# Unit-I

## 8085 Microprocessor

### Hardware Architecture of 8085:

### Features of Microprocessor - 8085

* 8085 is developed by INTEL.
* 8 bit microprocessor: Can accept 8 bit data simultaneously.
* Designed Using NMOS Technology.
* It provides on chip clock generator, hence it does not require external clock generator.
* operates on 3 MHz clock frequency.
* It includes 8 bit multiplexed address / data bus which reduce the number of Pins.

→ **Main Units of 8085**

1) Control Unit
2) ALU.
3) Register.
4) Interrupt
5) Internal Data bus.

**Address Bus:** Address bus is a group of 16 lines generally identified as $A_0$ to $A_{15}$. The address bus is unidirectional and bits flow in one direction from the MPU to peripheral devices.

**Data Bus:** data bus is a group of 8 lines used for data flow. These lines are bi-directional and data flow in both directions between the MPU and memory and peripheral devices. (Memory Protection Unit)

Control bus It Carries Synchronization Signals, Si

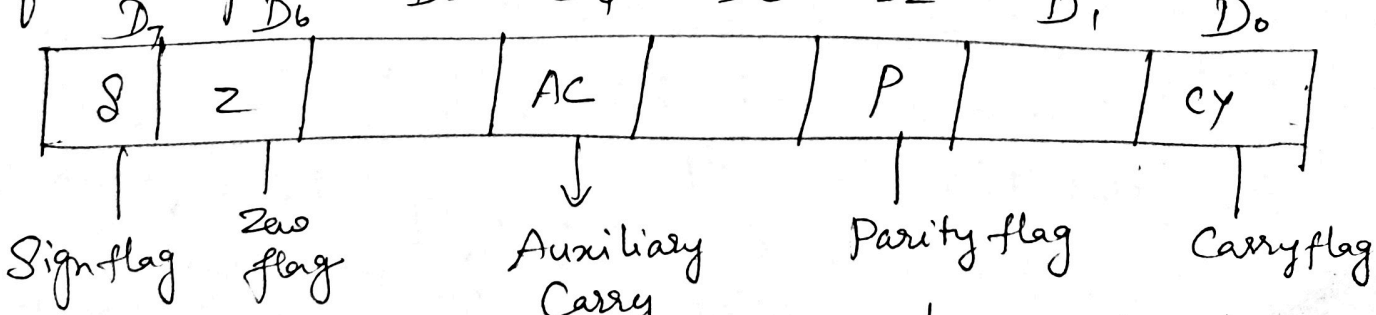
Providing timing Signals.

Registers of 8085 The 8085 have Six general purpose registers to store 8-bit data during program execution. These registers are identified as B, C, D, E, H and L. They can be combined as register pairs - BC, DE, and HL to perform some 16-bit Operations.

Accumulator (A) It is an 8 bit register that is part of the ALU. This register is used to store 8 bit data and to perform arithmetic and logical Operations. The result of an Operation is stored in the accumulator.

Flags: Microprocessor Uses 5 flags. for testing the data conditions. They are Zero(Z), Carry(CY) Sign(S), Parity(P), and Auxiliary Carry(AC). flags.

→ flags are set or reset according to the result of an Operation




→ It is an 8 bit register having five 1bit flipflops which holds either 0 or 1 depending upon Result Stored in the accumulator.

Sign flag: If $D_7$ of the result is 1 the sign flag is Set. Otherwise it is reset $D_7$ is reserved for indicating the sign.

If $D_7$ is 1, the number will be viewed as negative number

If $D_7$ is 0 the number will be viewed as positive number

2) Zero flag (Z). If the result of arithmetic and logical Operation is zero. then Zero flag is Set. Otherwise it is reset.

3) Auxiliary Carry flag (Ac) If $D_3$ generates any Carry when doing any arithmetic and logical Operation, this flag is set. otherwise it is reset.

4) Parity flag. (P) If the result of arithmetic and logical Operation Contains even number of 1's then this flag will be Set and if it is odd number of 1's it will be reset.

5) Carry flag (CY) : If any arithmetic and logical Operation result any Carry then Carry flag is set otherwise it is reset.

Arithmetic and Logic Unit (ALU)

It is Used to perform the arithmetic Operation like addition, Subtraction, multiplication, division, increment and decrement and logical operations like AND, OR, and Ex-OR.

**Program Counter:** The function of the Program Counter is to point to the memory address of the next instruction to be executed. When an opcode is being fetched, the Program Counter is incremented by one to point to the next memory location.

**Stack Pointer:** It is also a 16 bit register used as a memory pointer. It points to a memory location in R/W memory, called the Stack.

**Temporary Register:** It is used to hold the data during the arithmetic and logical operations.

**Instruction Register:** When an instruction is fetched from the memory, it is loaded in the instruction register.

**Instruction Decoder:** It gets the instruction from the instruction register and decodes the instruction. It identifies the instruction to be performed.

**Timing and Control Unit:**

It has Three control signals, ALE, RD (Active low) and WR (active low) and Three status signals I/O M (Active low), S0, S1.

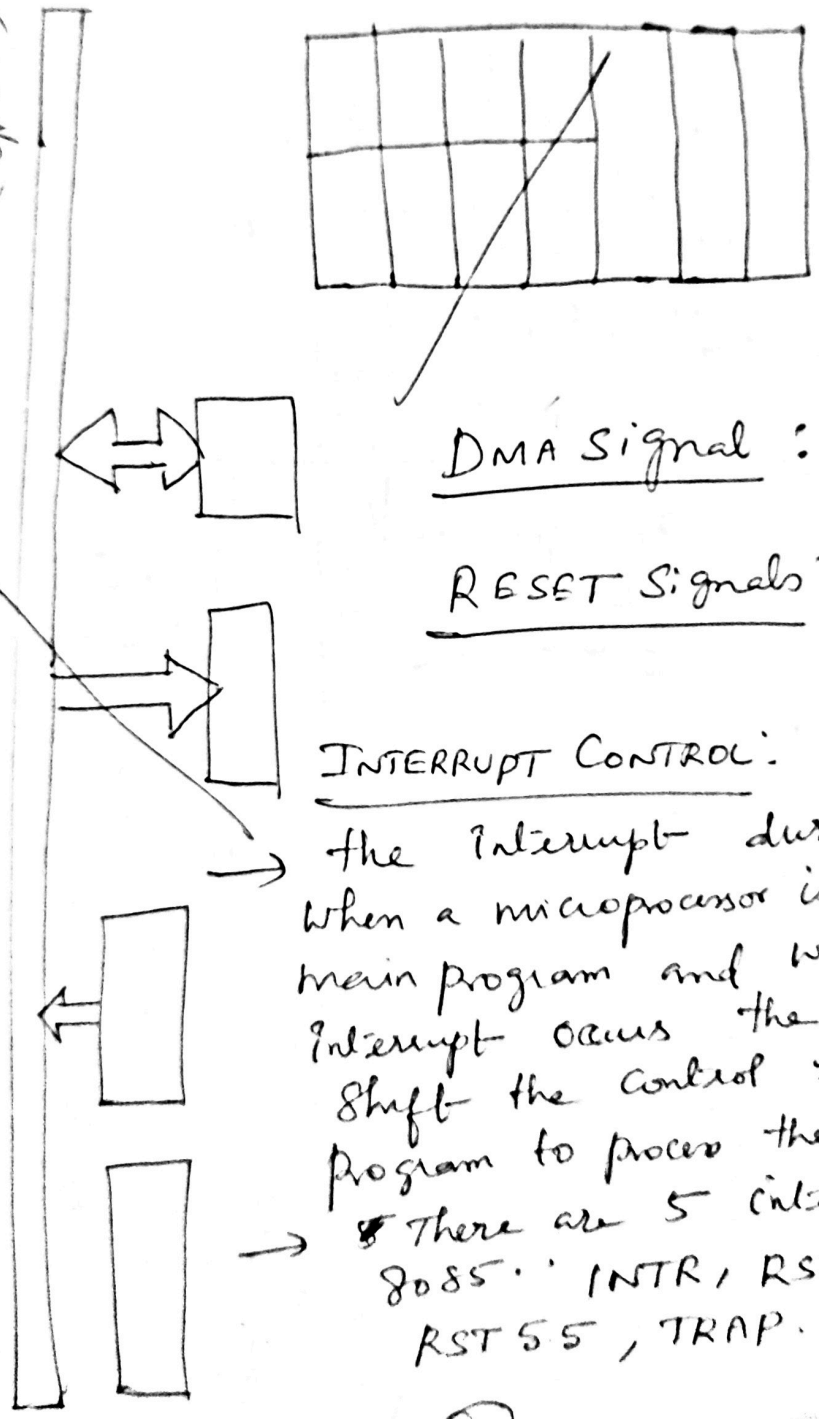
**ALE** (Address Latch Enable) It is used for provide control signal to Synchronize the components of microprocessor and timing for instruction to perform the operation.

When ALE = 1, it makes address bus Enable.

When ALE = 0 means data bus Enable

RD/ (active low) WR are used to indicate whether the operation is reading the data from memory or writing the data into memory respectively.

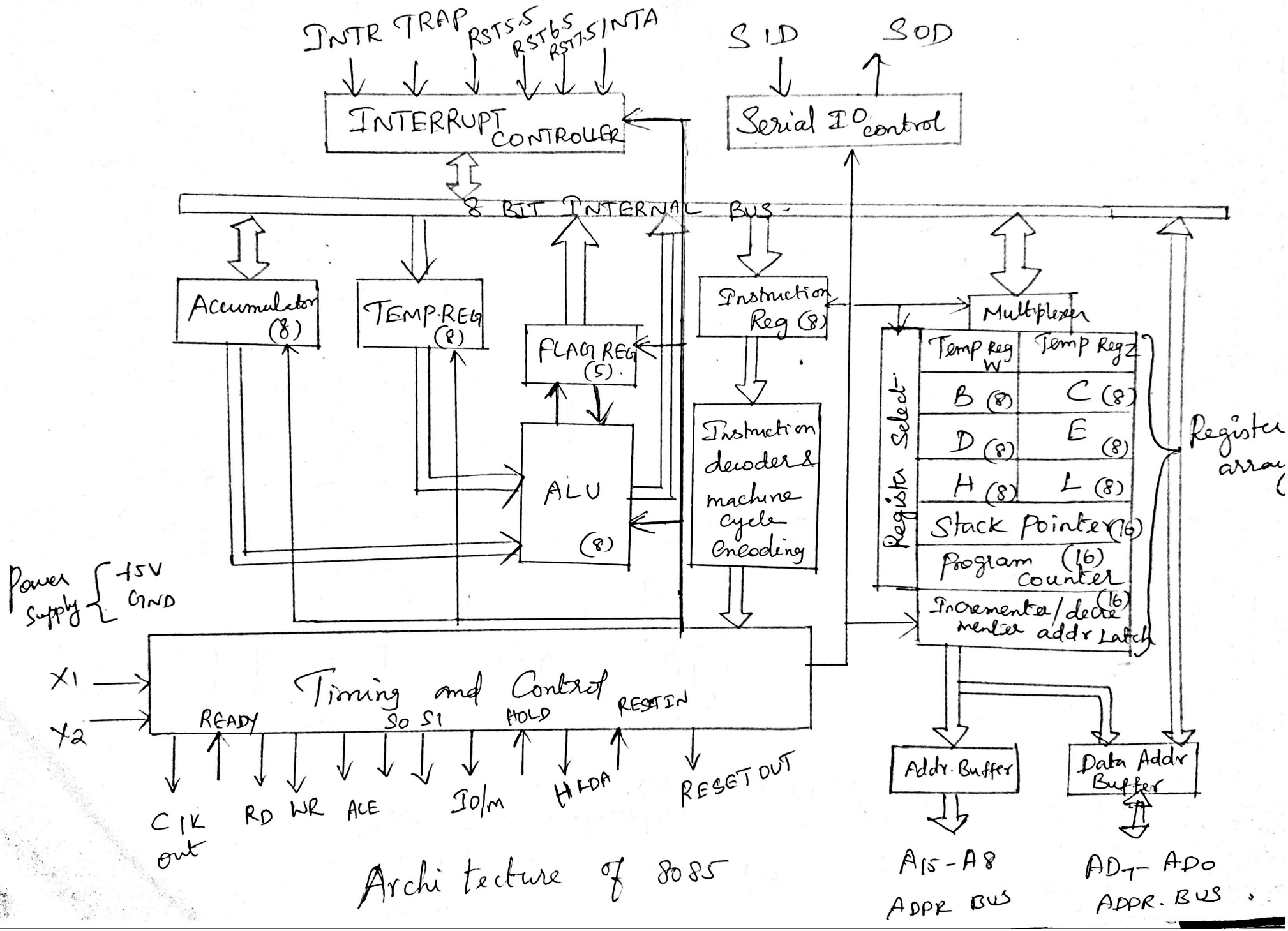
IO/M is used to indicate whether the operation is belongs to the memory or Peripherals.




DMA Signal : HOLD, HLDA (HOLD Acknowledge)

RESET Signals: RESET IN
RESET OUT

INTERRUPT CONTROL: It controls the Interrupt during a Process. When a microprocessor is executing a main program and whenever an Interrupt occurs the microprocessor shift the control from the main program to process the incoming request. There are 5 interrupt signals in 8085 : INTR, RST7.5, RST6.5, RST5.5, TRAP.

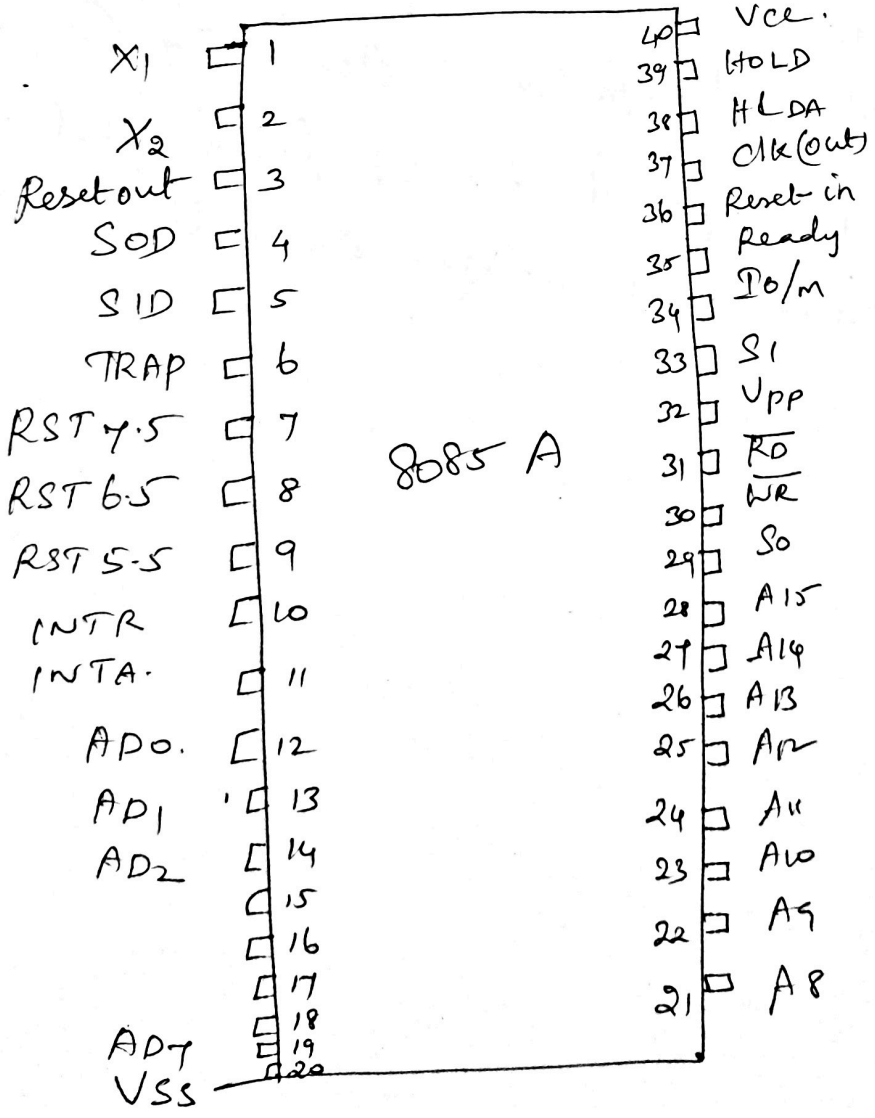



Architecture of 8085

# PIN diagram. (or) Signal Configuration of 8085

8085 is a 40 Pin IC, DIP package. The signals from the pins can be grouped as follows

* Power Supply and clock signals.
* Address bus.
* Data bus
* Control and Status signals.
* Interrupts and externally Initiated Signals
* Serial I/O Ports.
* Direct Memory Access (DMA).



## Power Supply and clock frequency Signals:

$V_{cc}$ : +5 volt power supply.

$V_{ss}$ : Ground

$X_1, X_2$ : Crystal or R/C network or LC network Connections to Set the frequency of internal clock generator.

CLK(Out) : clock Output is used as the System clock for peripheral and devices interfaced with the microprocessor.

**Address Bus:** $A_{15} - A_8$ it carries the most significant 8 bits of memory I/O address.

**Data Bus** $AD_7 - AD_0$ – it carries the least significant 8 bit address and data bus.

**Control and Status Signals.**

These signals are used to identify the nature of operation. There are 3 Control signals and 3 Status signals.

Three Control signals are RD, WR, ALE.

**RD →** This signal indicates that the selected I/O or memory device is to be read and is ready for accepting data available on the data bus.

**WR** This signal indicates that the data on the data bus is to be written into a selected memory or I/O location.

**ALE →** It is a positive going pulse generated when a new operation is started by the microprocessor. When the pulse goes high, it indicates address. When the pulse goes low it indicates data.

Status signals are $IO/\overline{M}$, $S_0$, $S_2$.

$IO/\overline{M}$ → it is used to differentiate between I/O and memory operation.

$IO/\overline{M}$ → "high" indicates I/O operation

$IO/\overline{M}$ → "low" indicates memory operation.

**$S_1$ & $S_0$.** → It is used to identify the type of current operation.

**Clock signal** : There are 3 clock signals. ie)
$X_1, X_2$, CLK OUT

$X_1, X_2$ → A crystal is connected at these two pins and is used to set frequency of the internal clock generator.

CLK OUT → This signal is used as the system clock for devices connected with the microprocessor.

**Interrupt & Externally Initiated Signals:**

Interrupts are the signals generated by external devices to request the microprocessor to perform a task. There are 5 interrupt signals ie)
TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR

**INTA** → It is an interrupt acknowledgement signal.

**RESET IN** → It is used to reset the microprocessor

**RESET OUT** → It is used to reset all the connected devices.

**READY** → This signal indicates that the device is ready to send or receive data.
→ If READY is low then CPU has to wait for READY to go high.

**HOLD** → This signal indicates that another master is requesting the use of the address and data buses.

**HLDA** → HOLD Acknowledge
it indicates that CPU has received the HOLD request.

## Serial I/O signals (7)

There are 2 signals i.e) SID and SOD. and these signals are used for Serial Communication.

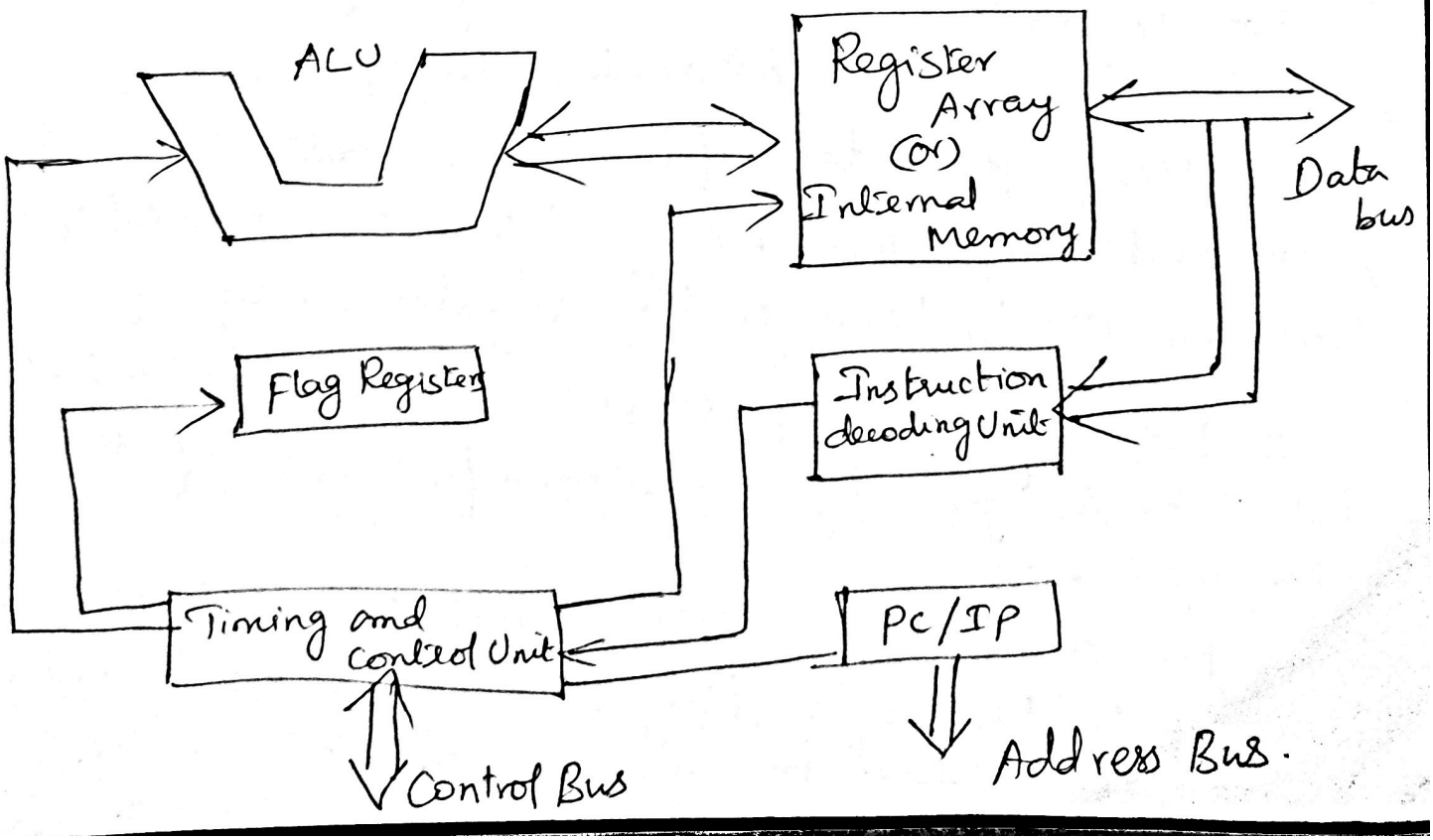
SOD → (Serial Output data line)

SID → Serial input data line.

## Functional Building Block of Processor.

→ A microprocessor is a Programmable IC which capable of performing arithmetic and logical operations.

→ The basic functional blocks of a microprocessor are ALU, flag register, register array, Program Counter (PC), Instruction pointer (IP), Instruction Decoding Unit and Timing and Control Unit.



## ALU Unit

* It is the Computational Unit of the microprocessor which perform arithmetic and logical Operation on binary data.

* Various conditions of the result are stored as Status bits called flags in the flag register

* for example Consider the Sign Flag one of the bit positions of the flag register is called the sign flag and it is used to store the sign of the result of the ALU operation

* If the result is negative then "1" is stored in the sign flag and if the result is positive the "0" is stored in the Sign flag.

## Register array.

* It is the Internal Storage of device and so it is also called Internal memory

* The Input data for ALU, Output data of ALU, and any other binary Infomation needed for Processing are stored in the register Array.

* For any microprocessor there will be a set of instructions given by the manufactures of the microprocessors.

* for doing any Useful work with the microprocessor we have to write a Program Using these instructions and Store them in memory device

→ Instruction Pointer generates the address of the Instructions to be fetched from the memory and send through the data bus

→ Instruction Codes are decoded by the decoding Unit which send the information to the timing and Control Unit.

→ The data is stored in the register array for processing by the ALU.

→ Control Unit will generates the necessary Control Signals for the internal and external operations of the microprocessor.

## Memory and I/O Interfacing

→ There are Two kinds of interfacing are available.

1. Memory Interfacing.
2. I/O Interfacing

Several memory chips and I/O devices are connected to a microprocessor.

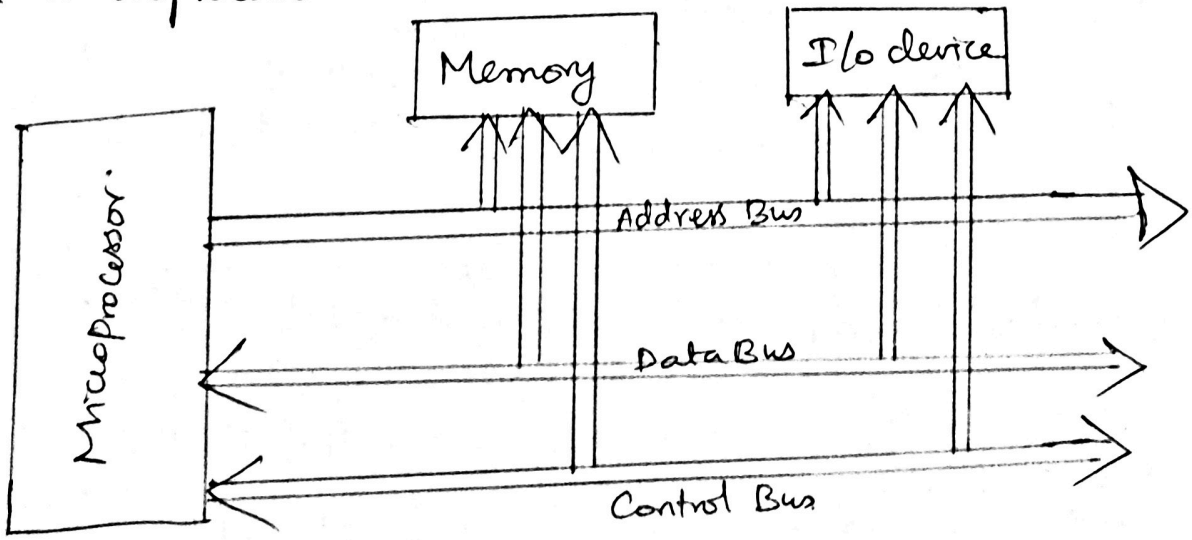



Diagram for Memory and I/O Interfacing

→ The above figure shows a schematic diagram to Interface memory chip or I/O devices to a microprocessor.

Memory Interfacing

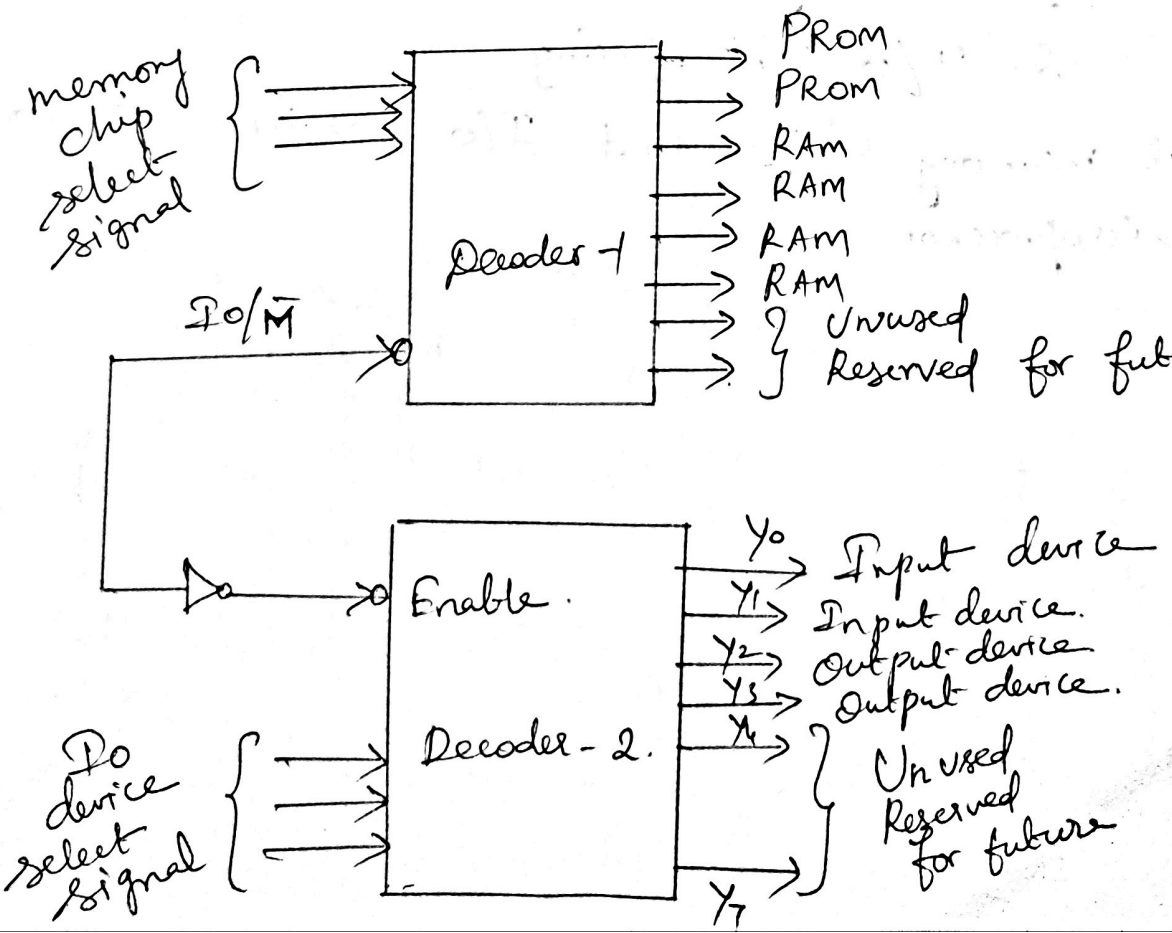
Following is the list of 8085 pins used for Interfacing with other devices.

$A_{15} - A_8$ → higher address Bus

$AD_7 - AD_0$ → lower address / data bus

ALE, RD, WR, READY.

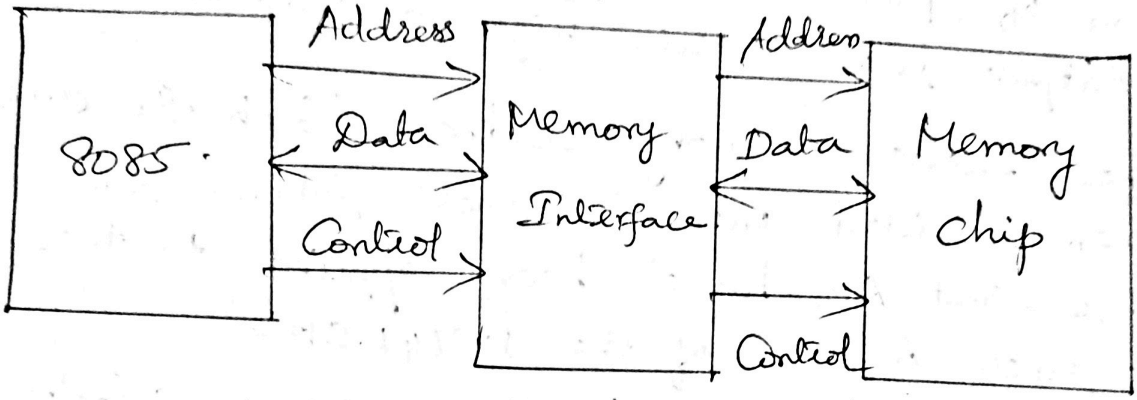
→ The address of a memory location or an I/O device is sent out by the microprocessor. The corresponding memory chip or I/O device is selected by a decoding Unit



* If $IO/\overline{M}$ is high the decoder 2 is activated and required I/O device is selected.

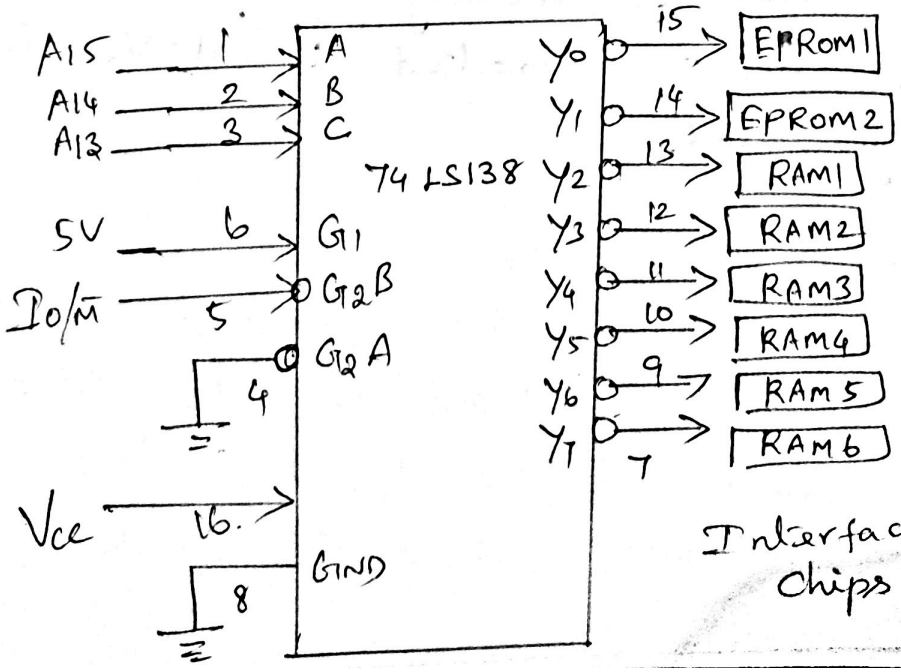
* If $IO/\overline{M}$ is low the decoder 1 is activated and the required, memory chip is selected.

* A few MSBs of the address lines are applied to the decoder to select a memory chip or an I/O device




8085 Interfacing with Memory Chip

→ It shows the interface of memory chips through 74S138 and it is a 1 to 8 line decoder.




Interfacing of memory chips Using 74LS138.

→ $G_{1}$, $G_{2B}$, $G_{2A}$ are enable signals. To enable 74LS138, $G_1$ should be high and $G_{2A}$ and $G_{2B}$ should be low.

→ There are three select lines A, B, C, applying proper logic to select lines any of the output can be selected.

→ $Y_0, Y_1 \cdots Y_7$ are 8 output lines. At the time of selection of particular output line will go low and other output lines remain high of particular.

→ The entire memory address (64K for 8085) has been divided into 8 zones. Address lines $A_{15}$, $A_{14}$, and $A_{13}$ have been applied to the select lines A, B, C of the 74LS138.

→ The logic applied to these lines select a particular memory device, in EPROM or a RAM. Other address lines $A_0, A_1, A_2, \ldots A_{12}$ go directly to memory chip.

→ $IO/\overline{M}$ goes low for memory read/write operation $G_{2B}$ goes low, $G_1$ is connected to +5Vdc supply and $G_{2A}$ is grounded.

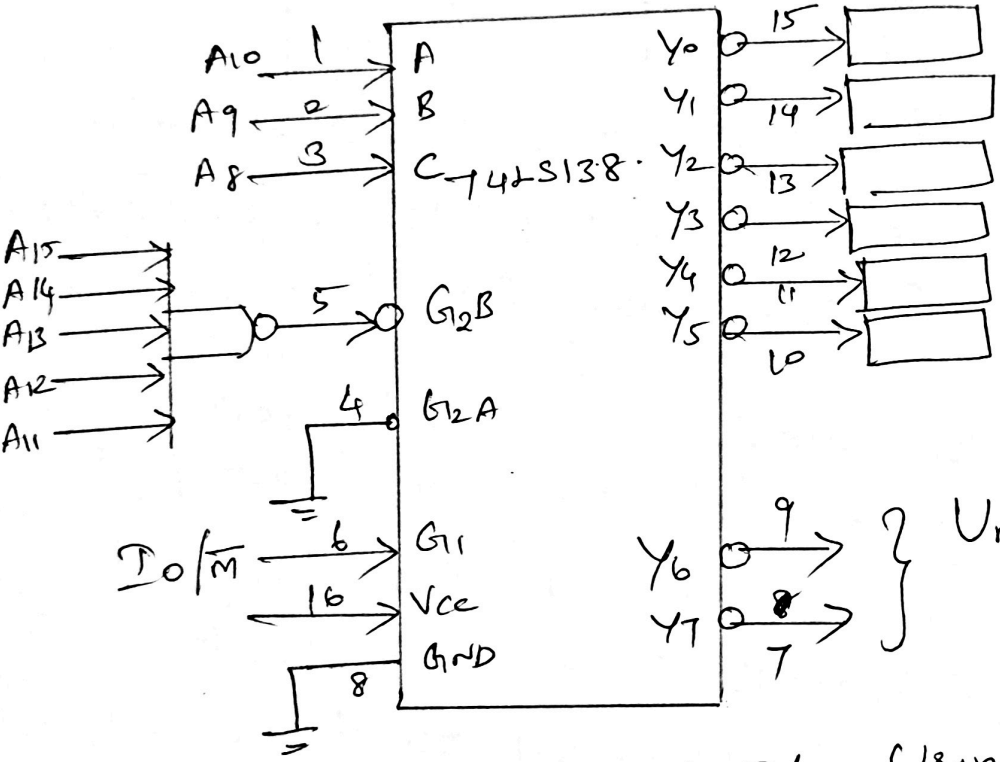
## I/O Interfacing

→ An I/O device is a 8 bits only $A_8 - A_{15}$ lines of address bus are used for I/O addressing.

→ The address lines $A_8$, $A_9$, $A_{10}$ have been applied to select lines A, B, and C of the 74LS138.

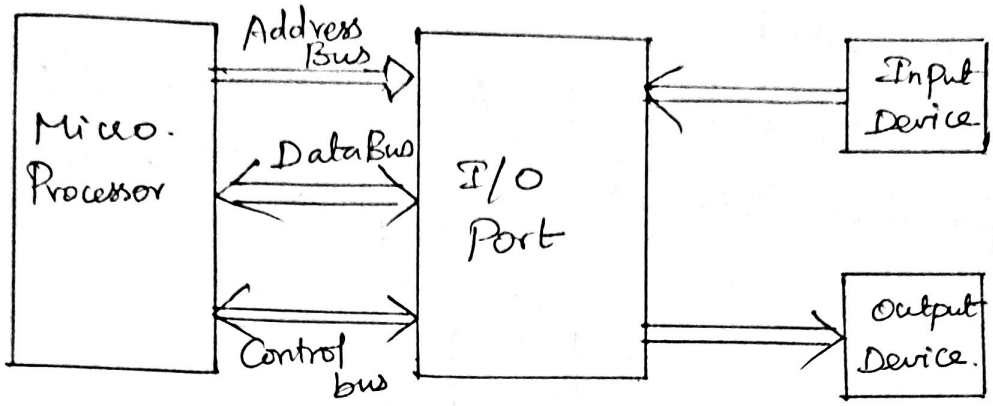
→ The address lines $A_{11}-A_{15}$ are applied to $G_{2B}$ through a NAND gate $G_{2B}$ becomes low only when all address lines $A_{11}-A_{15}$ are high. $G_{2A}$ is grounded.

→ $IO/\overline{M}$ is connected to $G_1$, when $IO/\overline{M}$ goes high for I/O read/write operation, $G_2$ goes high




Interfacing of I/O Using 74LS138.

# I/O Ports:




Interfacing of I/o Device through I/o Port.

→ An input device is Connected to the microprocessor through an input port, an input port is a place of Unloading data.

→ An input device unloads data into the port the microprocessor reads data from the input port

→ Thus the data are transferred from the input device to the accumulator via input port.

→ An Output device is Connected to the microprocessor through output port

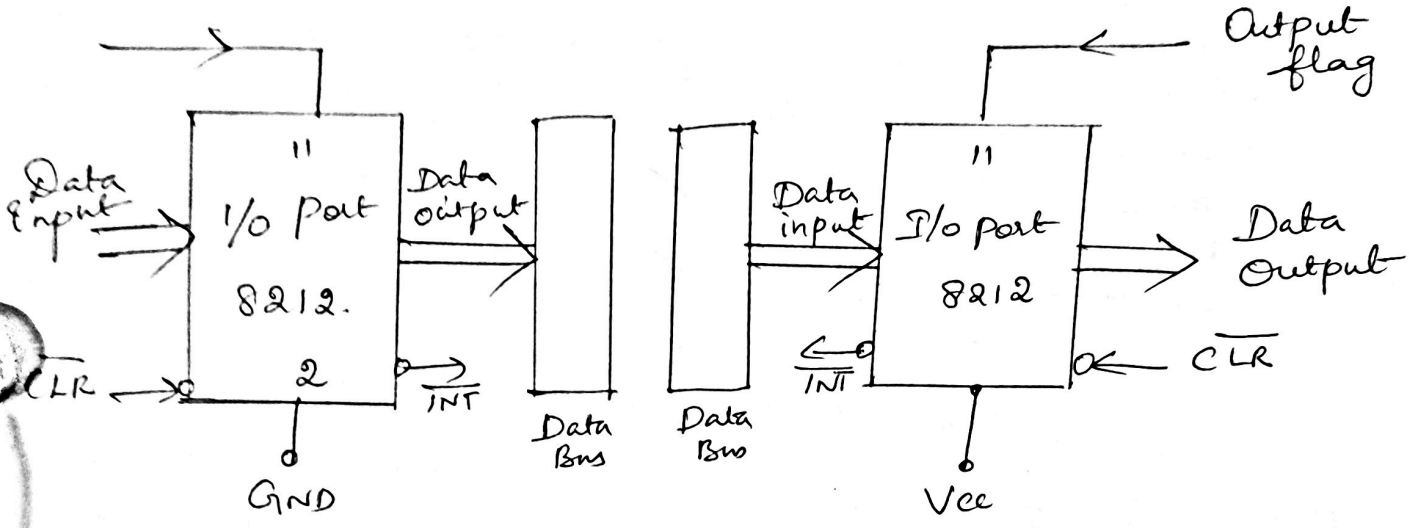
→ O/p port is Connected to the Output device, data are transferred to the Output device.

→ An I/o Port may be Programmable or non Programmable A non Programmable port act as an input port if it has been designed and Connected in Input mode.

(11)

→ A port connected in Output mode acts as an Output port. But a Programmable I/O port can be Programmed to act either as an Input port or Output port.

→ The Intel 8212 is an 8-bit non Programmable I/O port. It can be connected to the microprocessor either as an Input port or an Output port.




Interfacing of Intel 8212.

→ Intel 8255 which is a programmable peripheral interface (PPI). The intel 8255 is more powerful than Intel 8155

## Programmable Peripheral Interface (PPI)

A Programmable peripheral interface is a multiport device. The ports may be Programmed in a variety of ways as required by the Programmer

## INTEL 8255

→ It is a Programmable Peripheral Interface (PPI) it has Two kind of Versions.

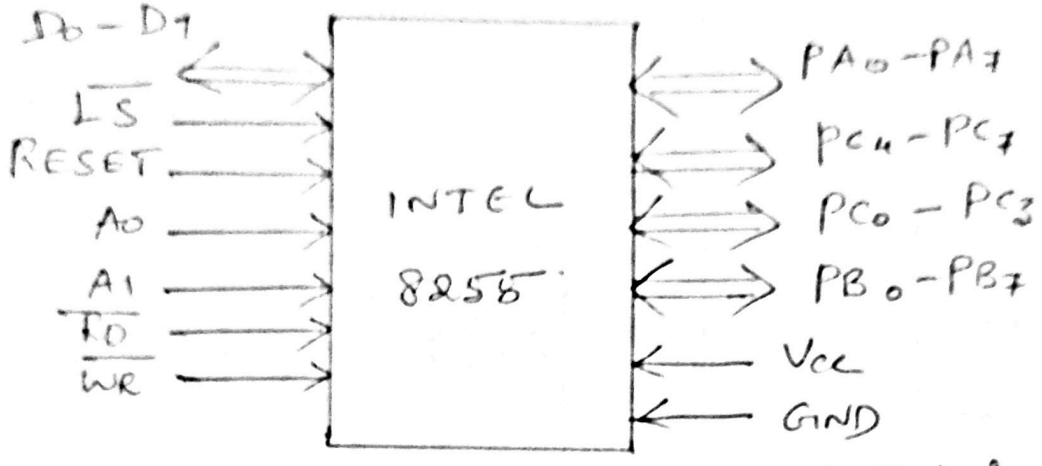
① Intel 8255 A, ② INTEL 8255 A-5.

→ Main function of Intel 8255 is to Interface peripheral devices to the microcomputer It has three kinds of 8 bit ports.

① Port A ② Port B ③ Port C

Port C has to be divided into two of 4-bit Port, Port C Upper, and Port C Lower.

## Architecture of INTEL 8255




Schematic Diagram of Intel 8255

Intel 8255 is a 40 Pin Ic Package. It operates on a single 5V dc power supply. It is having the following important characteristics

1. Ambient Temperature 0 to 70°C
2. Voltage on any Pin : 0.5V to 7V
3. power dissipation 1 watt.

$V_{IL}$ = Input low voltage = minimum 0.5V and maximum 0.8V

$V_{IH}$ = Input high voltage = Minimum 2V and maximum Vcc

$V_{OL}$ = Output low voltage = 0.45V

$V_{OH}$ = Output high Voltage = 2.4V

$I_{DR}$ = Darlington Dive Current = Minimum 1mA maximum 4mA, of any 8 pins of the port.

The Pins for Various ports are as follows

(1) $PA_0 - PA_7$ 8 pins of port A
(2) $PB_0 - PB_7$ 8 Pins of port B
(3) $PC_0 - PC_3$ 4 Pins of port C lower.
(4) $PC_4 - PC_7$ 4 Pins of port C upper.

Control Signals.

① $\overline{CS}$ (chip Select).
It is a chip select signal. The low Status of this signal. The low Status of this signal enables Communication between CPU and 8255.

② $\overline{RD}$ (READ): When this pin goes low, the 8255 Sends out data or Status information to the CPU on the data bus.

③ $\overline{WR}$ (WRITE) when $\overline{WR}$ goes low, the CPU writes data or Control word into 8255. CPU writes the data into the o/p port of 8255 & Control word into control word register.

④ A0 and A1 The Selection of Input port and Control word register is done using A0 and A1 in Conjunction with $\overline{RD}$ and $\overline{WR}$.

⑤ Operating Modes of 8255
Intel 8255 has the following three modes of Operation. which are Selected by Software.
(1) Mode 0 - Simple Input/output.
(2) Mode 1 - Stored Input/output.
(3) Mode 2 - Bi directional port.

: In mode 0 → programmed to be either an input or output port.

In mode 1 → port A and port B both are designed to operate in this mode.

## Control groups

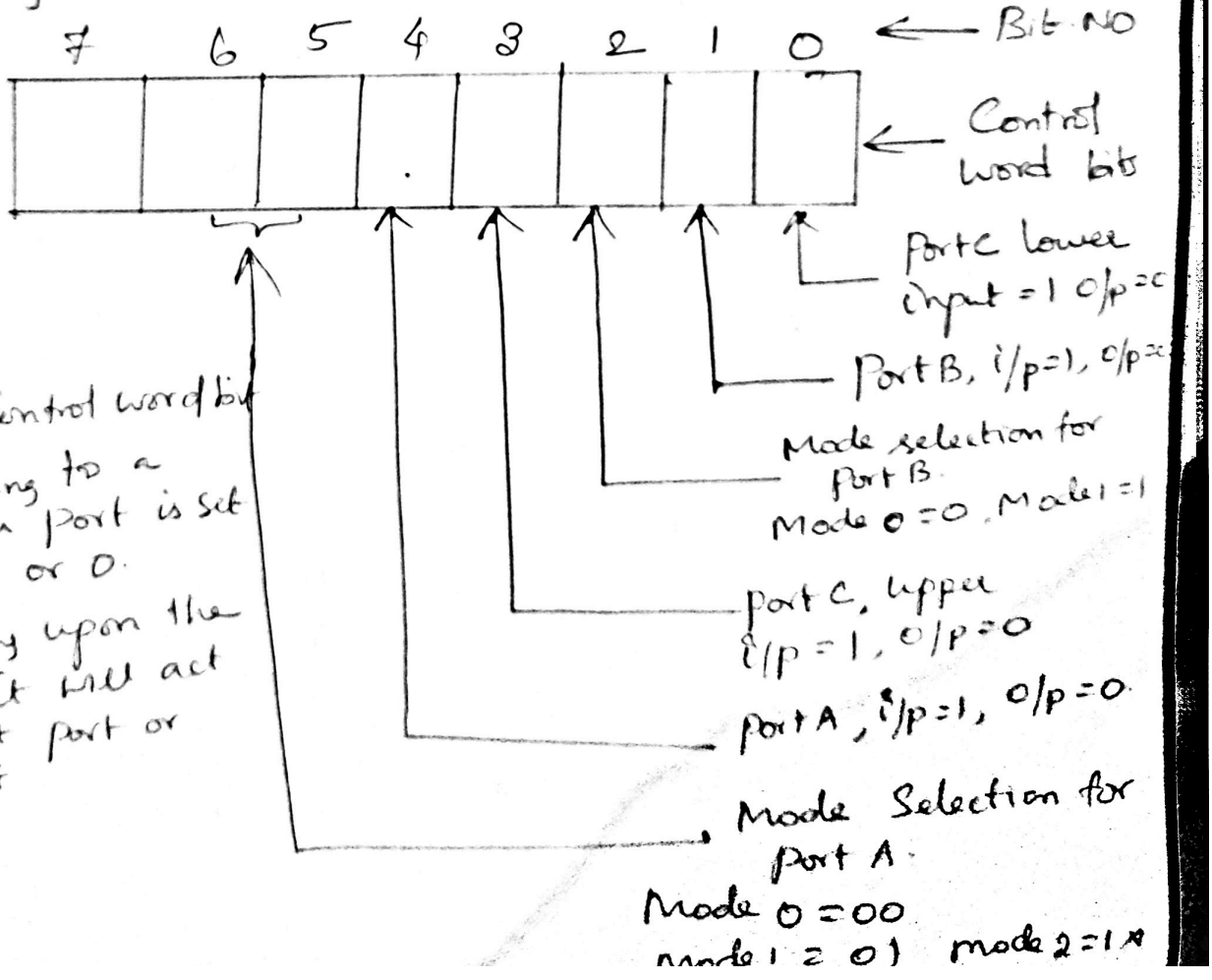
→ Intel 8255 has 24 I/O lines of I/O ports for the control purpose and this has to be divided into two groups ① Group A ② Group B

↓ This Contain port A & port C upper

↓ This Contain the Port B and Port C lower

## Control word

→ It is formed which contains the information regarding the function and modes of the ports



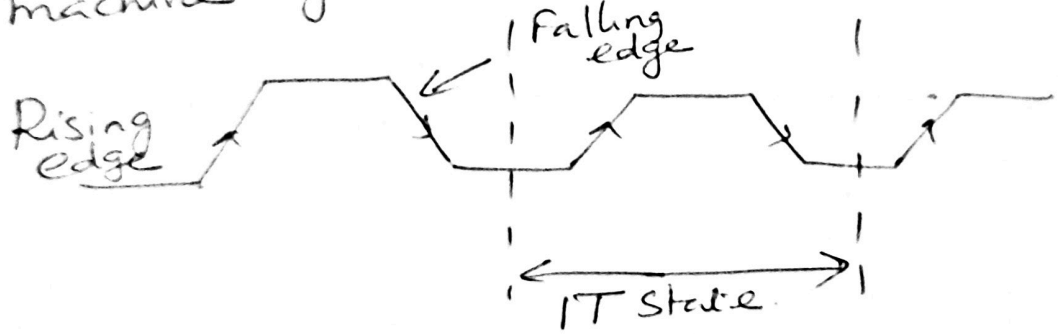

* Each Control word bit Corresponding to a particular port is set to either 1 or 0.
* Depending upon the value it will act as input port or o/p port

# Timing Diagram of 8085

* To perform machine cycle we need to carry some necessary steps and it can be represented graphically such a graphic representation is called timing diagram.

* 8085 Processor has 7 basic machine cycles
  1. Opcode fetch cycle (4T or 6T)
  2. Memory Read cycle (3T)
  3. Memory Write cycle (3T)
  4. I/O Read cycle (3T)
  5. I/O Write cycle (3T)
  6. Interrupt acknowledge (6T or 12T)
  7. Bus idle cycle (2T or 3T)

→ Time taken by the Processor to execute a machine cycle is called T-state.




T-state Diagram.

## Opcode fetch cycle

→ In a fetch cycle the microprocessor fetches the Opcode of an instruction from the memory.

→ $T_1$ $T_2$ $T_3$ and $T_4$ are Consecutive four Clock cycle.

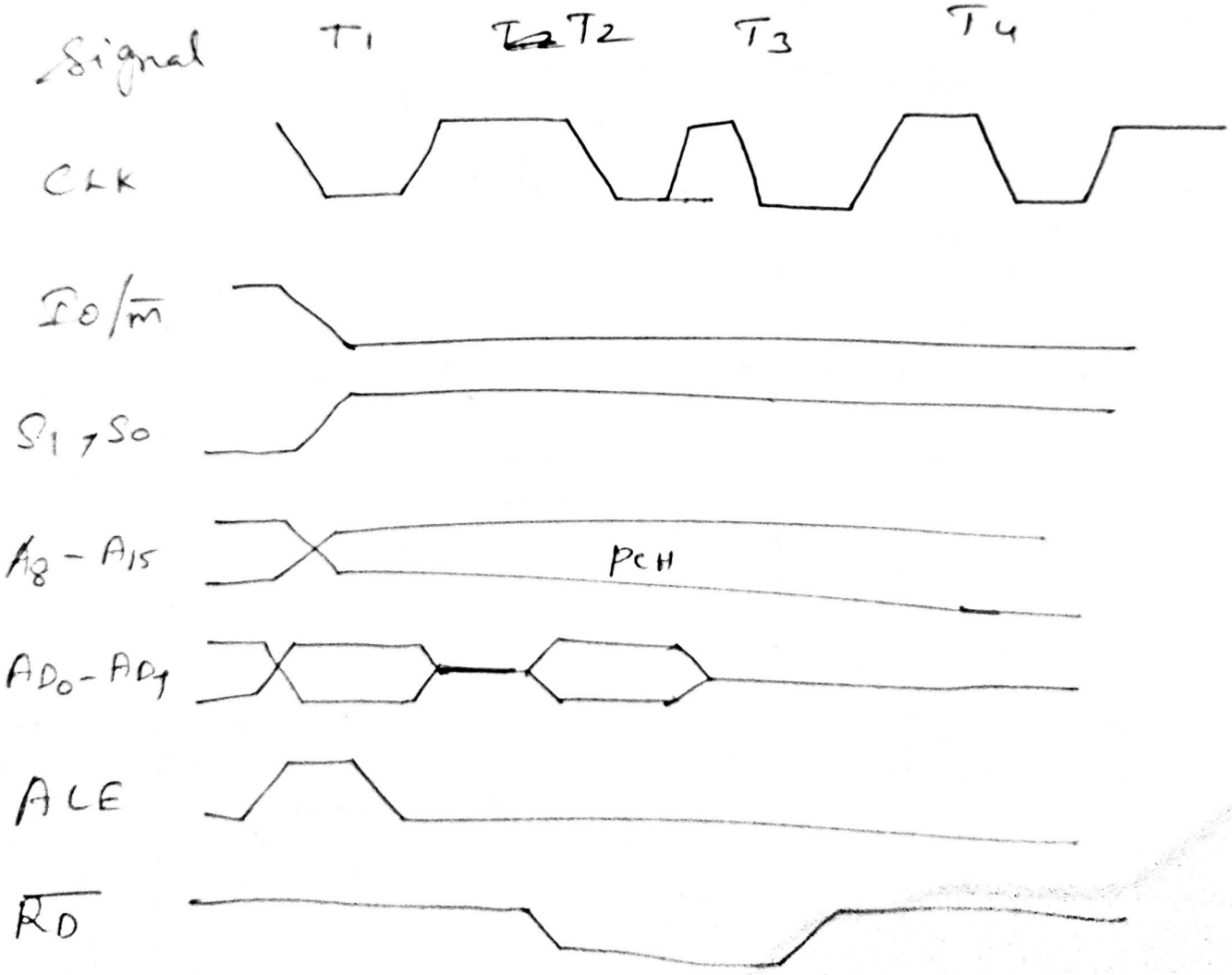
$IO/\overline{M} \rightarrow 0$. indicate that it want to make Communicate with memory

→ $S_0, S_1$ → high indicate that it is going to perform fetch operation

→ In first clk cycle $T_1$ µp sends out the address of memory location where the opcode is available

→ During $T_2$ AD bus becomes ready to carry data In $T_2$ µp makes $\overline{RD}$ low now memory gets opcode from the specified memory location and places it one the data bus.

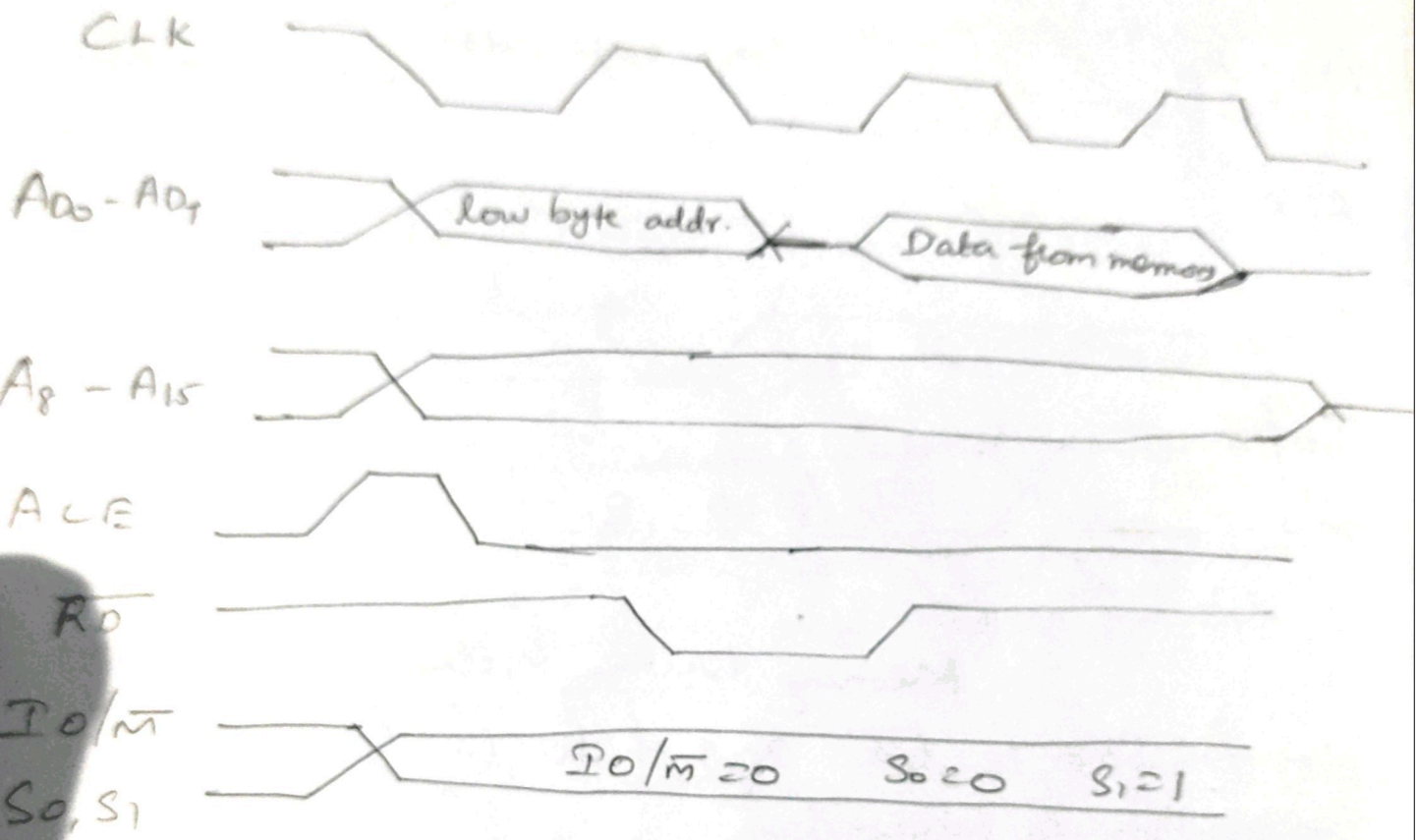
→ During $T_3$ the opcode is placed in the Instruction reg




Timing Diagram for opcode fetch operation

## Memory Read cycle.

→ In a memory read cycle the micro processor reads the content of the memory location. The content is placed either in the accumulator or any other register of the CPU

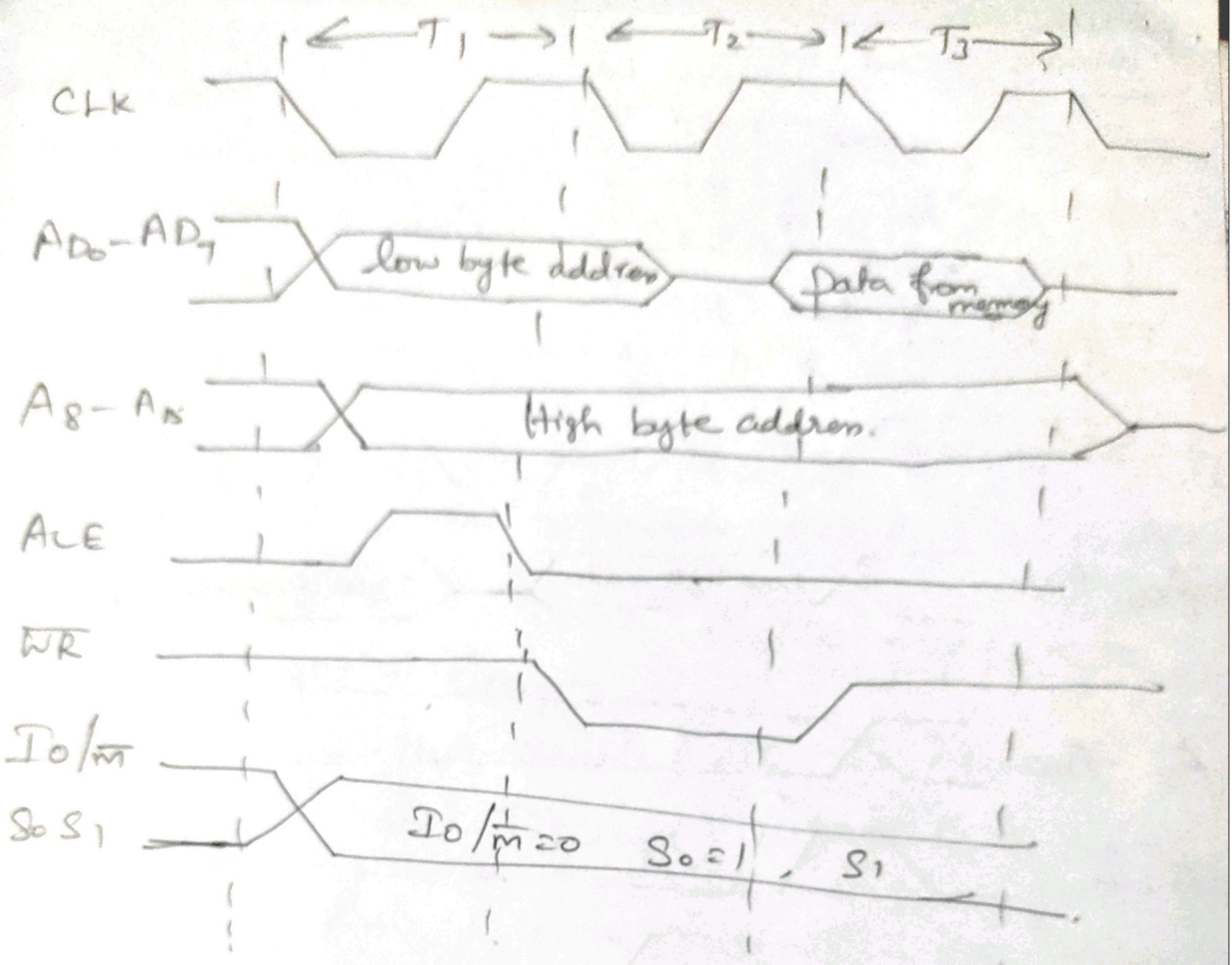



Memory read cycle.

## Memory Write cycle (3T)

→ In a memory device cycle the CPU sends data from the accumulator or any other register to the memory.

→ The Status signal $S_0$ and $S_1$ are 1 and 0 respectively for write operation. $\overline{WR}$ goes low in $T_2$ indicates that write operation is to be performed.




Memory write cycle.

→ As soon as $\overline{WR}$ goes high in $T_3$ the write operation is terminated.

# UNIT-2

# Programming of 8085 Processor

## Introduction.

* A Computer is Capable of doing what the Programmer ask to do. To perform a particular task the Programmer prepares a sequence of instructions Called a Program.

* The Program is Stored in RAM. The cpu takes one instruction of the program at a time from RAM and Executes it. A Program written in the form of 0's and 1's is Called machine language Program.

* The language in which the instructions are represented by mnemonics is Called assembly language The assembly language have to be Converted to machine language for execution. This Conversion is performed by a Software tool Called Compiler.

## Types of Programming

We Can Programme the Computer in following ways.

1. Machine language Program.
2. Assembly language Program.
3. High-level language Program.

## Instruction format:

An Instruction is a Command given to the Computer to perform a Specified operation on given data. Instruction set of a microprocessor is the Collection of instruction that the microprocessor is designed to execute.

These instructions have been classified into following groups

1. Data Transfer group.
2. Arithmetic group.
3. Logical group.
4. Branch Control Group
5. I/O and Machine Control Group.

**Data Transfer group** : * Data Transfer Instructions are used to move data b/w registers, register pairs and between memory and registers. examples are MOV, MVI, LXI, LDA, STA.

* When an instruction of data transfer group is executed, data is transferred from the source to the destination without altering the contents of the source.

Example MOV, A, B

The Content of reg B is copied into the reg A.

**Arithmetic group** These instruction groups will perform arithmetic operations such as addition, Subtraction, increment or decrement of the content of a register or memory. example ADD, SUB, INR etc.

**Logical group** : These instruction group Perform logical operation such as AND, OR, Compare, rotate. etc.

Example ANA, XRA, ORA, etc.

**Branch Control group**. This group includes the instructions for Conditional and Unconditional jump, Subroutine Call and return and reset.

Examples are JMP, JC, JZ, CALL, RST, etc.

**I/O and Machine Control Group**:

This group includes the instructions for I/P O/P Ports. Stacks, and machine Control. Examples are IN, OUT PUSH, POP, HLT, etc.

## Data formats

Intel 8085 is an 8 bit microprocessor. It handles 8 bit data at a time. One byte consists of 8 bits.

→ The various techniques to specify data for instructions are

① 8bit or 16 bit data may be directly given in the instruction itself.

② The address of the memory location, I/O port or I/o device.

③ In some instructions only one register is specified.

④ Some instructions specify two registers

⑤ In some instructions data is implied.

→ Due to different ways of specifying data for instructions, the machine codes of all instructions are not of the same length.

## Three Types of Intel 8085 instructions which are as follows

① Single byte instruction
② Two byte instruction
③ Three byte instruction.

## One byte instruction.

Example of one byte Instructions are

① MOV A, B.
Move the Content of register B to register A.

② ADD B. → Add the Content of register B to Content of the Accumulator.

③ RAL → Rotate the Content of the accumulator left by One bit.

## Two byte instruction.

In a two byte instruction the 1st byte of the instruction is its opcode and 2nd byte is either data or address.

① MVI B, 05, Move 05 to register B.

06, 05 ; MVI B, 05 in the Code form.

The 1st byte 06 is the opcode for MVI B and 2nd byte 05 is the data. which is to be moved to Register B.

② In 01 ; read data at Port B.

DB, 01. where DB is the opcode for In and 01 is the address of a Port.

## Three byte Instruction.

In a Three byte instruction the 1st byte of the instruction is its opcode and 2nd. and 3rd bytes are either 16-bit data or - 16 bit address Example

① LXI H, 2400H → Load H-L Pair with 2400H.

21, 00, 24. → The 1st byte 21 is the opcode for Instruction LXI H. The 2nd byte 00 is 8 LSB of the data (2400H) which is loaded into register L. The 3rd byte 24 is 8 MSB of data (2400H) which is loaded into register H.

② LDA 2500H. → 3A, 00, 25 → MSB of Data. → LSB of Data → Opcode

## Machine Language Program.

→ A Program written in the form of 0's and 1's called a machine language Program. In the machine language there is a Specific binary code for each instruction.

→ The language in which a Programmer writes Programs is called Souace language. The langage in which a Compuiter works is called object or machine. langage. Machine codes are also called object codes. This machine langage is a low level langage.

## demerits of machine languages

① It is very difficult to understand or debug a Program.
② Programs are long. & Program writting is Difficult.
③ chances of careless errors in writing the Program.

## Assembly language Program

* To overcome the ~~dea~~ drawbacks of machine level langage, assembly language Program will be used.

* It can be easily write a Program in alphanumeric symbols instead of zeros and ones. A Program written in ~~bas~~ mnemonics is known as assembly language Program.

Example add for addition, SUB for Subtraction, CMP for Comparision etc. Such Symbols are Called mnemonics.

→ It Can be Used in following Situations.
① Small to moderate Size of Program.
② Real time Control applications.
③ for industrial applications.

→ Assembly language Program is much easier and faster than the machine language.

## Assembler

→ A Program which translates an assembly language Program into a machine language Program is called an assembler

→ Assembler Can be divided into two types. based on the function of it.
① self Assembler ② Cross assembler

① **Self assembler** → It is also known as resident assembler. This assembler runs on the microcomputer for which it Produces object codes. (machine Codes).

② **Cross assembler.** : When a Program Can be dveloped on other Computer means we need a Special assembler known as Cross assembler.

Disassembler. Assembler is used to convert the assembly language into machine language. Sometimes we need to perform vice versa. Process so we need a disassembler.

* It is also software that converts a machine language program to an assembly language

Passes of assembler

Assembler can take to passes to convert the assembly language program into machine language program such as.

① One pass assembler
② Two pass assembler

One pass assembler → An assembler which goes through an assembly language program only once is known as one pass assembler.

Two pass assembler → An assembler which goes through an assembly language program twice is called a two pass assembler.

→ During the first pass it collects all labels and during the second pass it produces the machine code for each instruction

advantages of assembly language over high level language.

① Computation time is less.
② It is faster to produce result.

## Disadvantages of assembly language Compared to High level language

(1) Programming is difficult and time Consuming
(2) assembly language is computer Oriented.
(3) Assembly language is not Portable. (a) Program written for one micro Computer cannot be used for any other because each micro computer has its own assembly language.
(4) Assembly language is longer as Compared to high level language Program.

## High level language

→ difficulties of assembly languages can be overcome using high level languages. Instructions written in high level languages are called statements. Examples of high level languages are COBOL, BASIC, C, PASCAL, C++.

→ In micro Computer System high level language is mostly used due to the following reasons.
(1) Hardware and memory are becoming less expensive.
(2) Software and Programmes are becoming more costly.
(3) Compilers are easily available.

Advantages (1) instructions are very clear
(2). It is faster and easier than assembly language.

(3) It is portable (ie) they can run on any Computer

(4) Easier documentation.

Disadvantages

* One has to learn the special rules for writting programs in a particular high level languages
* low speed.
* High level language takes more time to Produce result.
* Extensive hardware and software supports are required.

# Addressing Modes of 8085

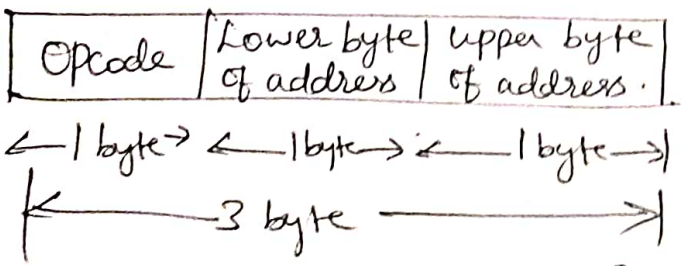
* The method of Specifying the data to be Operated by the instruction is Called Addressing.
* Addressing mode is the method by which the microprocessor can access the register or memory
* 8085 has four basic addressing modes to address the data Stored in memory or register

1. Direct addressing.
2. Register addressing
3. Register indirect addressing
4. Immediate addressing
5. Implicit addressing

## (1) Direct addressing.

* The direct addressing mode is used when the Operand is available at some memory or an IO location.
* In this addressing mode the address of the Operand (data) is given in the instruction itself.
* Which contains the 16 bit address of a memory location where from the data is to be accessed.

The instruction format of direct addressing mode is




Example STA 2400H – Store the Content of the accumulator to memory location 2400H

② IN 02 → Input the data from input port 02H in accumulator.

③ LDA 1050H → Load the data available in memory location 1050H in accumulator.

④ JMP 2010H → Jump to memory location 2010H.

## ② Register addressing.

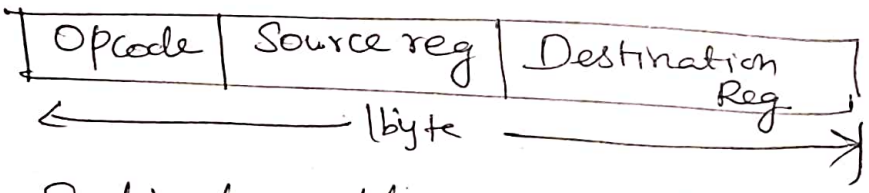
→ In register addressing the instruction specifies the name of the register in which the data is available.

→ In this mode the operands are in the general purpose register.

eg: MOV A,B → Move the content of B register to A register

ADD B → Add the content of B register to that in the accumulator.

CMP B → Compare the content of B register with that of accumulator.




## ③ Register Indirect addressing mode:

→ In the register indirect addressing the instruction specifies a register pair which contains the 16 bit address of a memory location where from the data is to be accessed.

→ The first register H, B, D contains the higher order byte and second register of register pair contains the lower-order byte of the address.

Example LXI H, 2500H – load HL pair with 2500H.

MOV A,M. → Move the Content of the memory location whose address is in H-L pair (i.e 2500 H) to the accumulator.

HLT → Halt.

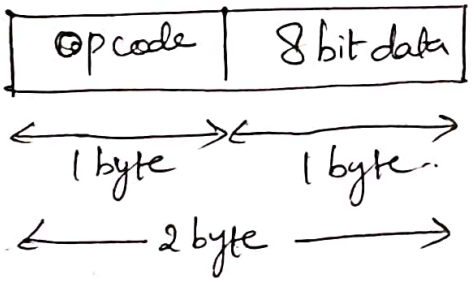
(4) Immediate addressing

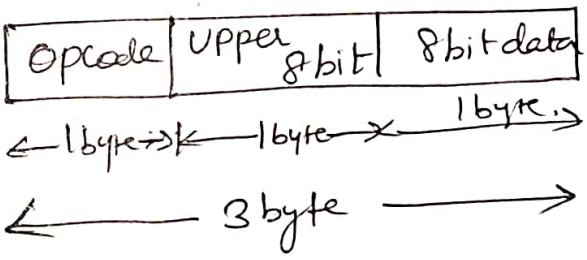
→ In immediate addressing mode the operand is Specified within the instruction itself. (i.e) the data is Stored along with the instruction.

MVI, A, 05 → Move 05 in register A.

ADI 02H → Add the data 02H with accumulator Content.

→ Instruction format of instructions with immediate addressing mode. is



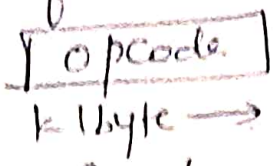




Implicit Addressing (or) Implied Addressing

→ In implicit addressing mode the instruction itself Specifies the data to be operand.

→ There are certain instructions which operate on the content or the accumulator directly these instructions don't require Specifying the operands.

instruction format of implied addressing mode is.




Eg. CMA → Complements the Content of accumulator
RAR → Rotate the Contents of the accumulator right by one bit
RAL → Rotate the Contents of the accumulator left by one bit.
STC → Set Carry. - Operate only on the Carry flag.

## Assembly language format:

→ The microprocessor development System Consists of a set of hardware and software tools

→ Software tools are also Called Program development tools.

→ Software tools can be run on PC in order to write, assemble, debug, modify and test the assembly language Program.

→ Assembly language is an example - A Programmer can comfortably write a Program in Specific alphanumeric Symbols instead of binary Codes.

→ The alphanumeric Symbols for different Command in meaning fully performance Oriented and easy to remember.

eg. ADD for addition.
SUB for Subtraction.
CMP for Comparision

→ These Symbols are Called ⑨ Mnemonics. The Program written with mnemonics are Called assembly language Programs.

→ Assembly is a software which translates an assembly language Program into a machine language Program. There are two types of Assembler.

① One pass assembler ② Two Pass assembler.

→ An assembler which Complies an assembly language Program only once is known as One-pass assembler.

→ An assembler which passes through an assembly language Program twice during Compilation is called a two pass assembler.

## Advantages of Assembler.

① The assembler translates mnemonics into binary Code with speed and accuracy thus eliminating human errors in looking up the Codes.

② The assembles assign appropriate Values to the Variables Used in a Program.

③ It is Easy to insert or delete instructions in a Program and reassemble the entire Program.

④ The assembler checks Syntax errors, such as Wrong Labels, opcodes, expressions etc.

## Format of assembly language in 8085

Mnemonics - which represents the actual instruction

Opcodes - opcode stands for operation code. (e) a machine Code instruction

Operands → Additional information required by the opcode (ie) data.

Labels → The label is used in the Program as a reference for a memory location.

→ Each Instruction has Two parts.

① One is the task to be performed Called operation code (opcode).

② Second is the data to be operated on called the Operand.

## Programming Techniques : Looping, Counting and Indexing.

→ Programming techniques are the most important in any microprocessor. Because Using the program only we can perform all kind of tasks.

**Looping** : * The programming technique used to instruct to the microprocesor to repeat task is called looping.

* A loop is a set up by instructing the microprocesor to change the sequence of execution and perform the task again. This process can be achieved by jump instruction

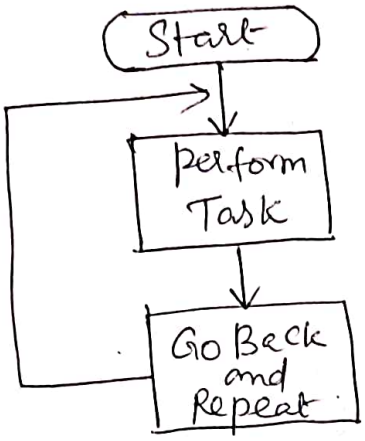
**Types:** It can be divided into two types

① Continuous loop → repeats a task continuously

② Conditional loop → repeat a task until Certain data Conditions are met.

## Continuous loop:

A Continuous loop is set up by using the Unconditional Jump Instruction. It is shown below




flow chart of a Continuous loop.

A program with a Continuous loop does not stop the repeating process until the system is reset.

Example * Continuous Counter
* Continuous Monitor System.

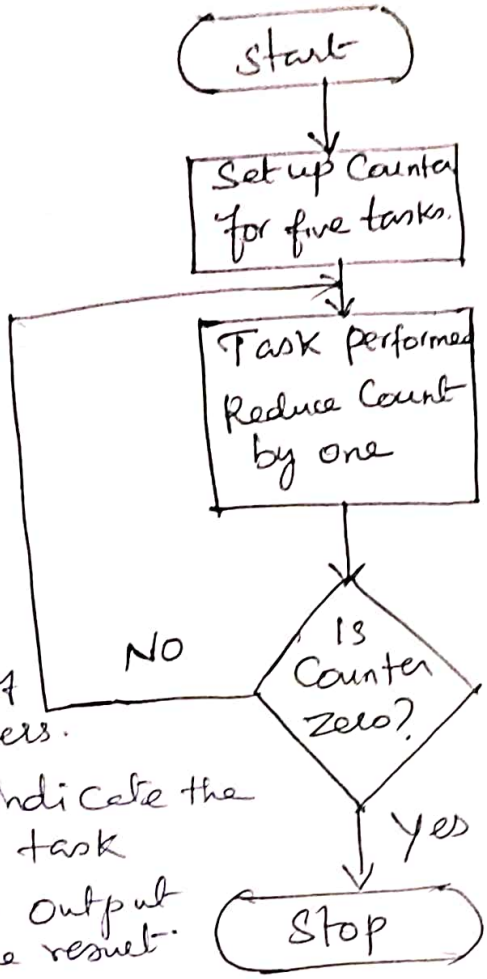
## Conditional loop.

* This loop can be set by the Conditional Jump instructions. These instructions check flags and repeat the specified tasks if the Condetions are Satisfied.
* It includes the Zero flag, Carry flag etc.

## Conditional loop with Counter.

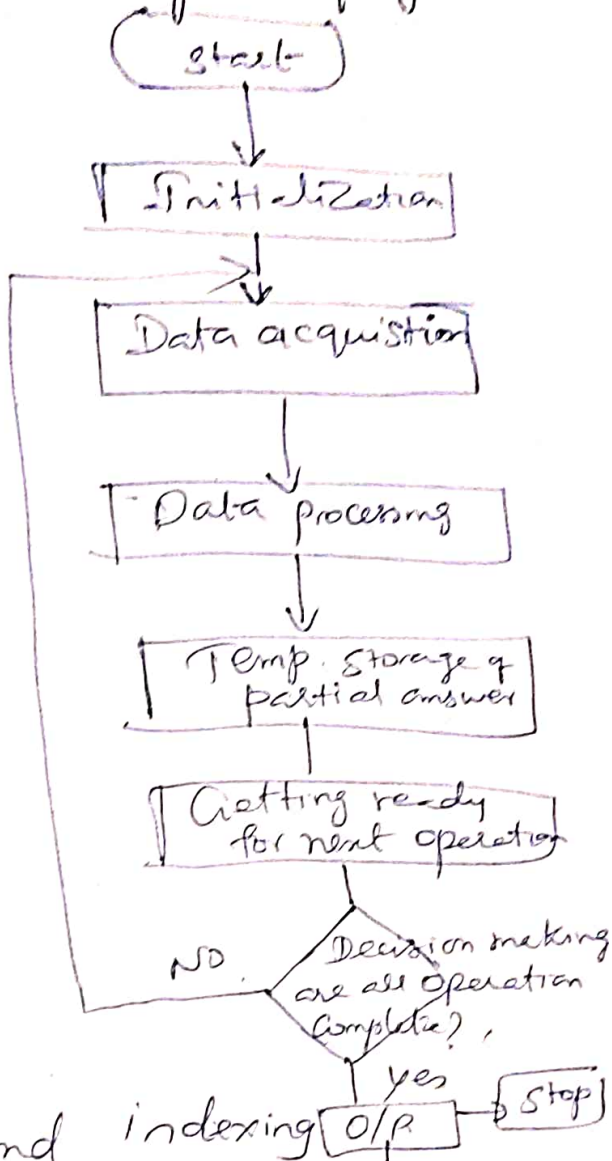
* Counter is a Variable used to Count the event or task done in the Program. A Counter is a typical applications of the Conditional loop.
* µp needs a Counter to perform looping task. When the counting is Completed, it needs a flag.
* following flowchart will have the following tasks.
  ① Counter is set-up by loading an appropriate Count in a register.

(2) Counting is performed by either incrementing or decrementing the Counter.
(3) Loop is set-up by a Conditional Jump instruction
(4) End of Counting is indicated by a flag.




(5) Registers for temp. Storage of partial answers.
(6) A flag to indicate the Completion of task
(7) To store or output the result.

Task Repetitions.




## Conditional loop with Counter and indexing

* Another type of loop includes indexing along with a Counter
* Indexing is the process of referring objects with sequential numbers for example in a lib, books are arranged according to numbers and they are referred to or sorted by numbers. This is called Indexing.

**Example** Write steps to add ten bytes of data stored in memory locations starting at a given location and display the sum. Draw a flow chart also

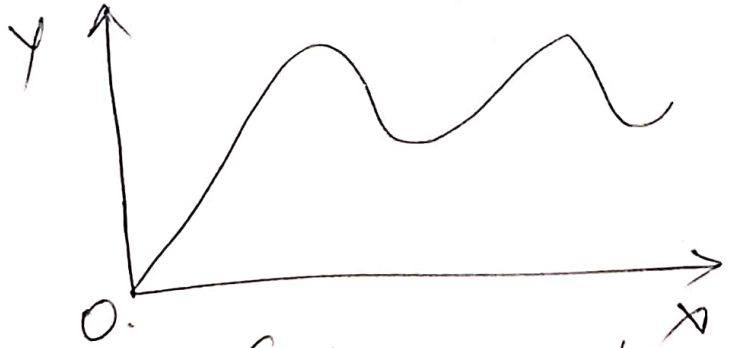
**Procedure**
(1) A Counter to Count 10 data bytes
(2) An index or a memory pointer to locate where data byte are stored. (3) To Transfer Data from memory location to µp (4) To perfom addition.

## Lookup Table

* The lookup table in a microprocessor is a system that helps replace the runtime computation.
* It replaces them with an easier index operation
* It can be used to search for both letters and numbers.
* It means the memory location placed in the register of a microprocessor.
* Table containing various combination of input and output desired for that particular combinations. It is similar to truth table. These lookup tables abbreviated as LUT.
* It is widely used to define data for non-linear expression.

for example

① If we need to get the output Y for the input (X) in the below mentioned plot.




Continuous non linear wave-form.

② The above plot can't be represented by simple or linear

③ The given plot is non-linear function.

④ Suppose the o/p of the plot is already known means those values saved in a table for the particular input.

(5) The Programmer can access those o/p values for the particular inputs.

→ Lookup table is a reference table.

⇒

## Program 1 for lookup table

Square of a number using lookup table.

Algorithm: (i) Initialize HL pair to point lookup table, get the data.

(ii) Check whether the given input data is less than 9.

(iii) Add the desired address with the accumulator Content.

(iv) Store the result.

| Label | Mnemonics | | Comments |
|---|---|---|---|
| | LXI H, | 5000 | Initialize lookup table address. |
| | LDA | 5050 | Get the input data. |
| | CPI | 0A | Check input > 9. |
| | JC | Level. | Condition Satisfied go to Level. |
| | STA | 5051 | |
| | HLL | | |
| Level. | MOV | C, A. | Add the desired address. |
| | MVI | B, 00 | |
| | DAD | B | |
| | MOV | A, M | |
| | STA | 5051 | Store the result. |
| | HLT | | Terminate the Program. |

## Lookup Table

| Memory location. | Data. |
|---|---|
| 5000H | 01 |
| 5001H | 04 |
| 5002H | 09 |
| 5003H | 16 |
| 5004H | 25 |
| 5005H | 36 |
| 5006H | 49 |
| 5007H | 64 |
| 5008H | 81 |

**Result** → Input Data 05H in memory location 5050H.

→ Data 25H (square of 5) in memory location 5051H.

**Explanation.** → In the above example by using lookup table to find the square of the number

→ The result of square is stored into the memory

→ The square numbers are stored in lookup table with memory location from 5000 to 5008.

## Subroutines.

→ A Subroutine is a group of instruction that will be used repeatedly in different locations of the program. Rather than repeat the same instructions several times, they can be grouped into a Subroutine that is called from the different locations.

→ In assembly language a Subroutine can exist anywhere in the code.

→ Intel 8085 has 2 instructions for dealing with Subroutines.

① The CALL instruction.
② The RET instruction.

→ CALL instruction is used to redirect Program. Execution to the Subroutine.

→ The RET instruction is used to return the execution to the calling routine.

→ Call is used to Call a Subroutine and the Call instruction is written in the main Program.

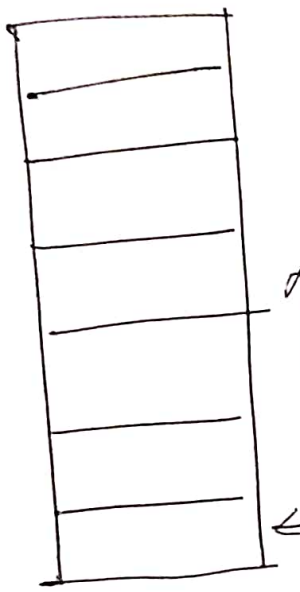
→ Return is used to return from the Subroutine and return instruction is written in the Subroutine to return to the main Program.

## STACK

→ The Stack is a group of memory locations in the R/W memory that is used for temporary storage of binary information during the execution of a Program.

→ The Stack is a LIFO Structure. - Last in First Out

→ Stack normally grows backwards into memory It means the Programmer defines the bottom of the stack and stack grow up into reducing address range Memory.




→ In the 8085, the stack is defined by setting the stack pointer register. LXI SP, FFFFH, This sets the Stack pointer to location FFFFH

→ The Size of the Stack is limited only by the available memory.

## Storing Data on the Stack

→ Information is Saved on the stack by Pushing it on

→ Information is retrieved from the stack by Poping it off.

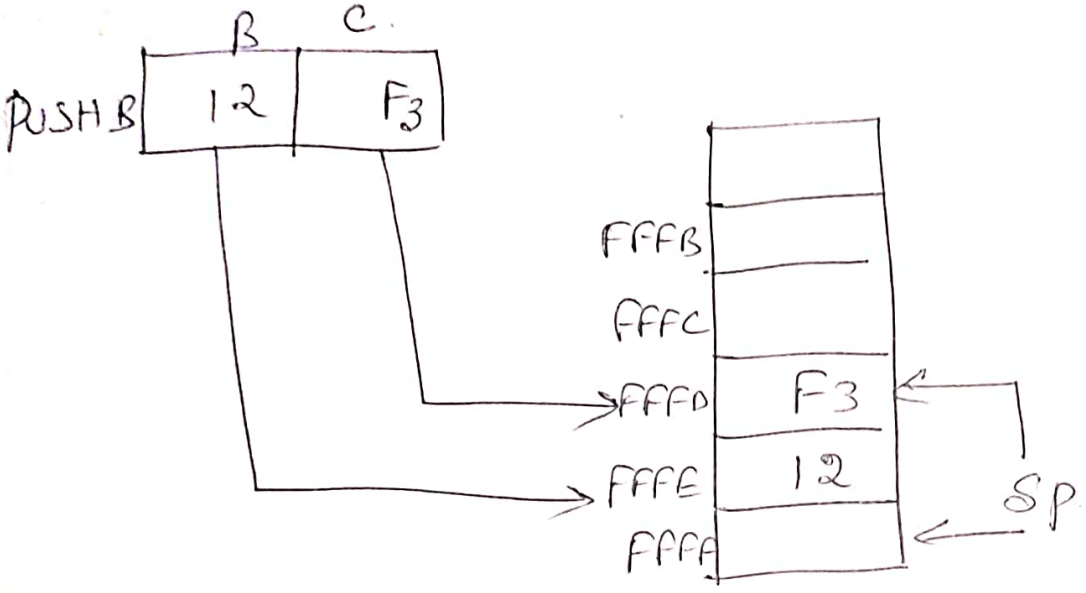
→ 8085 provides two instructions

① PUSH and → it is used to Store the data

② Pop → and Pop is used to retrieve the data from the stack

PUSH instruction → it will perform the following task

1. Decrement the stack pointer.
2. Copy the contents of register B to the memory location pointed to by stack pointer
3. Decrement the stack pointer.
4. Copy the contents of register C to the memory location pointed to by stack pointer




PUSH Instruction.

# Data Transfer Schemes.



In a microprocessor based system or in a Computer data transfer takes place b/w two devices such as

① Between microprocessor and Memory.
② Between Microprocessor and I/o devices.
③ Between Memory and I/o devices.

→ A microProcessor based system or a Computer may have several I/o devices of different Speed.

Data Transfer scheme

- Programmed Data Transfer scheme
  - Synchronous Data Transfer scheme
  - Asynchronous Data Transfer scheme
  - Interrupt driven Data Transfer scheme.
- DMA Scheme
  - Burst mode of DMA data Transfer
  - cycle Stealing Technique of DMA Data Transfer.

→ So a slow I/o devices cannot transfer data when microprocessor issues Instruction for the same. because it takes some time to get ready.

→ To Solve Such Kinds of Problem of Speed mismatch b/w a microprocessor and I/o devices a number of data transfer techniques have been developed.

The data transfer schemes are classified into Two types

① Programmed Data Transfer schemes.

② DMA (Direct Memory Accen) data Transfer schemes.

## Programmed Data Transfer Schemes.

* These schemes are Controlled by the CPU. Data are transferred from an I/O. devices to the CPU or vice versa.

* Data are transferred under the Control of Programs which reside in the memory. These programs are executed by the CPU when an I/O device is ready to transfer data.

* Microprocessor executes the Program to Transfer data. This scheme is suitable for transferring, small amount of data.

## Synchronous Data Transfer scheme.

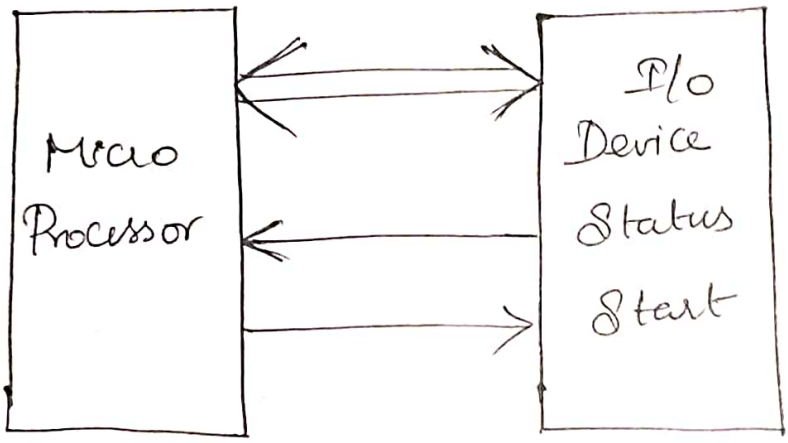
→ Synchronous means "at the same time". The data sending device and recaving device are Synchronized with the same time.

→ When the speed of the CPU and I/O devices will be matched, this technique of data transfer is used.

→ The transfer of data with I/O devices. is Performed by executing IN or OUT instruction for I/O mapped I/O devices.

* IN instruction is used to read data from an input device or input port.
* OUT instruction is used to send data from the CPU to an Output device or Output port.
* The I/O devices Compatible with microprocessors in Speed are usually not available so this data Transfer technique is rarely used for I/O devices.

## Asynchronous Data Transfer Scheme.




Asynchronous Data Transfer.

* Asynchronous means "at irregular intervals" In this method data b/w Input and output device is not based on pre determined timing pattern.
* This method is used when the Speed of an IO Device does not match the Speed of microprocessor and the timing characteristic of I/O devices is not Predictable.

## Handshaking Mode:

* In this method microprocessor will check the status of the I/O devices whether the devices is ready or not.
* The microprocesor initiates the I/o device to get ready and then Continuously checks the status of the I/o device till the I/o Device becomes ready to transfer data.
* When I/o device becomes ready, the micro Processor sends instructions to Transfer data. This method of data transfer is called "Hand Shaking Mode".

## Hand Shake Signals

* In hand shaking mode data transfer takes place because of some signals are exchanged b/w I/o device and microprocessor before the actual data Transfer takesplace.
* μp issues an initiating signal to the IO device to get ready.
* When I/o device becomes ready it sends signals to the Processor to indicate that it is ready. Such signals are called hand shake signals.

## Draw back

→ It is used for slow I/o devices. It is an inefficient because the precious time of microprocessor is wasted in waiting.

# Interrupt Driven Data Transfer

→ When microprocessor receives the interrupt it Completes the current instruction and then attend the I/O device.

→ Once microprocessor receives the interrupt it saves the Contents of the Program Counter on the Stack first and then takes up a Subroutine called ISS. (Interrupt Service Subroutine).

→ After Completing the data transfer the microprocessor returns back to the main program which it was executing before the interrupt occured.

## Advantages.

→ It is used for slow I/o devices

→ It is an efficient technique Compared to asynchronous data transfer method. because precious time of the microprocessor is not wasted in waiting while and I/o device is getting ready

## Multiple Interrupts:

→ In a microComputer System Several I/o devices Can use interrupt driven data transfer scheme

→ While interfacing I/o devices Using interrupts the following situations may arise.

① A microprocessor may have only one interrupt level and Several I/o devices are to be Connected to it.

# DMA Data Transfer scheme

→ In this method CPU does not participate Data are directly transferred from an I/O device to the memory or vice versa

→ Data transfer is controlled by the I/O Device or DMA Controller. In this scheme used when large amount of data are to be Transferred.

→ An I/O device wants to send data using DMA technique means it sends the HOLD signal to the CPU

→ CPU will receive the HOLD signal from an I/O device and CPU gives up the Control of buses as soon as the Current machine cycle is Completed

→ CPU sends a hold acknowledge signal to the I/O device to indicate that it has received the HOLD request and it has released the buses

→ I/O device takes over the Control of buses and directly transfers data to the memory or reads data from the memory.

## Advantages

→ It is faster than Programmed data transfer scheme

→ It is used for high Speed Printers

→ It is used to transfer data from mass Storage devices Such as floppy disks, hard disks etc.

## Burst mode of DMA Data Transfer.

* In DMA data transfer the I/O device withdraws the DMA request only after all the data bytes have been transferred is called burst mode of data transfer.
* Using this method we can transfer the block of data and it is used in magnetic disks drives

## Cycle Stealing Technique

* In this technique a long block of data is transferred by a sequence of DMA cycles.
* In this method after transferring one byte or several bytes, the I/O device withdraws DMA request.

## Drawback

* In DMA data transfer the I/O device must have registers to store memory addresses and byte count.
* It must have electronic circuit to generate necessary control signal for DMA operations.
* But I/O devices do not have these facilities To solve the Problem DMA Controllers have been designed.

DMA Controlla Chip are Intel 8237 A, 8257 etc.

# Terms Used in Microprocessor

Byte : 8 bit (8 digit) binary number or code is called byte.

Word : 16 bit.

Double word : 32 bit (32 digit).

Multiple word : 64, 128 - bit/digit

## Evolution of micro processors:

**First Generation** Micro processor This technology provided low cost, slow speed and low o/p current

→ it requires a lot of additional support IC to form a system.

→ INTEL 4004, INTEL 4040, MICRO SYSTE INTL. MC-

INTEL 8008 ROCKWELL PPS-8

**Second Generation** → (8 bit micro processor) INTEL 8080, INTEL 8085. MOTOROLA M6800., ZILOG Z80.

→ larger chip size., 40 pins.

**Third Generation.** (16 bit Micro Processor) INTEL 8086, INTEL 8088, INTEL 80186 INTEL 80286.,

→ high speed, easier to Program.

**Fourth Generation** → This fourth generation Processors are 32-bit processors.

INTEL 80386. INTEL 80486., MOTOROLA M68020

→ It supports increased number of addressing modes.

→ physical memory Space of 16 MB

# Definition of Micro Processor

- Microprocessor is a Programmable, clock driven, Register based electronic device. that reads instruction from a Storage device, takes the data from input unit and Process the data according to the instructions and Provides the result to the output unit.

# UNIT-III

## 8051 Micro Controller

### Introduction

**Microprocessor** → it Can be Understood as the heart of the Computer System. It is a Processor where the memory and I/o Component are Connected externally The circuit is Complex due to external Connection. It Can't be Used in Compact System. It is not efficient. It has less number of registers. Most of the Operations are based on memory. It has a Zero Status flag. It is generally Used in Personal Computers.

**Micro Controller** → It Can be Understood as the heart of the Embedded System. It is a Controlling device wherein the memory and I/o output Component are Present internally. It is present on chip memory. The memory and I/O Components are available. The circuit is less Complex It is efficient. It is generally Used in Washing machine and air Conditioners.

### 8051 Micro Controller

→ The 8051 is the first micro Controller of the MCS-5 family. Introduced by Intel corporation

→ Microcontroller Can be classified on the basis of their bits Processed like 8 bit MC., 16 bit MC

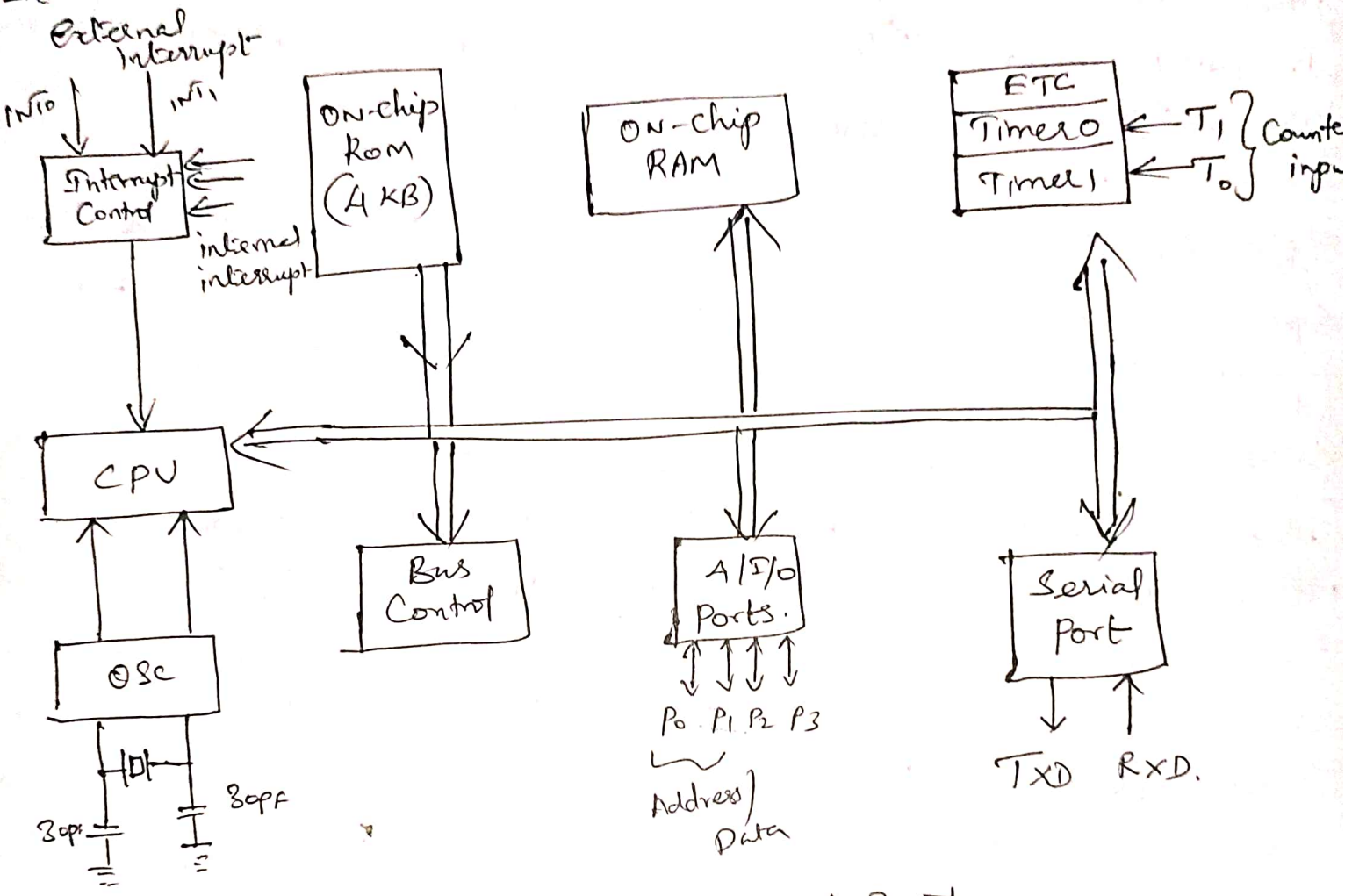
### Features.

→ 8 bit CPU
→ On chip Oscillator
→ 4 KB of Rom. (Program memory).
→ 128 bytes of RAm (Data Memory).

→ 21 special functions registers
→ 32 I/O lines.

## Block diagram of 8051

→ 8051 Micro controller is a 8 bit micro controller.
It can read, write and process 8 bit data.




Block diagram of 8051.

→ Micro controller 8051 has 128 byte RAM for data Storage. RAM is nonvolatile memory. During execution for storing the data, the RAM is used.

→ In 8051 4KB ROM is available for Program storage. This is used for permanent data storage or data which is not changed during the processing like Program or algorithm.

→ **Serial port** : There are two pins are available for Serial Communication TXD and RXD.

**Input output port** → There are four input output ports are available. $P_0, P_1, P_2, P_3$., each port is 8 bit wide and has special function register $P_0, P_1, P_2, P_3$.

**Oscillator** → It is used for providing the clock to MC 8051 which decides the speed or baud rate of MC

**Interrupts** It is defined as requests because they can be refushed (masked) if they are not used, that is when an interrupt is acknowledged.

**Pin Diagram of 8051**

→ 8051 is available in 40 pin dual in line package

**Power Supply** 8051 is operated by using +5V supply and Vcc is the power supply pin and Vss is the ground pin

**Pin diagram of 8051**

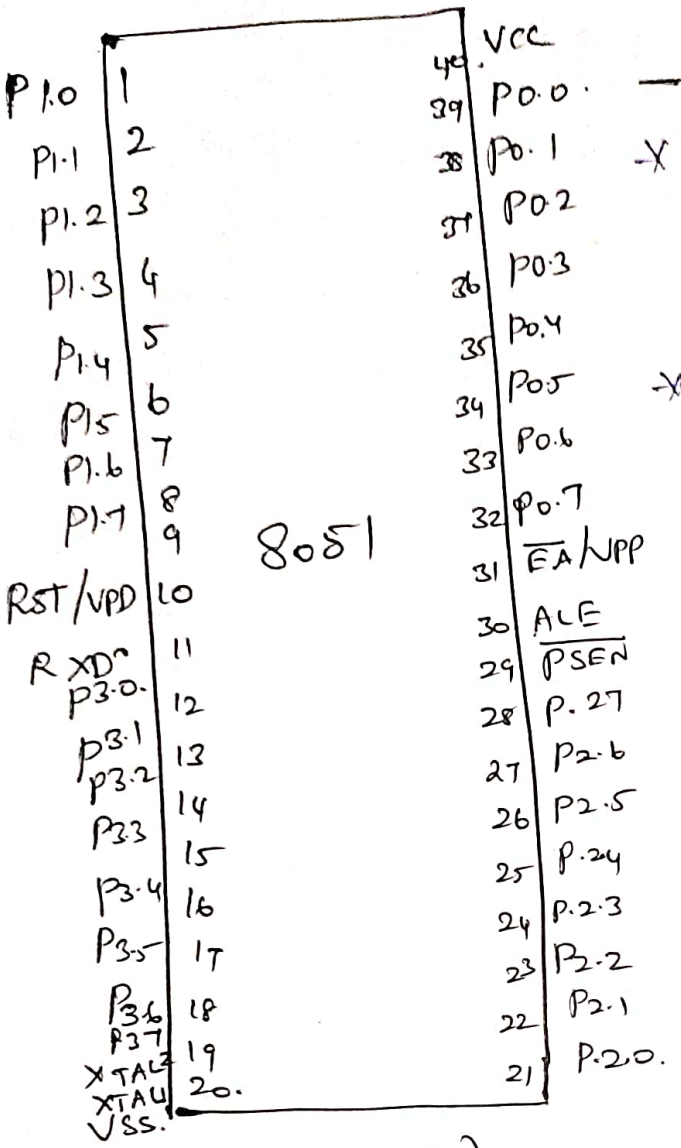
→ 8051 is available in 40 Pin dual in line Package. Various pins are given

Power Supply (Vcc and Vss).

8051 is operated by using +5V supply and Vcc is the power supply pin and Vss is the ground pin.

Port C P0.0 to P0.7).

→ There are 8 bit bidirectional input/output port pins. They are bit addressible.




## Port 1 (P1.0 to P.1.7)

- Port 1 is an 8 bit bidirectional input/output with the internal pull-ups.
- It plays an important role during the Programming of the internal memory of 8051 and 8751.

## Port 2 (P2.0 to P.2.7)

→ This is also an 8 bit bidirectional I/O with internal pull ups.

→ This also has an alternate function of higher order address byte while accessing the external memory

## Port 3 (P3.0 to P.3.7)

→ This is also an 8 bit bidirectional I/O with internal pull-ups

→ It also serves many other important alternate functions which enable the 8051 to be called a micro Controller

→ The alternate functions of Port 3 pins are as follo

P3.0 → RXD (Serial Input).
P3.1 → TXD (Serial Output).
P3.2 → $\overline{INT_0}$ (External Interrupt).
P3.3 → $\overline{INT_1}$ (External Interrupt)
P3.4 → $T_0$ (Timer 0 External Input)
P3.5 → $T_1$ (Timer 1 External Input)
P.3.6 → $\overline{WR}$ (External Data memory write Strobe)

# Central Processing Unit (CPU)

It is the heart of the micro controller. It consists of
(1) Accumulator (Acc) (2) B register (3) Stack pointer
(4) Program Counter (5) Program Status word (PSW).
(6) Data Pointer register (DPTR)

## Accumulator

→ It is an 8 bit register. It is usually referred as A.
It is a very important register to store the result of data operations such as addition, Subtraction multiplication, division. and for logical operations such as AND, OR, XOR..
→ It is bit and byte addressable.

**B register** This register is mainly used to multiply and divide operations in which this register acts as one of the operand and after the operations a part of the result is stored in this register.

## Stack pointer (SP).

→ It is an 8 bit register. This pointer can point to any location in

## Arithmetic Logic Unit (ALU)

→ It is used to perform arithmetic and logic instructions, such as addition, Subtraction, multiplication division and logical AND, OR, EXOR, bit operations.

## PSW - Program Status word

* It is an 8 bit register. PSW bit pattern

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| C | AC | F0 | RS1 | RS0 | OF | - | P |

P3.7 → $\overline{RD}$ (External Data memory Read strobe).

**ALE / $\overline{PROG}$** → ALE means Address Latch Enable.
PROG means program pulse.

**XTAL1 and XTAL2** The 8051 has an Internal Clock Circuit hence a crystal of Proper frequency Can be Connected directly to these two Pins.

**PSEN** → It means Program Story Enable. This Pin gives out active low output pulses.
→ This signal is Used for fetching the data from the External program memory.

**EA / VPP** → It means external access / Programming voltage
→ This Pin if Connected to 5V, will indicate to the Processor that the 4 KB of the Program memory within the chip should be used.
→ Address 0000H to 0FFFH will be within the chip

**RST / VPD**

A high on this Pin for more than 24 Oscillator cycles will reset the chip.

## Architecture of 8051

Functional organization of the micro Controller 8051
This is a basic architecture for all the members of 8051 family But they may have more or less Comparators. ADC Circuits, Ports. etc.

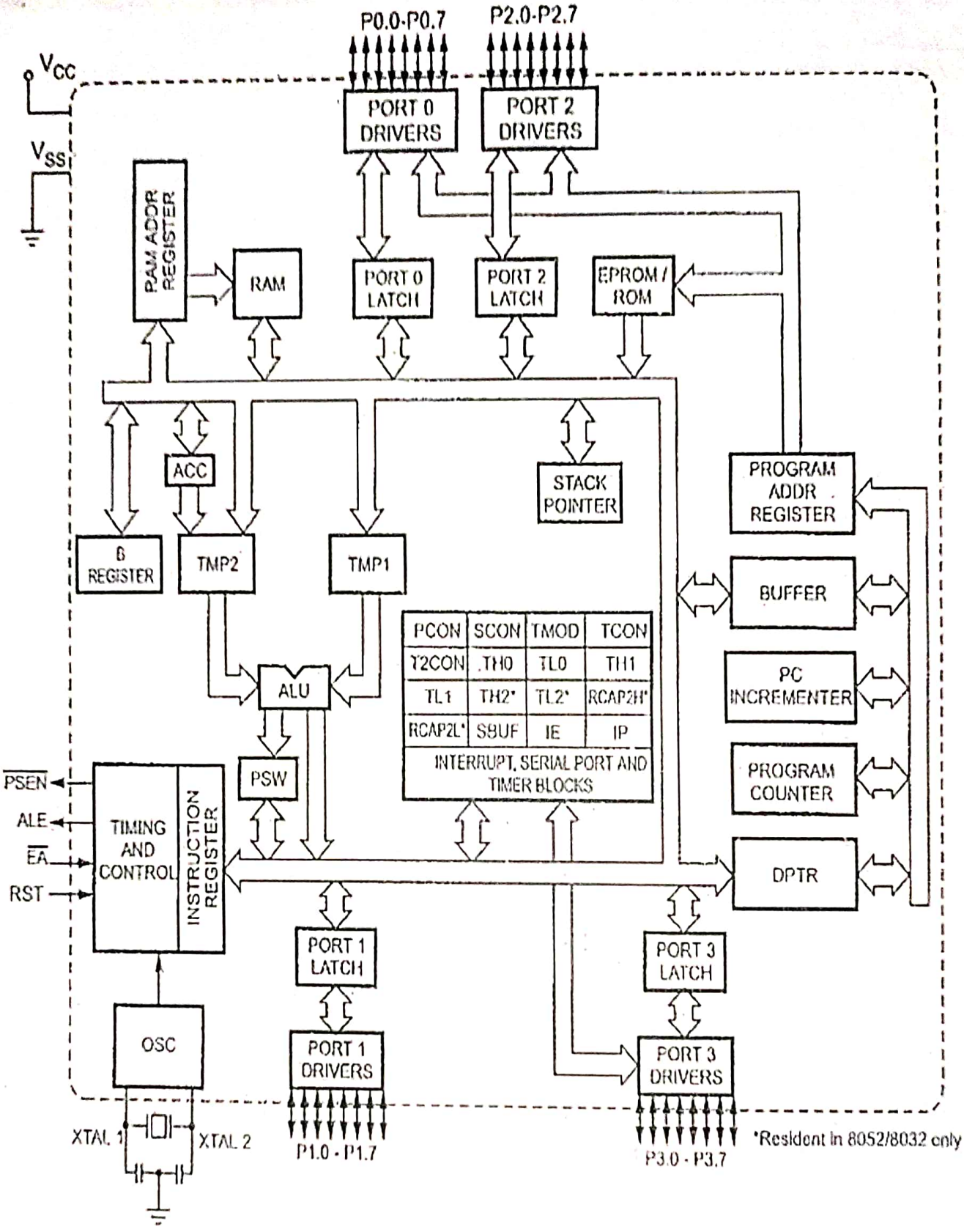



Fig. 12.2 Intel 8051/8031 architecture

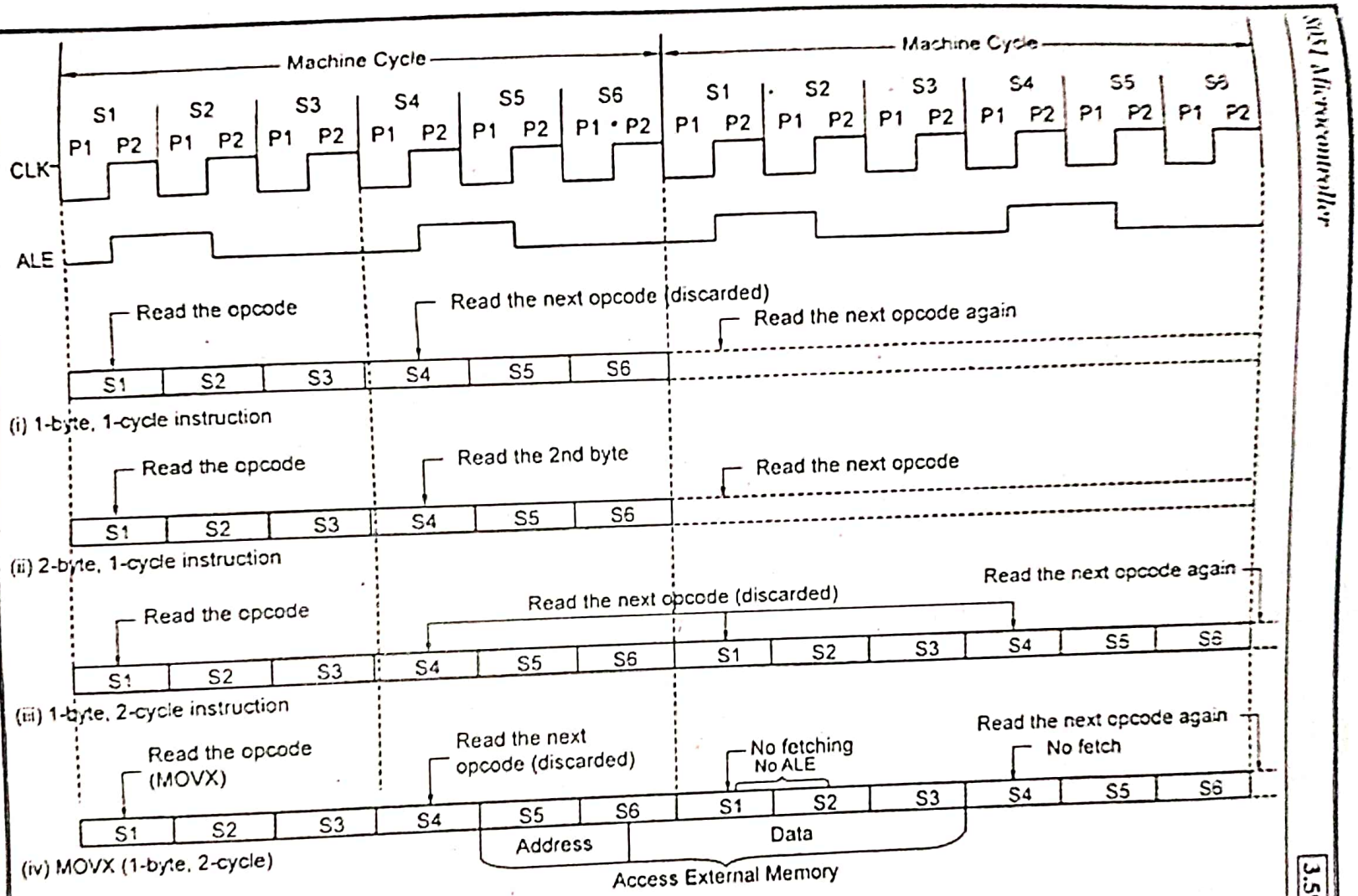



*Fig. 3.13. Timing Diagram for Instructions executed from Internal Memory*

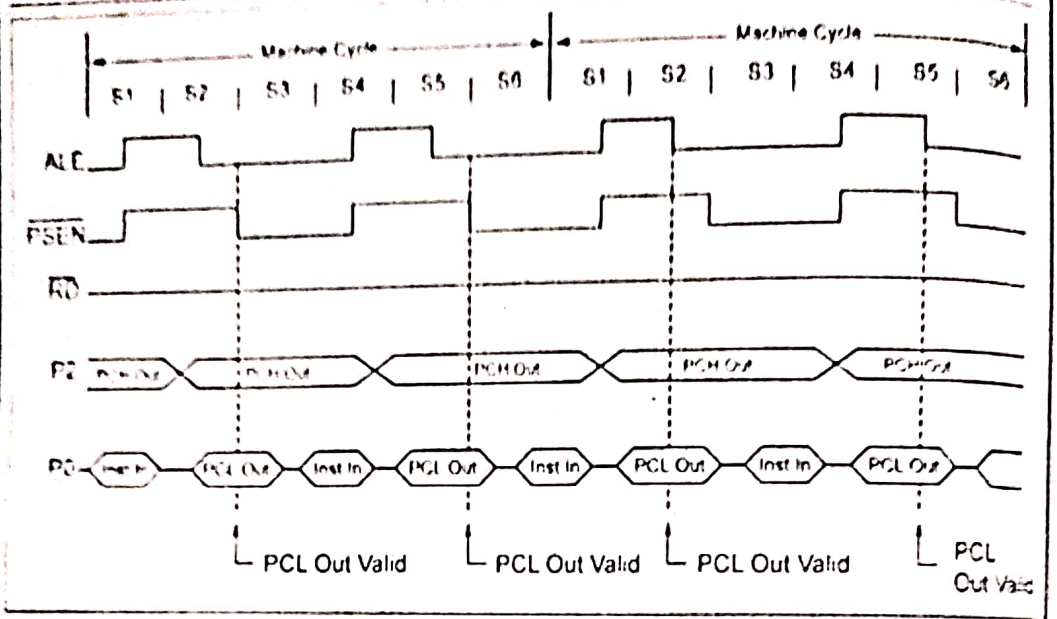



*Fig. 3.14.(i) Execution of Instructions other than MOVX*

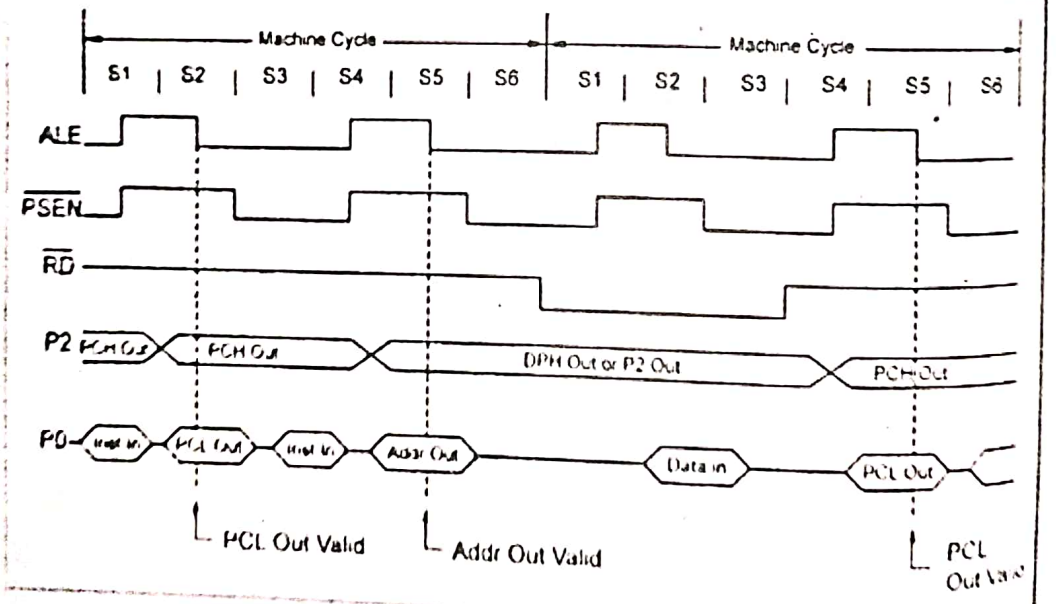



*Fig. 3.14. (ii) Execution of MOVX Instructions*

P0 = Port 0

P2 = Port 2

**Carry flag:** It is set C=1 if the result contains Carry.
It is reset C=0 if the result does not Contain any Carry.

**Auxiliary Carry flag.** It is used in BCD operations.

**Overflow flag.** If there is any overflow in the result then OF is set (OF=1) otherwise it is reset (OF=0).

**Parity flag** If the result contains odd number of one's then P is 1 (set) otherwise it is reset.

**RS0, RS1** RS means Register Bank Select bits. It is used to select the register banks.

**Program Counter** It is a 16 bit register. It is used to store the address of instruction which is to be executed next.

**Data pointer.** It is a 16 bit register. It can also be used as two numbers of 8 bit registers.
It is used to fetch any 8 bit data from the data memory space.

**Input/output ports.** In 8051 totally 4 number of 8bit ports. are available. These are port0, port1, port2, port3.

**Timer/Counters.** 8051 contains 16 bit timer/ counters.
TMOD is a mode control register. TCON is a timer control register.
→ These are used to configure the timer/ counters. in various ways.

# 8051 I/O Ports

→ In 8051, totally 4 Ports are available. These are $P_0$, $P_1$, $P_2$, $P_3$. These are 8 bit ports. If the first 0 is written to a port it will become o/p port. If the first 1 is written to a port it will become input port.

## Port 0

It can be used as input or output port. Each pin must be connected externally to a 10kΩ Pullup resistors because Po is an open drain bidirectional I/o port.

it can be used in two ways.

1. It can be used as simple I/o port
2. It can be used for external memory interface for lower order address and data bus.

## Port 0 as Input

If we want to use port 0 as input port then the port is to be programmed by writting 1 to all the bits. By using the following Program port 0 is Configured as input port.

## Port 1

It is not necessary to connect any Pull up resistors because it is having Pull up resistors already.

## Port 1 as input

Upon reset Port 1 is Configured as an input port. If it is used as an o/p port then 0 is written to a port. Then once again if we want to use it as an input port, then 1 is written to all its bits.

# Memory Organization.

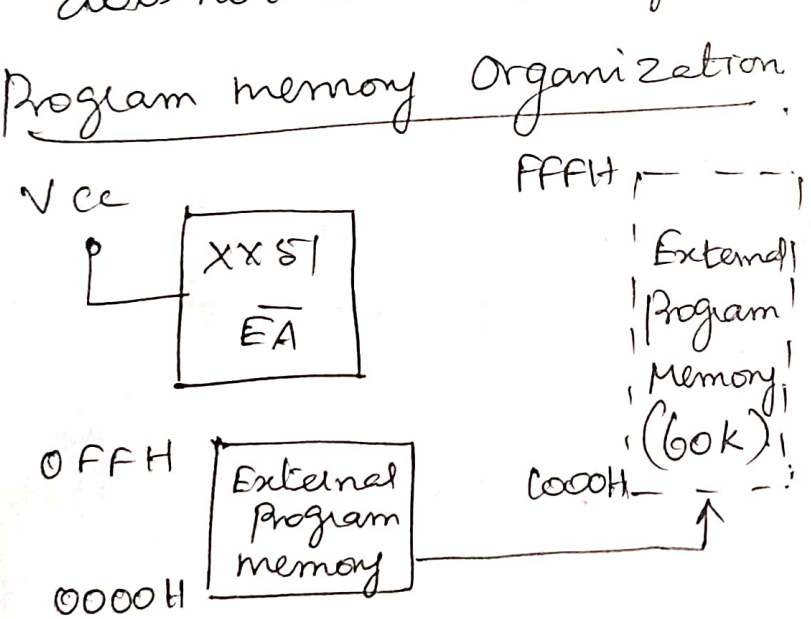
Micro Control 8051 memory Can be organised into two Variations

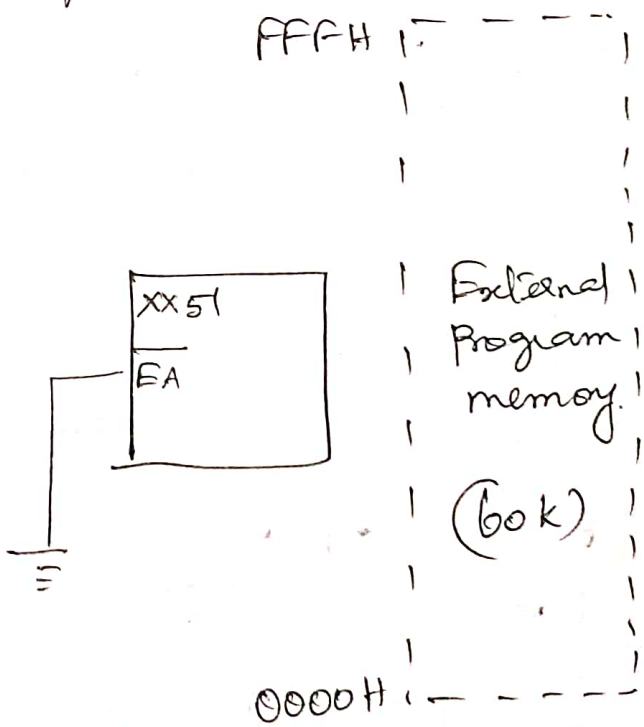
① Princeton Architecture
② Harvard architecture.

① Princeton architecture → it treats address memory and data memory as a single unit, µp 8085 is an Example for Princeton architecture.

② Harvard architecture → it treats Program memory and data memory as separate entities. Thus Harvard architecture demands address, data and control bus for accessing them seperately. where as Princeton architecture does not demand any such Separate bus.

## Program memory Organization




Program memory Arrangement.

→ It has an internal program of 4K size if needed an external memory Can be added of size 60k. maximum. so total we have 64 kb. Size memory available in 8051.

→ By default the external access pin should be connected Vcc so that instructions are fetched from internal memory initially.

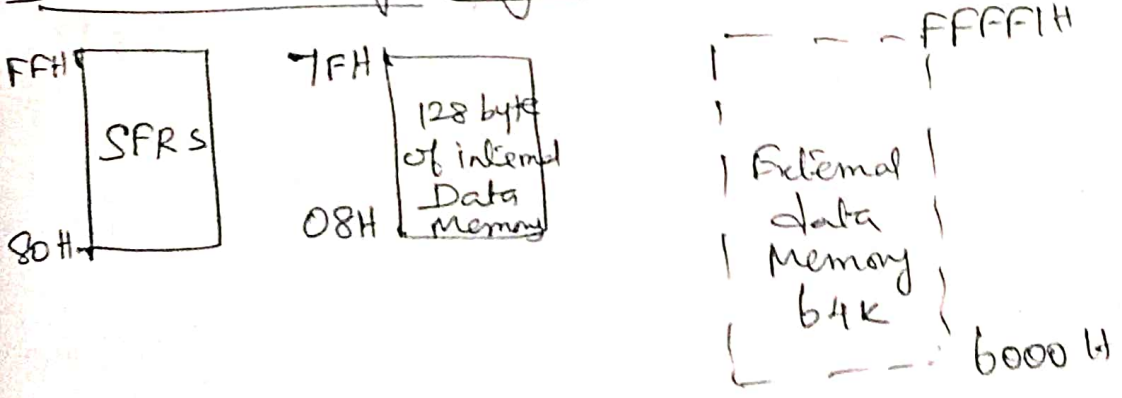
→ When the limit of internal memory is crossed control will automatically move to external memory to fetch remaining instructions.

## Features of Program memory

→ Once the Program lock bits are Programmed contents of internal memory cannot be accessed by using an external circuitry. Only internal memory can be locked and protected. Once locked these bits can be unlocked only by a memory erase operation.

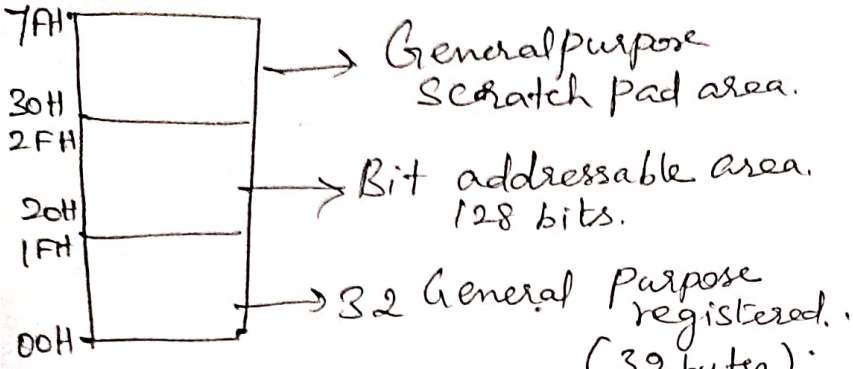
**Pipelining** :→ Micro controller 8051 is capable of Pipelining. Pipelining makes a processor capable of fetching the next instruction while executing previous instruction, doing more than one operation at a time.

## Data Memory Organization




Internal and External Data memory.

Data memory can be separated three parts.
① Register banks ② Bit addressable area
③ Scratch pad area.

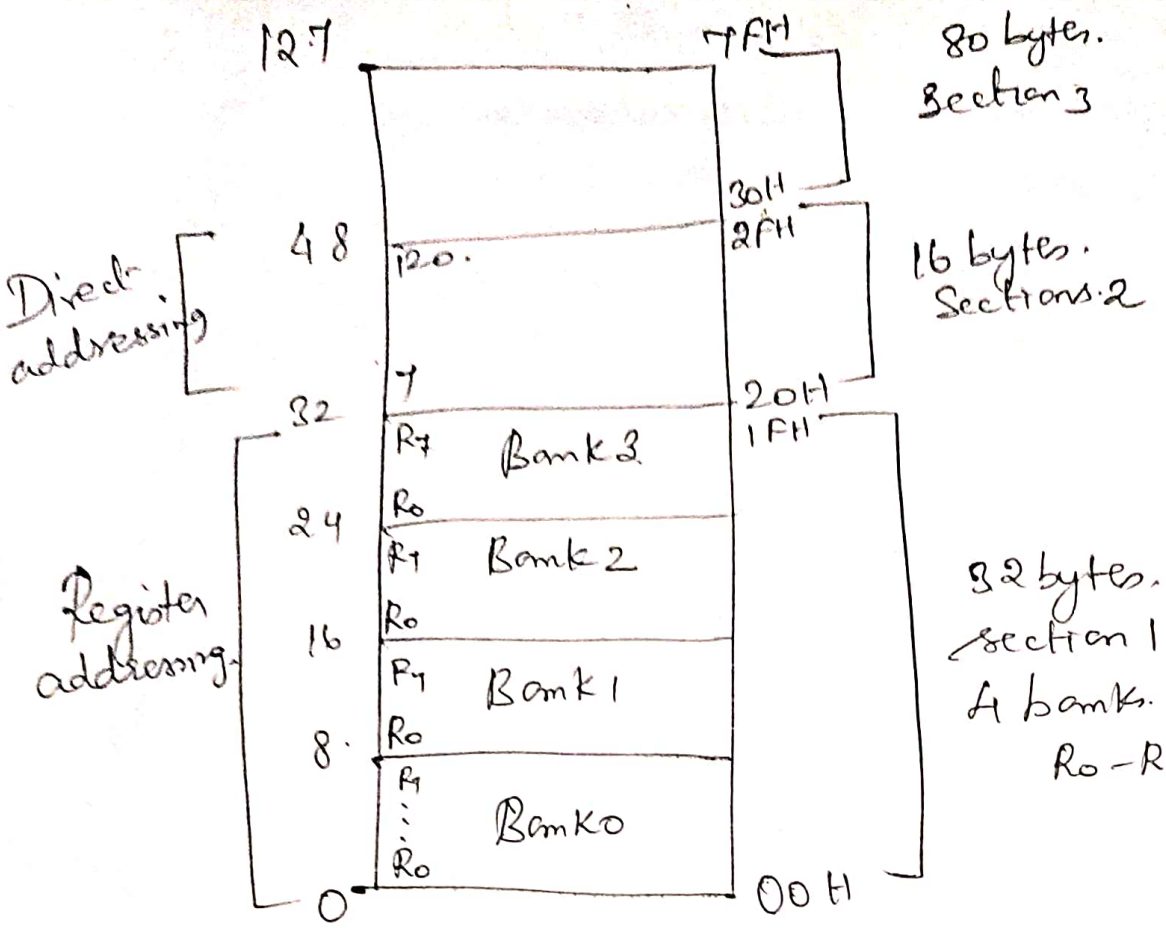



Area of Data memory

① Register bank → It have 4 Register bank designed #0, #1, #2, #3. each bank has 8 registers. $R_0, R_1, \ldots R_7$. at a time only one register bank is selected for operation.
→ Reg's are used to store data or Operands during executions.

② Bit addressable area: → Usually used to store bit variables.
→ They are designated from address 20H to 2FH.
→ It is mainly used to store the bit variable from application Program like status of an o/p device. like LED or Motor (ON/OFF) etc

③ Scratch pad area: → it is used for general purpose storage
→ Scratch pad area is from 30H to 7FH and this includes stack too.




Organization of Internal RAM

## 8051 I/O ports

→ In 8051 totally 4 ports are available.
These are $P_0$, $P_1$, $P_2$, $P_3$ these are 8 bit port.

→ If the first 0 is written to a port it will become output port. If the first 1 is written to a port it will become input port.

Port 0:

**Port 2** It is also 8 bit port. It can be used as input or O/p. It is not necessary for any Pull up resistors

**Port 3** : It can be used as input or output port. It is not necessary to connect any pull up register. Port 3 pins can be used for doing alternate function. These functions are given below.

| Port 3 bits. | functions. | |
|---|---|---|
| P.3.0 | RxD | Serial Communication |
| P3.1 | TxD | Signals. |
| P3.2 | $\overline{INT0}$ | External Interrupts. |
| P3.3 | $\overline{INT1}$ | |
| P.3.4 | T0 | Used for Timer 0 and Timer 1. |
| P3.5 | T1 | |
| P3.6 | $\overline{WR}$ | Write and read Signals. |
| P3.7 | $\overline{RD}$ | Used for External memories. |

## Interrupts.

The 8051 has five interrupt sources. Each of these interrupts can be programmed to two Priority levels. The interrupt sources are.

| S.NO. | Source | Destination. |
|---|---|---|
| 1. | $\overline{INT0}$ | External request from P3.2 Pin |
| 2. | Timer 0 | Overflow from Timer 0 activates the Interrupt Request Flag TF0. |
| 3 | $\overline{INT1}$ | External request from P.3.3. |
| 4 | Timer 1 | Overflow from Timer 1 activates the Interrupt Request Flag TF1. |
| 5 | Serial Port | Completion of Transmission or reception of a Serial frame activates flag TI or RI |

→ Each of these Interrupt Sources can be individually Enabled or disabled by Setting or clearing a Interrupt Enable Register (IE).

→ All these Sources can be Programmed either to a high Priority level or to a low priority level by Setting or clearing the Interrupt Priority Register. (IP)

## Priority of Interrupt

| Source | Priority Level |
|---|---|
| External interrupt 0 | Highest |
| Timer 0 overflow | |
| External Interrupt 1 | |
| Timer 1 overflow | |
| Serial Port | Lowest. |

## Response Time

→ The Response time to an External Interrupt is a very Critical factor in Control applications where an Immediate action is required.

→ The response time will depend on the above mentioned three Cases or Conditions

→ Once an Interrupt is not blocked by any of the above three Condition then a hardware Call is generated Internally by the Processor. after the Program branches to a predefined location.

# Interrupt Control Registers

→ The Interrupt Request flags are in two different registers. Such as.

1. IE → Interrupt Enable Register.
2. IP → Interrupt Priority Register.

→ External Interrupts control bits are in two formats.

IT0 in TCON → 0.
IT1 in TCON → 2.

## IE → Interrupt Enable Register.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EA | X | X | ES | ET1 | EX1 | ET0 | EX0 |

→ If EA = 0. all interrupts are disabled and if EA = 1 each interrupt can be individually set or cleared.

→ If ES = 0 Serial port interrupt is disabled and if ES = 1 Serial port interrupt is enabled.

→ If ET1 = 0 Timer 1 interrupt is disabled if ET1 = 1 Enabled.

→ If EX1 = 0. External interrupt 1 is disabled. and if EX1 = 1 it is enabled.

→ If ET0 = 0 Timer 0 interrupt is disabled

## IP → Interrupt Priority Register.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | PS | PT1 | PX1 | PT0 | PX0 |

PS Serial port Interrupt priority level. PS = 1 Programs it to the higher. Priority level.

→ PT1 → Timer 1 Interrupt Priority level PT1 = 1 Program it to the higher priority level.

PX1 External Interrupt 1 Priority level PX1 = 1 Program it to the higher Priority level.

PT0 → Timer 0 Interrupt priority level PT0 = 1 Program it to the higher Priority level.

PX0 → External Interrupt 0 Priority level PX0 = 1 Programs it to the higher Priority level.

## Data Transfer Instructions.

Data Transfer Instructions are divided into Three classes.

1. General Purpose transfer.
2. Accumulator Specific transfers.
3. Address object Transfer.

### (1) General Purpose Transfer.

It contains Three data Transfer operation such as follows.

(1) MOV → performs a bit or a byte transfer from source operand to the destination operand.

(2) PUSH → Increment the stack pointer register and then transfer a byte from source operand to Stack element currently addressed by Stack Pointer.

(3) POP → Transfer a byte operand from the Stack element addressed by the Stack Pointer register to the destination operand and then decrement Stack pointer.

## Accumulator Specific Transfers:

There are accumulator Specific transfer operations are available.

XCH → Exchange the byte Source Operand with register A.

XCHD → Exchange the low order nibble of the byte Source operand with the low order nibble of register A.

MOVX → Perfoms a byte move b/w the external data memory and registers A. The external address. Can be Specified by the DPTR register or R1 or R0 register.

MOVC → Perfoms the move of a byte from the Program memory to register A.

## Address object Transfers.

MOV, DPTR # data loads, 16 bits of Immediate data into a pair of destination register DPH, and DPL.

1. MOV DPTR # data 16 (load data pointer with a 16 bit Constant).

The data Pointer is loaded with the 16 bit Constant Indicated. NO flags are affected.

(DPTR) ← #data 16.

## CONTROL TRANSFER INSTRUCTION

There are three classes of Control transfer Operations

1. Unconditional Calls, return and Jump.
2. Conditional Jumps.
3. Interrupt.

## Unconditional Calls, Return and jump.

→ These transfer the control from the current value of the program counter to the target address.

→ Both direct and Indirect transfers are supported. The three transfer operations are follows.

### ACALL and LCALL.

→ ACALL and LCALL push the address of the next instruction onto the stack and then transfer the control to the target address.

→ ACALL - Absolute Call is a 2 byte instruction and used when the target address is in the current 2k page.

→ LCALL → long call is a 3 byte instruction that addresses the full 64K program space.

### RET and RETI

RET → Return from Subroutine and RETI - Return from Interrupt transfer control to return address saved on the stack. and decrement stack pointer register by 2

### Conditional Jumps

There are two Complex Conditional Jump Instructions.

1. The CJNE Compares the first operand with the second operand and performs a jump if they are not equal
2. The DJNE decrements the source operand and returns the result to the operand.

# Comparision To Programming Concepts of 8051 with 8085

→ A clear Understanding of the 8051 hardware features is essential for its Programming and its interaction with the external world. The following resources would be used in Programming of the 8051 such as

**Memory.** The 8051 can have separate Program memory and data memory each of 64 KB. In Program memory out of 64 KB, 4 KB of memory is present on chip as ROM
- 128 bytes internal data RAM.
- 21 Special function Register (SFRs).

## Internal Data RAM

128 bytes of internal data RAM can be divided into
1. 32 bytes of for 4 banks of registers R0–R7. The register bank is selected by using two bits RS0 and RS1 in the Program status word. These registers can be used as general purpose registers.
2. 16 bytes of direct addressing bits. These can be directly referred and used in programs.
3. 80 bytes as general purpose RAM.

## Special function Registers.

Using the SFRs the different resources of the 8051 like timers/counters, serial ports, interrupts may be programmed and controlled.

## Program Status Word

The Program Status word Contains Several Status bit that reflect the Current State of the CPU. The PSW is one of the Special function registers. and Contains the carry bit, auxiliary carry bit, two register Select bits the overflow flag bit, a parity bit.

## 8085 Programming

The Programming of a microprocessor is required to make it perform a required job.

**Memory** In 8085 Processor there is no Inbuilt memory and registers are Used to store the result, the following registers are Used in Programming 8085.

① Registers - A, B, C, D, E, H, L. ③ Program Counter

② Register pairs - BC, DE, HL. ④ Stack Pointer.

⑤ Conditional flag - Z, S, Cy, AC, P.

**Special Function Registers.** There is no Special function registers in 8085.

**Program Status word** The Program Status word is a 16 bit register. The higher-order 8 bits Contain. accumulator Contents and the lower order 8 bits have five Condition flags.

# Addressing Modes of 8051

The various ways of accessing data is known as addressing modes. There are five ways of addressing modes are available.

1. Register addressing
2. Direct addressing.
3. Register Indirect addressing.
4. Immediate addressing.
5. Base register plus Index register Indirect addressing.

## Register addressing

→ In this addressing mode, registers are involved. Source register and destination register must be of the same size.

→ It Permits access to eight registers. ($R_0$-$R_7$) of the register bank.

→ Example. MOV A, $R_1$ : move the content of $R_1$ to A

MOV $R_2$, A : move the content of A to $R_2$.

ADD A, $R_3$ add the content of A and $R_3$.

## Direct Addressing

→ In this mode, the data is in a RAM memory location. Its address is given as a part of the Instruction. No Symbol is used here.

→ In this mode, the address of the Operand is specified in the Instruction and it has Operands as byte or bit.

→ Direct addressing of byte Provides operation on ~~this~~ one of the following

1. Lower 128 bytes of Internal data RAM.
2. Special functions registers.

## Register Indirect addressing

→ In this addressing the address of the operand is not specified directly Instead the address of the operand is specified as the contents of the register mentioned in the instruction.

→ Here the register is used as a pointer to the data if the data is inside the CPU only R0, R1 registers are used.

The symbol @ indicates the register - indirect addressing mode.

Example MOV, A @ R1 :
MOV @ R0, B

## Immediate Addressing

→ In this case the operand on which the operation has to be performed according to the instruction. Specified in the instruction itself.

→ In this mode the operand comes immediately after the opcode and the symbol of immediate addressing mode is #

→ Example MOV A #20h; load data 20h into A
MOV R2 #80h ; load data 80h into R2

## Base Register plus Index Register Indirect Addressing.

→ This is an indirect instruction used to access the Program memory. In the instruction, the operand on which the operation is to be performed is not specified directly

→ Summation of contents of base register and index register determines the operand address.

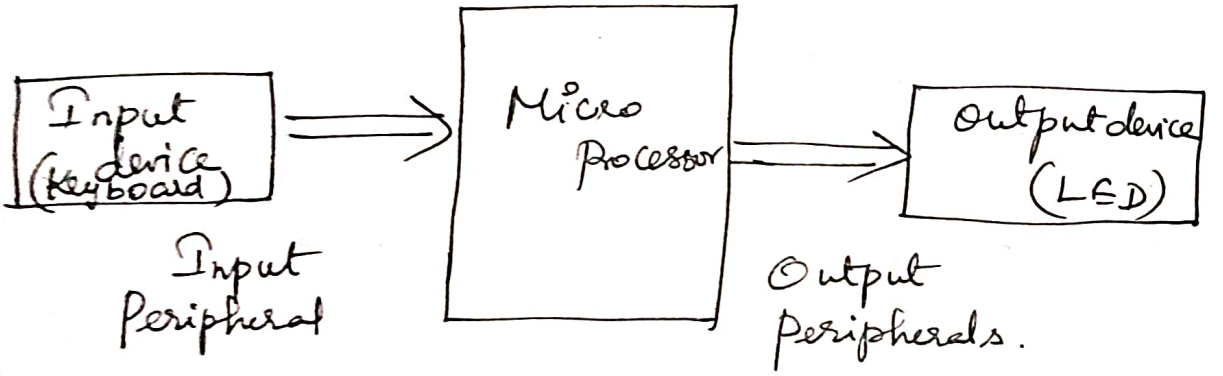
→ DPTR or pc register may act as the base register and accumulator act as the Index register.

# UNIT- 4

# Peripheral Interfacing

## Introduction.

To Communicate with the Outside world Microprocessor Use peripherals. (I/o devices). Commonly Used input devices are keyboard, A/D Converters. etc. and Output devices are CRT, Printer, LEDs etc. These input and output devices are called peripherals (or) I/Os. Peripherals are Connected to the microprocessor through electronic circuits known as Interfacing Circuits.




Microprocessor Unit with I/o devices.

Some peripheral interfacing chips of 8085 and 8086. microprocessor.

(i) programmable peripheral Interface Intel 8255
(ii) Programmable Interrupt Controller PIC Intel 8259
(iii) Programmable Communication Interface (PCI) Intel 8251.
(iv) Keyboard Display Controller Intel 8279.
(v) A/D and D/A Converter Interfacing.

# Programmable Peripheral Interface (PPI) Intel 8255

→ A Programmable peripheral Interface is a multiport device. The port may be programmed in a variety of ways as required by the Programmer. The device is very useful for interfacing Peripheral devices.

→ The 8255 is a general purpose Programmable I/O device used for parallel data transfer. It has 24 I/O pins which can be grouped as 3x8 bit Parallel ports of Port A, Port B, Port C.

→ The eight bits of Port C can be used as individual bits (or) be grouped in two 4-bit Ports as $C_{upper}(C_U)$ and $C_{Lower}(C_L)$.

→ fumetions of the 8255 are classified according to 2 modes. (i) Bit set/Reset (BSR) mode
(ii) I/O mode.

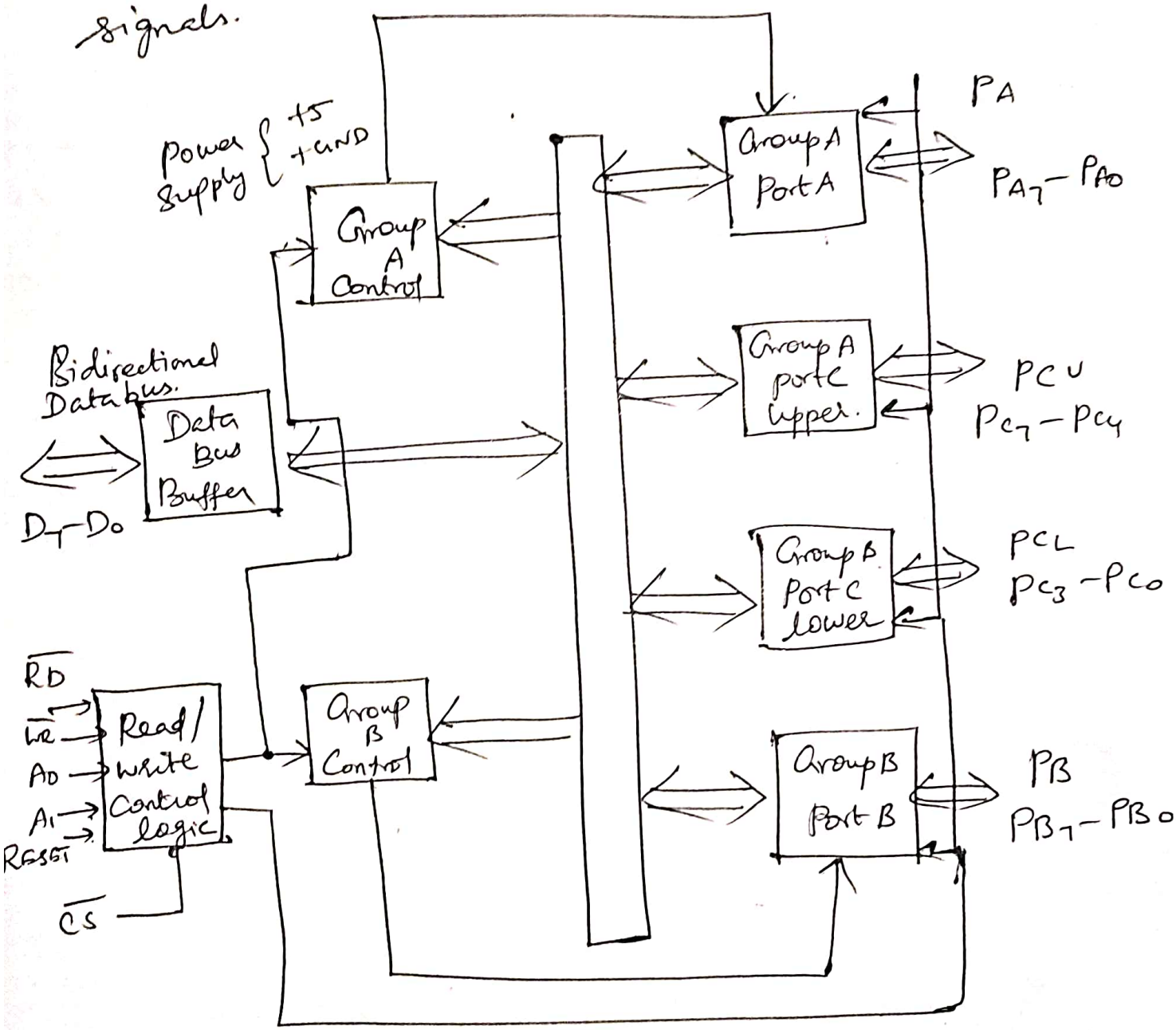
## Features

* The 8255 is a widely used Programmable, Parallel I/O device.
* It can be Programmed to transfer data under Various Condutions from Simple I/O to Interrupt I/O.
* It is compatible, with all Intel and most other Microprocessor.
* it Can operate in 3 I/O modes.
(i) Mode 0 (ii) Mode 1 and (iii) Mode 2.

# Block diagram of 8255

→ The Internal block diagram of 8255. It consists of two 8 bit ports of Port A and Port B, two 4 bit ports of Port $C_U$ and Port $C_L$, data bus buffer, control logic, group A and group B controls.

→ Also includes all the elements of a Programmable device, port C performs functions similar to that of the status register in addition to providing hand shake signals.




Block diagram of 8255.

## Data bus Buffer

* It is a tri state bi directional buffer Used to interface the internal data bus of 8255 to the system data bus.

## Control logic

→ The Control logic block accepts Control bus signal as well as input from the address bus and generates the commands to the Individual group Control of Group A and Group B.

→ The Control logic has six lines

**$\overline{RD}$ (Read)** : This control signal Enables the Read operation. when the signal is low, the CPU reads data from a selected I/O port of the 8255.

**$\overline{WR}$ (write)** : This Control signal Enables the write Operation. When the signal is low the CPU writes data or control word into 8255.

**A0 and A1** : The selection of input port and Control word register is done by using A0 and $A_1$

| $A_1$ | A0 | selected ports. |
|---|---|---|
| 0 | 0 | Port A |
| 0 | 1 | Port B |
| 1 | 0 | Port C |
| 1 | 1 | Control word register. |

**RESET**: This is an active high signal; it clears the control register and sets all ports in the input mode.

- **$\overline{CS}$ : (chip select)** . It is a chip select signal. The low status of this signal Enables Communication b/w CPU and 8255. 0 → 8255 is selected
1 → 8255 is not selected

# Programming and Operation of 8255

## (i) Programming in mode 0:

→ The ports A, B and C can be configured as simple input or output ports by writing the appropriate control word in the control word register.

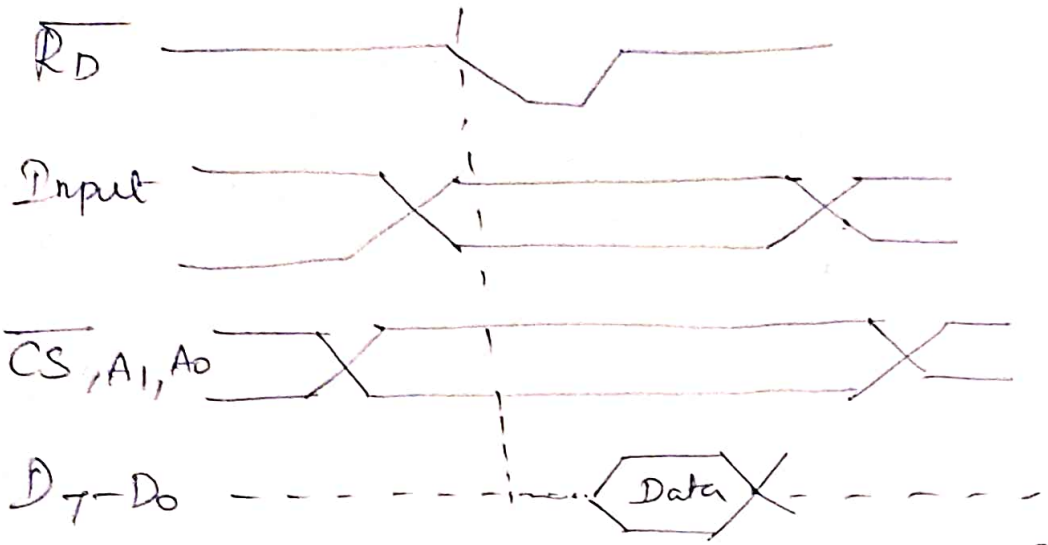
→ In the control word $D_7$ is set to 1 and $D_6$, $D_5$, $D_2$ are all set to '0' to configure all the ports in mode 0 operation.

→ Status of bits $D_4$, $D_3$, $D_1$ and $D_0$ then determine whether the corresponding ports are to be configured as input or output.

mode 0 Configuration.

| A | | B | | Group A | | Group B. | |
|---|---|---|---|---|---|---|---|
| $D_4$ | $D_3$ | $D_2$ | $D_1$ | Port A | Port C (upper) | Port B | Port C (lower) |
| 0 | 0 | 0 | 0 | O/P | O/P | O/P | O/P |
| 0 | 0 | 0 | 1 | O/P | O/P | O/P | I/P |
| 0 | 0 | 1 | 0 | O/P | O/P | I/P | O/P |
| 0 | 0 | 1 | 1 | O/P | O/P | I/P | I/P |
| 1 | 0 | 0 | 0 | I/P | O/P | O/P | O/P |
| 1 | 1 | 0 | 0 | I/P | I/P | O/P | O/P |
| 1 | 1 | 1 | 0 | I/P | I/P | I/P | O/P |
| 1 | 1 | 1 | 1 | I/P | I/P | I/P | I/P |

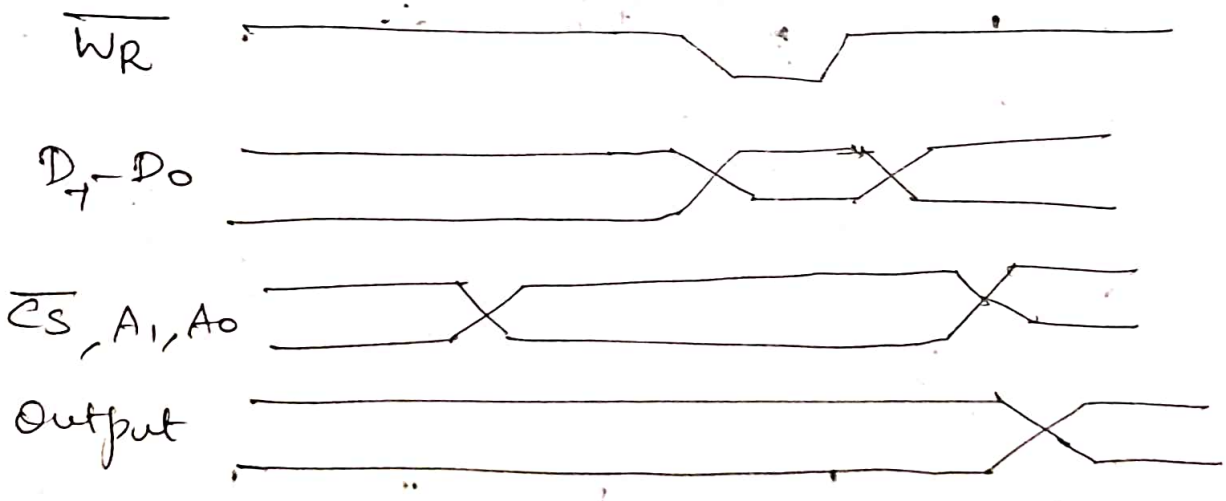
## Input Mode




Timing Diagram for mode 0 input mode.

→ Read Command activates $\overline{RD}$ signal. Upon activation of $\overline{RD}$ signal, CPU reads the data from the selected input port into the CPU register.

## Output Mode

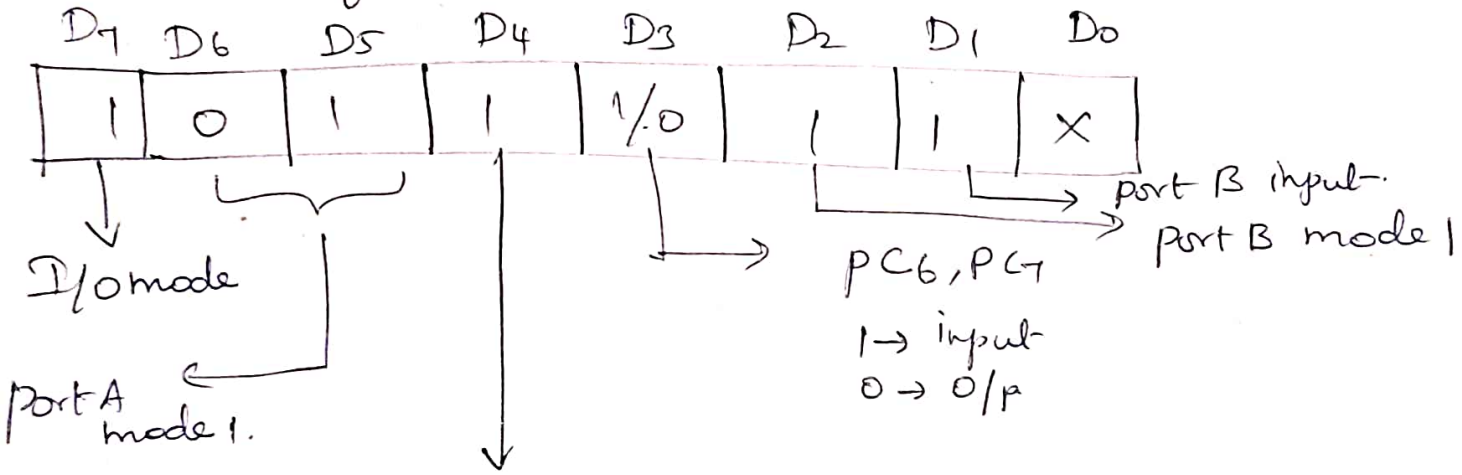



→ CPU can write data into the O/P port by initiating write command with proper port address. The CPU sends data on the data bus and upon activation of $\overline{WR}$ signal, data on the data bus gets latched on the selected O/P port.

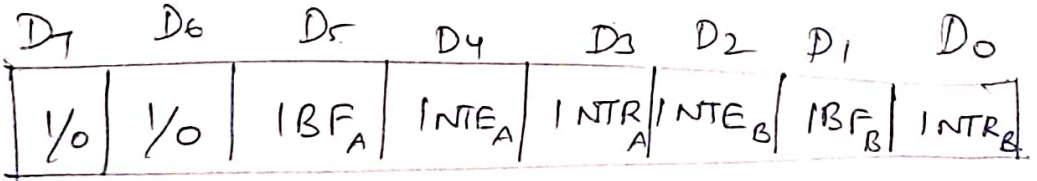
## (ii). Programming in Mode 1

→ Both the Group A and Group B can be operate in mode 1, either together or individually with each port containing an 8 bit latched input or output data port and a 4 bit port which is used for control and status of the 8 bit port.

→ Port A uses the upper three signal $PC_3$ $PC_4$, $PC_5$
Port B uses the lower three signals $PC_0$, $PC_1$, $PC_2$ used for control.




Control word




8255 Mode 1 Configuration.

IBF → Input Buffer full. → This signal is an acknowledgement by 8255 to indicate that input latch has received the data byte.

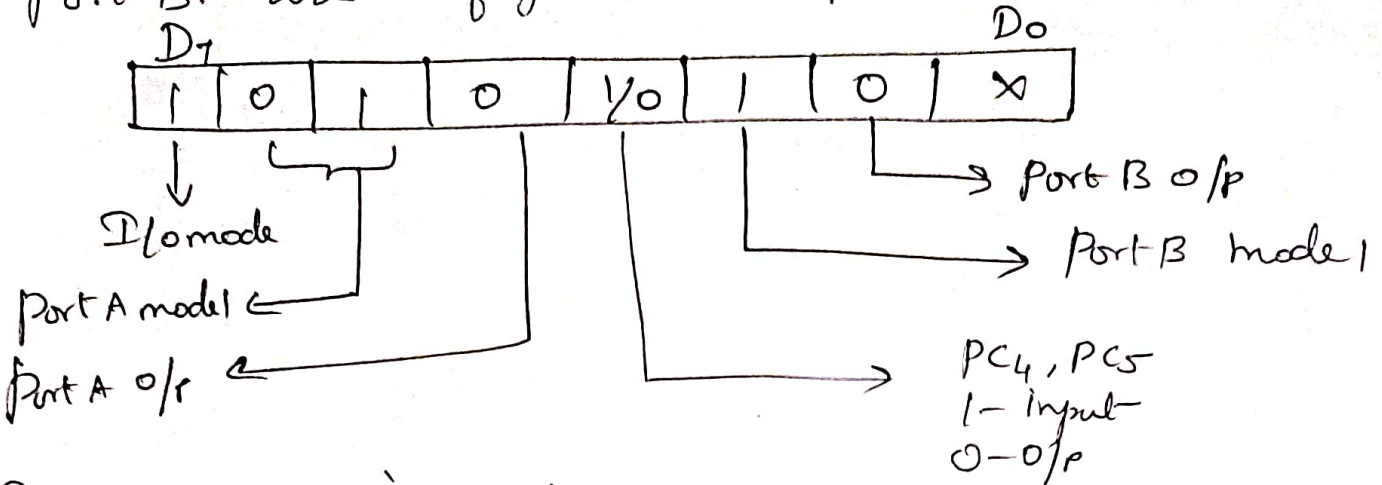
INTR → Interrupt request.

INTE → Interrupt Enable. → This is an internal flipflop used to enable or disable the generation of INTR signal.

→ The two flip flop. $INTE_A$, $INTE_B$.
↓ Enabled or disabled by $PC_4$
↓ $PC_2$.

## Mode 1 O/P & Control signals.

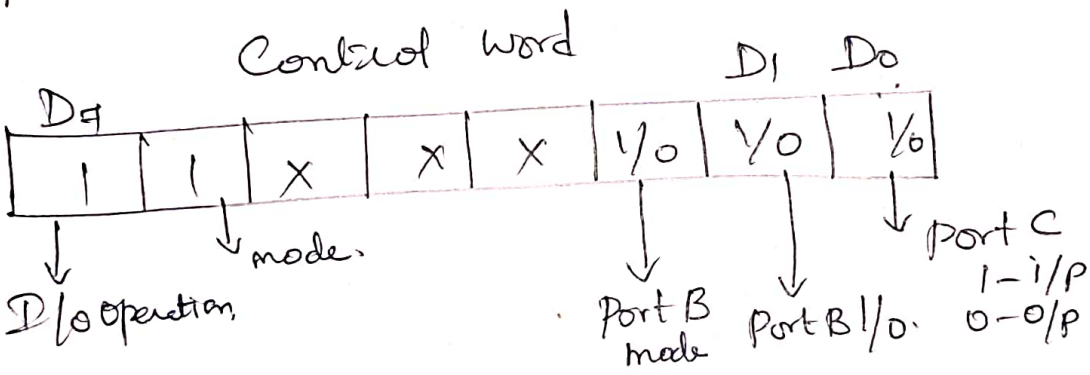
The Control signals which is used when Port A and Port B are configured as O/P. ports.

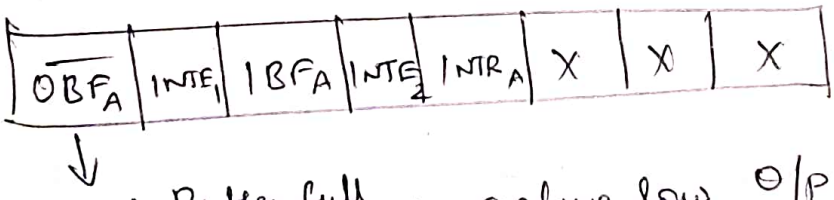



## Programming in Mode 2. (Bi directional Bus I/O).

→ When 8255 is operated in mode 2, port A can be used as bi-directional 8 bit I/O. bus using for handshaking.

→ Port B can be programmed in mode 0 or in mode 1




OBF$_A$ → Output Buffer full → active low O/p which indicates that CPU has written data into Port A.

IBF$_A$ → This is an active high O/p which indicates that data has been loaded into input latch of Port A.

# Programmable Communication Interface (INTEL 8251)

→ Programmable Communication Interface (PCI) is used for Serial data transmission

→ It is a programmable chip designed for Synchronous and asynchronous data communication Packaged in a 28 pin DIP.

→ It is a Universal Synchronous / asynchronous Receiver/ Transmitter (USART).

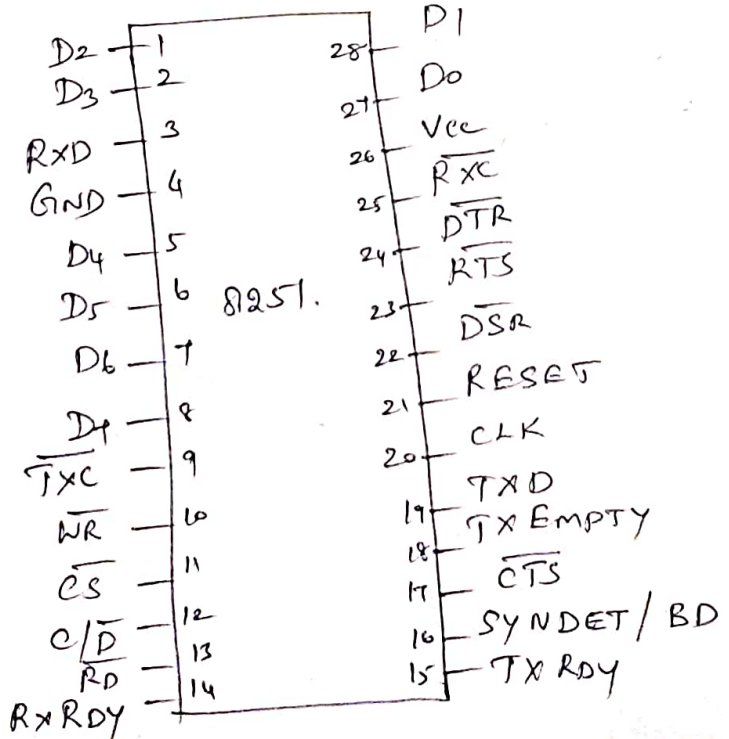
→ It is Compatible with 8085, 8086, 8088, and 8748 system

→ 8251 Can be used to transmit/receive Serial data

→ It accepts data in parallel format from the microprocessor and Converts them into the serial data for transmission.

→ It also receives Serial data and Converts them into parallel and sends the data in parallel format of the CPU.

## Pin Diagram of 8251



## Description

$\overline{CS}$ - Chip select → when this signal goes low 8251 is selected by microprocessor Unit for Communication.

C/$\overline{D}$ - Control / Data → when this signal is high, the control register or status register is addressed. When it is low data buffer is addressed.

$\overline{WR}$ - Write : when it is low the CPU writes data into Intel 8251.

$\overline{RD}$ - Read : when it is low CPU reads data from Intel 8251.

RESET → A high on this input resets the 8251 and forces it into the idle mode

CLK - Clock → This is the clock input usually connected to the System clock. This clock does not control either the transmission or reception rate. The clock is necessary for Communication with the microprocessor

$\overline{DSR}$ (Data set Ready) : When it is low, the 8251 has ready to set data

$\overline{DTR}$ (Data Terminal Read) : When it is low, the 8251 has ready to set data terminal.

$\overline{RTS}$ (Request to send). When it is low 8251 Sends request to send the data.

$\overline{CTS}$ (clear to Send) : A low on this pin Enables 8251 to transmit Serial data.

$\overline{TXC}$ (Transmitter clock) : It governs the rate of data transmission.

# Programmable Interrupt Controller (PIC). (4)

## INTEL 8259

Intel 8259 is a single chip Programmable interrupt Controller. It is used when several I/O devices transfer data using interrupt and they are to be connected to the same interrupt level of the microprocessor.

→ It has 28 Pin DIP IC and it is Compatible with 8086, 8088, 8085 microprocessors.

**Features** : → it is Compatible with 8085, 8086, 8088 microprocessor

→ it can manage 8 prionty interrupt

→ it can be programmed to accept either the level triggered or edge triggered Interrupt request

→ it can be expanded to 64 Prionty levels by Cascading additional 8259's.

**Pin diagram**

**Description**

**$D_0 - D_7$** : Bidirectional data bus. Control status and Interrupt Vector Infomation are transferred via this bus.

**$IR_0 - IR_7$** : Interrupt request I/O devices Send interrupt request through these lines.

**$CAS_0 - CAS_2$** : Cascade lines.

**$\overline{CS}$** : Chip select when it is low, Enable the Communication b/w CPU and 8259.

$\overline{WR}$ : (Write) → when it is low Enables to 8259 to accept Command word from CPU.

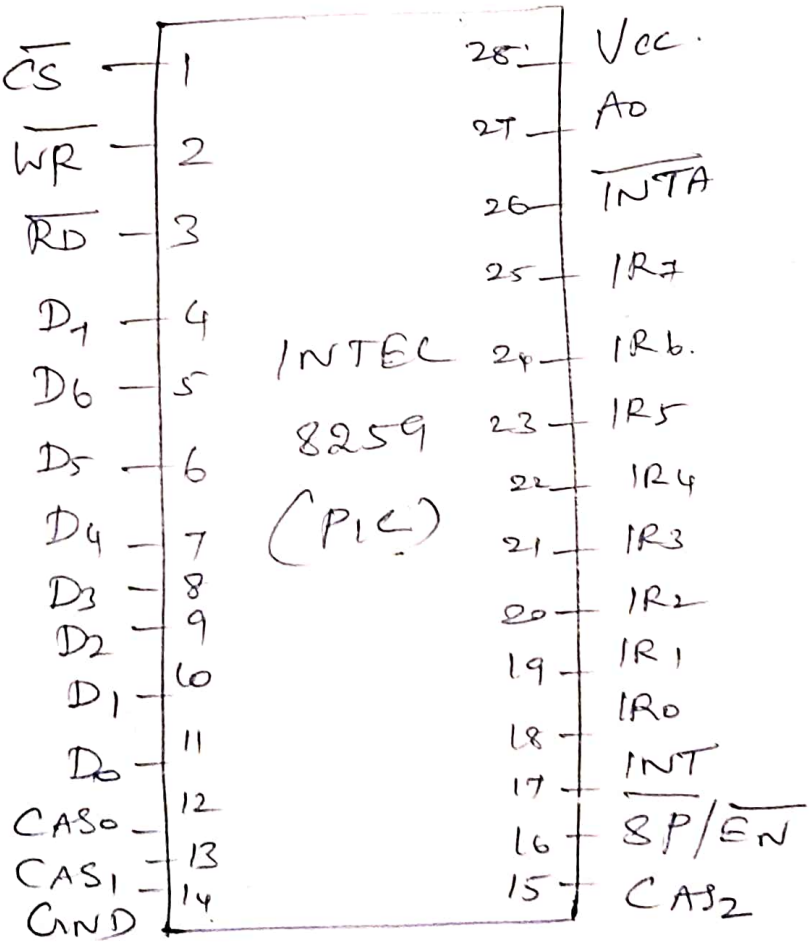
$\overline{RD}$ (Read) → when it is low Enables 8259 to send various Status Signals on data bus for CPU.

$\overline{SP}/\overline{EN}$ : Slave Program / Enable buffer. It is related to Cascade control.

INT : Interrupt. It is used to interrupt the CPU.

$\overline{INTA}$ : Interrupt Acknowledge.

$A_0$ : Address line It acts in Conjuction with $\overline{RD}$ $\overline{WR}$ $\overline{CS}$.



# Block diagram of 8259

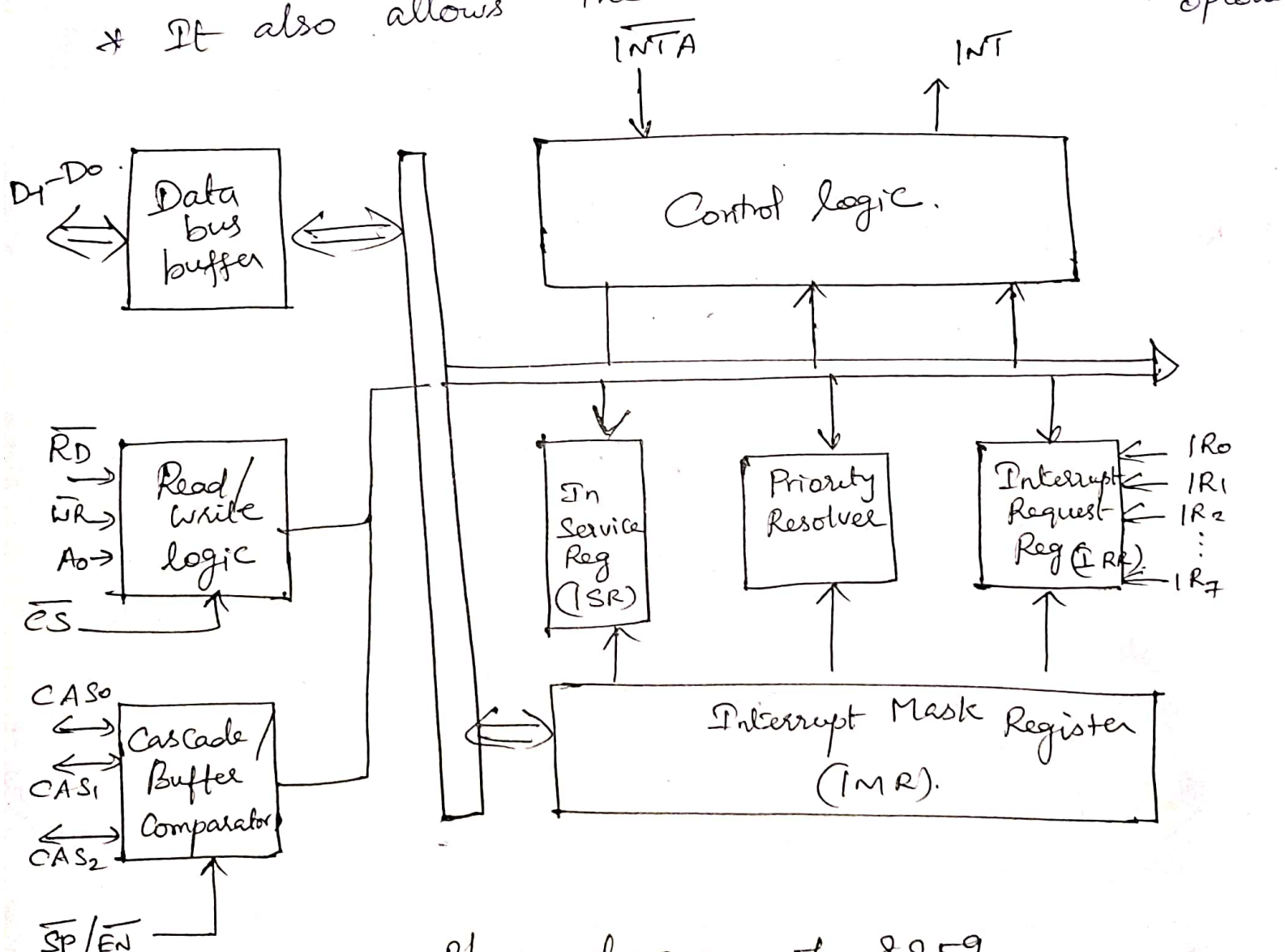


→ Internal block diagram of 8259 includes 8 blocks. They are Control logic, Read/write logic, Data bus Buffer, three registers (IRR, ISR, and IMR) Priority Resolver and cascade buffer.

## Data bus Buffer

* It is an eight pin bidirectional data bus.
* It also allows the 8259 to send interrupt opcode

Control, Status and interrupt vector information are transferred via this bus.




Block diagram of 8259.

## Read/Write Logic

The processor uses the $\overline{RD}$, $\overline{WR}$ and A0 to read or write 8259. when it is selected by $\overline{CS}$. Chipselect ($\overline{CS}$) and A0 determine the port address of the Controller.

**Control logic** → This block has two pins INT (Interrupt) as an output and $\overline{INTA}$ (interrupt acknowledge) as an Input.

→ INT is connected to interrupt pin of the microprocessor Unit. whenever valid interrupt is asserted this signal high.

→ $\overline{INTA}$ is the Interrupt ack. signal from the Microprocessor Unit

**Interrupt Request Register**

→ it has 8 input lines (IR0 - IR7) for interrupts. When these lines go high the requests are stored in the register.

**Interrupt Service Register (OR) In Service Register**

→ It keeps track of which interrupt inputs are currently being serviced. and corresponding bit will be set in this register.

**Priority Resolver (PR)**

PR examine the above three registers (IRR, ISR, IMR) and determines whether INT should be sent to the processor or not

IMR .. Interrupt Mask reg.
it stores the masking bits of interrupt lines to be masked
→ interrup input can be masked by setting corresponding bit in IMR

TXRDY (Transmitter Ready) : It ensure the transmitter ready.

RXRDY (Transmitter Empty) : It ensures the receiver ready.

RxD (Receive data) : line for receiving data

TxD (Transmit data) : line for Serial data communication

RxC (Receiver Clock) : It governs the rate at which characters are received.

$D_7 - D_0$ : 8 bit data bus

## Block diagram of 8251

## Programmable Interval Timer (8253/ 8254)

A Programmable Interval timer is used in real time applications for timing and Counting functions such as BCD / binary Counting, generation of Square wave of desired frequency.

→ Popular Used Programmable Interval timer chips are Intel 8253 and 8254. It includes Three identical 16 bit counters.

### Features

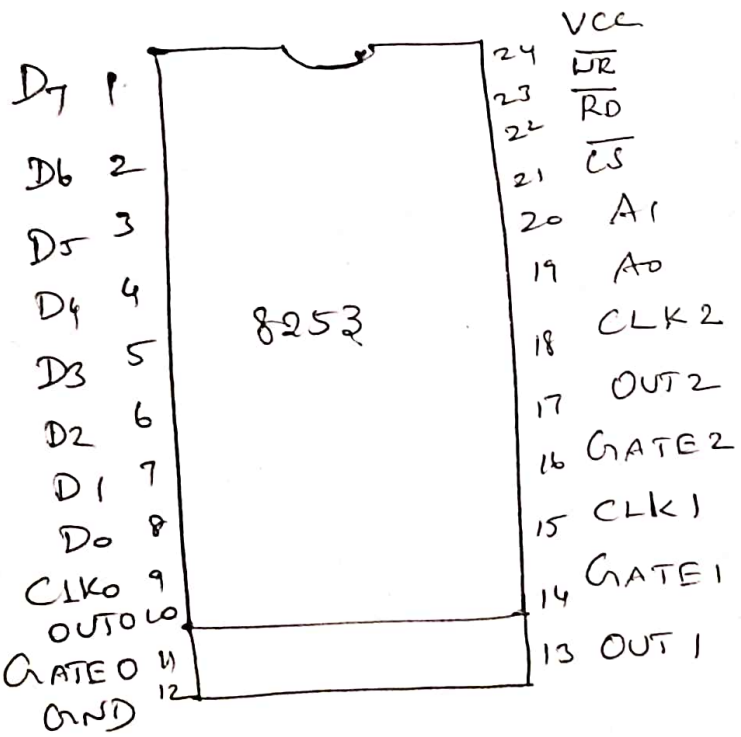
1. Three Independent 16-bit down Counter
2. 8254 can handle inputs from DC to 10MHz whereas 8253 can operate up to 2.6 MHz.
3. Counter can be Programmed in Six different modes.
4. Compatible with all Intel and most other microprocessors.

It Can Operate in Six Operating modes.

Mode 0 Interrupt on terminal Count
Mode 1 Programmable one Shot
Mode 2: Rate generator
Mode 3: Square wave generator
Mode 4: Software Triggered mode
Mode 5: Hardware Triggered Mode.

## Pin diagram of 8253

8253 is 24 pin IC and Operates at 5V dc. It Contains three Independent 16 bit Counters. It can be Operated in any one of the Six operation modes.




### Description

$\overline{CS}$ → Chip select when it is low Enables the communication b/w CPU and 8253.

WR Write: when it is low CPU O/P data in the form of mode information are loading Counters.

RD (Read): when it is low CPU reads data

A0 – A1 Address lines } These Pins are Connected to address bus These are used to select one of the Three Counters.

D0 – D7 : These are tri state bidirectional data bus Used to Interface 8253 to the System data bus.

CLK0, CLK1, CLK2 } These are clock signals for Counter 0, and Counter 1 Counter 2 respectively.

GATE0, GATE1, GATE2 } These are gate terminals for Counter 0, and Counter 1 and Counter 2. respectively

OUT0, OUT1, OUT2 } These are O/p terminals for Counter 0, Counter 1 Counter 2 respectively.

## Block diagram of 8253

→ Block diagram of 8253 includes three Counters a data bus buffer, Read/write control logic and control register.

→ Each counter has two input signals namely CLOCK and GATE and one output signal OUT.

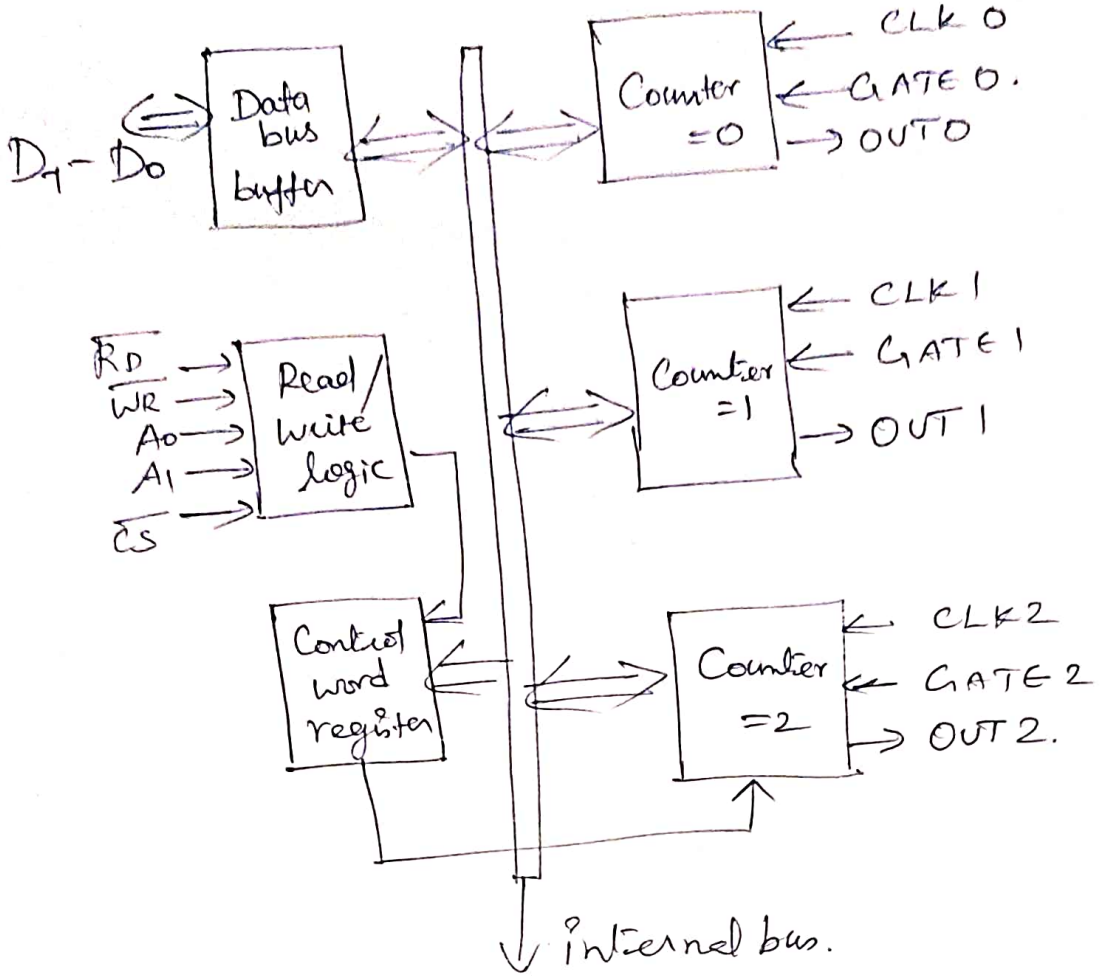
## Data bus buffer.

→ This is a tri state bidirectional, 8 bit buffer is used to interface the 8253 to the system data bus through $D_0$-$D_7$. The data bus has three basic functions.

→ The data bus has three basic functions
(i) Programming the 8253 in various modes.
(ii) Loading the Count registers.
(iii) Reading the Count Values.

## Read/write logic

The 8253 contains a read/write logic which accept input from the system bus and then generates control signals for operation of 8253. The following table shows the status of Pins associated with read/write logic.




Block diagram of 8253

Control word register

When the pins $A_0$, $A_1$ are 11 the control word register is selected. The control word format is shown below:

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| $SC_1$ | $SC_0$ | $RL_1$ | $RL_0$ | $M_2$ | $M_1$ | $M_0$ | BCD |

The bits $D_7$ and $D_6$ of the control word are to select one of the 3 counters. $D_5$ and $D_4$ are for loading/reading the count $D_3$, $D_2$, $D_1$ are for the selection of operating mode of the selected counter.

## Operation of 8253

Each Counter of the 8253 is individually Programmed by writting a Control word into the Control word register. ($A_0 A_1 = 11$).

→ Bits $SC_1$ and $SC_0$ select the Counter, bits $RW_1$ and $RW_0$ select the read, write or latch command. bits.

$M_2, M_1, M_0$ → select mode of operation.

**Write operation** → Write a Control word into Control register.

→ load the low order byte of a count in the Counter register.

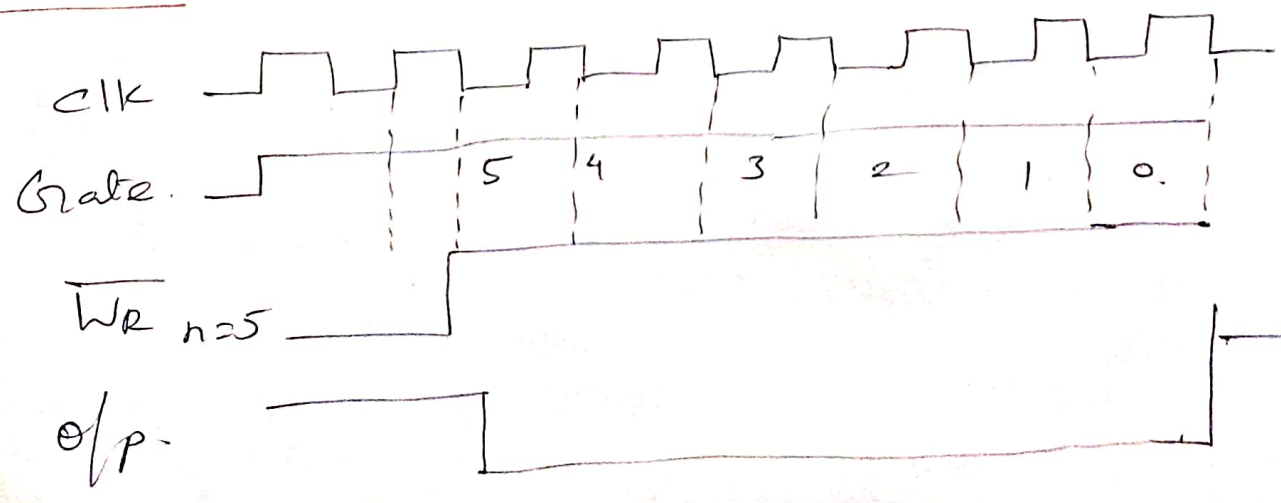
→ load the high order byte of Count in the Counter register.

**READ operation** → In some applications especially in event Counters. it is necessary to read the Value of the count in Process. This Can be done by three possible methods.

## Modes of Operation.

The 8253/54 Can Operate in Six different modes.

### mode 0: Interrupt on Terminal Count



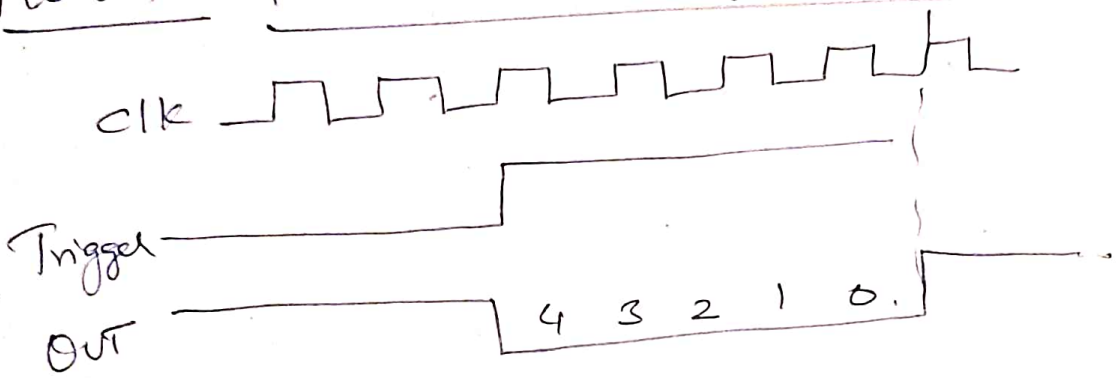
## Mode 1. Hardware - Retriggerable one-shot

In this mode initially the OUT is low once a Count is loaded in the register, the Counter is decremented every clock cycle. and when the Count reaches zero OUT goes high.

→ This can be Used as an interrupt.

→ OUT remains high Until a new Count

## Mode 1 Hardware- Retriggerable One-Shot




→ OUT is initially high

→ when the Gate is triggered, the OUT goes low and at the end of the Count out goes high again.

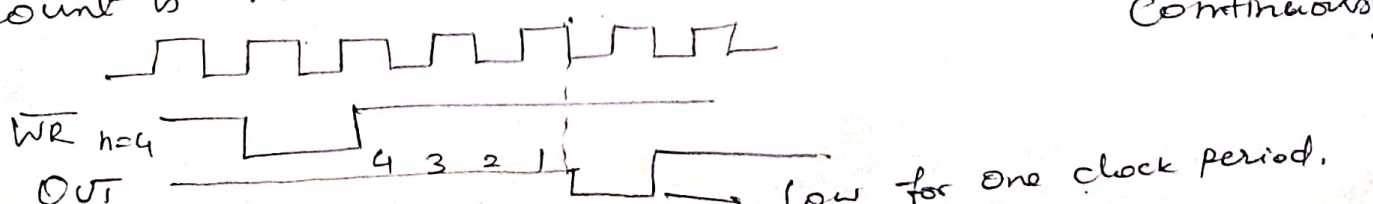
→ Thus generates one shot pulse

## Mode 2 Rate Generator

→ This mode is used to generate a Pulse Equal to the clock Period at a given interval.

→ when Count is loaded the OUT stays high, Until it reaches 1 and then the OUT goes low for one clock period.

→ Count is reloaded automatically and Pulse is generated. Continuously

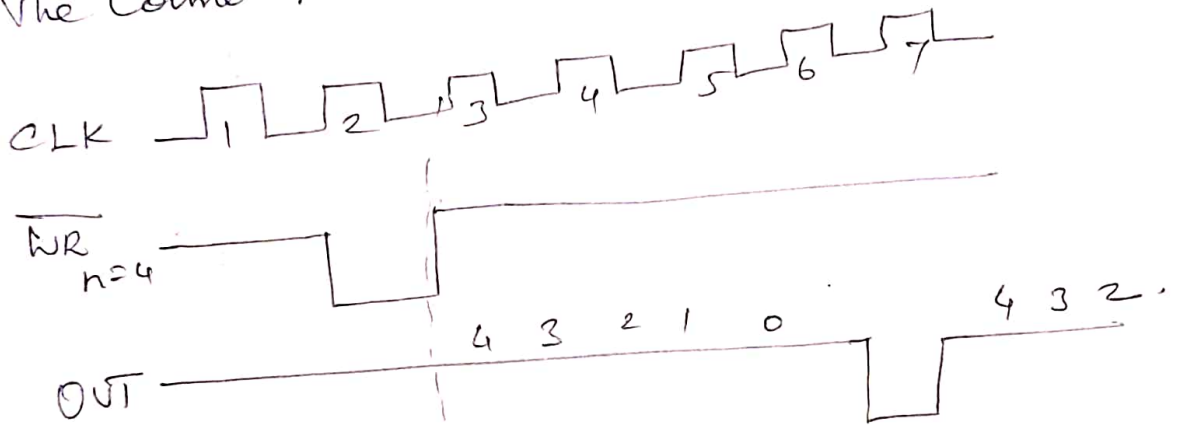


## Mode 3 — Square Wave Generator

→ it is similar to rate generator,
→ when a count is loaded, the OUT is high, The count is decremented by two. at every clock cycle.

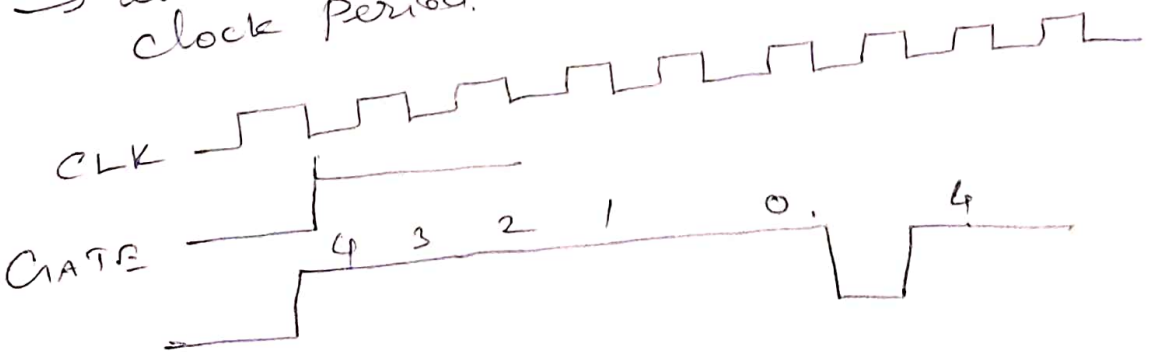
## Mode 4 — Software Triggered Strobe

→ OUT is initially high goes low for one ~~cycle~~ clock Period. at the end of the count.
→ The Count must be reloaded for Subsequent O/P.




## Mode 5 — Hardware Triggered Strobe

→ OUT is initially low, when the GATE is triggered, it goes high and the count begins.
→ at the end of count OUT goes low for one clock Period.



# Programmable keyboard / Display Interface (8279)

8279 is a programmable keyboard / display interface. It is used to interface a matrix keyboard and a multiplexed display.

→ it has 40 pins. with two major segments.

## Features of 8279

(i) It Provides Simultaneous keyboard and display operations.

(ii) It has built in hardware to provide key debounce

(iii) It provides a scanned Interface to a 64 Contact key matrix with two more keys. CONTROL and SHIFT.

(iv) It Provides three input modes for keyboard interface
- (a) Scanned keyboard mode
- (b) Scanned Sensor Matrix mode.
- (c) Strobed Input mode.

(v) It provides two output modes for display Interface
- (i) Left Entry (Typewriter Type)
- (ii) Right entry (Calculator Type)

## Pin description.

**DB0 - DB7 – Bi directional data bus**

It Communicates all data, Commands and status Information b/w the CPU and the 8279 are transmitted on these bi directional 8 bit data bus.

**$\overline{RD}$ : READ**

When it is low, the CPU reads the contents of selected register from 8279

WR (Write) When it is low, the CPU loads the data into Selected register depending on the status of A0.

RESET When it is low, it resets 8279. After being reset 8279 is configured in the following mode.

(i) Sixteen 8 bit character display.
(ii) Encoded Scan Keyboard
(iii) The program clock prescalar.

A0 : Address line

When A0 is high signals are Interpreted as a Command or Status. When A0 is low signals are Interpreted as a data.

$\overline{CS}$ : chip select When it is low, enables the Communication between the CPU and 8279.

CLK clock This signal is usually driven by the System clock and Used to generate Internal timing.

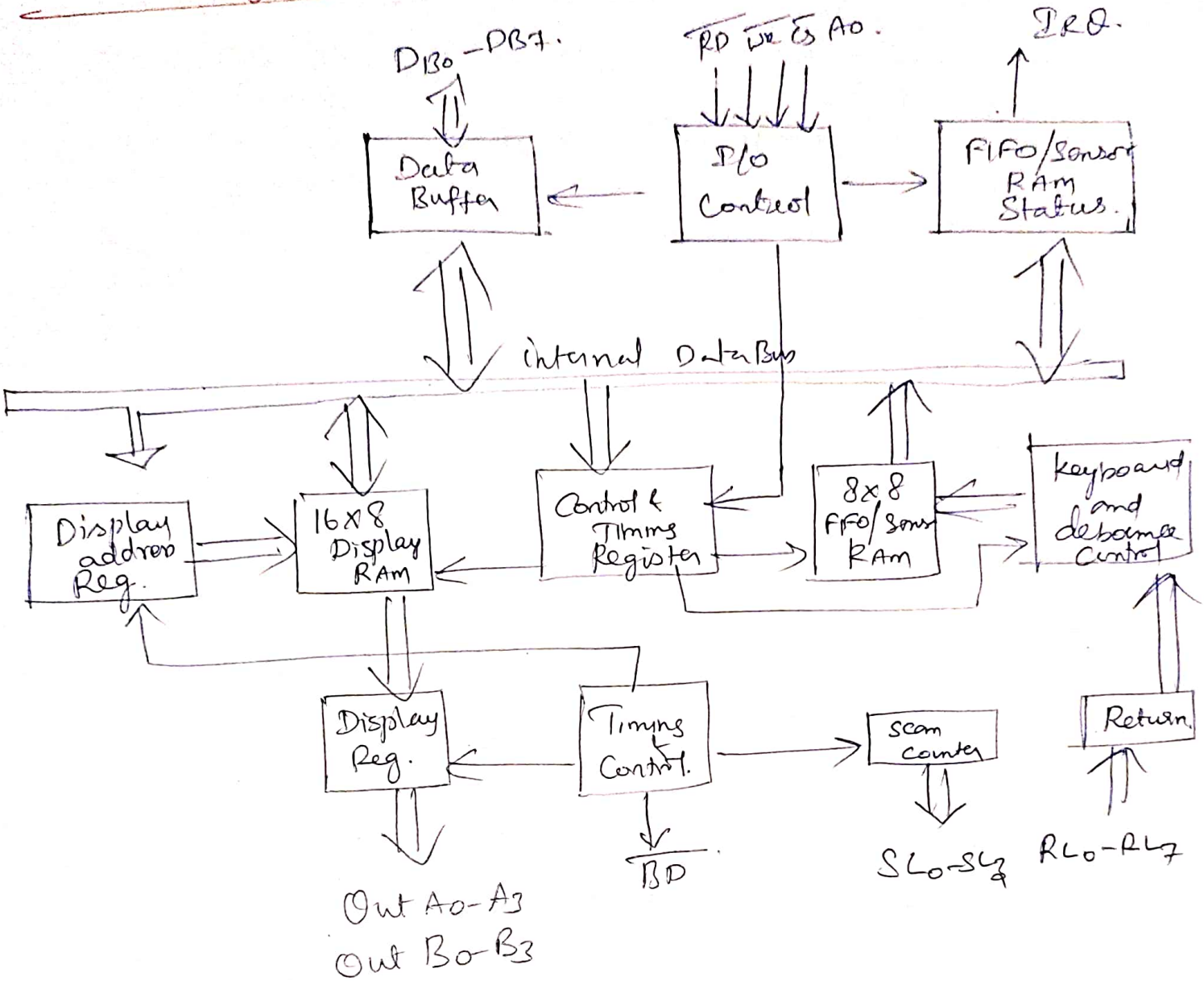
$\overline{BD}$ : Blank display This is active low output is used to blank the display during digit switching

Out $A_3$-$A_0$ and OUT $B_3$-$B_0$ These are two four bit Output port which can be considered as an One 8 bit port These are used for sending data to display drivers for display RAM and segment i/p of Seven segment display

IRQ Interrupt Request

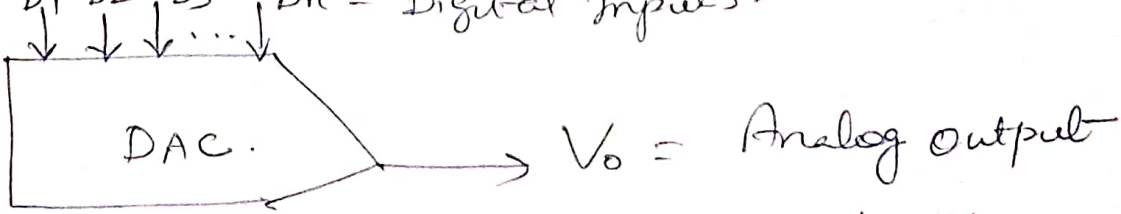
This signal is used to implement Interrupt driven input System. In Scanned Keyboard mode, the Interrupt line goes low when there is data in the FIFO/Sensor RAM.

# Block diagram of 8279




## Programmable Digital to Analog Converter (DAC)

Digital to Analog Converters are used to Convert a digital quantity to Analog quantity. D/A Converter Produces an o/p current or voltage proportional to digital quantity applied to its o/p.

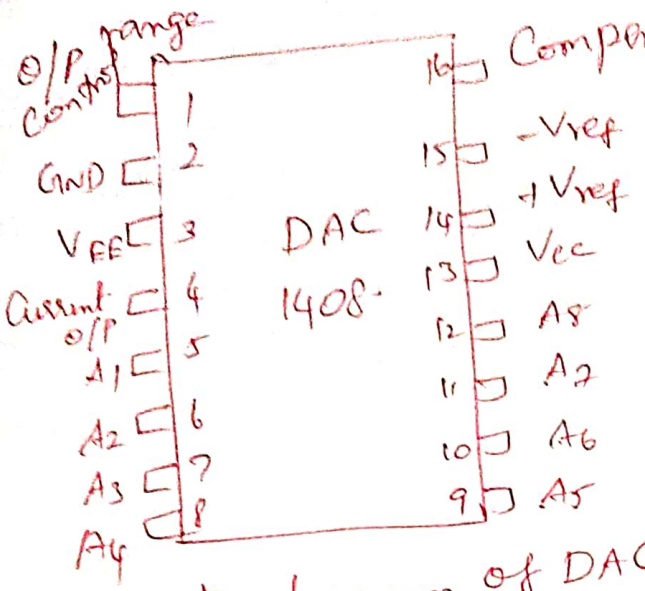



DAC Circuit of 4 bit Length.

# DAC 1408

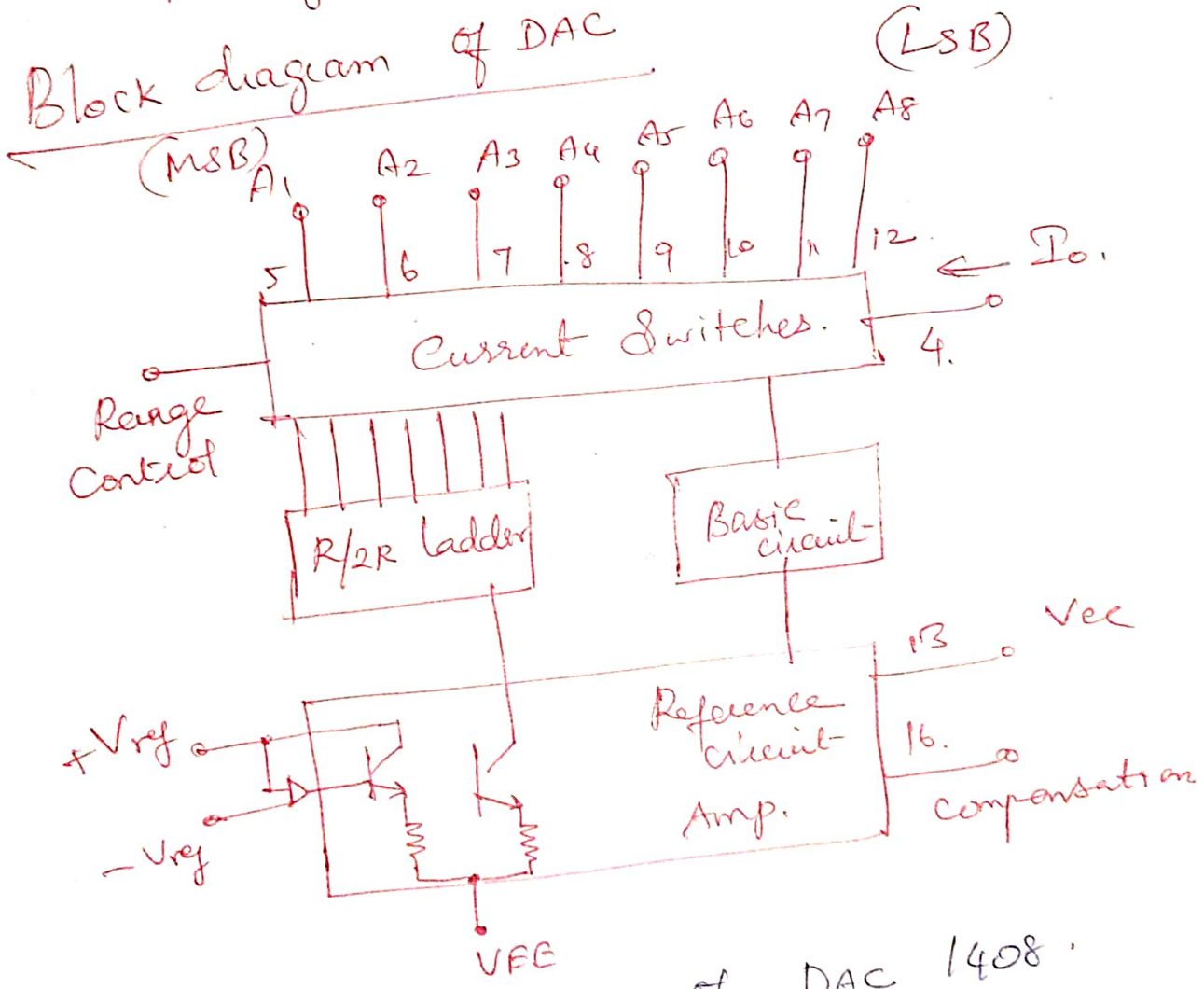
→ DAC 1408 is an 8 bit R/2R ladder type D/A Converter to the 8085 System using an interfacing 8255.

→ The Pin diagram of DAC 1408 is shown in fig.




Pin diagram of DAC 1408.

## Block diagram of DAC




Block diagram of DAC 1408.

→ The DAC 1408 consists of a reference Current amplifier OR R/2R ladder and eight high speed Current Switches. It has eight input line A1 (MSB) through A8 (LSB). which Control the position of Current Switches.

The voltage at Vref and resistor $R_{14}$ determines the total ref current source and $R_{15}$ is generally equal to $R_{14}$ to match the i/p impedance of Ref current amplifier.

$$I_o = \frac{V_{ref}}{R_{14}} \left[ \frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256} \right].$$

Input $A_1$ through $A_8$ can be either 0 or 1. Therefore for typical circuit full scale current. Can be given as

$$I_o = \frac{5}{2.5K} \left[ \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} \right].$$

$$I_o = \frac{2mA \times 255}{256} = 1.992\ mA.$$

→ It shows that the full scale Output current is always 1LSB less than the reference Current Source of 2mA. This output current is converted into voltage I to V Converter. The output voltage for full scale input can be given as.

$$V_o = 1.992 \times 2.5K = 4.98V.$$

$$\boxed{V_o = 4.98V}$$

## Interfacing of DAC with 8051

The digital to analog Converter is a device widely used to Convert digital pulse to analog signal There are two methods available for creating a DAC such as

1. Binary Weighted
2. R/2R Ladder.

DAC 0808

→ In DAC 0808, the digital inputs are converted to Current ($I_{out}$). and by Connecting a register to the $I_{out}$ Pin.

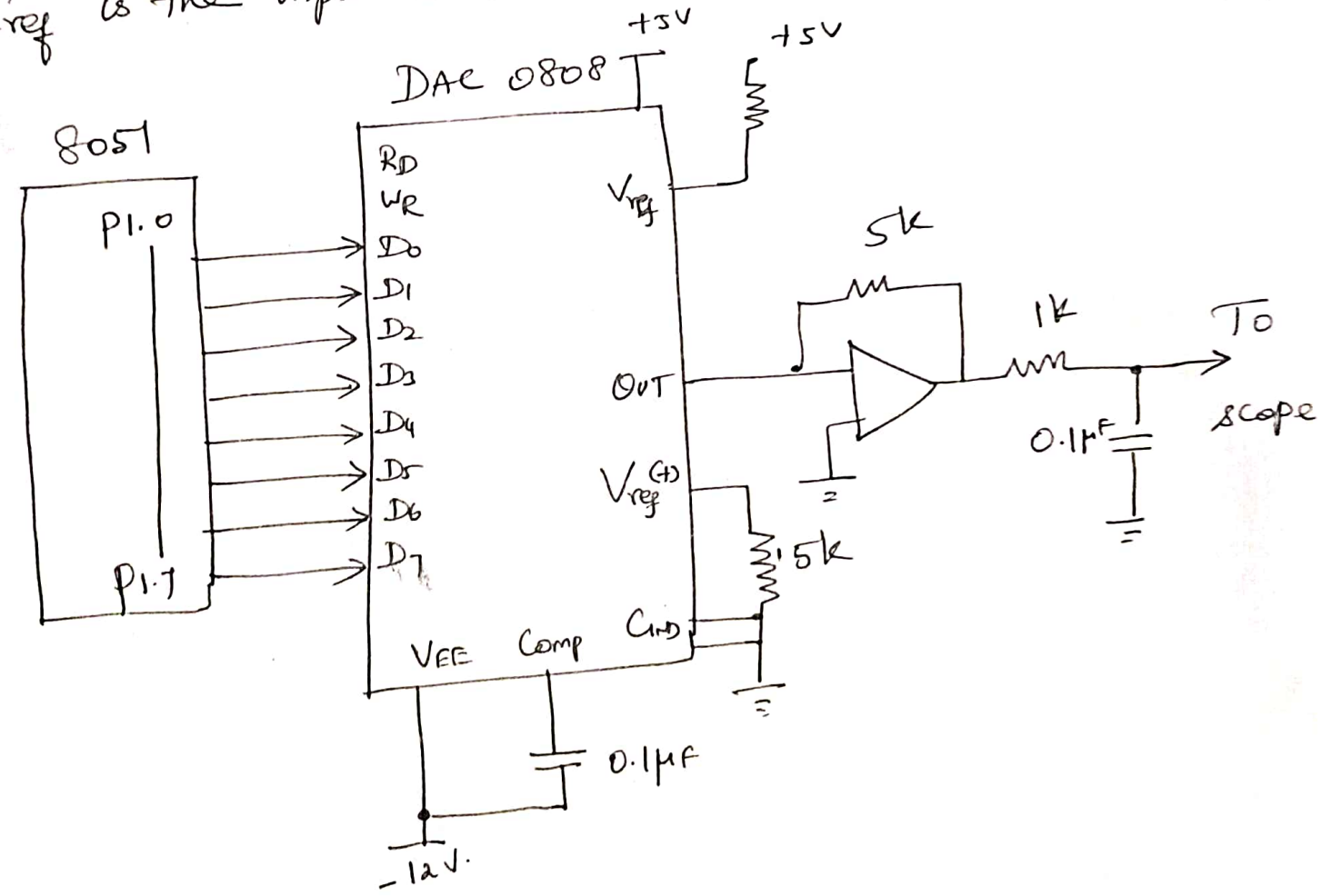
→ The total current Provided by the $I_{out}$ Pin is a function of the binary numbers at the $D_0$-$D_7$ inputs of the DAC 0808 and reference Current ($I_{ref}$).

$$I_{out} = I_{ref}\left(\frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256}\right).$$

Where $D_0$ – is the LSB of the input.

$D_7$ – is the MSB of the input.

$I_{ref}$ is the input current that must be applied to Pin 14.




8051 Connection to DAC 0808.

→ The Iref Current is generally set to 2.0 mA. It shows the generation of current reference by Using standard 5V power supply and 1k and 1.5k ohm standard resistors.

→ Some DAC's also Use the Zener diode, which DAC's Using the Zener diode will Overcome any fluctuation associated with the power Supply voltage.

# UNIT-5

## Micro Controller Programming & Applications

### Key board Interfacing

→ Key board and LCD are most widely used input/output devices of the 8051. Keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports. Therefore with two 8 bit ports an 8×8 matrix of keys can be connected to the microprocessor.

→ When a key is pressed a row and a column make a contact, otherwise there is no connection b/w rows and column.

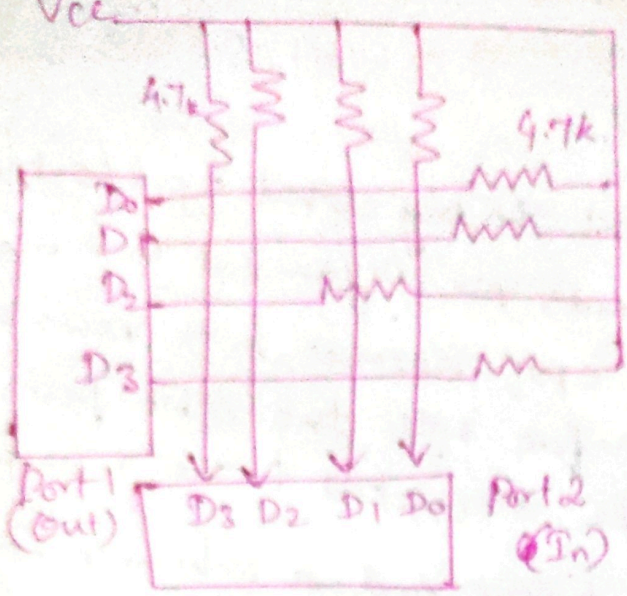
### Scanning and identifying the key

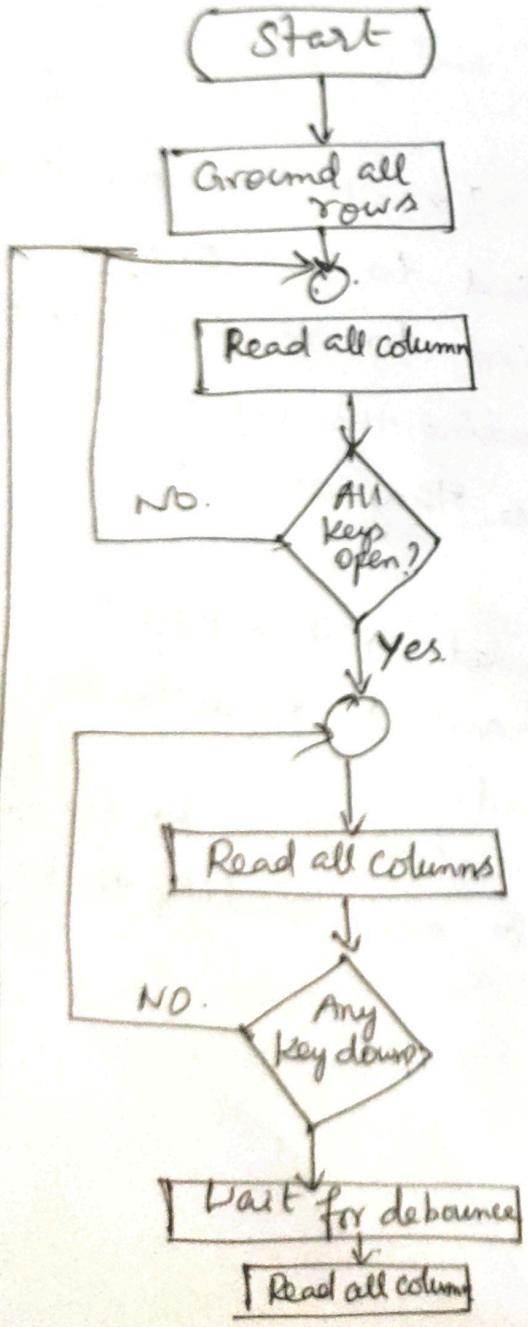
→ The following fig. shows the 4×4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port.

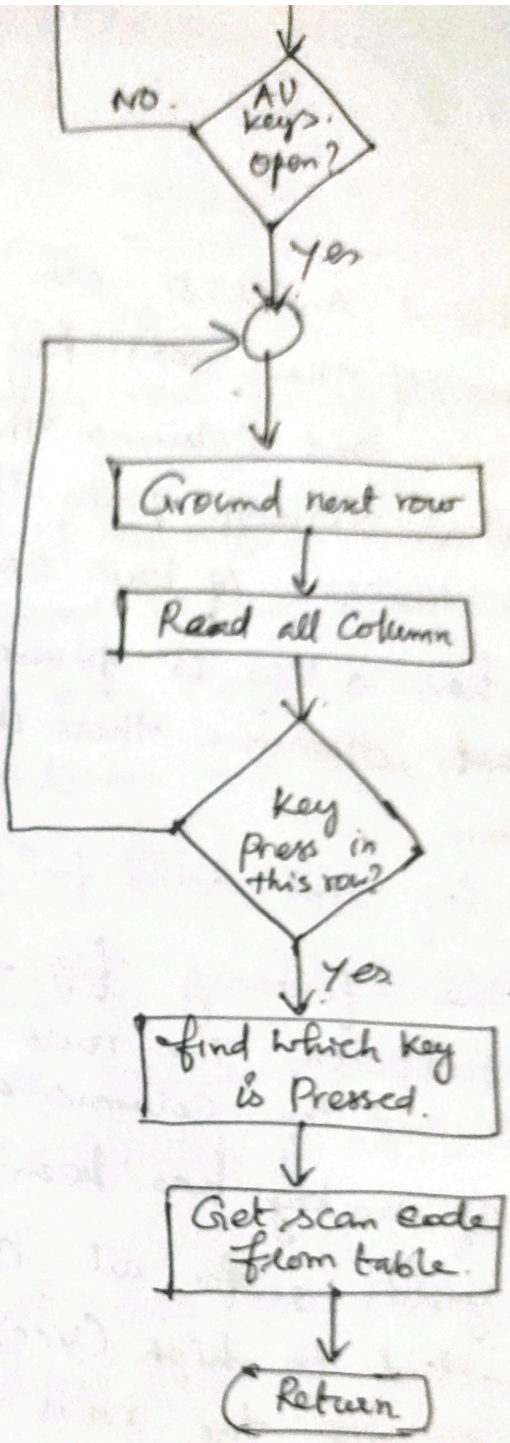
→ If no key has been pressed reading the input port will yield 1s for all columns, since they are all connected to high (Vcc).

→ If all the rows are grounded and a key is pressed one of the columns will have 0 since the key pressed provides the path to ground.

→ It is the function of the micro controller to scan the keyboard continuously to detect and identify the key pressed.




## Grounding rows and reading the Columns

* To detect a Pressed key the micro controller grounds all rows by Providing 0 to the output latch and then it reads the column. If the data from read from the column is D3 - D0 = 1111, no keys has been Pressed.

And the Process Continues until a key is press is detected. If one of the Column bits has a Zero this means that a key Press has occurred. -for example. If $D_3-D_0$ : 1101. This means that a key in the D1 Column has been Pressed. after Key Press is detected. the microcontroller will go through the Process of identifying the key.

→ If the data read is all. 1s no key in that row is activated and the Process is moved to the next row. It grounds the next row, reads the Columns and checks for any Zero. This Process Continues Until the row is identified.

## Stepper Motor Control

→ A Stepper motor is a digital motor used to translate electrical pulses into mechanical movements The following fig. shows the typical 2 phase motor interfaced Using 8255.

→ Motor shown in the circuit has two phases with Center tap winding. The Center taps of these windings are Connected to the 12V Supply. Due to this motor can be excited by grounding four terminals of the two windings

→ Motor Can be rotated in steps by giving Proper Excitation Sequence of these windings.

**Rotor** Stepper motor has a permanent Magnet Rotor It is also known as Shaft. It is Surrounded by Stator.

**Step angle** It is the minimum degree of rotation associated with a Single Step.

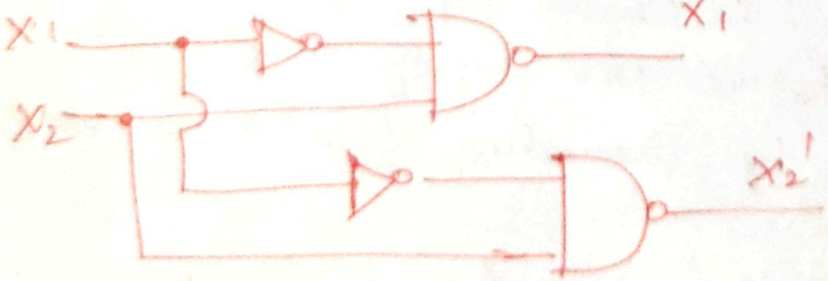
**Steps Per revolution.**

This is the total number of steps needed to complete One rotation or 360°

$$\text{Steps per second} = \frac{\text{rpm} \times \text{Steps per revolution}}{60}$$

→ The following table shows the typical excitation. Sequence. The given excitation. Sequence rotates the motor in clockwise direction. To rotate motor in anticlockwise direction we have to excite motor in a reverse sequence.

→ The excitation sequence for Stepper motor has many changes due to change in Winding Connections However it is not desirable to excite both the ends of the same winding simultaneously.



Digital Locking System.

Excitation Table

| Step | $X_1$ | $X_2$ | $Y_1$ | $Y_2$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |

# Application to automation Systems.

## Application to Automation System.

→ A top loading automatic washing machine is shown in the following figure. On the top right hand - there are four knobs for machine Programming. As we open the top cover, we see a washing basket which can rotate.

**Input Setting** There are four knobs on the top right hand side for Programming the machine.

**Load Select** Load basically means the number of cloth intended to be washed together. There are three setting high, medium and low. Based on the load selected the machine decides the amount of water required.

**Water Inlet Select** Machine can take either hot tap or mix water. All the back of the machine there are two inlet pipes for hot and tap water. The knob setting on mix allows 50% tap and 50% hot water as input.

**Modes** Though this knob the machine can be in a normal or Save mode. In the normal mode (i) the clothes are washed (ii) the detergent is drained. (iii) The fresh water is put (iv) the clothes are rinsed (v) the water is drained. (vi) Using Spin the moisture from clothes is taken out to a large extent.

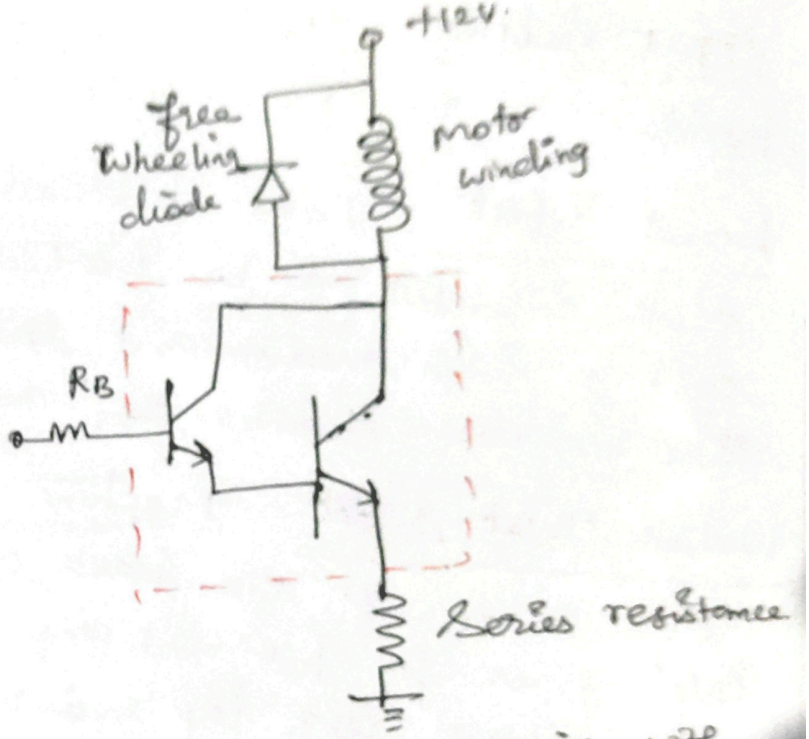
**Program select** Using this knob, the machine is programmed to wash the clothes of different kinds.
→ The setting are extra heavy, normal, light, and delicate. Extra heavy clothes are very different clothes like bedsheets. pillows covers. and curtains etc.

→ The excitation sequence given in the above table is called a full step excitation sequence. In which excitation ends of the phase are changed in one step.

→ The excitation sequence given in the following table takes two steps to change the excitation ends of the phase. Such a sequence is called half step sequence and in each step the motor is rotated by 0.9°.

| Stp | $x_1$ | $x_2$ | $y_1$ | $y_2$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 0 |
| 8 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |

Half step excitation sequence




Excitation Circuit with Series resistance.

→ Stepper motor is stepped from one position to the next by changing the currents through the fields in the motor. The winding Inductance oppose the change in current and this puts limit on the stepping rate.

→ for higher stepping rates and more torque. It is necessary to use a higher voltage source and current limiting resistors. By adding series resistance, we decrease L/R time constant which allows the current to change more rapidly in the windings.

→ Heavy clothes are clothes with a lesser dirt level than that of extra heavy.
→ Normal clothes are day today personal wears.

## Control System Design

**Measurement** Quantity of water is being filled.

**Control** : following are the control requirements.

→ Inlet Control of water.
→ Water quantity Control
→ Agitator Control.
→ Spin Control
→ Drain Control.

**Indications** :- The indications provided in the machine are

* Machine on Indication (LED).
* Washing Complete (LED + BUZZER).

## Water Quantity Measurement.

The machine has to measure the water quantity during the fill operation. The measurement scheme may be based on the level in the basket or strain caused by the level of the water

**Level Measurement** Considering that water is a conductive liquid, electrodes are placed at particular levels in a metal container. When the water touches the electrodes, the circuit is completed and a signal is generated.

**Strain Measurement** → when a wire is stretched within its elastic limit. it will increase in length and correspondingly the diameter will decrease. Thus the resistance of the wire changes due to the strain this is called the piezo electric effect.

$$\Delta R/R = k \frac{\Delta L}{L} = k \frac{\sigma}{E}.$$

R → Original Resistance.
L → Original Length.
E → young's modulus.
σ = Stress = force / area.

**Water Quantity Control** The quantity of water to be filled is dictated by the load selected by the user. (ie) High, medium or low. As soon as the water quantity reaches the desired level the signal will be received at high, medium or low. At that instant, the water flow must be disabled by making H = 0. and T = 0.

**Control of Servo Motor.**

→ Servo motors are Self Contained mechanical device that are used to control the machine with great precision. These are found in many applications from toys to industrial automation.

→ **Working Principle**

→ servo motor works on PWM (Pulse width Modulation) Principle means its angle of rotation is controlled by the duration of applied pulse to its control PIN.

→ Basically servo motor is made up of DC motor which is controlled by a variable resistor. (potentiometer). and some gears.

→ High speed force of DC motor is converted into torque. by Gears.

→ potentiometer is connected to the output shaft of the servo. to calculate the angle and stop the DC motor. on required angle.

→ Servo motor can be rotated from 0 to 180 degree but it can go up to 210 degree. depending on manufacturing.

→ This degree of rotation can be controlled by applying a Logic Level 1 Pulse for duration b/w 1ms to 2ms.

## Circuit diagram and working explanation.

→ Servo motor has three wires Red for Vcc (Power Supply) Brown for Ground, and Orange is control wire. Control wire can be connected to 8051. we have connected it to Pin 2.1 of 8051

→ Now we have to keep this pin to Logic 1 for 1ms to rotate it to 0 degree. 1.5 ms for 90 degree. 2ms for 180 degree. we have used on chip Timer of 8051 to create delay. we have created delay of 50μs. through the function "Servo delay" and used "for" loop to create delay in multiple of 50μs.

## Servo motor Application.

→ It is used in press machine for cutting the pieces to size

→ It is used in the sugar filling station.

→ It is used in Labeling application.

→ It is used in aeroplanes.