Ex:- HLT
END
INT
INTO
NOD

8086 has 256 software interrupts.

INTO 0 to INT(FF)
255

Maskable & Non Maskable Interrupt:-

* The MP has the facility for accept (or) reject the interrupt.
* The processor has to accept the interrupt is called Non-Maskable interrupt.

Ex:- NMI

* The program processor has to reject the interrupts is called maskable interrupt.

Ex:- INTR

* In 8086 all 256 (int 00 to INT FF) software.

Vectored and Non Vectored Interrupt:-

In 8086 all the interrupts are vectored interrupts. When an interrupt signal is accepted is accepted by the up if the program contrd automatically brances to a specific address called the vectored address. them the interrupt is called Vectored Interrupt.

The vector address is predefined by the manufacturers of processors.

In Non vectored interrupts, the interrupt device should supply the address of the ISR to be executed in response to the interrupt.

INTEL Predefined Interrupts:-

8086 has 5 predefined interrupts.

1. Type 0 Interrupt (Division by zero error)
2. Type 1 Interrupt (single step)
3. Type 2 Interrupt (Non-Maskable Interrupt)
4. Type 3 Interrupt (Break point Interrupt)
5. Type 4 Interrupt (Interrupt on overflow)

Type 0 Interrupt:-

Type 0 interrupt is divide by zero interrupt. When the Quotient result of division operation is too large to fit in the destination register. The 8086 performs type 0 Interrupt.

Ex:- INT 0

Type 1 Interrupt:-

Type 1 interrupt is single step interrupt.

Single step control is used for debugging.

In this the processor executes one instruction and then after that the contents of registers and memory locations can be examined to see the result.

Ex:- INT 1

Type 2 Interrupt:-

Type 2 is the Non maskable interrupt.

* When an interrupt is received on the pin NMI of the µp (microprocessor).

Ex:- INT 2

Typ 3 Interrupt:-

Type 3 is a break point is inserted in a program for debugging.

The program is executed upto the break point.

Ex:- INT 3.

Type 4 Interrupt:-

Type 4 is an over flow interrupt.

* If the overblow flag is set (OF=1) this interrupt occurs.

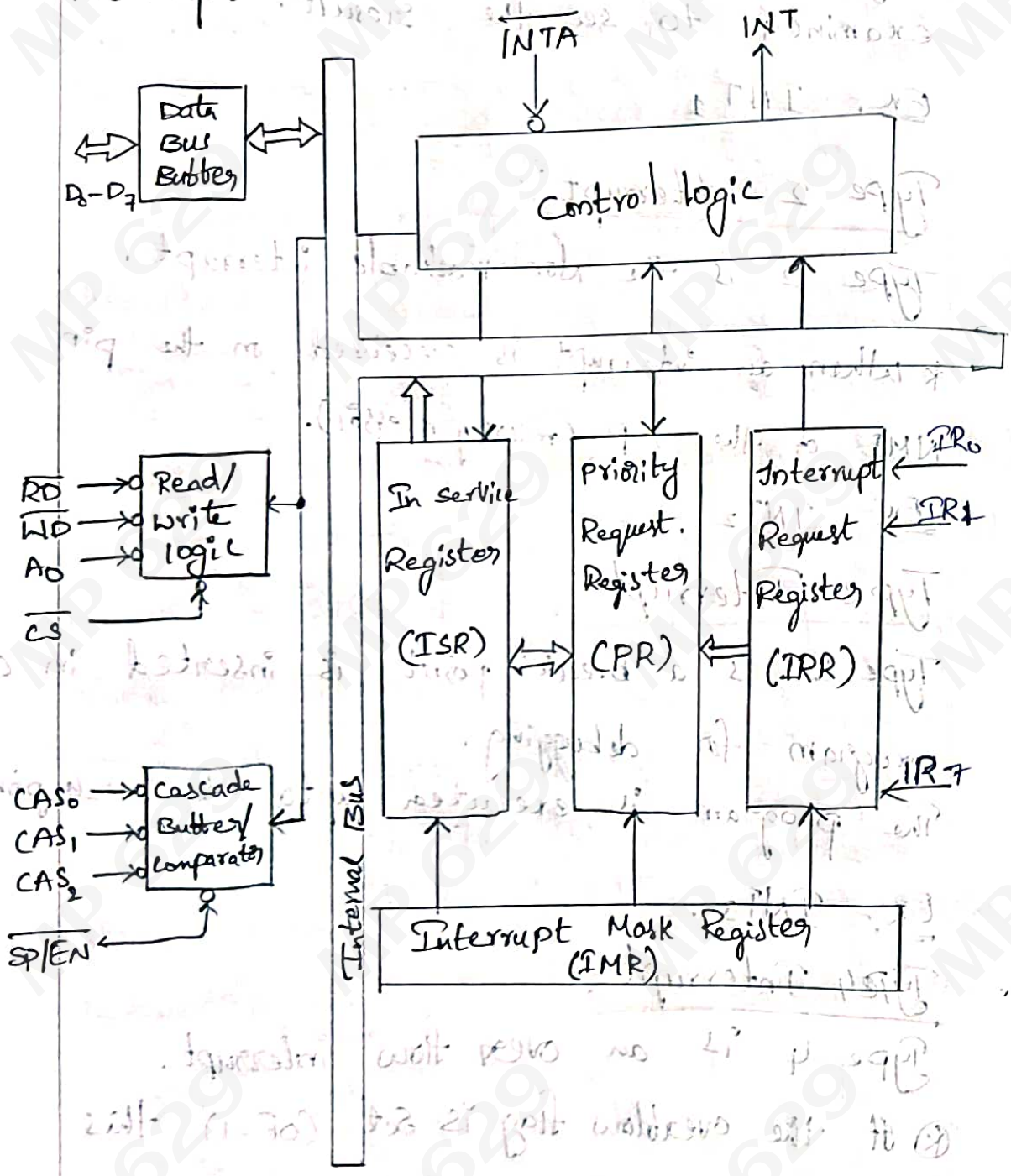
Ex:- INT 4

INTO.

PIC - programmable Interrupt Controller.

[8259] → Pin Number.

* Draw the functional block diagram of PIC (8259) (8259) programmable Interrupt Controller.



1. Data Bus Buffer:-

→ It is a bidirectional data-bus buffer.

⇒ The Singhals $D_0 - D_7$ is connected with 8086 up data lines.

2. Read/write logic:-

* This block contains the signals $\overline{RD}$, $\overline{WD}$, A0, $\overline{CS}$
* The 8259 is selected by $\overline{CS}$ ($\overline{chipselect}$)

3. Cascade Buffer/comparator:-

* This block contains 3 I/O pins CAS_0, CAS_1, CAS_2
* These 3 pins are o/p's when the 8259 is used as Master.
* These 3 pins are i/p's when the 8259 is used as slave.

4. Control logic:-

* This block
* It consists of two signals $\overline{INTA}$ and INT.
* Which is connected to 8086 INTR Pin

5. IRR (Interrupt Request Register):-

* It receives 8 interrupt signals through the lines $IR_0 - IR_7$
* If IR_0 has the highest priority means IR_7 is the lowest Priority.

6. PR (Priority Request Register):-

* It is used to store the highest priority interrupt & Then next.

7. ISR (In Service Register:-

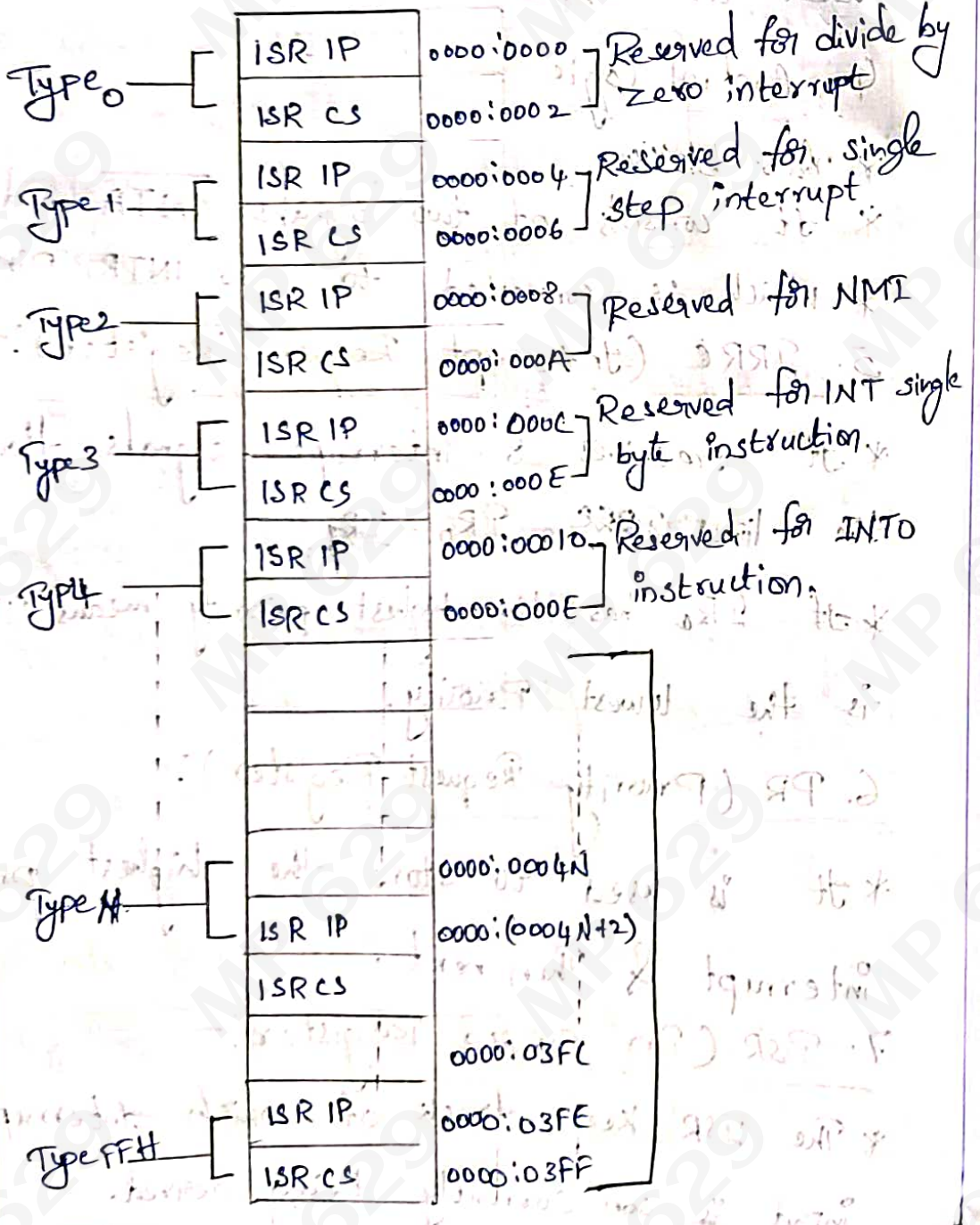
* The ISR keep track of which interrupt input is currently being served.

8. IMR (Interrupt Mask Register)

* It is used to decide whether the interrupt is masked or not.
* It store the masking bits of the Interrupt.

⇒ 8259A is a pic used to interface with 8086 µP.

* Explain interrupt vector tables or Explain Implementing Interrupt?



* The 8086 has 256 types of software interrupts.
* The number of interrupts to be implemented and use in a system depends on system designer.
* The addresses stored in this table are called vector addresses.
* As shown in the table, the starting address of each ISS is stored i.e two bytes for IP and two types for CS.
* The 1 KB memory space is required for this interrupt vector table. ie. from 0000 to 03FF.

* Describe the procedure for executing an ALP (Assembly Language Program) with assembler?

A. We need to check the procedure for execution of MASM/TASM assembler in your system. The following process is to be noted.

Steps:-

1. Click "start" button.
2. Select "run" option.
3. Type "command"
4. Then we will enter into MS-DOS prompt.
5. We will see c\windows\system.
6. Type cd\
7. We will see c:\>
8. Type CD-MASM
9. We will see c:\MASM

10. Once the Software has been loaded follow the following steps.

11. Type "Edit" command" to open.

12. Type CD:\MASM> edit.

13. Once the program is being enter save the program with a → filename.ASM

14. After the file has being saved in the DOS-prompt, Then follow the following steps to run your program.

15. Type MASM <filename.ASM>

16. Type CD:\MASM; Link <filename.ASM>

17. If we will see zero warning errors, then the above display.

18. Type DOS-prompt you will get result of the program.

A. Type CD:\MASM > Debug <filename.ASM.exe>

Then will get - symbol below the above command, if the file was correct.

.ASM, .EXE, Link, Debug → Software developmen tools.

Page.....

```
Title.....
DOSSEG
.MODEL
.STACK
.CODE
MAIN PROC
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -

MAIN ENDP
DATA
- - - - - -
- - - - - -

END MAIN
```

→ Program Title

} Actual programming Instruction

} Data used as per the program.

* Illustrate CALL, RETURN instructions and Parameter passing?

CALL :-

⇒ CALL means CALL the subroutine.

* CALL instruction is used to transfer program control to a subroutine (or) procedure.
* The return address is saved on the stack.
* There are two types of CALL:
  1. NEAR CALL
  2. FAR CALL.

NEAR CALL :-

* It refers to a procedure CALL which is in the same code segment as the CALL instruction.

* The address of a near CALL is 2 bytes long.

* FAR CALL:-

* It refers to a procedure CALL which is in the different code segment as the CALL instruction.
* The address of a far CALL is 4 bytes long.

RETURN (RET):-

* Return means Returns from the subroutine
* The RETURN instruction is used to the transfer program control procedure back to the calling or called procedure. (main program)
* Return is also two types:
  1. Near RET
  2. FAR RET

*1. NEAR RET:-

* It means the transfer of control is with in the segment.

*2. FAR RET:-

Transfer of control is in the other segment.

* When we want to call the procedure in the main program. The instruction used is CALL.

* When the after executing subroutine the control is back to the main program. The instruction used is RETURN (RET).

Parameter Passing :—

* How the data will be passed, from main program to subroutine is generally called Parameter Passing
* To pass the information about address, variables or data from main program to subroutine is called parameter passing.
* Passing of parameters is a mechanisim for communication of data and information between the calling and called procedures.
* There are four main ways of passing prameters to and from procedures.

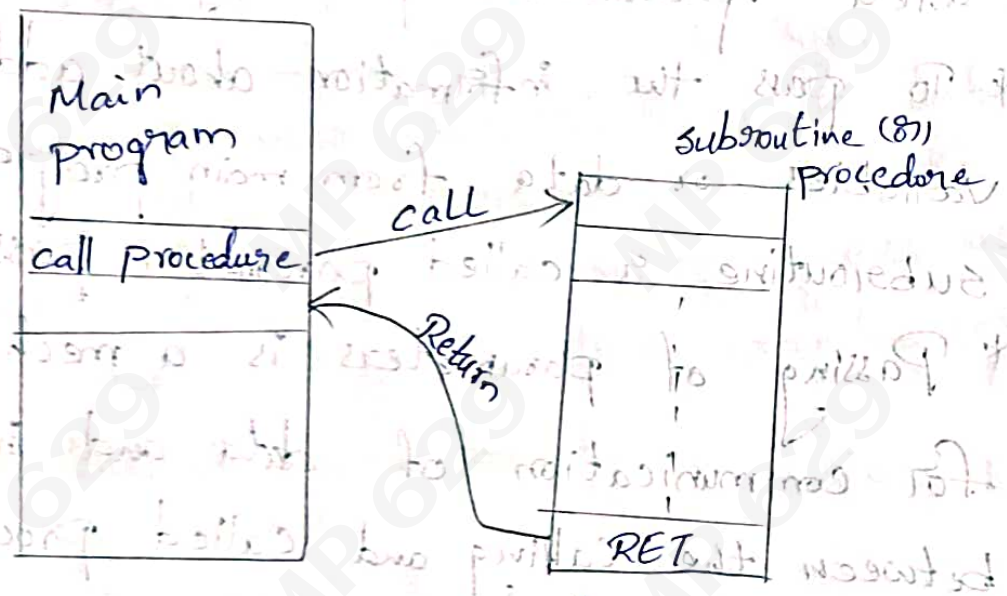
1. Passing parameters using register.
2. Passing parameters using stack
3. Passing parameter using memory address.
4. The parameter that are used to transfer data to a procedure are known as input parameters.

=> The parameters to transfer result from the procedure to the calling procedure

are known as output parameters.

* The parameters used to transfer data in both the directions are called input-output parameters.

* Explain procedure programming in 8086?
(or)
Subroutine programming?



When a ~~prog~~ group of instructions are to be used several times to perform a same function in a program, then we can write them as a seperate sub program called procedure or subroutine. whenever required the procedures can be called in a program using CALL instructions.

The procedures are written and assembled as seperate program modules and stored in memory. When a procedure is called in the main program the program control is transferred

to procedure and after executing the procedure the program control is transferred back to main program. In 8086 processor, the instruction CALL is used to call a procedure in the main program and the instruction, RET is used to return the control to main program.

* 8086 assembler directives related to procedure is PROC.
* PROC :- Beginning to the procedure.
* ENDP :- End of the procedure.
* The usage of procedure provide many advantages. That is program design development and implementation to slove complex programs.
* When a procedure is called the current value of IP is pushed on to the stack.
* When a procedure returns the value of IP is popped from the stack.

Understand the Significance of assembly language program?

A. Assembly Language Program:-

An assembly language is a low-level programming language for computers, microprocessors, micro-controllers, and other integrated circuits. It implements a symbolic representation of the binary machine codes and other constants needed to program a given CPU architecture. This representation is usually defined by the hardware manufacturer, and is based on mnemonics that symbolize processing steps. (instructions), processor registers, memory locations, and other language features.

Assembly language programming requires good knowledge of machine architecture, operating system and programming principles. Assembly language reveals the secret of the computer's hardware and software. The program written in this language are compact and efficient. A program has to be converted to machine code for execution and it is performed by the translator known as "Assembler".

Assembly language, unlike high level languages, is machine dependent. Each microprocessor has its own set of instructions, that it can support. Using with 8086

instructions, an assembly language program can be implemented in the following methods :-

(i) standard coding form.
(ii) using microsoft assembler (MASM)
(iii) using DEBUG program on the IBM PC
(iv) using turbo assembler (TASM)

To implement the assembly language programs on the laboratory microprocessor trainer kits, the 'standard coding form' is only the suitable method to execute the programs. MASM, TASM and DEBUG methods are useful to implement the assembly ~~langua~~ language programs on the IBM PC without using the microprocessor trainer kits. In this chapter let us study the first two methods.

Standard Coding Form method :-

In standard coding form method, the assembly language program is consists of a sequence of instruction will written using labels, mnemonics, symbolic addresses and operands. In 8086 assembly language programming the machine codes should be constructed for each instruction by hand, particularly in this method. This is major disadvantage in this method of programming.

Memory address	Machine code	Label	Mnemonics	operand	Comments

CHAPTER-4

INTEL ADVANCED MICRO PROCESSORS

80x86 families
80186
80286
80386
80486
80586 (or) pentium

* List the features of 80286 processor?

* 80286 is a 16-bit processor.
* It has 24 address lines
* It has 16 data lines
* It support 2^{24} = 16MB of physical memory.
* It support 1GB of virtual memory.
* It provides multiuser and multitasking.
* It requires +5 volts dc supply for operation.
* It operates on 10MHz clock frequency.
* 80286 is 6 times faster than 8086.
* 80286 operates in 2 modes
  1. real address mode
  2. protected virtual address mode.
* 80286 is 68 pin IC package.
* It is introduced in the year 1982.

* List the features of 80386?

A * 80386 is a 32-bit processor introduced in the year of 1985
* It has 32 address lines.
* It has 32 data lines
* It support 2^{32} = 4gb of physical memory.
* It requires +5 volts dc supply for its operation.
* It support 64 TB of virtual memory.
* It operates on 12.5, 16, 20, 25 (or) 33 MHz of clock frequency.
* It is a 132 pin PGA (Pin Grid Array)
* It provides :-
multi tasking
memory management
high speed bus Interface
pipelined Architecture
cache memory in a single IC chip.

* List the features of 80486?

A * 80486 is a 32 bit processor introduced in 1989.
* It has 32-bit Address lines.
* It has 32-bit data lines.
* It requires +5 volts dc supply for its operation.
* It supports 2^{32} = 4GB of physical memory.

* It supports 64TB of virtual memory.
* It is a 168 pin PGA [Pin Grid Array]
* It operates on 25 - 100 MHz of clock frequency
* It provides :—

Multi tasking,
Memory Management,
Highspeed bus interface,
pipelined Architecture,
Cache memory in a single IC chip,
floating point unit.

Pentium processors :—

Pentium-pro
pentium - II
pentium - III
pentium - IV
} Advanced pentium processors

* List the feautures of pentium processors?

A. ① It is labelled as P5 (or) 80586
② pentium processor is a 32-bit processor. Introduced in the year of 1993.
③ It has a 32 address lines.
④ It has a 32-bit - 64 bit data lines.
⑤ It supports 2^{32} = 4GB of physical memory.
⑥ It operates on 100 - 200 MHz clock frequency.
⑦ It is a 237 pin IC package.
⑧ It requires +5 volts dc power supply for internal operation

⑨ It consist of 3.5 million transistors.

⑩ It provides => super scalar Architecture.

=> Two integer pipe line unit

=> Floating point point

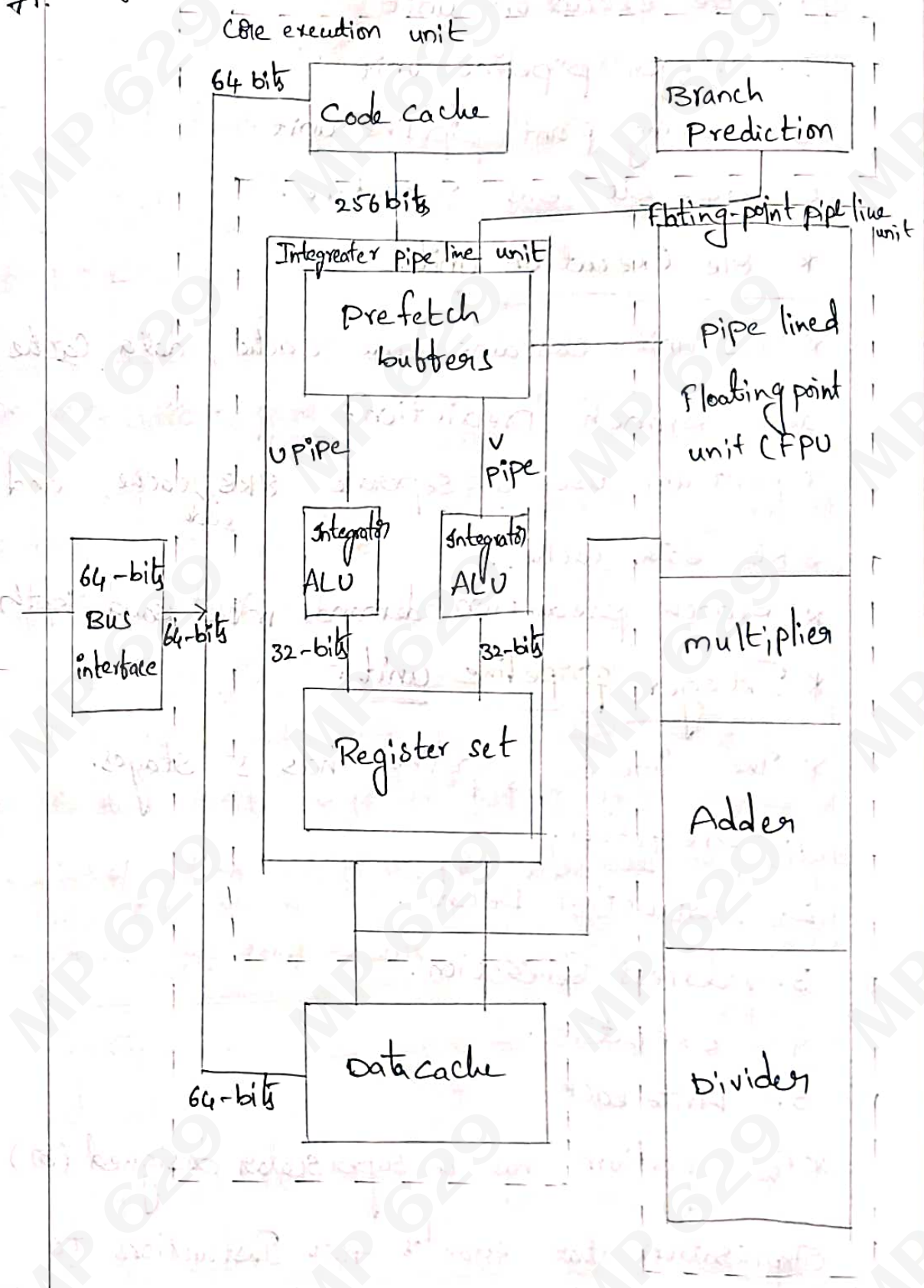
=> Two cache unit in a single IC chip.

* Comprehend (Compare) the features of 80286, 80386 and 80486 processor?

Parameter	80286	80386	80486
1. year	Introduced in the year 1982	Introduced in the year 1985	Introduced in the year 1989
2. No. of bits	it is a 16-bit processor	32-bit processor	32-bit processor.
3. Data lines	16-bit data lines	32-data lines	32-data lines
4. Address lines	24 Address lines	32 Address lines	32. Address lines
5. physical memory	2^{24} = 16MB	2^{32} = 4GB	2^{32} = 4GB
6. No. of pins	68 pins	132 pins	168 pins
7. clock frequency	10 MHz	12.5 to 33 MHz	25-100 MHz
8. power supply	+5 volts dc	+5 volts dc	+5 volts dc.
9. Virtual memory	1GB	64 TB	64 TB
10 Functional units	BIU, EU, IU, AU	BIU, EU, SU, PU, CPU, IDU	It is a combination of 80386 + 8KB of cache memory.

* Explain pentium processor with a block diagram?

A:



Internal block Diagram of Pentium Microprocessor.

The pentium processor contains the following units (or) blocks.

1. Core execution unit
2. Integer pipeline unit
3. Floating point pipeline unit
4. 64-bit bus Interface.

* Core Execution unit:-

* This unit contains code cache, data cache, and Branch prediction.
* Pentium uses a seperate 8KB code cache and 8 KB data cache.
* Branch prediction demands more Bandwidth

* Integer Pipeline unit:-

* The integer pipeline has 5 stages.
  1. pre fetch.
  2. Instruction Decode.
  3. Address Generation.
  4. Execution.
  5. Write back
* The pentium has a superscalar designed (or) Organisation that enables two Instructions to execute in parallel.
* As shown in the figure the pentium has 2 integer pipeline units. They are U and V

pipeline Integers.

* Each pipeline has a seperate ALU's.

* After the execution of the Instructions the result is stored in the registers.

3. Floating Point Pipeline unit:-

* This unit has 8 stages.

1. prefetch
2. Instruction Decode.
3. Address generation.
4. operand fetch
5. First Execute.
6. Second Execute
7. Write float
8. Error Reporting.

* This unit performs floating addition, Subtraction, Multiplication, division, and compare...

4. 64-bit Bus Interface

* This unit is used to interface and external I/o devices with the pentium processor, using 64-bit data bus.

* Question

comprehent and compare the features of advanced pentium processors. (8T) compare features of pentium pro, PII, PIII and PIV.

Parameter	Pentium pro.	Pentium II	Pentium III	Pentum IV
No. of bit	It is a 32-bit Processor	32-bit	32-bit	it is 64-bit Processor.
Year	Introduced in the year 1995	1997	1999	2000
No. of Data lines	64-data lines	64	64	64-bit data lines
No. of Address lines	36-address lines	36	36	36
Power supply	+5V dc	+5V dc	+5V dc	+5V dc
Clock frequency	150-200 MHz	233-450 MHz	500 MHz - 1 GHz	1.4 GHz - 2.6 GHz
No. of pin	387	242	370	478
No. of transistors	5.5 million	7.5 million	9.5 million	42 million Transistors
Physical Memory.	2^{36} = 64GB	64 GB	64 GB	64 GB
Cache memory	256 KB of secondary level cache L2	256 KB of secondary level cache L2	16 KB Instruction cache 16 KB data Cache	8 KB data cache 256 KB of secondary cache.

CHAPTER-5

Architecture of 8051 MICRO CONTROLLER

* Comparision between Microprocessor and Micro controller?

A

Micro processor	Micro Controller.
Single chip cpu is called as microprocessor	A single chip micro computer is called as micro controller
A microprocessor contains ALU, register, clock circuit, control unit and interrupts	A micro controller contains ADC, DAC, ROM, RAM, clock circuit and microprocessor.
Simple architecture	Complex architecture
Ex:- 8086, 80286, 80386, 80486, 80585 or pentium (relatively slower)	Ex:- 8051 faster
Microprocessor based system requires more hardwares	It requires less hardware.
Multifunctional pins are less.	It has more number of multifunctional pins.
used for general purpose applications like used in digital computing system or computers	used for specific purpose applications like used in embedded systems
It requires large number of interfacing devices	It requires less number of interfacing devices.
It has single memory map for data and code memory	It has seperate memory map for data and code memory
It has one or two bit handling instructions	It has many bits handling instructions.

* Define Microcontroller?

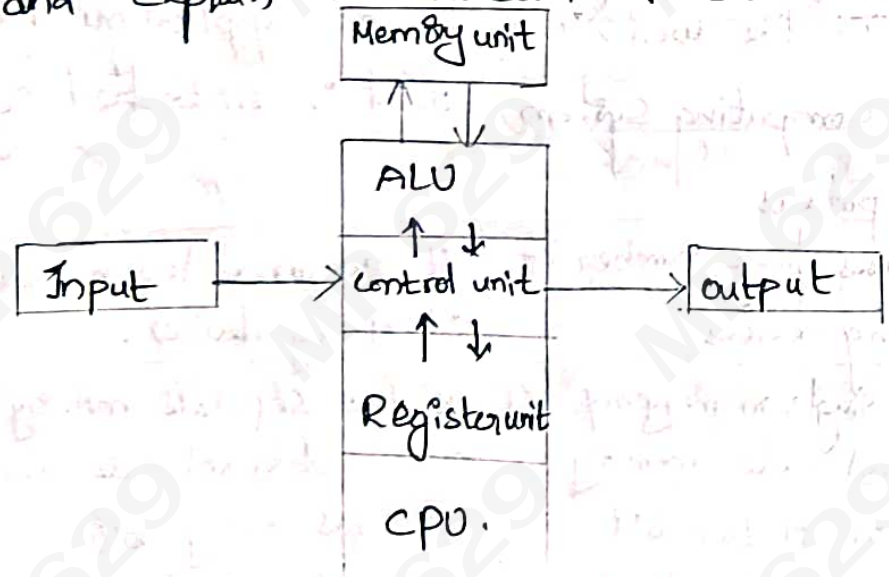
A single chip microcomputer is called as microcontroller, includes a processor, Memory and I/o pheripherals on single chip.

Ex:- Intel 8051 MC(or)MC

* List the features of 8051 MC?

A. 1. Intel 8051 MC is a 8-bit, introduced in the year of 1981.
2. It is a 40-pin Ic chip.
3. It has 4 register banks
4. Data lines 8-bit
5. Address lines -16
6. Timer/counter → 02
7. Internal RAM - 128 bytes.
8. Internal ROM - 4KB
9. It requires +5 volts dc supply for it's operations.
10. It requires the clock frequency is $12MH_z$
11. It has 32 I/o Lines.

* Draw the block diagram of micro computer and explain the function of each unit?



Microprocessor :- Microprocessors performs various operations on digital signals. There are three functional components which forms the cpu of a microcomputer. They are.

1. Register Array
2. Arithmetic logic unit
3. Timing and control unit.

Interface :-

Interface is needed to communicate microprocessor with outside world.

Memory :-

It is used to store instruction and data. The memory block has two sections. ie., ROM, RAM.

Input :-

The Input unit transfers the binary instruction and data from the outside world to the microprocessor.

Output :-

The output unit transfers the processed data from the microprocessor to the output device.

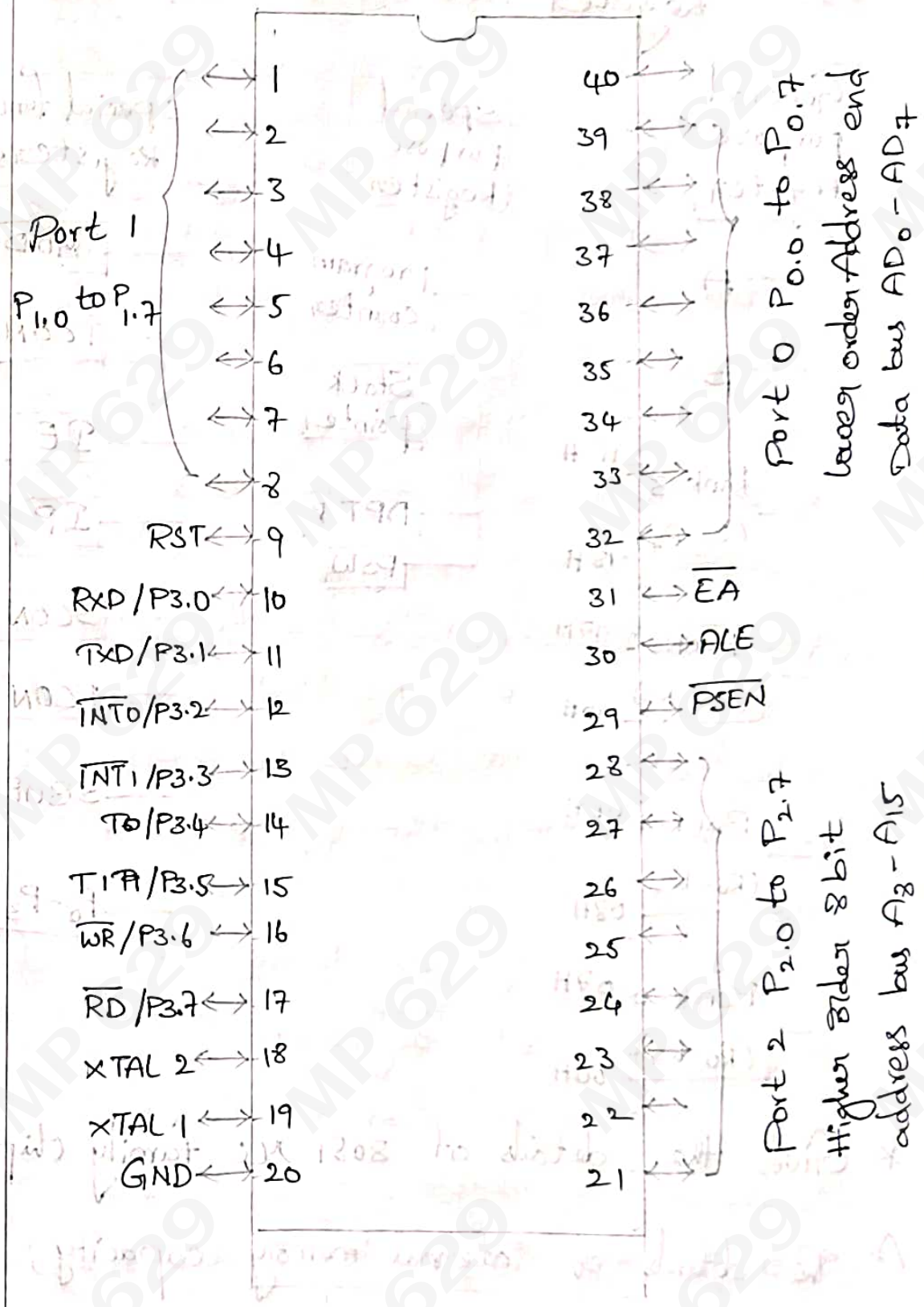
System Bus :-

A group of conducting wires which carries the electrical signals the data. Basically it is a communicating path between the microprocessor and I/o device and memory.

* Draw the block Diagram of 8051 MC?

A.

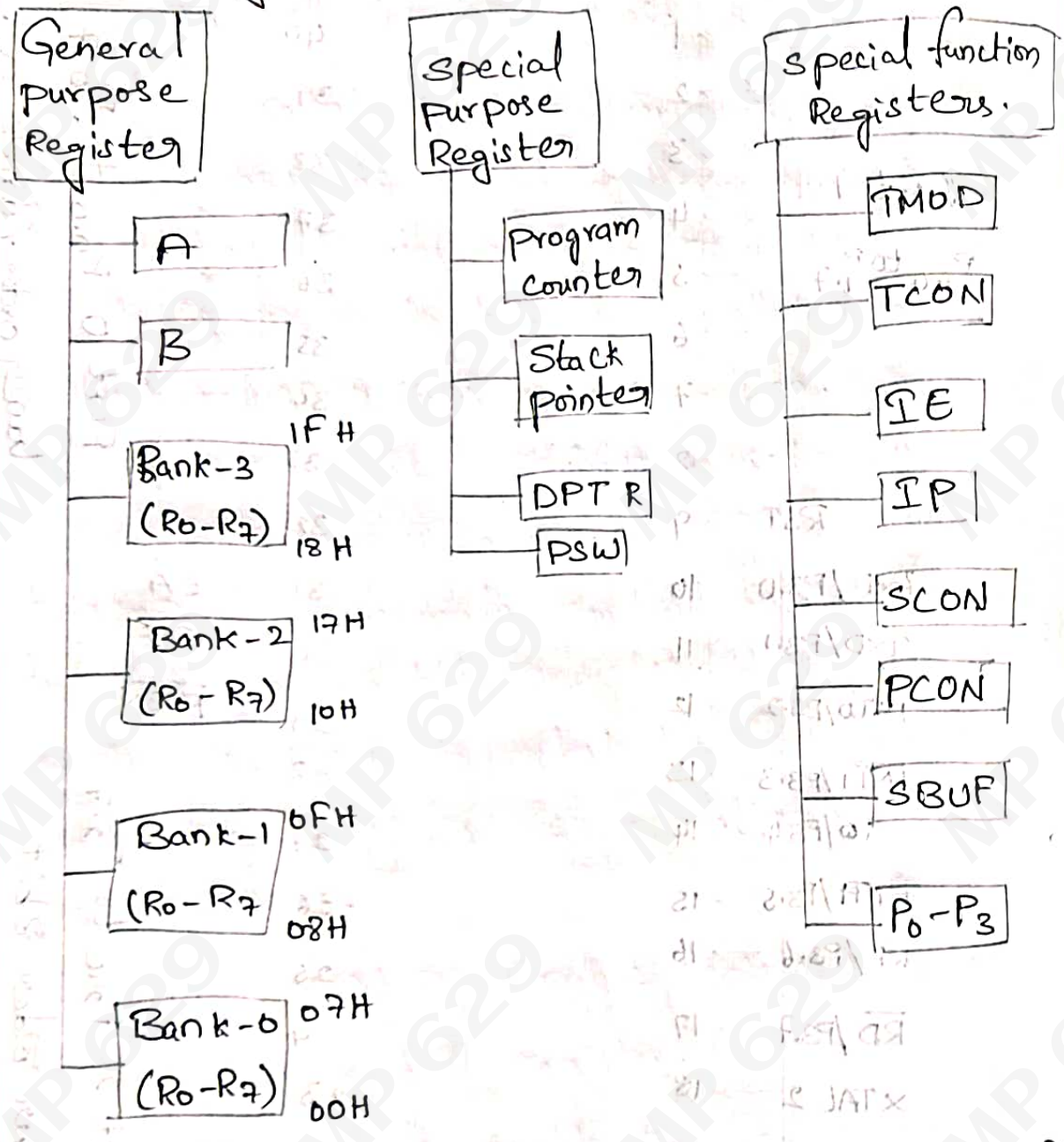
* Draw the pin Diagram of 8051 MC?



* Draw the Register structure of 8051 MC?

A.

Register structure of 8051



* Give the details of 8051 MC family chips?

A. The details of internal memory capacity, no. of timers and Interrupts of various microcontrollers are given below table.

	Device	on-chip Data memory	on-chip program memory	16-bit Timer/counter	No. of vectored Interrupts
1	8031	128 B	None	2	5
2	8032	256	None	2	6
3	8051	RAM 128B	4k ROM	2	5
4	8052	256	8K	3	6
5	8751	128	4k EPROM	2	5
6	8752	256	8 4EPROM	3	6
7	AT 89C51	128	4k Flash memory	2	5
8	AT89C52	256	8k Flash memory	3	6

* Explain the ~~features~~ special function registers in 8051?

A PCON :- (Power CONtrol Register)

* PCON is a 8-bit register.
* It is used for power control and baud rate selection.
* It is a byte addressable register.
* It also consists of general purpose user flag.

SCON :-

* It is a 8-bit register.
* It is serial port control and status register.
* This register contains not only the mode selection bits, but also the specific bit for transmit and receive.
* It has serial port interrupt bits.

TCON :- (Timer/counter control register)

* It is a 8-bit register.
* This register contains the timer control flags and intd interrupts flags.

SBUF :- (Serial Data Buffer register)

* The 8051 has serial data communication circuit that uses register SBNF to hold the data.
* It is used to hold the parallel data during transmission and reception.
* SBUF is an 8bit register used for serial communication.

TMOD :- (Timer mode control Register)

* It is an 8-bit register.
* It is used to select the operating mode and the timer/counter operation of the timers.

IE :- (Interrupt Enable Register)

* It is a 8-bit register.
* It is used to enable (or) disable all interrupts is ~~disable~~ of 8051.
* If the bits are 0, the corresponding interrupt is disable.
* If the bits are 1, the corresponding interrupt is Enable.

IP :- (Interrupt priority register)

* It is an 8-bit register.
* It is used to make the priority of the interrupts of 8051.
* If the bit are 0 :- The corresponding interrupts is lower priority.
* If the bits are 1 :- The corresponding interrupts is higher priority.

* Describe the various parts of 8051?

A * 8051 has 4 ports P_0, P_1, P_2, P_3.
* A total pins of 32 reserved for 4 ports.
* All 4 ports are bidirectional I/o ports ↔

Port-0 :-

* Port-0 can be designated as P0.0 - P0.7
* The internal ram address of port-0 is 80-kB
* It is used as simple inplit / output port.
* The Bidirectional lower order address and data lines are functioning in this port.

i.e ($AD_0 - AD_7$)

Port-1 :-

* Port-1 can be designates as ($P_{1.0} - P_{1-7}$)
* It is used as dedicated I/o port.
* It does not have any alternate function.

* The internal RAM address of Port-1 is _____

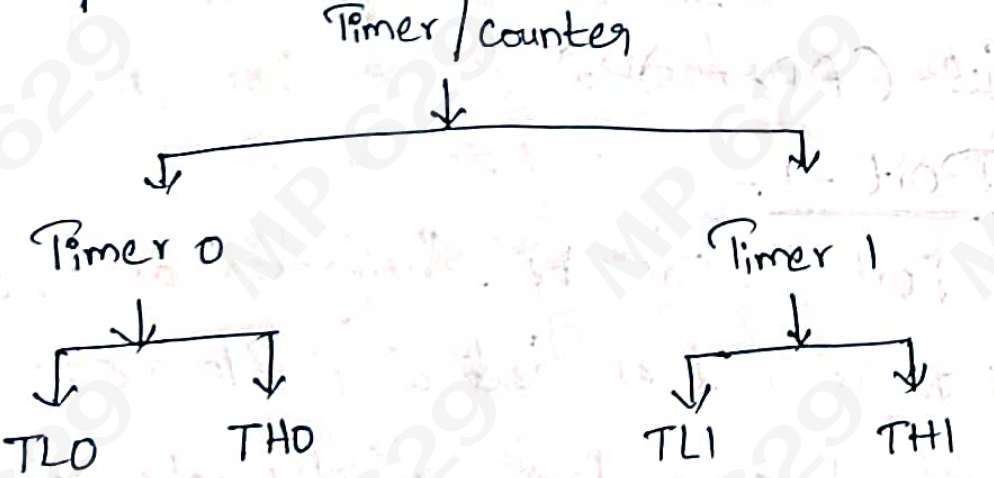
Port-2 :-

* Port-2 can be designated as ($P_{2.0}$ - $P_{2.7}$)
* It can be used as Input/output port.
* The alternate function of Port-2 is _____
* The higher order address lines i.e (A_8 - A_{15})
* The internal RAM address of port is _____

Port-3 :-

* Port-3 can be designated as $P_{3.0}$ - $P_{3.7}$
* It can be used as simple I/O port.
* The alternate functions such as (RxD, TxD, $\overline{INT_0}$, $\overline{INT_1}$, T_0, T_1, $\overline{WR}$, $\overline{RD}$) can be performed.
* The internal RAM address of Port-3 is _____.

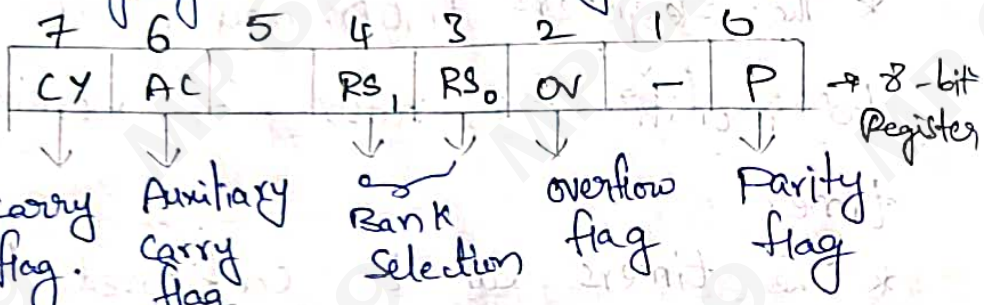
* Explain timer/counters in 8051?



* Counters are used to count the number of events.
* Timers are used to maintain time delays between the actions.
* A counter always counts up.
* 8051 has two 16-bit timer/counter
  Timer-0
  Timer-1
* The registers TMOD and TCON are used to monitor the nature of working of timer.
* The timers can be configured to operate either as timers (or) event counters.
* The ~~count~~ control bit $C/\overline{T}$ is devices the either it is timer operation (or) counter-operation.
* If $C/\overline{T} = 0$, it performs timer operation.
* If $C/\overline{T} = 1$, it performs counter operation
* The 8051 requires 12MHz clock frequency for its operations.

Special purpose Register:-

* PSW (program status word)
* It is called as flag register.
* PSW is a 8-bit register.
* Flag is a condition to repeat the status of the arithmetic logical operation.
* It is going to be hold by a flipflop.



RS_1	RS_0	Register Bank
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3.

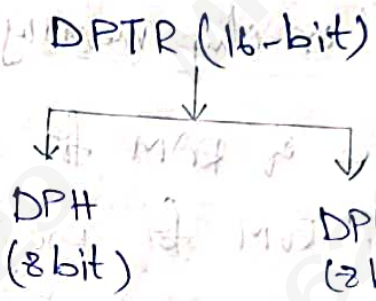
The 5th bit indicates user flag.

* PC (program counter.)
* It is a 16-bit register.
* It is used to hold the address of the next instruction which is to be executed.
* It is automatically incremented to point the next instruction in the program sequence after the execution of the current instruction.

* This is the only register that does not have an internal address.

* DPTR :- (Data Pointer)

* It is a 16-bit register, used to hold the 16-bit address.



* Stack Pointer :-

* It is a 8-bit register which controls a protion of Internal RAM known as STACK.

General Purpose Registers :-

Register A :- A (Accumalator)

* It is a 8 bit register, used for arithmetic and logical operations. After operation the result is stored in accumalator.

* It acts as a source as well as destination.

Register B :- B (Base)

* It is a 8-bit register.

* It is used for multiplication and division operations.

Ex :-

```
MUL BL.   DIV BL.
```

* Describe internal memory of 8051 MC?

A. Memory Organization of 8051

Internal memory (on chip memory):-

* 8051 has 128 bytes of RAM for data memory.
* 8051 has 4KB of ROM for program memory, Inside the IC chip.
* The figure shows the Internal RAM Organization:-

General purpose RAM Memory space	30 to 7F
Bid Addressable Area	20 to 2F
Register Bank$_3$	18 to 1F
Register Bank$_2$	10 to 17
Register Bank$_1$	08 to 0F
Register Bank$_0$	00 to 07

INTERNAL RAM ORGANIZATION

* The 128 bytes of RAM is divided in to three sections.

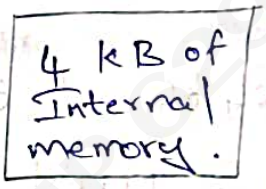
1. Register Banks from 00 to 1FH
2. Bit Addressable Area is from 20 to 2F
3. 30H to 7FH or called general purpose area.

1. Register Bank requires 32 bytes of RAM
2. Bit addressable area requires 16 bytes of RAM.
3. General purpose of 80 bytes of RAM.

32
16
80
122 bytes

Internal ROM:-

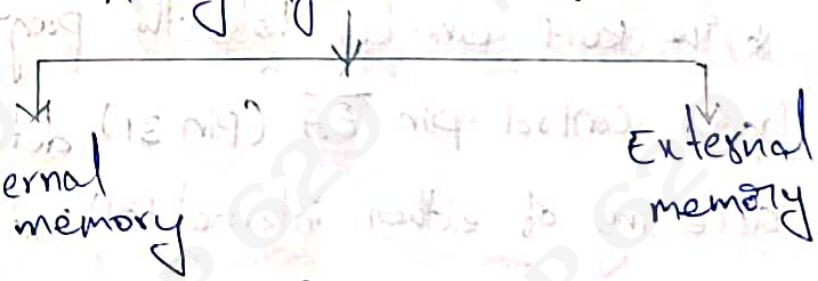
The 4KB of Internal ROM is organized to hold the user application program.



The memory space extends from 0000H to 0FFFH storing the program code.

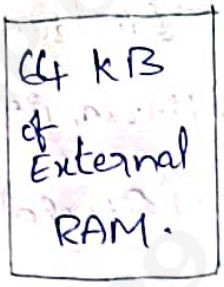
* Describe the External memory organization of 8051 MC?

Memory Organization of 8051



External memory (off chip memory):-

* The 8051 has 64 KB of external RAM.



As shown in the figure the range of external RAM is ~~from~~ from 0000H to FFFFH.

* If more space is required, then the external memory has to be connected.
* RAM can be used to hold data.
* This external is accessed through the use of DPTR.

External ROM:-

The 8051 has 64 KB of External ROM.

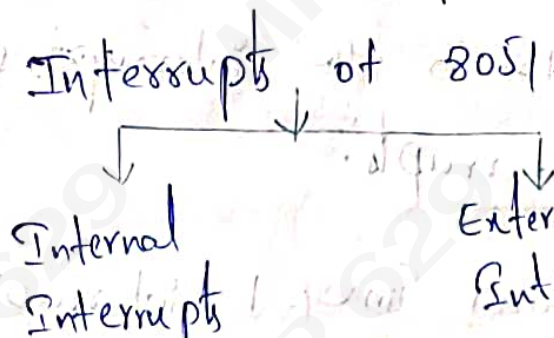
As shown in the figure the range of External ROM extends from 0000H to FFFFH.

* The ROM can be hold the program.
* The control pin $\overline{EA}$ (Pin 31) determines the accessing of either internal (or) External ROM.

* Explain the Interrupts of 8051?

A

Interrupts of 8051.



* 8051 has 2 External Interrupts. $\overline{INT_0}$, $\overline{INT_1}$
* 8051 has 3 Internal Interrupts. The 8051 has 5 Interrupts.

The Internal interrupts:-

1. Timer 0
2. Timer 1
3. Serial Port

* All the interrupts of 8051 are maskable and vectored interrupts.
* An interrupts is a signal either from hardware or from a program that interrupts the normal processing of the CPU.
* with the help of IE register the interrupt is enabled or disable with the help of IP register. The priority of Interrupt is decided.

External Interrupts:-

* The External interrupts are provided by the external circuitary connected to the pins of $\overline{INT_0}$ and $\overline{INT_1}$, of the 8051.

* The memory address of $\overline{INT_0}$ is 0003H, $\overline{INT_1}$, is 0013H in the interrupt vector table.

Internal Interrupts:-

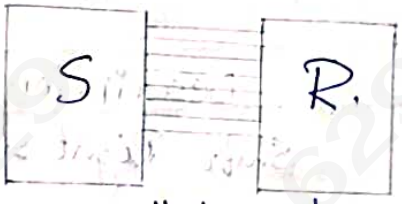
* Timer 0 and Timer 1 interrupts are generated by the internal timer/counter registers. These are TF_0 & TF_1.
* The memory address of TF_0 is 000BH and TF_1 is 001BH in the interrupt vector table.
* Serial port/communication is a single interrupt that belongs to both transmit and receive.
* The serial port interrupts is generated by the logical OR of R_I and T_I.
* The memory address of R_I and T_I is 0023H in the interrupt vector table.

Interrupt	Vector Address	Priority:
External Interrupt-0	0003H	Highest
Timer-0 Interrupt	000BH	
External Interrupt-1	0013H	
Timer-1 Interrupt	001BH	
~~External~~ Serial port Interrupt	0023	Lowest.

* Explain Serial Input / output of 8051?

A * The 8051 has one special communication interrupt.

* The 8051 is a 8-bit MC and it transfers 8-bit data simultaneously. Hence it requires 8 transmission lines between sender and receiver.



Parallel mode.

* The serial data coming from input device is to be converted in to parallel from and given to MC.

* parallel data from the MC is converted in to serial form and given to output devices.

* With the help of SBUF (Serial Port Buffer Register) the serial data transmission takes places.

* During transmission: parallel data is return in to SBUF. which is then converted into serial form.

* During reception serial data is converted into parallel and Enter SBUF.

* With the help of SCON register it contrd the serial data transmission and reception.

* The 8051 has serial data communication device. called [UART] universal Asyncronous Transistor Register.

* The register SCON is used to control data communication.
* The Register PCON is used to control, data rate.
* The 8051 provides 4 programmable modes for serial data communication.
* The modes are selected by SM_0, SM_1 bits in the SCON register.

SM_0	SM_1	Mode	Description.
0	0	0	Shift Right register.
0	1	1	8-bit UART
1	0	2	9-bit UART
1	1	3	9-bit UART

* Explain different modes of operation of 8051?

A. * The modes of operations depends on TMOD and TCON registers.
* The TMOD Register the bits M_0, M_1 specifies the modes

Mode 0:- ($M_1=0$, $M_0=0$)

In this mode timer can act as 13 bit timer.
T_L = 5bits
T_H = 8bits

Mode 1:- [$M_1=0$, $M_0=1$]

In this mode the timer can act as 16-bit timer.
T_L = 8 bits
T_H = 8 bits

Mode 2 :- ($M_1 = 1$, $M_0 = 0$)

This operation is called 8-bit auto reloaded mode.

* In this mode TH holds the reloaded values and TL holds the timer itself.
* The timer act as 8-bit timer.

Mode 3 :- ($M_0 = 1$, $M_1 = 0$)

This operation is called split mode. In this mode timer 0 is user as TL_0 and TH_0 and timer 1 is stopped.

* Explain condition & Loop statements?

A. 1. IF-THEN :-

The statement has the following.

```
IF condition THEN
    action
    action
```

The above statement says that, ~~THEN~~ IF the started condition to be true, THEN the series of actions will be executed.

* IF the specified condition is not true execution will skip over the actions after THEN.

Consider the following program by implementing this type of IF-THEN.

* Generally this statement is implemented with a conditional Jump instruction.

2. IF-THEN-ELSE :-

This is used to indicate a choice between two alternative forces of action.

```
IF condition THEN
    action
Else
    action.
```

3. Multiple IF-THEN-ELSE

```
IF    condition    THEN
        action
ELSE IF   condition   THEN
        action

  Else
        action.
```

* Assembler supports the use of conditional statements in the ALP.
* This conditional statements controls the flow of the program execution.

Loop statements

* Assembler supports the use of loop statements in the ALP.

WHILE - DO

```
WHILE    Somecondition is    DO
          present
            action
            action.
```

write a program to add two 16-bit quantities stored in the Ax and Bx registers. If no carry exists after the addition, increament SI register.

```
        ADD AX,BX
        JNC   THERE;
        NOP
        NOP
        NOP
AHEAD   INC   SI
```

* write a program to add two 16-bit numbers stored in the register Ax and Bx. If carry exists, then increment SI register; otherwise move the data 02H in to the register CL?

```
        ADD  Ax,Bx
        JC   AHEAD
        JMP  THERE
AHEAD   INC  SI
THERE   MOV  CL,02H
```

Multiple IF-THEN-ELSE

```
        ADD   AH, BH
        CMP   AH, 10H
        JB    AHEAD
        CMP   AH, 40H
        JB    THERE
        MOV   CL, 00H
AHEAD   INC   SI
        JMP   EXIT
THERE   MOV   CL, 04H
```

* Write assembly language programs using procedure and parameter passing?

A. Program Statement:-

Write an assembly language program to convert a given BCD number to a binary number using procedures and parameters passing.

* Program Methodology:-

During the conversion of BCD number to a binary number, the following sequence of operations is involved.

1. Nibble seperation.
2. Saving of lower nibbles.
3. Multiplication of AH with the upper nibble.
4. ADD the above product with the lower nibble.

Note that the program may use AL register to pass the parameters to the procedures.

Program :-

```
TITLE : BCD to Binary Conversion
.DOS.SEG
.MODEL small
.STACK 100H
.CODE
Main PROC
    MOV AX, @ data;
    MOV DS, AX
    MOV AL, num;
    CALL BCD Bin
    MOV LOC, AL;
    MOV AX, 4C00H;
    INT 21H
Main ENDP
    BCD Bin     PROC near
    PUSH F      ;   Save the flags
    PUSH BX     ;   and
    PUSH CX     ;   Registers
    MOV BL,AL   ;   Transfer the given BCD in BL
    AND BL, 0FH ;   To mask upper nibble.
    AND AL,F0H  ;   To mask lower nibble.
    MOV CL,04H  ;   Move upper BCD digit
                    to lower order.
```

```
    ROR AL,CL   ; Nibble position for multiplication
    MOV BH,0AH  ; Load conversion factor in BH
    MUL BH      ; multiply upper BCD digit
                  in AL by 0A in BH register.
                  product is in AL.
    ADD AL,BL   ; Add lower BCD digit to the
                  product.
    POP CX      ; Restore registers.
POP Bx
    POPF        ;
    RET         ;
    BCD Bin     ENDP
.DATA
    Num db 15H
    LOC db ?
END main
```