UNIT- III

8051 MC programming And Applications

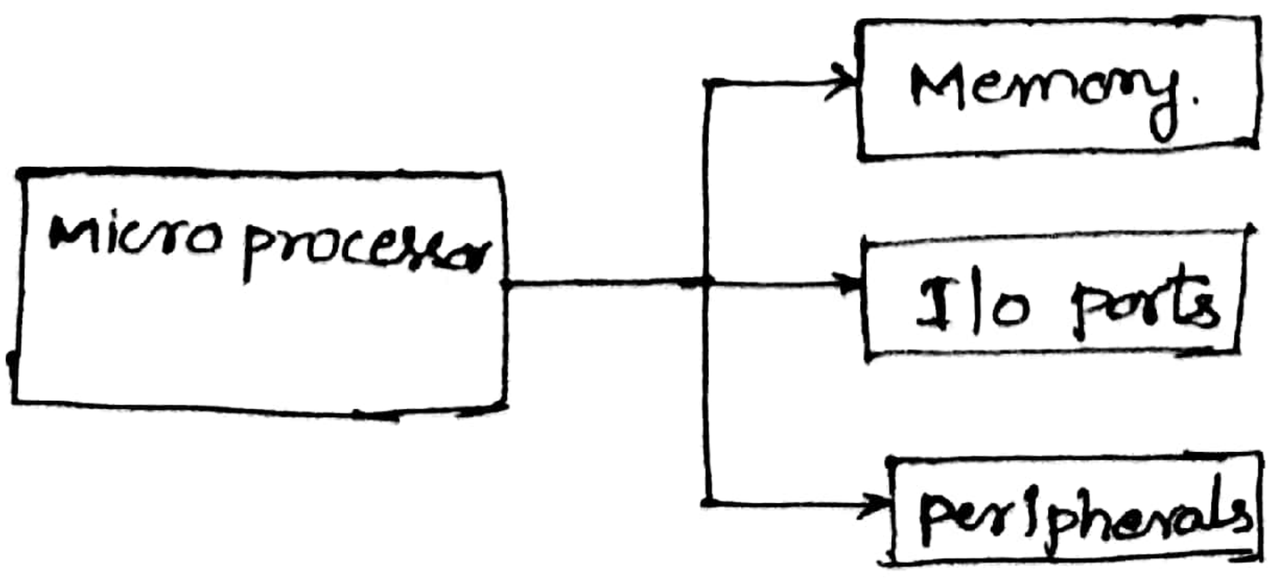
The micro processors is not self - sufficiant, it requires other components like memory and I/o devices to perform system work Configuration. But to Assemble them on a PCB, is usually not an exact solution for following reasons.

1. cost of system will be increase.
2. design on PCB requires a lot of effort and time
3. due to large size of PCB, Pysical Size is more.
4. due to discrete components the system is not reliable.

By consider all these problems intel decided to integrate a microprocessor along with I|o ports, memory and peripherals in a single system. and make self sufficiant. The single syste device is called "micro Controller"

design with microcontroller has following advantages:

1. The overall system cost is low.
2. The size of product is small
3. The system design requires little efforts and easy mainta
4. provides Software security features
5. All components available in a 40 pin package in an 8 bit p



Introduction:-

— To make a Complete microcomputer system, only up is not sufficient. It is necessaray to add other peripherals such as ROM, RAM, decoders, drivers, no.of I/o devices.

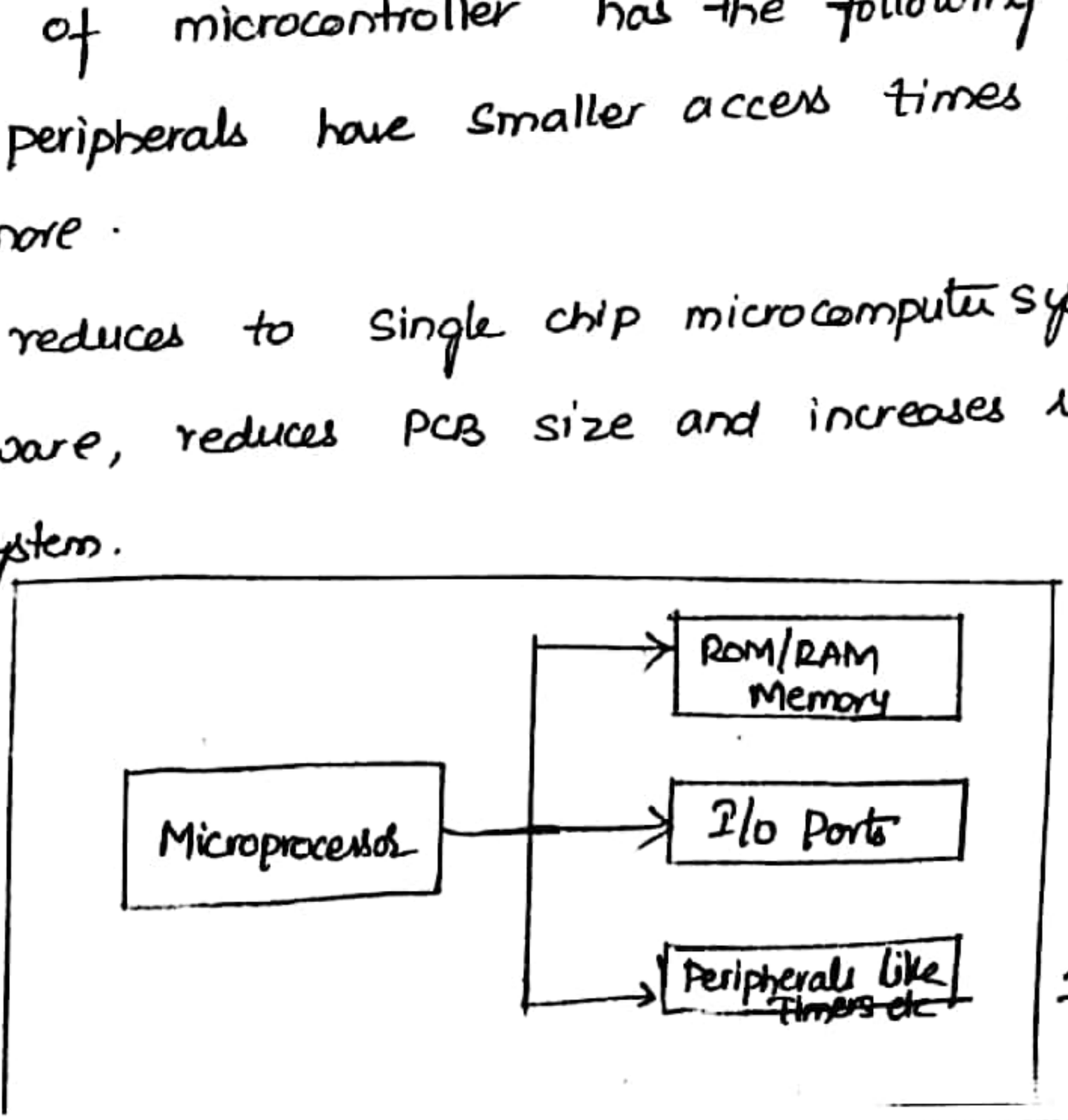
— The key feature of up based Computer system is that it is possible to design a system with a greater flexibility

— It is possible to configure a system as large or small System by adding suitable peripherals.

Microcontroller:— An integrated microprocessor along with I/o Ports and minimum memory into a single chip (ROM, RAM), Parallel I/o, Serial I/o, Counters and a clock circuit.

— The design of microcontroller has the following advantages:

1. Built-in peripherals have smaller access times hence Speed is more.
2. Hardware reduces to single chip microcomputer system.
3. Less hardware, reduces PCB size and increases reliability of the System.



Internal Block diagram of μc.

Overview of 8051 microcontroller:-

— Microcontroller contains most of the Components required to form a µp system. It is called a Single chip micro Computer since it has the ability to easily implement simple Control functions, it is called as microcontroller.

— The features of the 8051 family are as follows:

1) 4096 bytes on-chip program memory.
2) 128 bytes on-chip data memory.
3) Four register banks.
4) 128 bit addressable data memory. (16 bytes bit addressable data memory) ~~User defined software flags~~.
5) 64kBs each program and external RAM addressability.
6) One microsecond instruction cycle with 12MHz Crystal (11.05 MHz)
7) 32 bidirectional I/o lines organized as four 8-bit ports
8) Multiple mode, high Speed programmable serial port.
9) Two multiple mode, 16-bit Timers/counters.
10) Two-level prioritized interrupt structure.
11) Full depth stack for Subroutine return linkage and data storage
12) Direct Byte and Bit addressability.
13) Binary or Decimal arithematic.
14) signed-overflow detection and parity Computation.
15) Hardware multiply and Divide in 4µsec.
16) Integrated Boolean processor for Control applications
17) Upwardly Compatible with existing 8084 s/w.

Architecture of 8051:

B Register

A Register

Temp Reg

Temp Reg

ALU

PSW

Timing & control unit

OSC

Stack pointer

Program Counter

PC Incrementer

DPTR

RAM Address Register

RAM

Program Address Register

EPROM/ ROM

Instruction Reg.

Latch 0

Buffer 0

Port 0 o/p Driver

I/O A_0-A_7 D_0-D_7

Latch 1

Buffer 1

Port 1 o/p Driver

I/O

Latch 2

Buffer 2

Port 2 o/p Driver

I/O A_8-A_{15}

Latch 3

Buffer 3

Port 3 o/p Driver

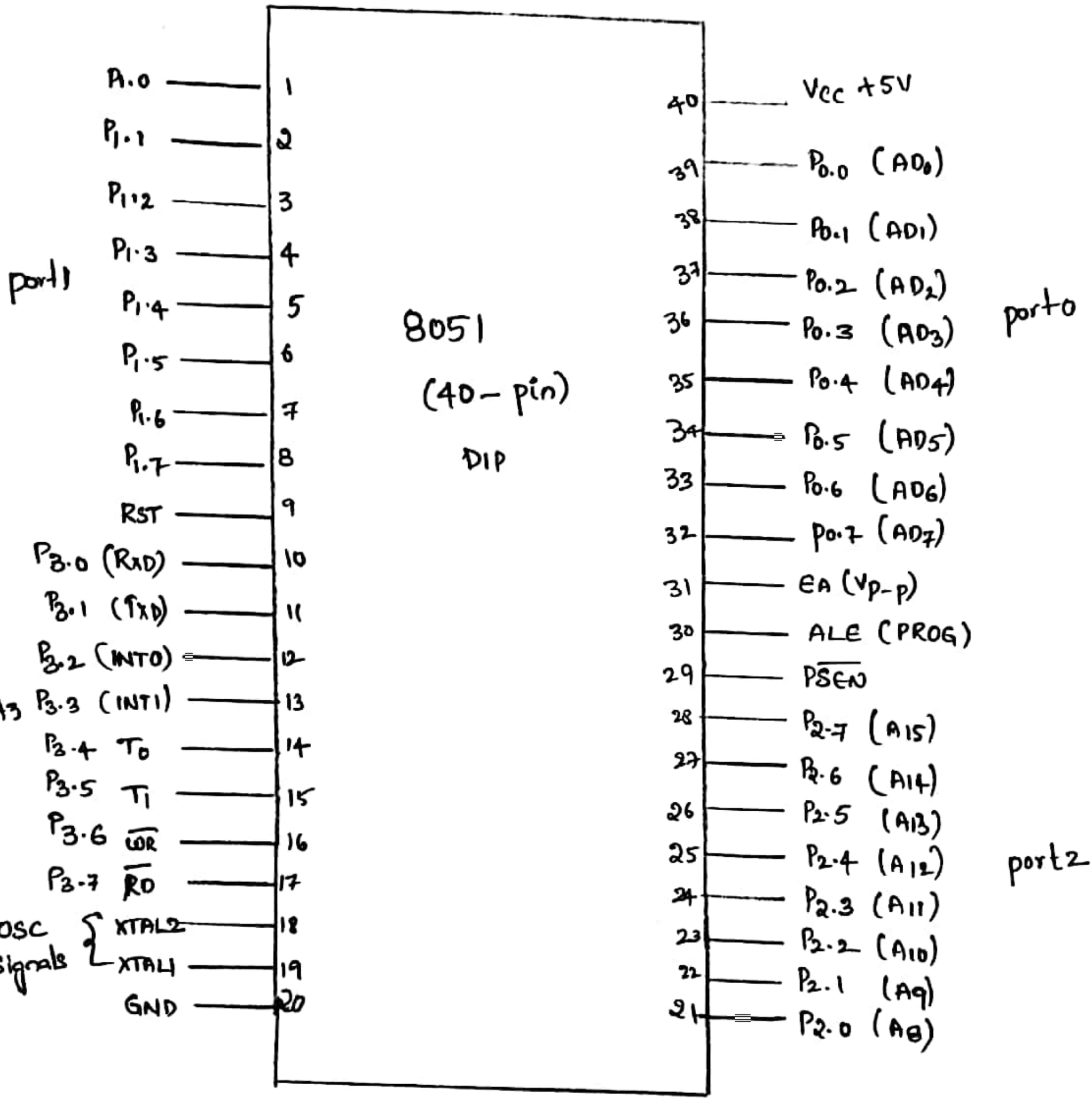
I/O Interrupt Controller Serial Data $\overline{RD}$ - $\overline{WR}$

16-bit address

Interrupt Serial port timer and memory read write Control

Special function Registers

PCON	SCON	TMOD	TCON
TH_0	TH_1	TL_0	TL_1
SBUF	IE	IP	



Pin-out of 8051

- The 8051 is packaged in a 40-pin DIP. It is important to note that many pins of 8051 are used for more than one fn.

- The 8051 has 32 I/o pins Configured as four eight bit parallel ports (P0, P1, P2 & P3). All four ports are bidirectional i,e each pin will be Configured as i/p or o/p (or both).

- All port-pins are multiplexed except the pins of Port 1. Each port Consist of a latch, an o/p driver and an i/p buffer.

Port 0 (pins 32-39):

- Port 0 pins Can be used as I/o pins. The o/p drives and I/p buffers of Port 0 are used to access external memory.

- Port 0 o/ps the low order byte of the external memory address, time multiplexed with the data being written or read. Thus port 0 can be used as a multiplexed address / data bus.

Port 1 (pins 1-8):-

Port 1 pins can be used only as I/o pins

Port 2 (pins 21-28):-

- The o/p drives of port 2 are used to access external memory. Port 2 o/ps the high order byte of external ... address when the address is 16 bit wide.

Otherwise port 2 is used as an I/o port.

Port 3 (pins 10-17):—

All port pins of port 3 are multifunctional. They have special fns.

power-supply pins V_{cc} (pin 40) and Vss (pin 20):—

— 8051 operates on dc power supply of +5V w.r.t ground

Oscillator pins XTAL2 (pin 18) and XTAL1 (pin 19):—

— for generating an internal clk signal, the external osc is connected at these two pins.

ALE (pin 30):—

AD_0 to AD_7 lines are multiplexed. To demultiplex these lines and for obtaining lower half of an address, an external latch and ALE s/g of 8051 is used.

RST (Reset, pin 9):—

— It is used to reset 8051. For proper reset operation, reset signal must be held high atleast for two m/c cycles while osc is running.

PSEN (program store Enable, pin 29):—

— It is the active low o/p control signal used to activate the enable signal of the external ROM/EPROM. It is activated every six oscillator periods while reading the external memory. Thus, this signal act as the read strobe to external program memory.

$\overline{EA}$ (External Access, pin 31):—

When $\overline{EA}$ pin is high (connected to Vcc), program fetches to addresses 0000H through 0FFFH are directed to the internal ROM and program fetches to addresses 1000H through FFFFH are directed to external ROM/EPROM.

— When $\overline{EA}$ is low (grounded), all addresses (0000H to FFFFH) are fetched by program are directed to the external ROM/EPROM.

CPU:—

— The CPU of 8051 consists of eight-bit ALU with associated registers like A, B, PSW, SP, the sixteen bit program counter and Data pointer (DPTR) registers. Along with these registers it has a set of SFRs.

— 8051's ALU can perform Arithematic & logic fns on eight bit variables.

— The arithematic unit can perform +, −, *, /. Logic unit perform logical operations such as AND, OR, EX-OR, rotate, clear, complement.

ALU also looks after branching decisions.

Note:— Unique feature of 8051 is that ALU can also manipulate one bit as well as eight bit data types.

— Individual bit may be set, cleared, complemented, moved, tested and used in logic computation.

Internal RAM:-

- 8051 has 128-byte internal RAM. It is accessed using RAM address register. Internal RAM of 8051 is organised into three distinct areas:
  - Working registers.
  - Bit addressable.
  - General purpose.

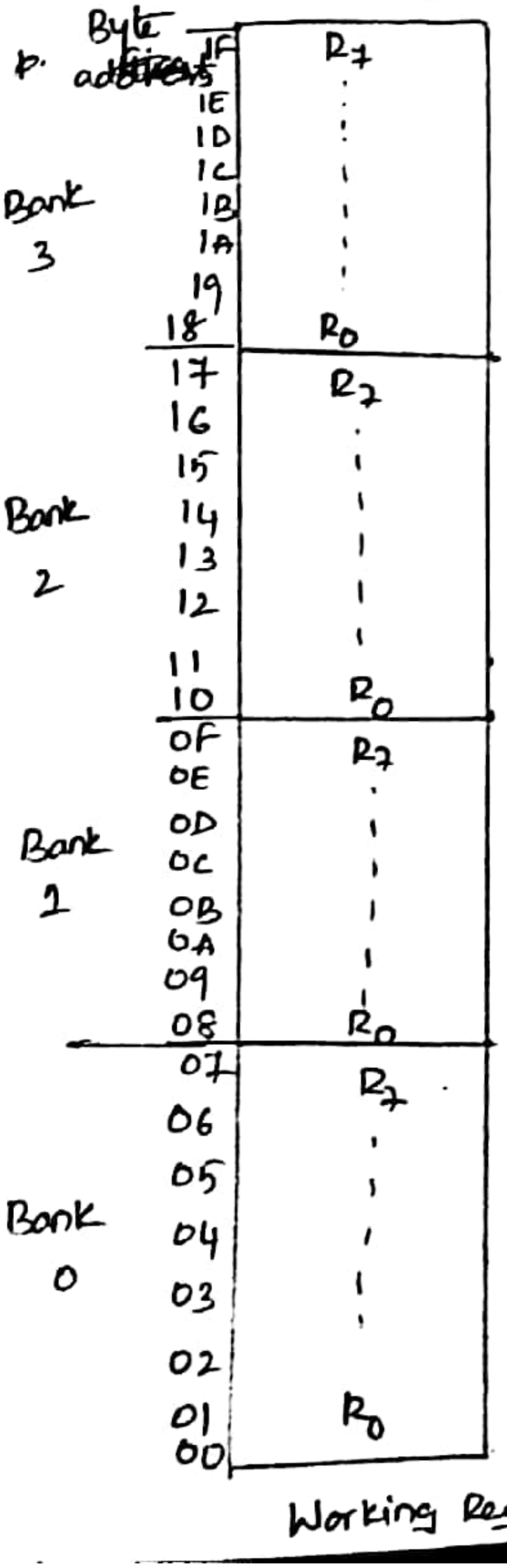
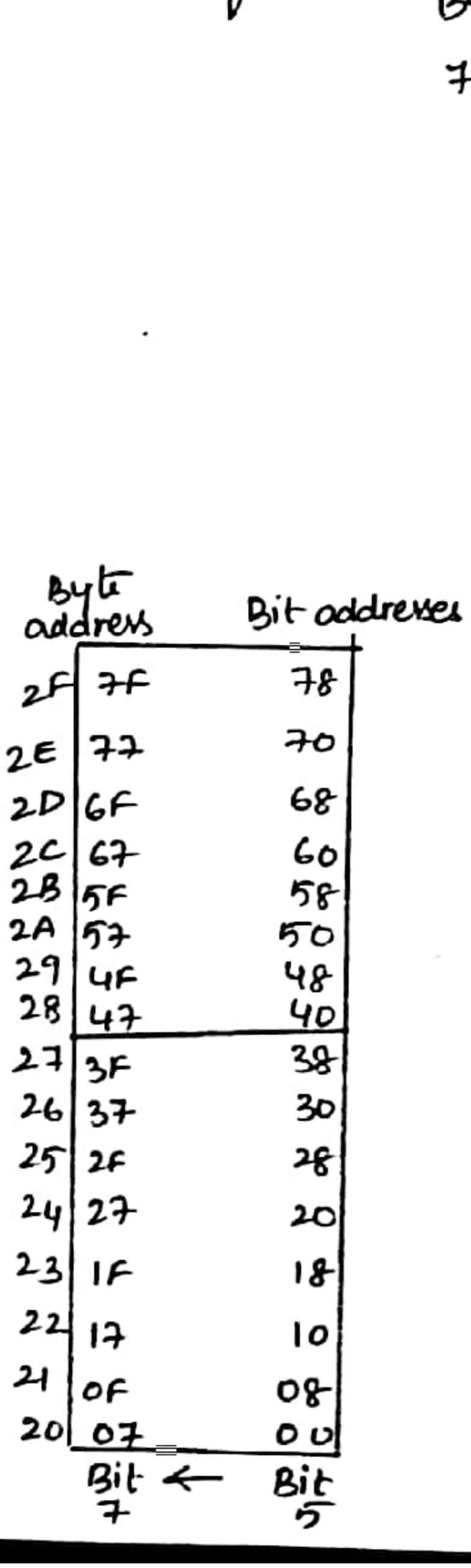
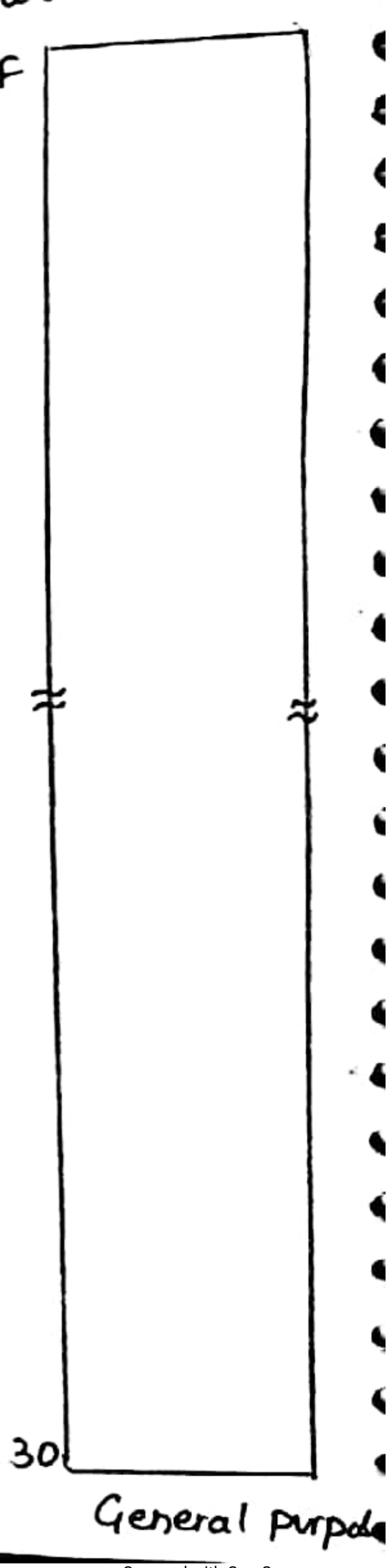


Fig: Organisation of Internal RAM of 8051





1. First 32 bytes from address 00H to 1FH of internal RAM Constitute 32 working registers.

— They are organised into four banks of eight reg each. The four register banks are numbered 0 to 3 and are Consist of eight registers named R_0 to R_7.

— Each register can be addressed by name or by its RAM address. Only one register bank is in use at a time.

— Bits RS_0 and RS_1 in the PSW determine which bank of registers is currently in use.

— Register banks when not selected can be used as general purpose RAM. On reset, Bank 0 is selected.

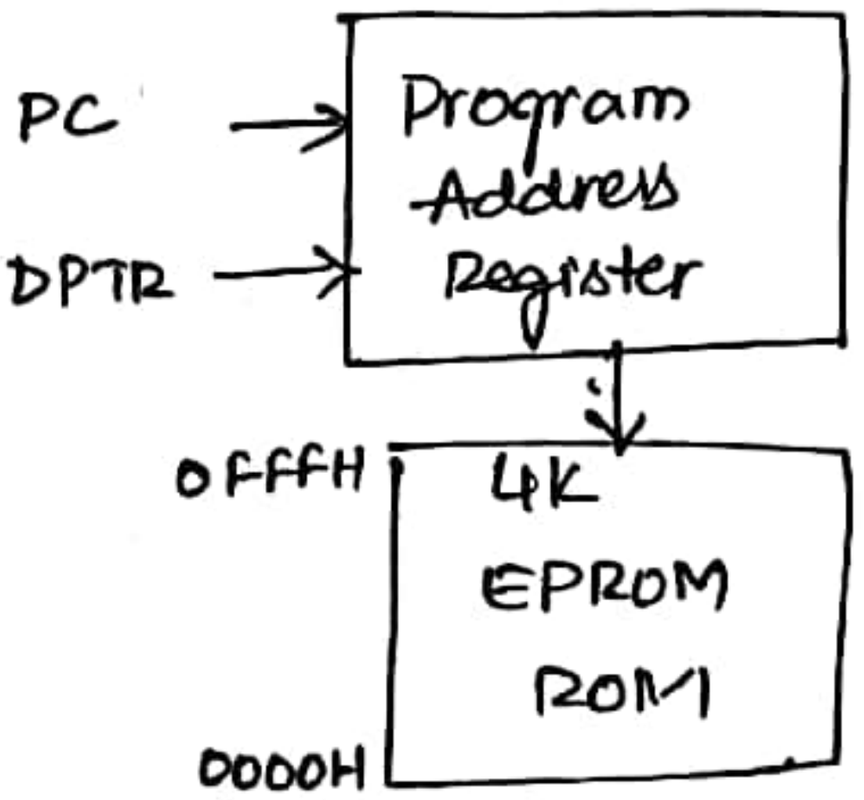
2. The 8051 provides 16 bytes of a bit addressable area. It occupies RAM byte addresses from 20H to 2FH, forming a total of 128 (16x8) addressable bits.

— An addressable bit may be specified by its bit address of 00H to 7FH, or 8 bit may form any byte address from 20H to 2FH.

3. The RAM area above bit addressable area from 30H to 7FH is called general purpose RAM. It is addressable as byte.

Internal ROM:-

- The 8051 has 4KB of internal ROM with address space from 0000H to 0FFFH.
- It is programmed by manufacturer when the chip is built. This part cannot be erased or altered after fabrication. This is used to store final version of the program.
- It is accessed using program address register. The program addresses higher than 0FFFH, which exceed the internal ROM capacity will cause the 8051 to automatically fetch code bytes from external program memory.
- However, code bytes can also be fetched exclusively from an external memory addresses 0000H to FFFFH by connecting the external access pin (EA) to ground.



Input/Output ports:-

- The 8051 has 32 I/o pins configured as four eight bit parallel ports. (P0, P1, P2, P3).
- All four port are bidirectional i,e each pin will be configured as I/P, o/P (or both) under software control.

- All port pins of Port 3 are multifunctional. They have Special fns including two external interrupts, two Counter I/ps, two Special data lines and two timing Control strobes.

Symbol	Position	Name & significance.
$\overline{RD}$	P3.7	Read data Control o/p. Active low pulse generated by hardware when external data memory is read.
$\overline{WR}$	P3.6	Write data Control o/p. Active low pulse generated by hardware when external data memory is written.
T1	P3.5	Timer/Counter 1 external I/p or test pin.
T0	P3.4	Timer/Counter 0 external I/p or test pin.
$\overline{INT1}$	P3.3	Interrupt 1 I/p pin. Low-level or falling edge triggered.
$\overline{INT0}$	P3.2	Interrupt 0 I/p pin. Low level or falling edge triggered.
TXD	P3.1	Transmit data pin for Serial port in UART mode. Clock o/p in shift register mode.
RXD	P3.0	Receive data pin for serial port in UART mode. Data I/o pin in shift reg mode.

Register Set of 8051:—

Register A (Accumulator): It is an 8-bit register. It holds a source operand and receives the result of the arithmematic instructions (+, −, *, /).

— The accumulator can be the source or destination for logical operations and no. of special data movement instructions, including look up tables and external RAM expansion.

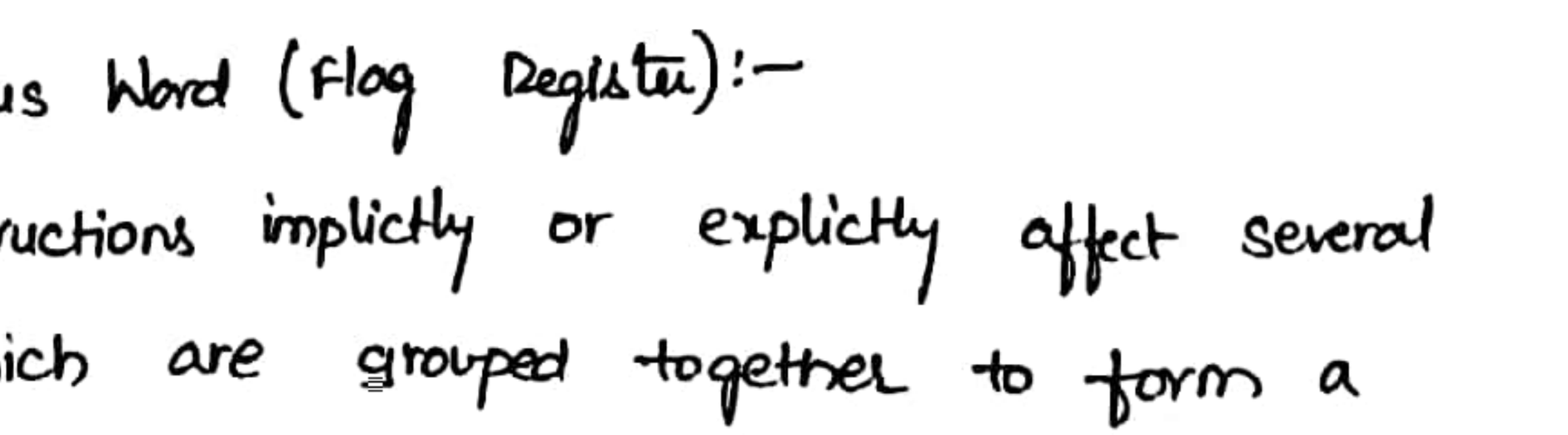
— Several fns apply exclusively to the accumulator: rotate, Parity Computation, testing for zero and soon.

Register B:—

In addition to accumulator, an 8-bit B-register is available as a general purpose reg when it is not being used for the hardware multiply/divide operations.

Program Status Word (Flag Register):—

Many instructions implictly or explictly affect several status flags, which are grouped together to form a program status word. PSW is 8-bit word, containing the following information.



RS1	RS0	Bank selection
0	0	00H-07H Bank 0
0	1	08H-0FH Bank 1
1	0	10H-17H Bank 2
1	1	18H-1FH Bank 3

RS1, RS0 are used to select the working registu bank.

Stack and Stock pointer:—

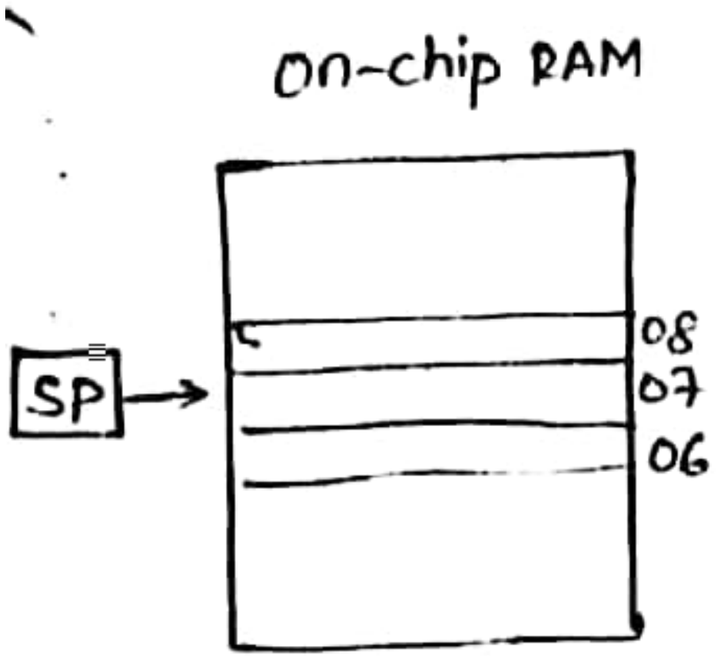
— The stack refers to an internal RAM that is used in Conjunction with opcodes data to store and retrieve data quickly.

— SP req is used by the 8051 to hold an internal RAM addres that is called top of stack.

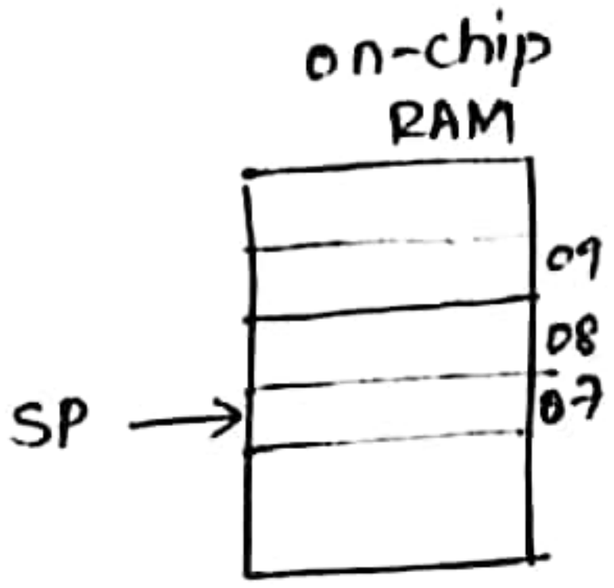
— SP is 8-bit wide. It is increased before data is stored during PUSH and CALL instructions and decremented after data is restored during POP and RET instructions

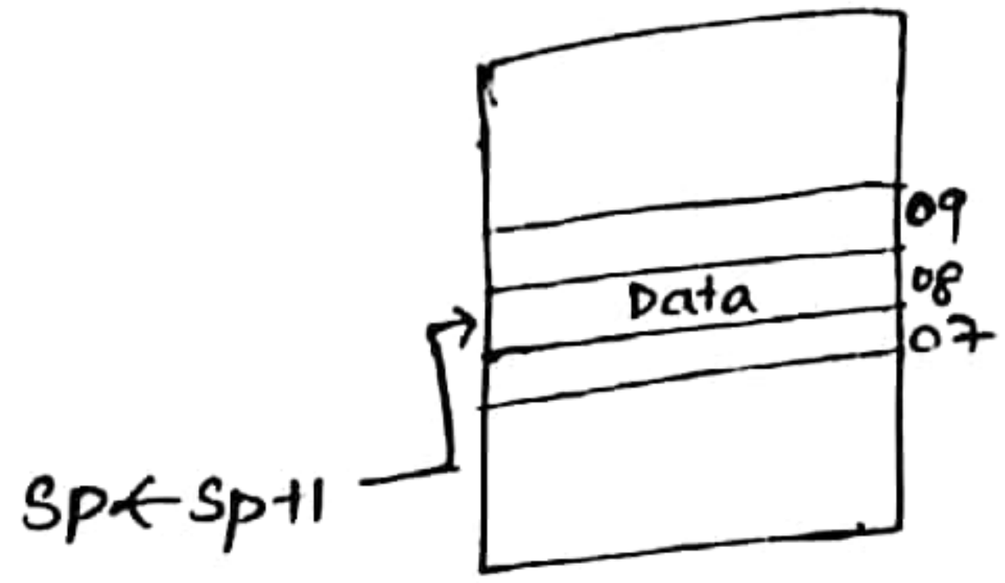
—Thus stack array can reside anywhere in on chip RAM. The SP is initialized to 07H after a reset

— This causes the stack to begin at location 08H.

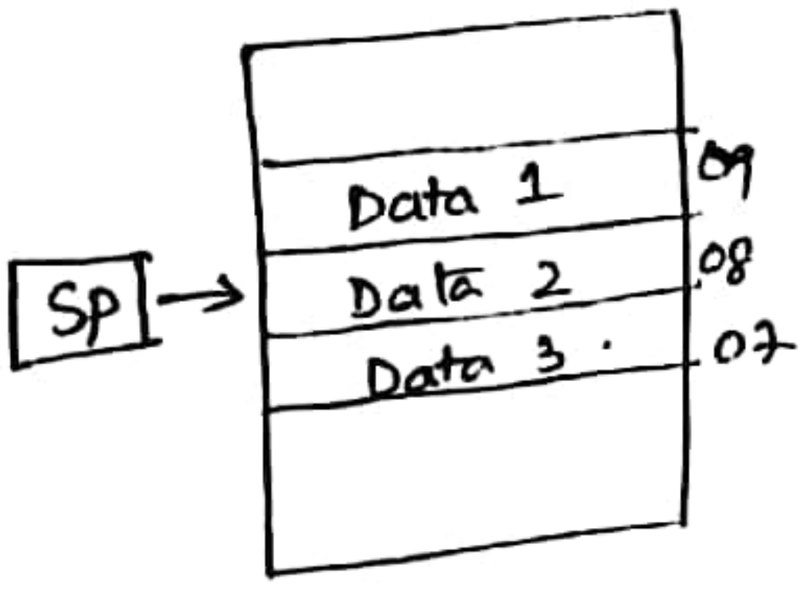


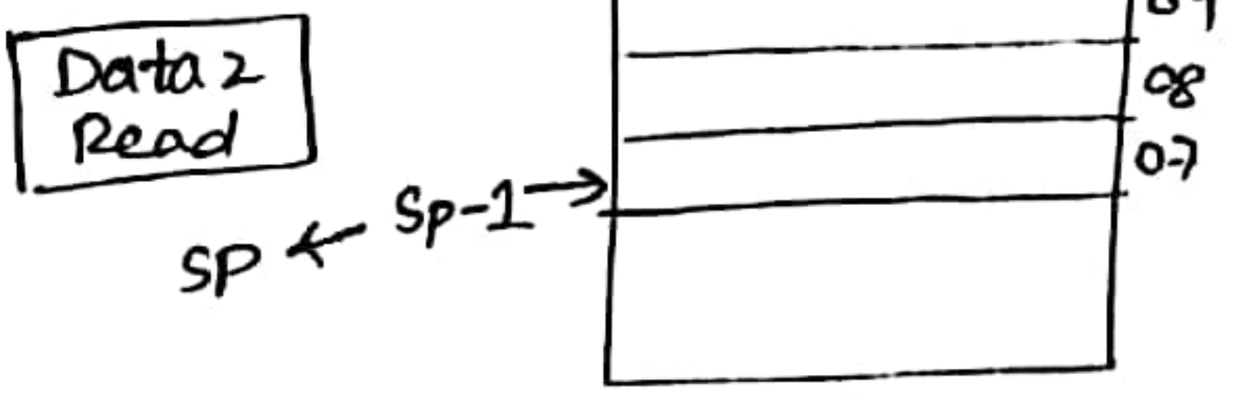
a) Status of stack and sp of reset.





b) Store operation.





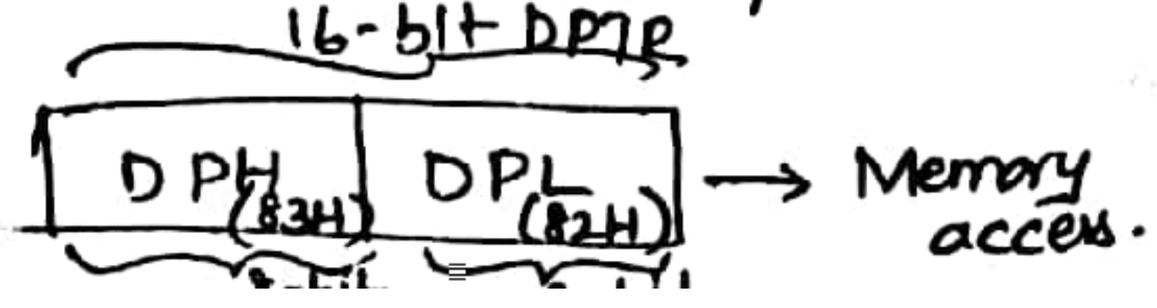
(c) Read operation.

Data pointer (DPTR) :—

— It Consists of a high byte (DPH) and a low byte (DPL). Its fn is to hold a 16 bit address. It may be manipulated as a 16 bit data register or as two independent 8 bit registers.

— It serves as a base register in indirect jumps, look up table instructions and external data transfer.

— The DPTR does not have a single internal address; DPH (83H) and DPL (82H) have separate internal addresses



Program Counter:—

— 8051 has a 16-bit PC. It is used to hold the address of memory location from which next instruction is to be fetched.

Due to this the width of PC decides the maximum program length in bytes.

Eg:- 8051 is 16-bit hence it can address upto 2^{16} bytes (64K) of memory.

— The PC is automatically incremented to point the next instruction in the program sequence after execution of current instruction.

— It may also be altered by certain instructions. The PC is the only reg that does not have an internal address.

Special Function Registers:—

— 8051 uses memory mapped I/o through a set of SFRs that are implemented in the address space immediately above the 128 bytes of RAM.

— All access to the four I/o ports, the CPU registers interrupt Control registers, the timer/Counter, UART and Power Control are performed through registers between 80H and FFH.

Port	Functions.
Port 0	– Used as an I/o port. – Used as a bi-directional low-order addres and data bus for external memor
Port 1	– Used as an Input/output port.
Port 2	– Used as an Input/output port. – Used as a higher order address bus for external memory.
Port 3.	– Used as input/output port or used for alternate fn as shown below. P3.0 – RXD Serial data input. P3.1 – TXD Serial data output. P3.2 – $\overline{INT0}$ External interupt 0. P3.3 – $\overline{INT1}$ External interupt 1. P3.4 – T0 – External Timer 0 I/p P3.5 – T1 External Timer 1. I/p. P3.6 – $\overline{WR}$ External memory write Sign P3.7 – $\overline{RD}$ External memory read sig

Addressing modes of 8051:-

— Data is stored at a source addres and moved to destination addres. The ways by which these addresses are specified are called addressing modes.

— The following four addressing modes are used to access data:

1. Immediate Addressing mode.
2. Registe Addressing mode.
3. Direct Addressing mode.
4. Indirect Addressing mode.

— Mov opcodes involve data transfer within the 8051. memory. The memory is divided into the following four distinct physical parts.

1. Internal RAM.
2. Internal Special-function Reg.
3. External RAM.
4. Internal & External ROM.

— finally the following five types of opcodes are used to move data:

1. MOV
2. MOVX
3. MOVC
4. PUSH and POP.
5. XCH.

Addressing modes of 8051 Microcontrollers

1. Immediate addressing mode
2. Directing adressing mode
3. Register adressing mode
4. Register Indirect adressing mode
5. Register specific adressing mode
6. Index adressing mode.

Immediate Adressing mode:

Data is provided in instruction itself immediatly. # symbol is used for immediate data

```
MOV A, #52H     ; 52 immediate move to A register
MOV Ri, #33H    33 value added immediate R of adress bank of memory
MOV DPTR, #AF00H  DPTR → external memory location
                  AF00 store in external DPTR memory.
```

Direct Addressing mode :-

Source or destination address is specified in instructions

```
MOV  Regis adress
     A, 50H   → move data in 50H to A Register
MOV  Ri, 42H
ADD  A, 51H
MOV  80H, Ri
```

Register Addresseing Mode :-

It is used to access eight working registers at selected registed bank to register

```
MOV  A, Ri
ADD  A, Ri
MOV  Ri, A
```

Register indirect adressing Mode :-

It can be used to access internal and external adress. The source or destination address is given in the register

It can uses the symbol " @"

MOV A, @Ri°

MOV @Ri°, DPTR

ADD A, @Ri°

Register specific addressing Mode :-

In this addressing mode, there will be single operand These Instructions can works specifically in particular register

Eg:- RL A

SWAP A

Indexed Addressing Mode :-

In this mode only program memory can be access i.e ROM only. In this Indexed registered are program counter and DPTR only

* Destination is always Accumulator register only.

MOVC A, @A + DPTR

MOVC A, @A + PC

Eg:-1 A = 30H

DPTR = 1125

1155H → This is adress, data of address 1155 is stored in accumalator register

2:- MOVC A, @A + PC

A = 50 H

P.C → 1120 H

1170H → this an adress.

The data of 1170 address can be moveand stored in A register.

* In this destination always A register

Instruction Set of 8051:-

Classification:-

1. Data Transfer Instructions.
2. Boolean Variable Manipulation Instructions.
3. Arithematic Instructions.
4. Logical Instructions.
5. Program flow Control instructions.
6. Interrupt Control instructions.

Data Transfer Instructions:-

MOV instructions:- MOV instructions within the registers, Internal RAM and SFR in 8051.

Instruction	Action.
MOV A, Rn	Move Rn into A. Rn - one of reg from selected bank $R_0 - R_7$
MOV Rn, A	Move A into Rn.
MOV A, # data	Move immediate 8 bit data into A.
MOV A, @Ri	Mov into A the byte from the address pointed by the Ri.
MOV DPTR, data 16.	Move 16 bit data into DPTR.

MOV C:— MOVC instruction means move (copy) the 8-bit data from one source at the program memory to the register A destination.

Eg:-

MOVC A, @A+DPTR → Copy the code byte, found at the ROM address formed by adding A and the DPTR, to A.

MOVC A, @A+PC → Copy the code byte, found at the ROM address formed by adding A and the PC, to A.

MOV X:— This is used for transfer of 8 bit data into A and from A using the external data memory addresses using DPTR or Ri as a pointer in 8051.

Eg:-

MOVX A, @DPTR → Copy the Content of external address in DPTR to A.

MOVX @R0, A → Copy data from A to 8-bit address in R0.

MOVX @DPTR, A → Copy data from A to external address in DPTR.

MOVX A, @Rp → Copy the Content of external address in Rp to A.

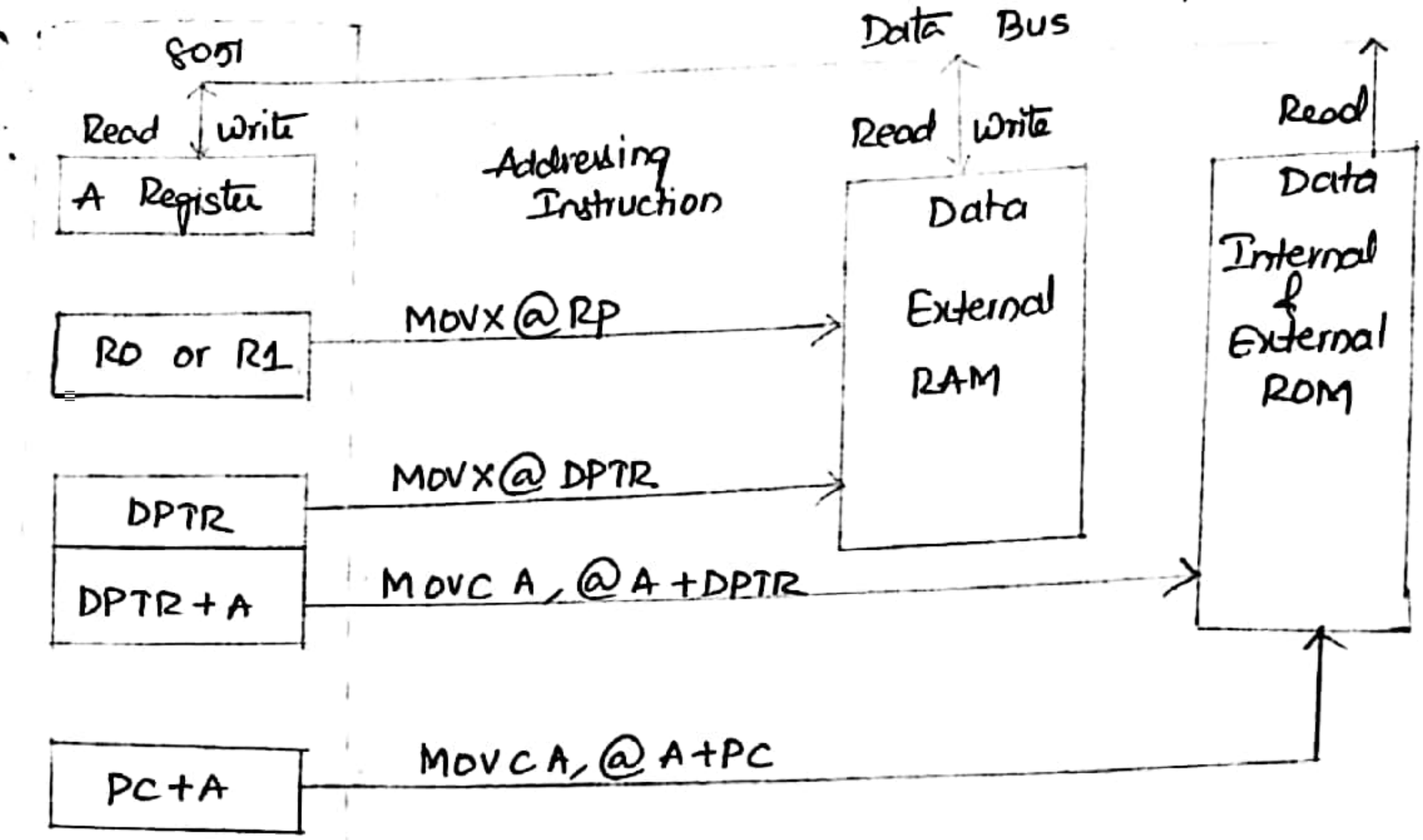


Fig: External Addressing Using MOVX & MOVC

PUSH & POP instructions:—

— A PUSH opcode copies data from source address to the stack. SP in incremented by 1 before the data is copied to the internal RAM location contained in SP so that the data is stored from low addresses to high addresses in the internal RAM.

— The stack grows up in memory as it is pushed. Excessive pushing can make the stack exceed 7Fh, after which pushed data is lost.

— A POP opcode copies data from stack to destination. SP is decremented by 1 after data is copied from stack RAM address to the direct destination to ensure that the data placed on stack is retrieved in the same order as it was stored.

Eg:-

PUSH add → Increment SP; Copy the data in add to the internal RAM address contained in SP.

POP add → Copy data from internal RAM address Contained in SP to add ; decrement SP.

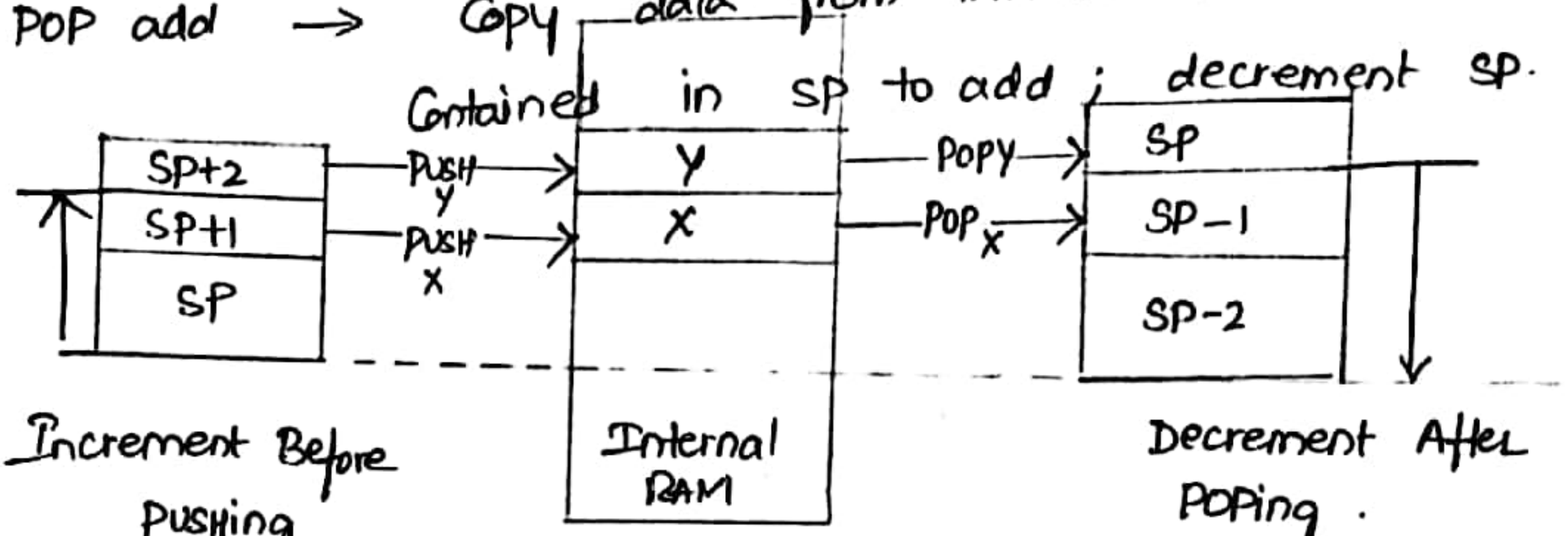


Fig: PUSH & POP the Stack.

XCH -Instructions (Exchange):—

— MOV, PUSH & POP instructions all involve Copying the da found in Source addres to destination address, the original data in the Source is not changed.

— Exchange instructions actualy move data in two directio from Source to destination and from destination to Source.

— All addressing modes except immediate may be used in XCH instructions.

Eg:- XCH A, Rr → Exchange data bytes b/w register Rr & A (Rr → Reg from R0-R7)

XCH A, @R1 → Exchange bytes b/w reg A and address in R1.

XCHD A, @R1 → Exchange lower nibble in A and the address in R1.

XCH A, 0F0h → Exchange bytes b/w reg A & reg B.
(0F0h → address of Reg B)

2. Data and Bit Manipulation instructions:—

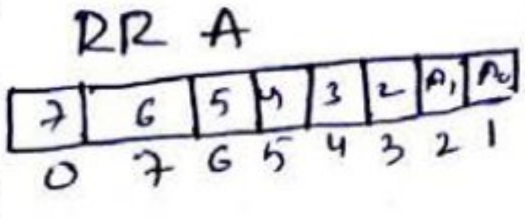
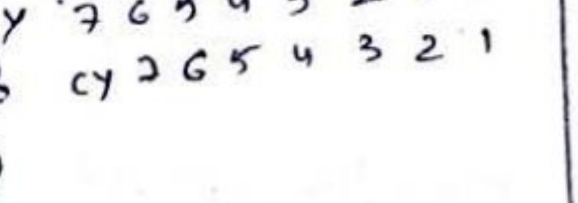
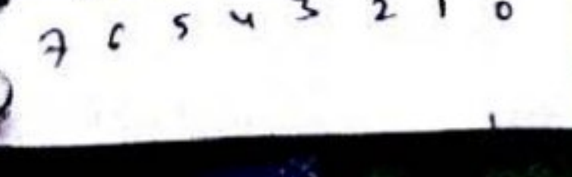
Byte Manipulation operations

CLR | SWAP | CPL | RL | RLC | RR | RRC

Bit Manipulation operations

CLR | SET B | CPL | ANL | ORL | MOV

Byte-Level Logical Operations:—

Mnemonic	Operation.
CLR A	Clear each bit of A register to 0.
CPL A	Complement each bit of A : every 1 becomes a 0, and each 0 becomes 1.
RR A 	Rotate the A register one bit position to right; bit A0 to bit A7, A7 to A6, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1, and A1 to A0.
RRC A 	Rotate the A register ~~one bit position to~~ and the Carry flag, as a ninth bit, one bit position to right; bit A0 to CY, CY to A7, A7 to A6, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to A1 & A1 to A0.
RL A 	Rotate the A reg one bit position to left bit A0 to A1, A1 to A2, ... A7 to A0.

RLC A	Rotate A reg & CY as a ninth bit, one bit position to left, bit A0 to A1, A1 to A2, ... A6 to A7, A7 to CY and CY to A0.
SWAP A	Interchange the nibbles of reg A; put the high nibble in low nibble position and the low nibble in the high nibble position.

Bit Manipulation Instructions:—

Mnemonic	Operation.

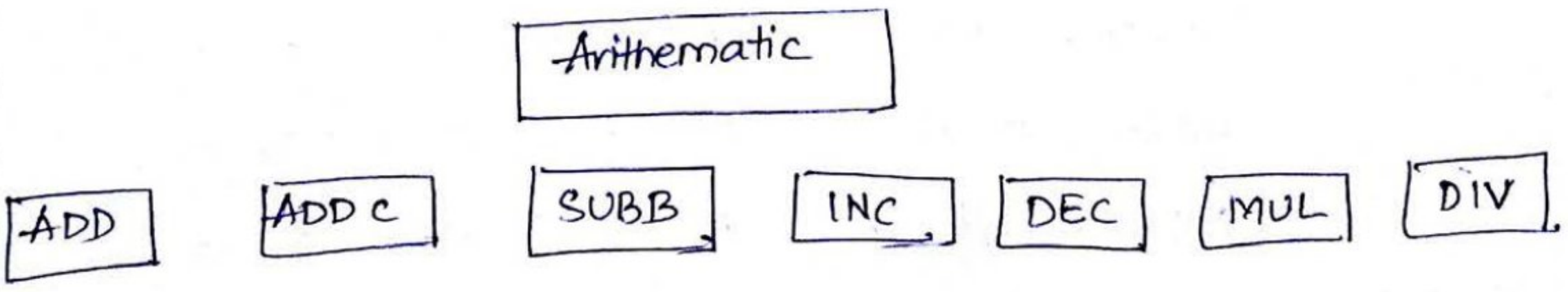
Mov C, bit
(or)
Mov C, b
	Copy the addressed bit to the C flag.

Mov b, C (bit)	Copy the C flag to the addressed bit.
CLR C	Clear the C flag to 0.
CLR b	Clear the addressed bit to 0.
CPL C	Complement the C flag.
CPL b	Complement the addressed bit.
SETB b	Set the addressed bit to 1.
SETB C	Set the C flag to 1.
ANL C, b	AND C and the addressed bit; put the result in C
ANL C, $\overline{b}$	AND C and the Complement of addressed bit; put result in C; the addressed bit is not altered.
ORL C, b	OR C and the addressed bit; put the result in C
ORL C, $\overline{b}$ (or) ORL C, /b	OR C and the Complement of addressed bit; put the result in C; the addressed bit is not altered.

Examples of rotate and Swap operations:—

Mnemonic	Operation Result.
MOV A, #0A5h	A = 10100101b = A5h.
RRA	A = 11010010b = D2h.
RR A	A = 01101001b = 69h.
RR A	A = 10110100b = B4h.
RR A	A = 01011010b = 5Ah.
SWAP A	A = 1010 0101b = A5h.
CLR C	C = 0 ; A = 10100101b = A5h.
RRC A	C = 1 ; A = 01010010b = 52h.
RRC A	C = 0 ; A = 10101001b = A9h.
RL A	A = 01010011b = 53h.
RL A	A = 1010 0110b = A6h.
SWAP A	C = 0 ; A = 01101010b = 6Ah.
RLC A	C = 0 ; A = 11010100b = D4h.
RLC A	C = 1 ; A = 10101000b = A8h.
SWAP A	C = 1 ; A = 10001010b = 8Ah.

3. Arithematic Instructions:—



Mnemonic	Operation.
ADD A, #n	Add A and the immediate number n; put sum in A.
ADD A, Rr	Add A and Reg Rr; put sum in A.
ADD A, add	Add A and the address Contents ; put sum in A
ADD A, @Rp	Add A and Contents of address in Rp; put sum in A
ADDC A, #n	Add Contents of A, the immediate number n, and C flag; put sum in A.
ADDC A, add	Add Contents of A, the direct address Contents, and c flag; put sum in A.
ADDC A, Rr	Add Contents of A, reg Rr, and c flag, put the Sum in A.
ADDC A, @Rp	Add Contents of A, the Contents of indirect address in Rp and c flag; put sum in A.

Note:- ADDC → used to add a carry after the LSBY addition in a multi-byte process.

ADD → used for LSBY addition.

Eg:-

```
MOV  A, #1Ch        A = 1Ch.
MOV  R5, #0A1h      R5 = A1h.
ADD  A, R5          A = BDh ; C=0, OV=0.
ADD  A, R5          A = 5Eh; C=1; OV=1.
ADDC A, #10h        A = 6Fh; C=0 ; OV=0.
ADDC A, #10h        A = 7Fh; C=0; OV=0.
```

Mnemonic	Operation.
SUBB A, #n	Subtract immediate number n and c flag from A; put the result in A.
SUBB A, add	Subtract the Contents of add and the c flag from A; put the result in A.
SUBB A, Rs	Subtract Rs and the c flag from A; put the result in A.
SUBB A, @Rp	Subtract the Contents of address in Rp and the c flag from A; put result in A.

Note:— To Set the Carry flag to 0 if it is not to be included as part of the subtraction operation.

Eg:—

MOV A, #3Ah	A = 3Ah.
MOV 45h, #13h	Address 45 = 13h.
SUBB A, 45h	A = 27h; c = 0, OV = 0.
SUBB A, 45h	A = 14h; c = 0; OV = 0.
SUBB A, #80h	A = 94h; c = 1, OV = 1.
SUBB A, #22h	A = 71h; c = 0; OV = 0.
SUBB A, #0FFh	A = 72h; c = 1, OV = 0.

MUL AB — Multiply A by B; put low-order byte of product in A, put high order byte in B.

Note:— There is no Comma b/w A & B in MUL mnemonic.

Eg:—

MOV A, #7Bh	A = 7Bh.
MOV 0F0h, #02h (→ Address of B Reg)	B = 02h.
MUL AB	A = F6h & B = 00h; OV flag = 0.
MOV B, #0FEh	B = FEh.
MUL AB	A = 14h and B = F4h; OV flag = 1.

Mnemonic	Operation.
DIV AB	Divide A by B; put the integer part of quotient in reg A and integer part of remainder in B.
Note:-	— The original Content of A and B are lost — There is no Comma b/w A & B in DIV mnemonic
Eg:- MOV A, #0FFh MOV 0F0h, #2Ch DIV AB	A = FFh (255d). B = 2C (44d). A = 05h (Quot) and B = 23h (Rem) [255d = (5 x 44) + 35].
DIV AB	A = 00h and B = 05h [05d = (0 x 35) + 5].
DIV AB	A = 00h & B = 00h [00d = (0 x 5) + 0].
DIV AB	A = ?? and B = ?? ; OV flag is set to 1
DAA	Adjust the sum of two packed BCD numbers found in A reg, leave the adjusted number in A.
Note:-	All numbers used must be in BCD form before addition. Only ADD & ADDC are adjusted to BCD by DAA.

Eg:- MOV A, #42h
ADD A, #13h
DAA
ADD A, #17h
DA A
ADDC A, #34h
DA A
ADDC A, #11h
DA A
	A = 42 BCD.
	A = 55h; C = 0.
	A = 55h; C = 0.
	A = 6Ch; C = 0.
	A = 72 BCD; C = 0.
	A = A6 h; C = 0.
	A = 06 BCD, C = 1.
	A = 18 BCD; C = 0.
	A = 18 BCD; C = 0.

Mnemonic	Operation.
INC A	Add a 1 to A reg.
INC Rr	Add a 1 to reg Rr.
INC add	Add a 1 to Contents of direct memory address.
INC @Rp	Add a 1 to Contents of memory address in Rp.
INC DPTR	Add a 1 to the 16-bit DPTR.
DEC A	Subtract a 1 from reg A.
DEC Rr	Subtract a 1 from reg Rr.
DEC add	Subtract a 1 from Contents of direct memory access.
DEC @Rp	Subtract a 1 from Contents of memory address in Rp.

Note:— No math flags are affected.
All 8-bit address Contents overflow from FFh to 00h.
DPTR is 16 bits; DPTR overflows from FFFFh to 0000h.
The 8-bit address Contents under-flow from 00h to FFh
There is no DEC DPTR to match INC DPTR.

Eg:—

MOV A, #3Ah	A=3Ah.
DEC A	A=39h.
MOV R0, # 15h	R0=15h.
MOV 15h, # 12h	Internal RAM addr 15h=12h.
INC @R0	" " " 15h=13h.
DEC 15h	" " " 15h=12h
INC R0	R0=16h.
MOV 16h, A	Internal RAM addr 16h=39h.
INC @R0	" " " 16h=3Ah
MOV DPTR, #12FFh	DPTR = 12FFh.
INC DPTR	DPTR = 1300h.
DEC 83h	DPTR= 1200h (SFR 83h is the DPH byte).

4. Logical Instructions:— ANL, XRL, ORL, CPL

Mnemonic	Operation.
ANL A, #n	AND each bit of A with the same bit of immediate number n; put the result in A.
ANL A, add	AND each bit of A with the same bit of the direct RAM addres, put the result in A.
ANL A, Rr	AND each bit of A with the same bit of Rr, put the result in A.
ANL A, @Rp	AND each bit of A with the same bit of the Contents of RAM addres contained in Rp; put the results in A.
ANL add, A	AND each bit of A with the direct RAM addres, put the results in the direct RAM address.
ANL add, #n	AND each bit of the RAM address with the same bit in the number n; put the result in the RAM address.
ORL A, #n	
ORL A, add	
ORL A, Rr	
ORL A, @Rp	
ORL add, A	
ORL add, #n	
XRL A, #n	
XRL A, add	
XRL A, Rr	
XRL A, @Rp	
XRL add, A	
XRL add, #n	
CLR A	Clear each bit of A reg to 0.
CPL A	Complement each bit of A; 1 ↔ 0.

5. Program-flow Control Instructions:-

i) Delay-cycle (NOP) instructions:-

NOP - No-operation, PC gets the address of next instruction on incrementing at NOP.

ii) Long, Absolute and Short Jumps:-

LJMP addr 16 : Jump to the next address given by two bytes in the instruction.

AJMP addr 11 : Jump to next address given by 3 higher bit in the instruction first byte and 8 lower in second byte.

SJMP rel : Jump in the range between -128 and +127 from address of the next instruction which would have executed in case of no jump.

JMP @ A+DPTR : Jump in next addresses given by addition of 8-bit of A with 16-bit of DPTR.

iii) Conditional Short Relative Jumps:-

JNZ rel : Jump to a relative address if A is not zero.

JZ rel : Jump to a relative address if A is zero.

JNC rel : Jump to a relative address if C is not 1 (on no carry).

JC rel : Jump to a relative address if C=1 (on Carry).

JB bit, rel : Jump to a relative address if addressed bit a (vi)
(bit set)

JNB bit, rel : Jump to relative address if addressed bit 0 (bit not set).

JBC bit, rel : Jump to a relative address if addressed bit 1 (bit set) and reset Carry (make c=0).

Note:- rel is a signed number and a signed 8-bit num has the decimal value b/w -128 to +127.

iv) Decrement and Conditional Jump on zero:-

DJNZ Rn, Rel : Decrement Rn, and jump if Rn is still not zero.

DJNZ direct, Rel : Decrement byte at the direct and jump if byte is still not zero.

v) Jump after Comparision:-

CJNE A #data, rel : Compare A and immediate data and jump if both not equal.

CJNE Rn, #data, rel : Compare Rn, and immediate data jump if both not equal.

CJNE A, direct, rel : Compare the bytes at A and direct and jump if both not equal.

CJNE @Ri, data, rel : Compare byte from the address pointed by Ri, and immediate data and jump if both not equal.

(vi) Call to a Routine – Unconditional and Return from Routine:-

LCALL addr 16 : Call to the next address given by two bytes in the instruction (lower address byte first).

ACALL addr 11 : Call the next address given by 3 higher bits in the instruction first byte and 8 lower bits in second byte.

RET ; Return to PC the saved PCL and PCH from stack.

6. Interrupt Control flow (RETI instruction):—

RETI : Return into PC the saved PCL and PCH from the stack.

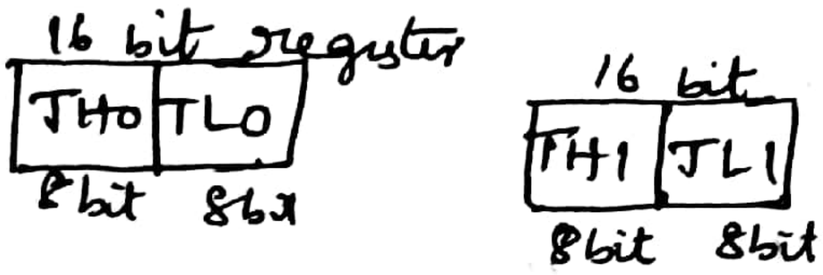
Timers of 8051 micro controller :-

8051 has 2 timer/counters circuits which can be used either as timers or counters.

* Timer → 1. Used to generate time delay
2. Used to generate/maintain the timings of an operation in syn. with clock signal.

* counter :- To count the no of events happening outside the controller.

* 8051 has two 16 bit timer registers
(i) Timer 0 (ii) Timer 1



TL – Timer low byte register
TH → Timer High byte register

2. Based on operation two registers used in Timer/counters
a) TMOD (Timer mode) Register
b) TCON (Timer Counter) Register.

TMOD Register :- It is 8 bit special Function register and responsible for following operations.
(1) Select the timer 0 as timer/counter
(2) Select the Timer 1 as timer/counter
(3) Select the modes of the timer.

MSB 7 6 5 4 3 2 1 LSB 0 — 8bit register

Gate	C/T	m_1	m_0	Gate	C/T	m_1	m_0
Timer 1				Timer 0			

1st 4 bits are Timer 0 and 2nd 4 bits are Timer 1 mode. Where as in 4 bits m_0, m_1 are used to select timer mod

m_1	m_0	operating mode
0	0	Mode 0 operation : 8 bit Timer/counter with 8 bit Prescalar. So it is 13 bit Timer.
0	1	Mode 1 : It is a 16 bit Timer/counter. Both THx & TLx are cascaded.
1	0	Mode 2 : It is 8 bit auto reload Timer/counter
1	1	This Function depends on the Timer 0 and Time 1.

C/$\bar{T}$:-

→ C/$\bar{T}$ = 0 For selecting Timer operation
= 1 For selecting counter operation

Gate (G) :→ 0

If G=0, Timer0 / Timer1 will be in run mode when TR_x (Run control) of TCON and $\overline{INT}_x$ (Intern of External bit) is also high

G=1 Timer0/ Timer1 will be in run mode when only TR_x Bit of TCON is high

TCON (Timer Control) Register :-

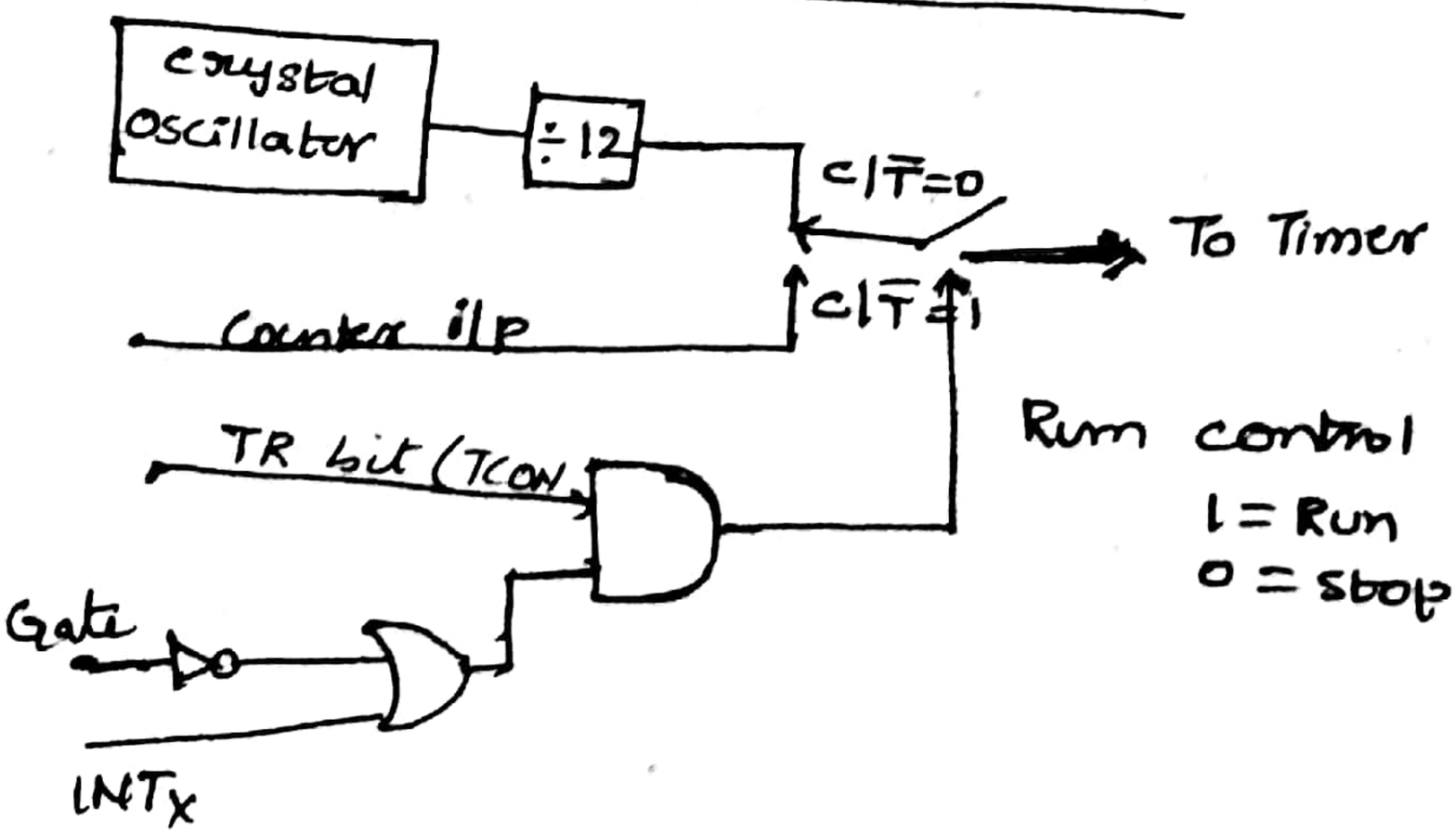
It is an 8 bit special Function register and can be used to control the Following operations.

1. It can start or stop of Timer
2. It can provide status of external interrupts INTx, This INTx has two external signals INT0 and INT1
3. To provide the status of Timer overflow register TF0 and TF1
4. It can be used to configure the interrupt either as level-triggerd (or) edge triggered.

MSB 7	6	5	4	3	2	1	LSB 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

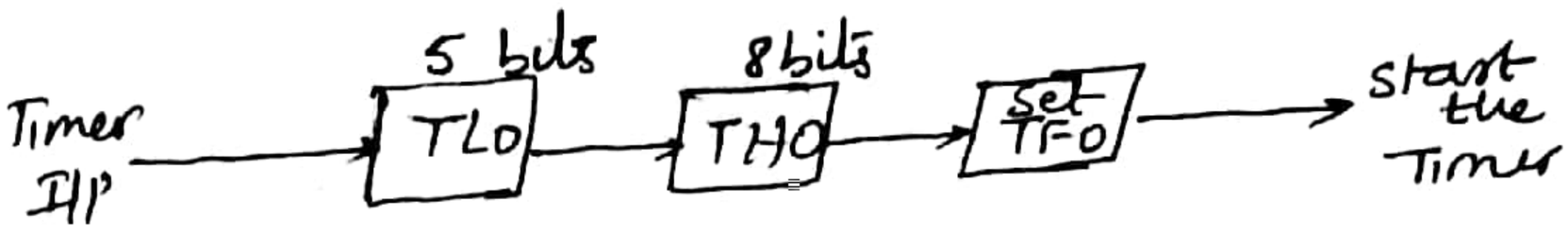
TF_1 → Timer 1 overflow Flag
TR_1 → Timer 1 Run control Flag
TF0 → Timer 0 over Flow Flag
TR0 → Timer 0 Run control Flag
IT0 → interrupt 0 type control bit
IE_1 → interrupt 1 type control bit It is edge triggered
IT_1 → interrupt 1 type control bit
IE_0 → Interrupt 0 Edge Flag, means enable Flag, i.e interrupt Signal INT (hardware interrupt

Control circuit for the timer



Timer modes

(i) **Mode 0**: It is a 13 bit TIC (8 bit TIC + 5 bit prescalar)



Take 5 bits from low byte register TL0 and 8 bits from TH0 and set Timer 0 overflow flag to 1 to start the operation of Timer.

```
MOV   TMOD,  #00H ;
MOV   THD,   #0F0H;
MOV   IE,    #82H ;
SETB  TRO;
```

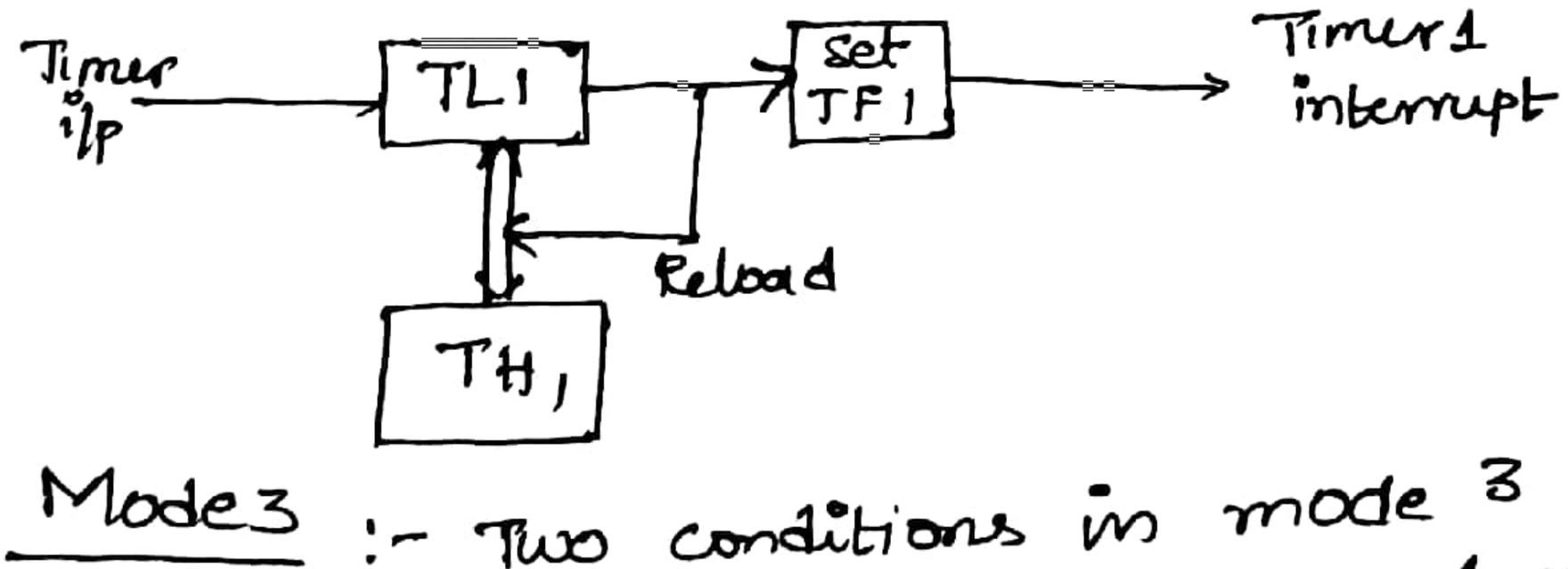
(ii) **MODE 1**:-

It is the 16 bit Timer/counter

consider both together TH0, TL0 as 16 bit data and set TF0. we get Timer 0 interrupt controller

```
MOV  TMOD, #01H
MOV  TL0, #0F0H
MOV  TH0, #0FFH
MOV  IE, #82H,
SETB TR0;
```

Mode 2 : It is the 8-bit auto reload timer/counter



Mode3 :- Two conditions in mode 3 operation

(i) when the timer 0 is working in mode 3

- TL0 used as an 8 bit Timer/counter
- It will be controlled by TF0 & INT0 i/ps

- TH0 is used as an 8 bit timer not counter, it will be controlled by TR1 & TF1.

(ii) When Timer 1 is working in mode 3:-

* It holds the count for the events.
* It does not turn.

8051 Serial communication

Serial communication is used to transfer data b/w two systems located at some distances. It will use single data line instead of 8 bit parallel lines. For serial communication the data will be received by parallel - series conversion. It can use to transfer data in 2 methods

1. Synchronous Serial Communication

It is transfer block of data at a time, there are special chips are used for serial data transfer such as UART or USART. The data will be transferred in duplex form hence it require two wire conductors one for transmission and other reception to transfer at a time

2. ASynchronous Serial Communication :-

widely used for character oriented transmission. each. character is placed b/w start and stop bits ex:- ASCII character placed between start and stop bit. The start bit always one. and stop bit always zero.

Data Transfer rate :-

mostly widely used data transfer rate formate is 1Kbps

RS-232 standards

RS-232 is interfacing standard and is mostly used I/O Interfacing Standard. It is used converts such Voltage Converter as MAX232 for convert TTL to RS232 Voltage level. Whereas RS-232 has 25 pins, many of pins are used for handshaking signals like DTR (Data terminal ready), DSR (data set ready), RTS (Request to send), CLR (Clear to send), RI (Ring Indicator).

8051 Serial port programming :-

It use SBUF register, SCON Register. Used to data Transfer.

following steps taken to transfer data

1. The TMOD register load the value 20H set data transfer rate

2. The TH1 is loaded with one of value
3. SCON is loaded with value of 50H → used for stop and stop bit
4. TR1 set to 1 to start timer1
5. TI cleard by "CLR" signal
6. NOW character being Transferred to SBUF register
7. NOW TI Flag set. check data completly transferred or not
8. To transfer next character go to step 5.

```
    MOV   TMOD, #20H
    MOV   TH1, #61
    MOV   SCON, #50H
    SETB  TR1
Again MOV  SBUF, #"A"
HERE: JNB  TI, HERE
      CLR  T,
      SJMP AGAIN.
```

Interrupts in 8051 Micro Controller :-

Interrupts stop the current execution or task of program. After perform ISR routin task again restart the program. 8051 has mainly 5 interrupt signals such as INT0, TF0, INT1, TF1, RI/TI.

Each interrupt can be enabled or disabled by bits of IE of TCON and whole interrupt system can be disabled by clearing EA bit of same register.

Among 5 interrupts two are external interrupts and 3 are the internal interrupts.

INT1 - External hardware interrupt
INT0 - External hardware interrupt
} External hardware

TF0 - Timer0 overflow interrupt
TF1 - Timer 1 overflow interrupt
RI|TI - Serial communication interrupt
} Internal produced by micro controller

External interrupts can be Level triggered or edge triggered. When interrupts occur then micro controller executes ISR. and memory location corresponding to interrupts enables it. The interrupt corresponds to memory location explain about the interrupt vector table. Shown below.

Interrupt Number	Interrupt Description	Address
0	External INT0	0003H
1	Timer 0 overflow	000BH
2	External INT1	0013H
3	Timer 1 overflow	001BH
4	Serial Port	0023H

* After "RESET" all interrupts get disabled. and all interrupts is enabled by software. From all five interrupts if anyone or all interrupts are activated it will set the corresponding interrupt flags.
* All interrupts can be set or cleared by Some special Fonction interrupt registers known as IE interrupt enabled and which can executed by interrupt priority register

IE (Interrupt enable Register):

It bit addressible register in which EA value must be set to one. and individual bits in particular interrupts from TCON for internal and external interrupts.

EA	-	-	ES	TF_1	EX_1	TF_0	EX_0

EA (IE_7) → EA = 0 all interrupts are disabled
EA = 1 all " are active

IE_6, IE_5 } not implemented. reserved for future

ES	IE_4	Enable Serial port interrupt
TF_1	IE_3	Enable / Disable Timer 1 O.F interrupt
EX_1	IE_2	Enable/disable External interrupt 1 (INT 1)
TF_0	IE_1	Enable / disable Timer 0 O.F interrupt
EX0	IE_0	Enable / disable Esternal interrupt 0 (INT0.)

Interrupt priority register :- (IP)

IP is possible to change the priority levels of an interrupt by clearing / setting individual bits priority. always low priority can interrupt high priority, but it prohibits interruption by another low prority interrupt, if in program not mentioned it will follow order INT0, TF0 INT1, TF1, SI.

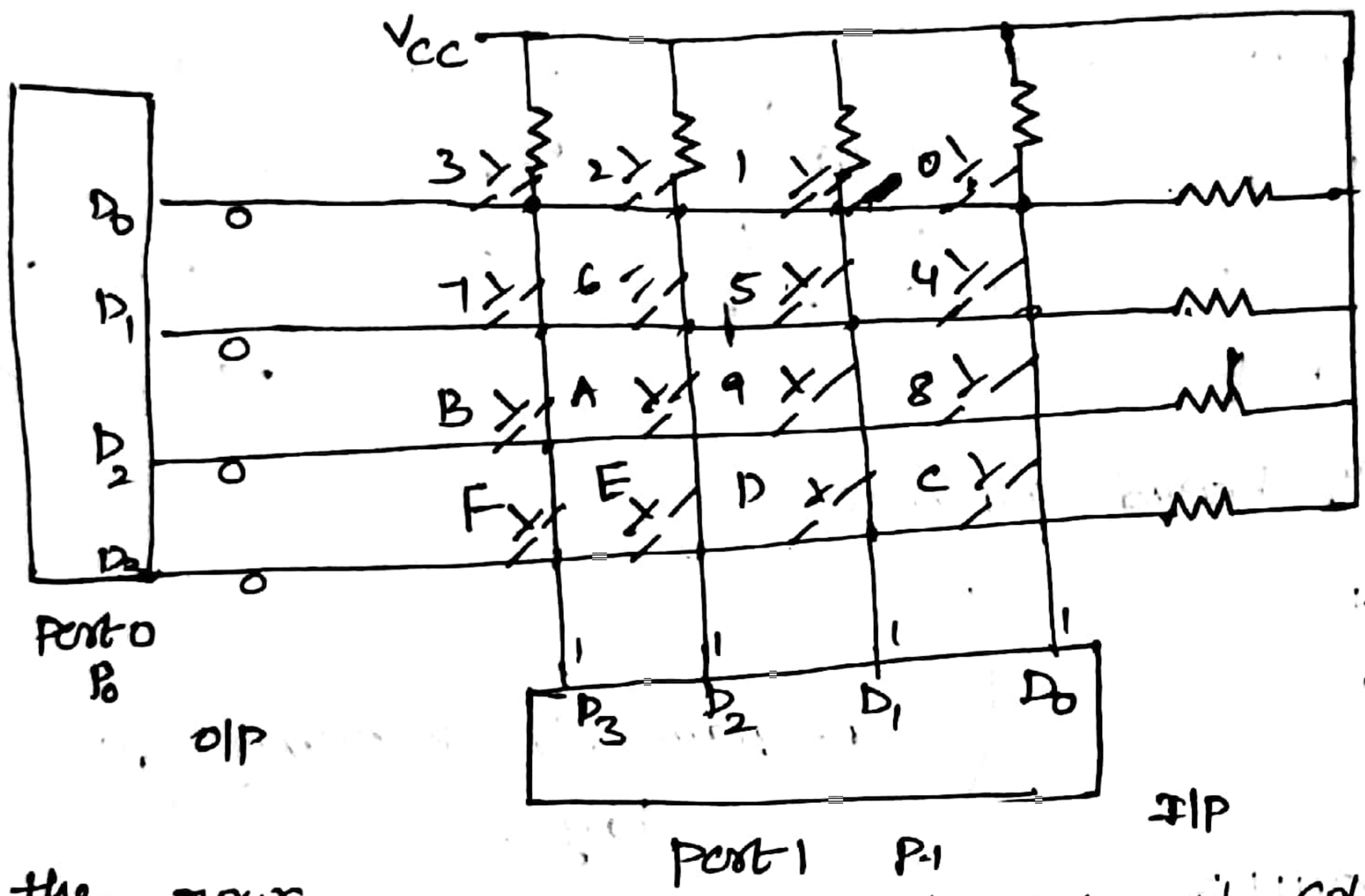
Interrupt programming in 8051 for external hardware.

```
ORG 0          → Start occurs here
Ljmp start     → Jump to interrupt vector
ORG 0003       → Interrupt vector of EX0
CPL. P1.2      → Compliment of P1.2
RETI           → Return to main program
ORG 0040H      → main program is here
START: MOV IE, #81h.  interrupt is enabled
HERE; SJMP HER, end   wait end
```

Simple Program for KeyBoard Interface :

Keyboard is arrangement of all buttons arranged in particular pattern. The pattern either in Rectangular or Square. The buttons has contain either Symbols, numbers or alphabets. If keyboard is working with only numbers then it is called " number keyboard" or "numeric keypad"

Consider 4×4 matrix keyboard.. i.e it has 4 columns and 4 rows. Internally keyboard is connected to the 8051 uc.



All the rows are connected to o/p port and columns are connected to i/p port of micro controller initially. All rows are initialized with '0's and columns are initialized with 1's of keyboard. Here the column's are connected to the i/p port. all four ports act as input and output ports. Hence column values as its input. hence it can read initially ....

* If controller reads all the one's on the column's of the keypad Then controller understand that no key is pressed.
* Now Assume that among all if one key is pressed then that particular column value is consider zero among four column's.
* Example :- If key 2 is pressed. Then switch 2 is closed. Then 2 is connected to 1st row and 2nd column.
* Now P_2 is column 2 and becomes zero. hence it understand that key is pressed.

* D_2 becomes zero means anyone of 4 keys 2, 6, A & E is pressed. But don't know exactly which is being pressed.
* Whenever D_2 is pressed than 1011 is activated as input
* To know exactly which key is pressed for that read the output port data.
* Here w.r.t column $D_2 \rightarrow 0$ D_0 of row is also zero and remaining becomes 1. This is Indication of key 2 is pressed.

ROWS			
D_0	P_1	D_2	D_3
0	1	1	1

D_3	D_2	D_1	D_0
1	0	1	1

1st Row & 2nd column → 1'x2 key is pressed → i.e 2nd key

Exc:-2

ROW			
D_0	D_1	D_2	D_3
1	0	1	1
1	1	0	0

columns				Keys	Rows and co
D_3	D_2	D_1	D_0		
1	1	0	0	5"	2nd row x 3rd colm
0	1	1	1	B.	3rd row x 1st colm

Program for key board interface

```
    MOV   P2, OFFH
K1: MOV   P1, #0H
    MOV   A, P2
    ANL   A, #00001111B
    CJNE  A, #00001111B, K1
K2: ACALL Delay.
    MOV   A, P2
    ANL   A, #00001111B
    CJNE  A, #00001111B, over
    SJMP  K2
over: ACALL Delay.
    MOV   A, P2
    ANL   A, #00001111B
    CJNE  A, #00001111B, over1
    SJMP  K2
```

(contd.)

program for checking all keys.

A, 00001111 → are equal no key is pressed otherwise it is pressed it is repeated until both are equal.

checking that any key pressed or not.

A = P_2 = 1110 1111
And A and 0000 1111 get
0000 1111 → compare with
0000 1111 → both equal hence no key pressed. hence go to K_2

Let us take press key is 6

Row 10 11
colm 11 01

Now P_2 → 0111 1101 xor mp
A = P_2 = 01111 1101

Now ANL for A & Acom → 0000 1101 (not equal)
0000 1111
hence key pressed and go for over

LCD interfacing with 8051 Microcontroller :

Display units are the most important output devices in electronic and embeded project. There are various display units 1X16, 2X16, 3X16 units. means

1X16 → Row has 16 characteristics.
2X16 → 2 Rows " " "
4X16 → 4 Rows has 16 characteristics

Each character takes 5X7 matrix space on LCD. LCD has 16 pins classified as 5 groups such as power pins, contrast pins, control pins, Data pins and backlight pins.

Category	PIN NO	Pin Name	Function
Power pins	1	Vss	Ground pin
	2	V_{DD}/V_{CC}	voltage pin +5V.
Contrast pin	3	VEE	connected to V_{CC} and V_{SS} through Resistor.
Control pins	4	RS	Register select pin RS = 0 → Command mode RS = 1 → Data mode.
	5	R/W	RW = 0 → Write mode. RW = 1 → Read mode.
	6	E	Enable pin High to Low pulse need to enable LCD.
Data pins	7-14	D_0-D_7	Stores the data displayed on LCD
Backlight pins	15	LED+ or A	Power the backlight +5V.
	16	LED or K	Backlight ground.

RS : set to 1 if we need to send some data on LCD.
set to 0 for sending command Instructions like clear screen

RW :- set to '0' going to write some dat on LCD.

And set to 1 if we want to Read data From LCD but we Don't require this

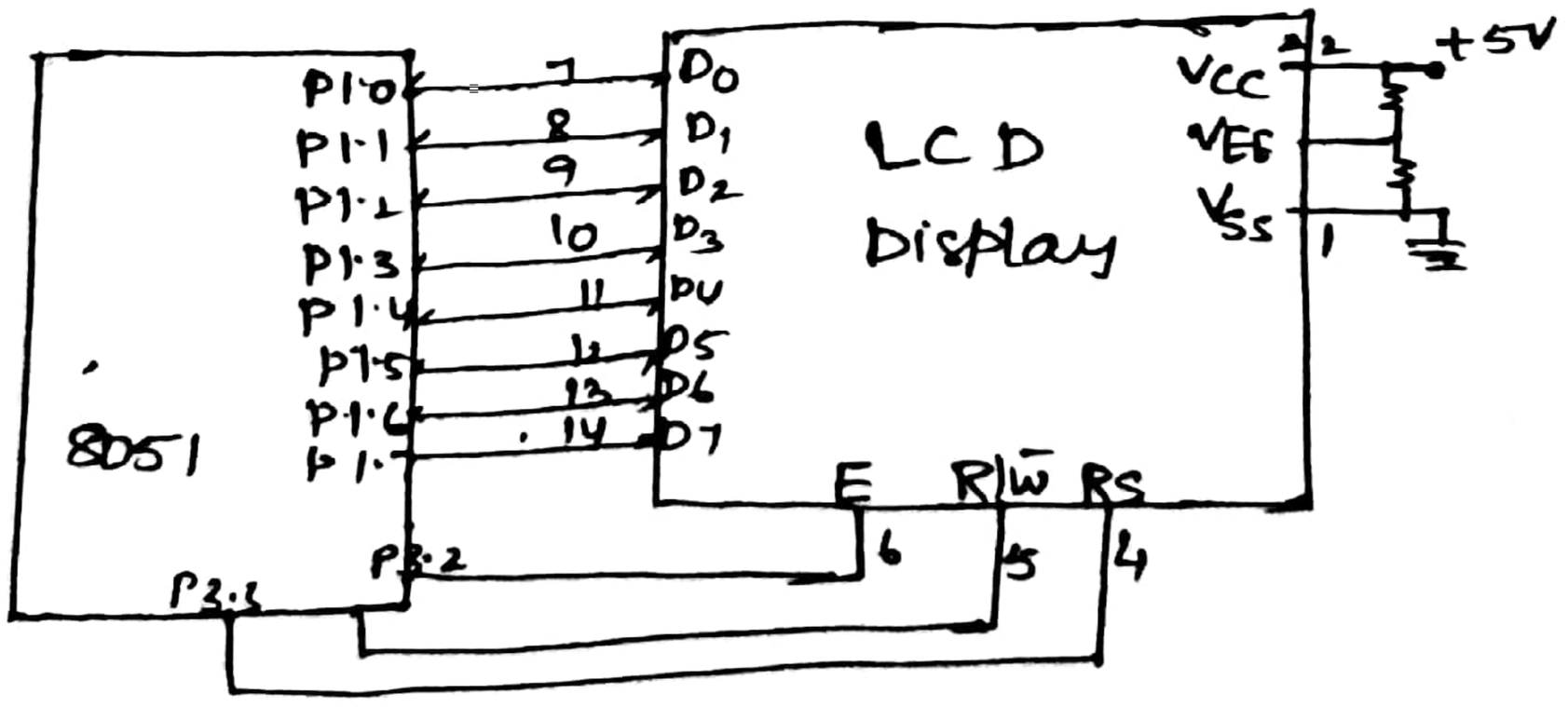
E :- Used to enable module when a High to low pulse is given to it with 450 ns.

Some of commands given in LCD are as ●

Hex code		Command to LCD Instruction Register
OF	→	LCD ON, cursor ON
01	→	clear display screen.
02	→	Return home.
04	→	Decreament cursor
06	→	Increament cursor
05	→	shift display right
07	→	Shift display Left
0E	→	Display ON, cursor blinking
08	→	Display OFF, cursor OFF
3c	→	Activate Second line
38	→	2 lines, 5X1 matrix
83	→	cursor line 1, position 3.
c0	→	Force cursor to beggining of 2nd line
c1	→	Jump to position 1, 2nd line
0c	→	display ON, cursor OFF
c2	→	Jump to position 2, 2nd line

command to obey.

Data to display



V_{EE} → control brightness of display

ALP for 2X16 LCD display to interface with 805,

it has mainly 3 objectives Command used to send information Data used to store data. Delay used to give time

```
ORG   0000 H
MOV   A, #38H   (2 line 5x7 matrix)
ACALL Command
ACall Delay.

MOV   A, #0EH   (Display ON Cursor blink
ACALL Command
ACAll Delay

MOV   A, #01H (clear display
ACALL Command
ACALL Delay.

MOV   A, #80H (beggin of 1st line)
ACAll Command
ACALL Delay.

MOV   A, #'K'
ACAll Data
Acall Delay.

MOV   A, #'S'
ACAIL Data
ACALL Delay

MOV   A, #'I'
ACALL Data
ACALL Delay.

MOV   A, #'T'
ACALL Data
ACALL Delay

STAY : SJMP  STAY
```

```
Command : CLR   P3.3
          MOV   P0, A  (Data to cursor)
          SETB  P3.2
          ACALL Delay
          CLR   P3.2
          RET

Data :    SETB  P3.3
          MOV   P0, A
          SETB  P3.2
          ACALL Delay
          CLR   P3.2
          RET

Delay :   MOV   R3, #40
BACK :    MOV   R4, #250
HERE :    DJNZ  R4, HERE
          DJNZ  R3, BACK.
          RET
          END.
```

Interfacing Servo motor to 8051 Microcontroller :-

Servo motors are self-contained mechanical devices that are used to control the machines. which is specially. design for specific angle position to control machines i.e 0-180° and 0-90° Based PWM Technique. It can be used in aeroplanes and robots, Sugar filling station, Labelling applications.

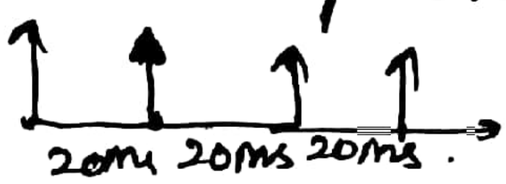
→ Basically Dc Motor is rotating in either Forward or reverse direction based on Supply.

→ But Stepper motor control the operation in angular positions from 0°-180° or 0°-90°
0°-180° design for clocks, watches etc. but for 0-90° or 0--90, used in applications of Robotics, aeroplanes etc.

→ It mainly consist of 3 wires → 2 are used for Supply one used as control Signal

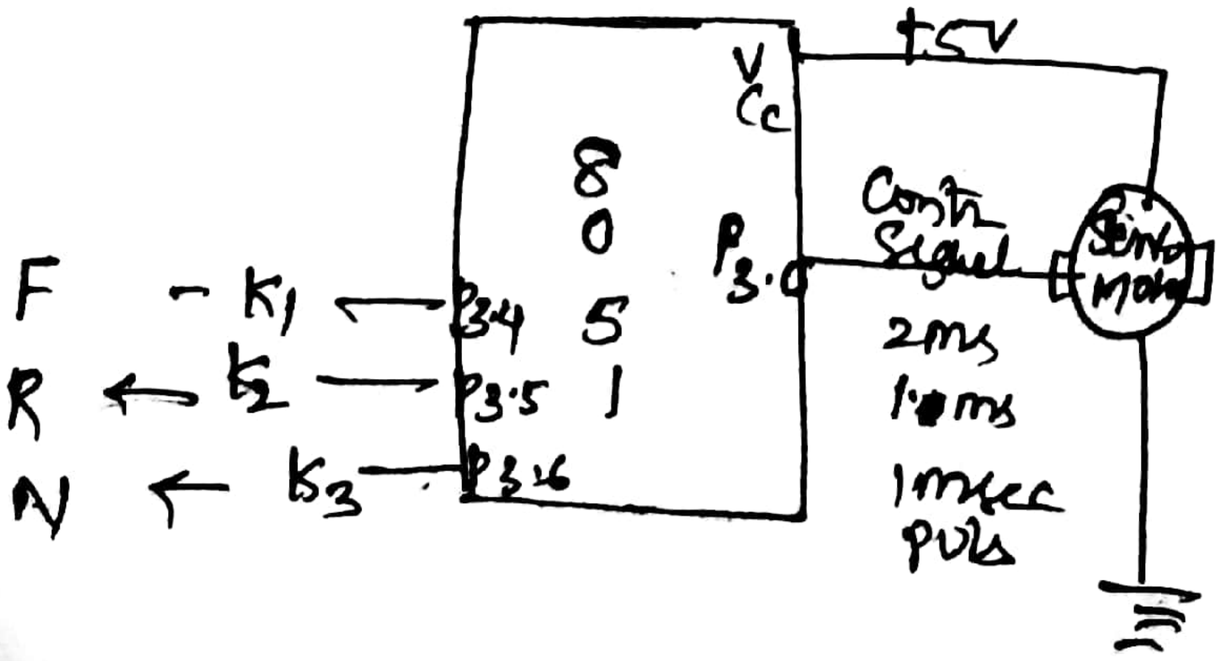
→ Control Signal in the pulse width (either 2ms, 1.5ms or 1ms) is connected pin of any port of Micro controller.

→ The pulse is applied for every ever 20ms otherwise it can not getting pulse.



→ different companies provides Various timings for forward pulse, reverse pulse or neutral pulse.
for > 1.5ms forward pulse (forward 90° → 2ms)
< 1.5ms reverse pulse (Reverse 90° → 1ms)
= 1.5ms neutral pulse.

If you want to applied twice forward then you can apply two signals with 20ms, & 20ms with time of 2ms and Vice versa



ALP for interfacing servo motor into 8051 MC

$P_{3.7}$	$P_{3.6}$	$P_{3.5}$	$P_{3.4}$	$P_{3.3}$	$P_{3.2}$	$P_{3.1}$	$P_{3.0}$	
0	0	0	1	0	0	0	0	10H k_1
0	0	1	0	0	0	0	0	20H k_2
0	1	0	0	0	0	0	0	30H k_3

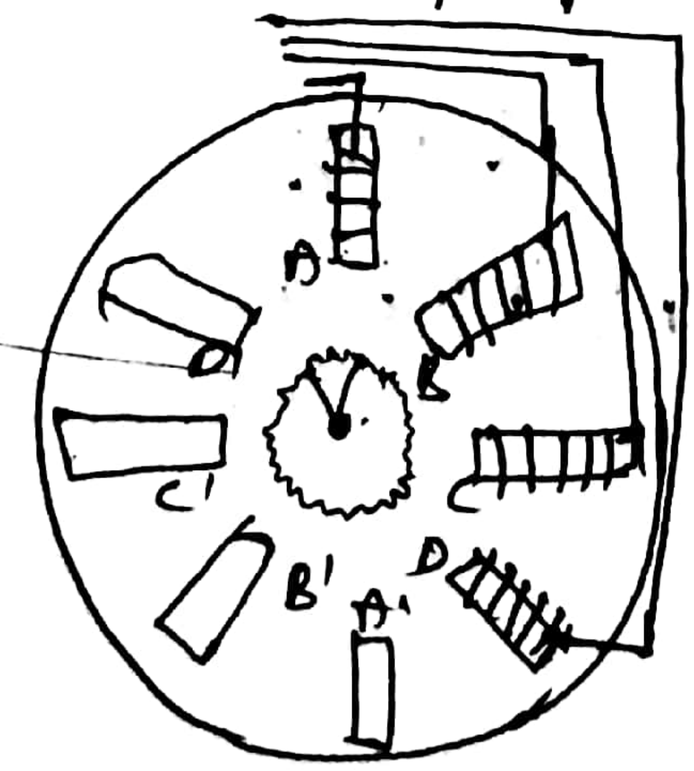
```
AGAIN: MOV A, P3
       AND A, #F0
       CJNE A, #10H, Next
Next: CJNE A, #20H, Next1
Next1: CJNE A, #40h.
       SJMP AGAIN

       SETB P3.0
       ACALL DELAY
       SJMP AGAIN
```

Interfacing stepper Motor to 8051 Micro Controller:

Stepper Motor is a special design motor. The Basic difference between stepper Motor and Normal Motor.

* Dc Motor has no control over on how much angle which can rotate. But there is control over on motor how much angle with which rotate.
* stepper Motor has no of pole magnets in stator and rotor



angle of rotation controlled

4 → AA', BB', CC', DD' poles.

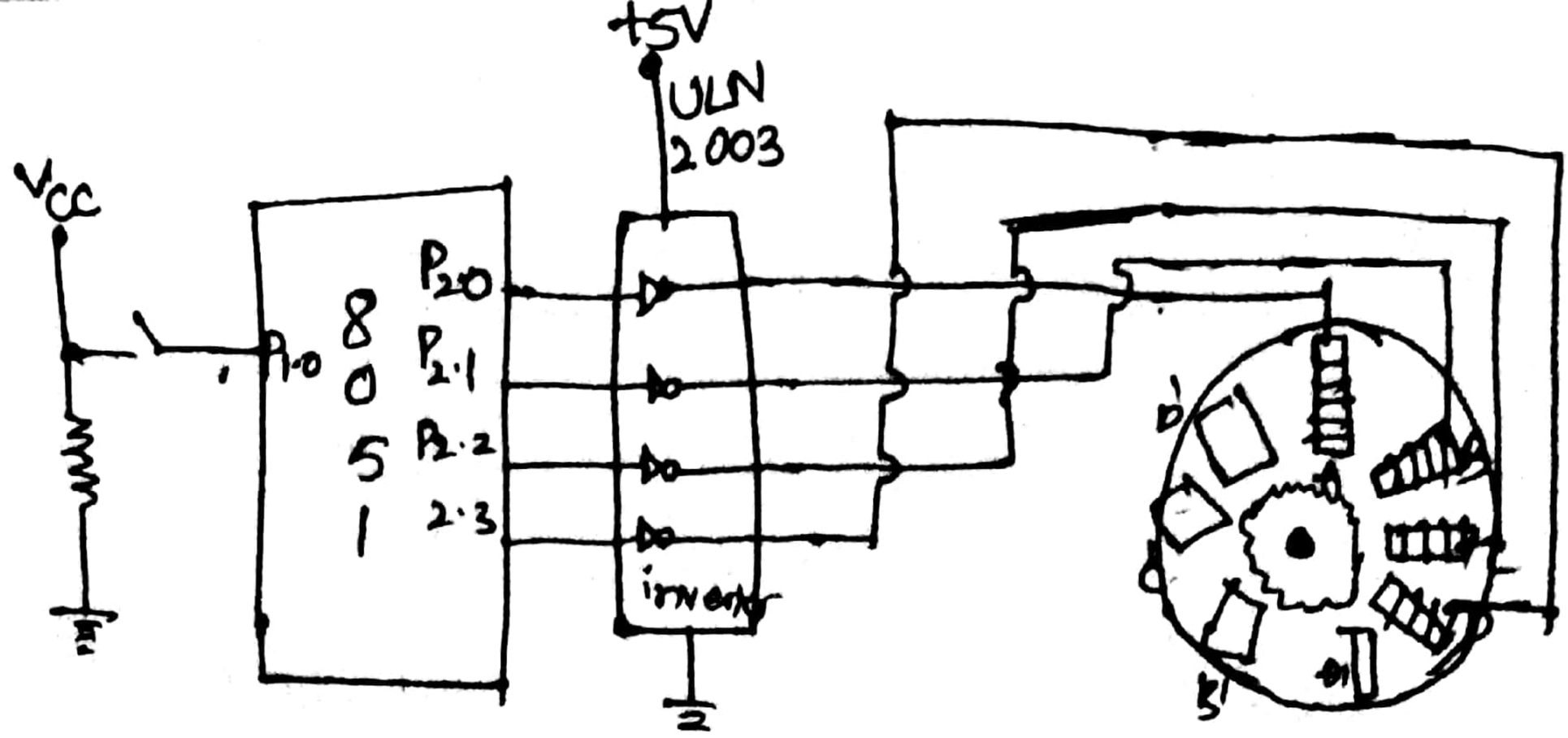
Rotor has no of skews or booths

Angle of rotation

$$= \frac{360}{\text{no of poles} \times \text{rotor teeth}}$$

$$= \frac{360}{4 \times 50} = 18°$$

mean it will rotate with 18° angle. If tooth decrease angle increase.



Interfacing Stepper Motor to 8051 Micro Controller

Stepper motor is interfacing with 8051 controller through IC called "ULN 2003"

Rotation of Stepper Motor.

	I/P				Inverter A B C D	Rotation
Step 1	0	1	1	1 (7)	1 0 0 0	From A
2	1	0	1	1 (B)	0 1 0 0	From A-B
3	1	1	0	1 (D)	0 0 1 0	B → C
4	1	1	1	0 (E)	0 0 0 1	C → D

* clockwise direction

A → B → C → D ← 7, B, D, E → I/P Value

for Anticlock wise direction

Step	I/P	After Inverter	Rotation of magnets
1	1 0 0 0	0 1 1 1	From A
2	0 0 0 1	1 1 1 0	A → D'
3	0 0 1 0	1 1 0 1	D' → C'
4	0 1 0 0	1 0 1 1	C' → B'
5	1 0 0 0	0 1 1 1	B' → A'

The values 7, B, D, E are used in program used to energize magnets to rotate.

Here P_1 and P_2 are used to connect the ports to interface and rotate stepper with program. by controller.

Port 2 is connected to 4 magnet pole stepper Motor through.

Port 1 is used as switch — 1 to rotate in clock wise
— 0 to rotate in Anti clock wise

Driving Pins of port 2 are for continue rotation.

$P_{2.0}$	2.1	2.2	2.3	2.4	2.5	2.6	2.7	
0	1	1	1	0	1	1	1	77
1	0	1	1	1	0	1	1	BB
1	1	0	1	1	1	0	1	DD
1	1	1	0	1	1	1	0	EE.

ALP for stepper motor interfacig to 8051 MC.

```
        ORG  0000H
        MOV  P1,#0FFH     Used as switch port1.0 i/p and port2 o/p
        MOV  A , #77H  ;  move 77H value in A
        MOV  P2 , A    ;  move A value in port 2 to rotate motor
TURN:   JNB  P1.0, ACW  i/p ; 1→ if bit = 1 condition fail go to part 1
        RR   A                0→ Jump if not bit condition satisfy go to
        ACALL DELAY             part 1
        MOV  P2, A
        SJMP TURN

ACW :   RL   A
        ACALL DELAY
        MOV  P2, A
        SJMP TURN

Delay:  MOV  R2, #255  → ms
Back :  MOV  R3, #255  → ms
Stay.   DJNZ R3, stay
        DJNZ R2, Back
        RET
        END
```

Part 1

part 2

only one coil is energized at a time, all four coil are energized one after another. It produce less torque. But power consumption is less