8051 Microcontroller

- Intel introduced 8051, referred as MCS- 51, in 1981.
- The 8051 is an 8-bit processor
- The CPU can work on only 8 bits of data at a time
- The 8051 became widely popular after allowing other manufactures to make and market any flavor of the 8051.

Features of 8051

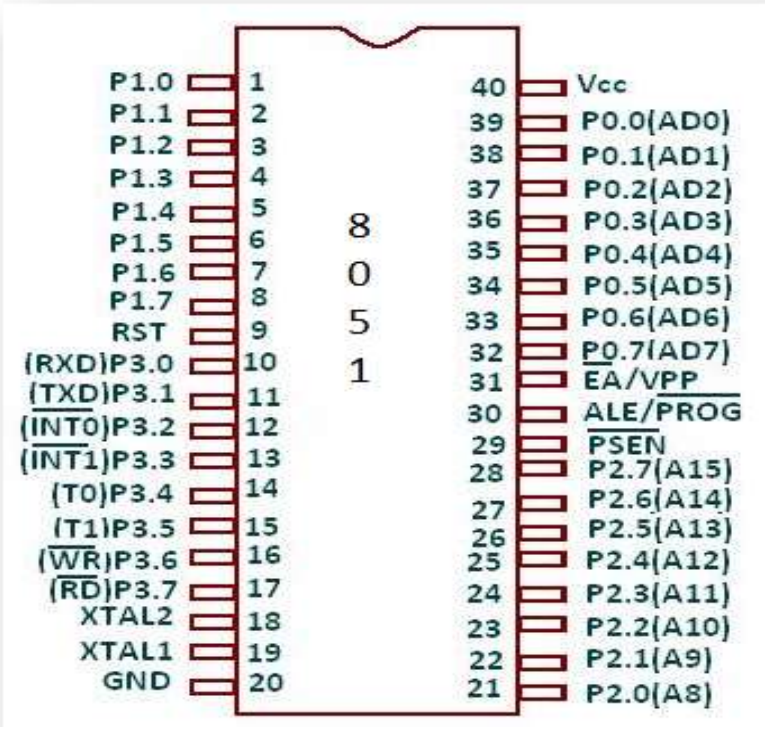
- 8 bit Processor
- 4KB Internal ROM
- 128 Bytes Internal RAM
- Four 8 BIT I/O PORTS (32 I/O LINES)
- Two 16 Bit Timers/Counters
- On Chip Full Duplex UART for Serial Communication
- 5 Vector Interrupts (2 External, 3 Internal - Timer0,Timer1,Serial)
- On Chip Clock Oscillator
- 16 bit Address bus
    - 64k External Code Memory
    - 64k External Data Memory
- 16-bit program counter to access external Code Memory and
- 16 bit Data Pointer to access external Data Memory
- 128 user defined flags
- 32 General Purpose Registers each of 8 bits

8051 Family

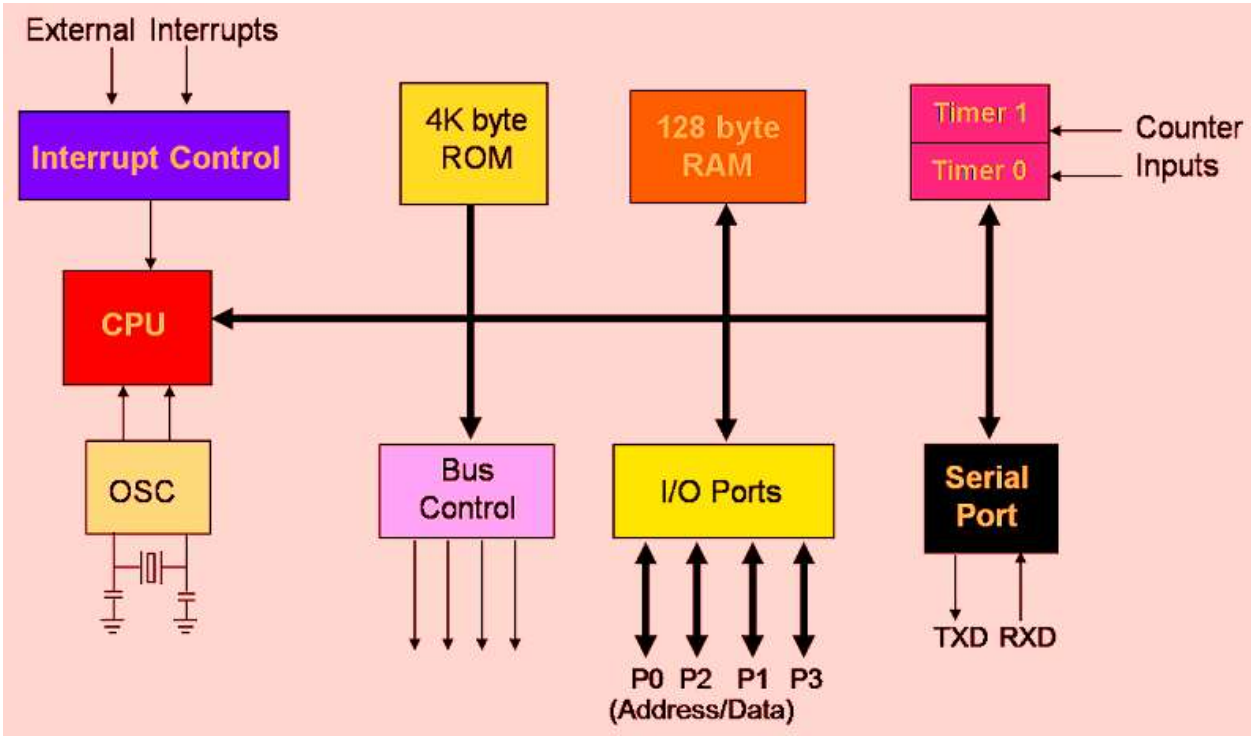
- The 8051 is a subset of the 8052
- The 8031 is a ROM-less 8051
    - Add external ROM to it
    - You lose two ports, and leave only 2 ports for I/O operations

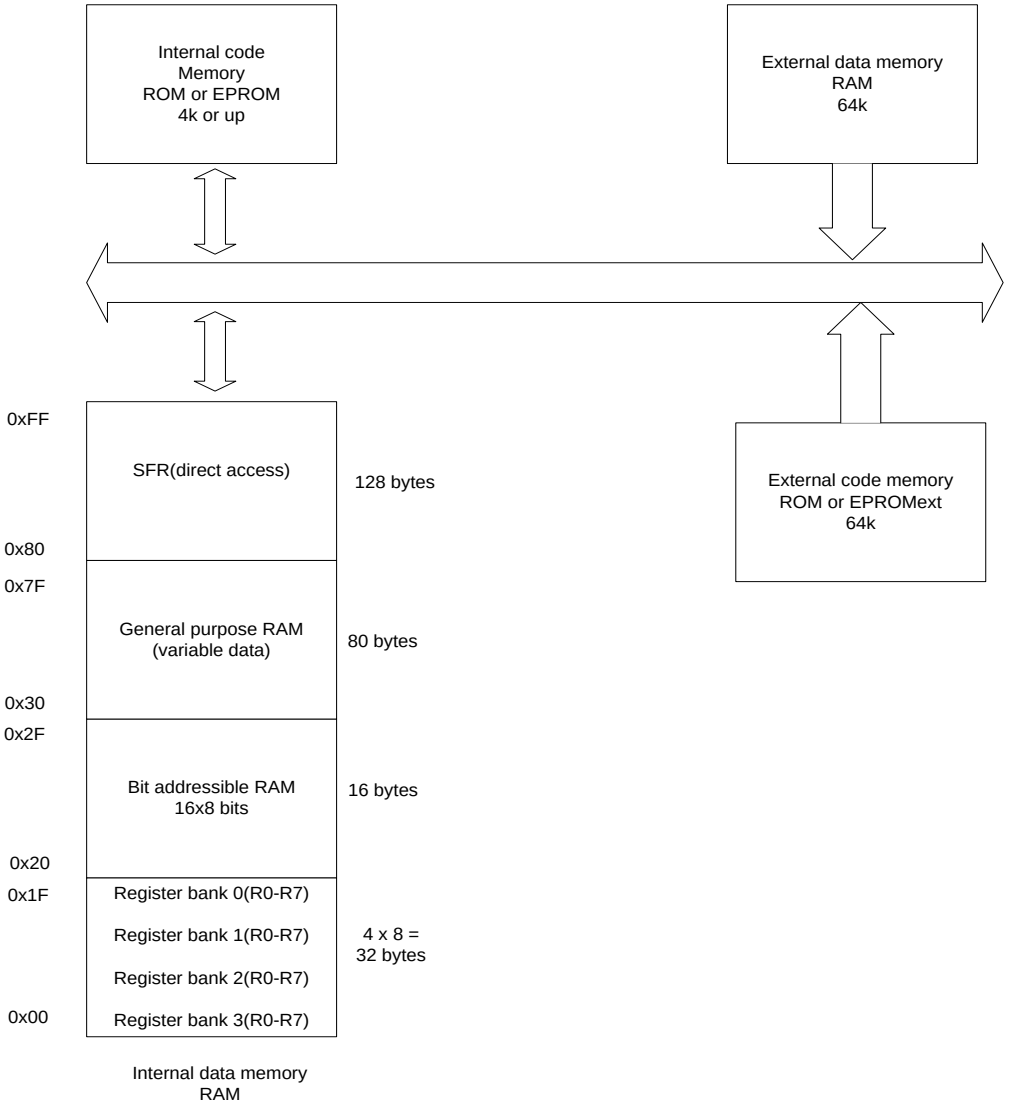
Feature	8051	8052	8031
ROM (on-chip program space in bytes)	4K	8K	0K
RAM (bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

Pin Diagram



Block Diagram of 8051



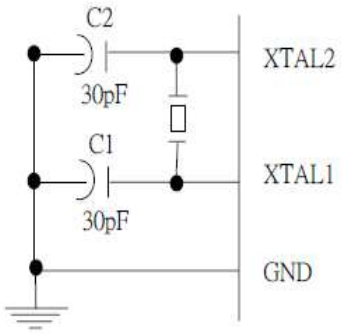


Pin Description of the 8051

- 8051 family members (e.g., 8751, 89C51, 89C52, DS89C4x0)
  - Have 40 pins dedicated for various functions such as I/O, RD, WR, address, data, and interrupts.
  - Come in different packages, such as
    - DIP(dual in-line package),
    - QFP(quad flat package), and
    - LLC(leadless chip carrier)
- Some companies provide a 20-pin version of the 8051 with a reduced number of I/O ports for less demanding applications

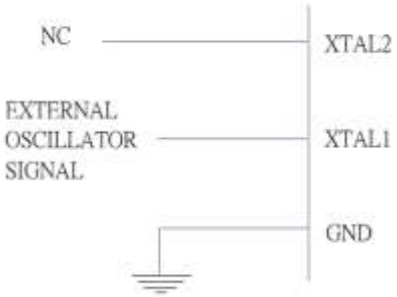
XTAL1 and XTAL2

- The 8051 has an on-chip oscillator but requires an external crystal to run it
  - A quartz crystal oscillator is connected to inputs XTAL1 (pin19) and XTAL2 (pin18)
  - The quartz crystal oscillator also needs two capacitors of 30 pF value
  - The original 8051 operates at 12 MHZ



XTAL1 and XTAL2 …..

- If you use a frequency source other than a crystal oscillator, such as a TTL oscillator:
  - It will be connected to XTAL1
  - XTAL2 is left unconnected



RST

- RESET pin is an input and is active high (normally low)
- Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities
- This is often referred to as a power-on reset
- Activating a power-on reset will cause all values in the registers to be lost

RESET value of some 8051 registers

we must place the first line of source code in ROM location 0

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

EA

- EA', **"external access''**, is an input pin and must be connected to Vcc or GND
- The 8051 family members all come with on-chip ROM to store programs and also have an external code and data memory.
- Normally EA pin is connected to Vcc (Internal Access).
- EA pin must be connected to GND to indicate that the code or data is stored externally.

PSEN' and ALE

- PSEN, **"program store enable''**, is an output pin
- This pin is connected to the OE pin of the external memory.
- For External Code Memory, PSEN' = 0
- For External Data Memory, PSEN' = 1
- ALE pin is used for demultiplexing the address and data.

I/O Port Pins

- The four 8-bit I/O ports **P0, P1, P2 and P3** each uses 8 pins.
- All the ports upon RESET are configured as output, ready to be used as input ports by the external device.

Port 0

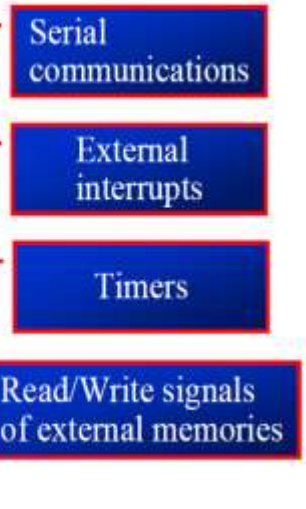
- Port 0 is **also** designated as **AD0-AD7**.
- When connecting an 8051 to an external memory, port 0 provides both address and data.
- The 8051 multiplexes address and data through port 0 to save pins.
- **ALE** indicates if P0 has address or data.
  - *When ALE=0, it provides data D0-D7*
  - *When ALE=1, it has address A0-A7*

Port 1 and Port 2

- In 8051-based systems **with no external memory connection**:
  - Both P1 and P2 are used as simple I/O.
- In 8051-based systems **with external memory connections**:
  - Port 2 must be used along with P0 to provide the 16-bit address for the external memory.
  - P0 provides the lower 8 bits via A0 – A7.
  - P2 is used for the upper 8 bits of the 16-bit address, designated as A8 – A15, and it cannot be used for I/O.

Port 3

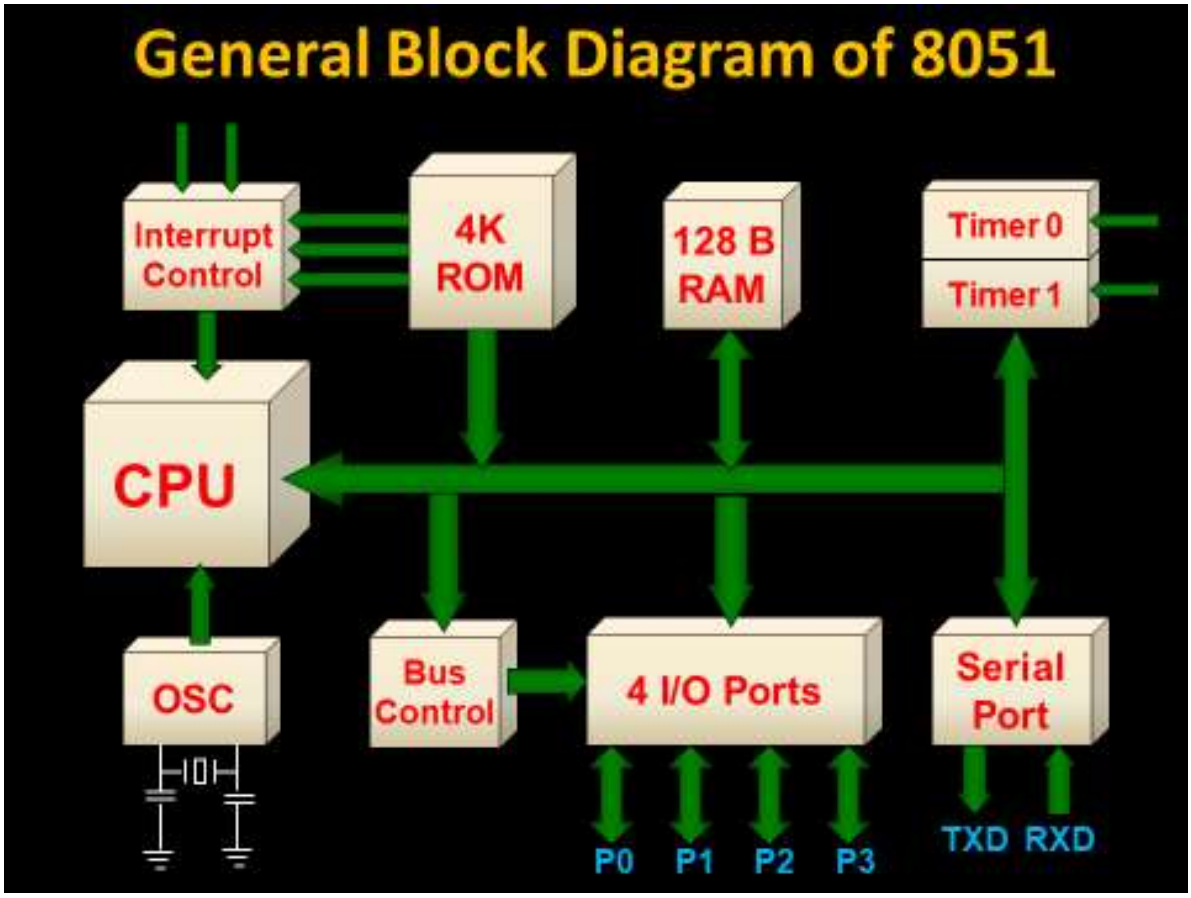
P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

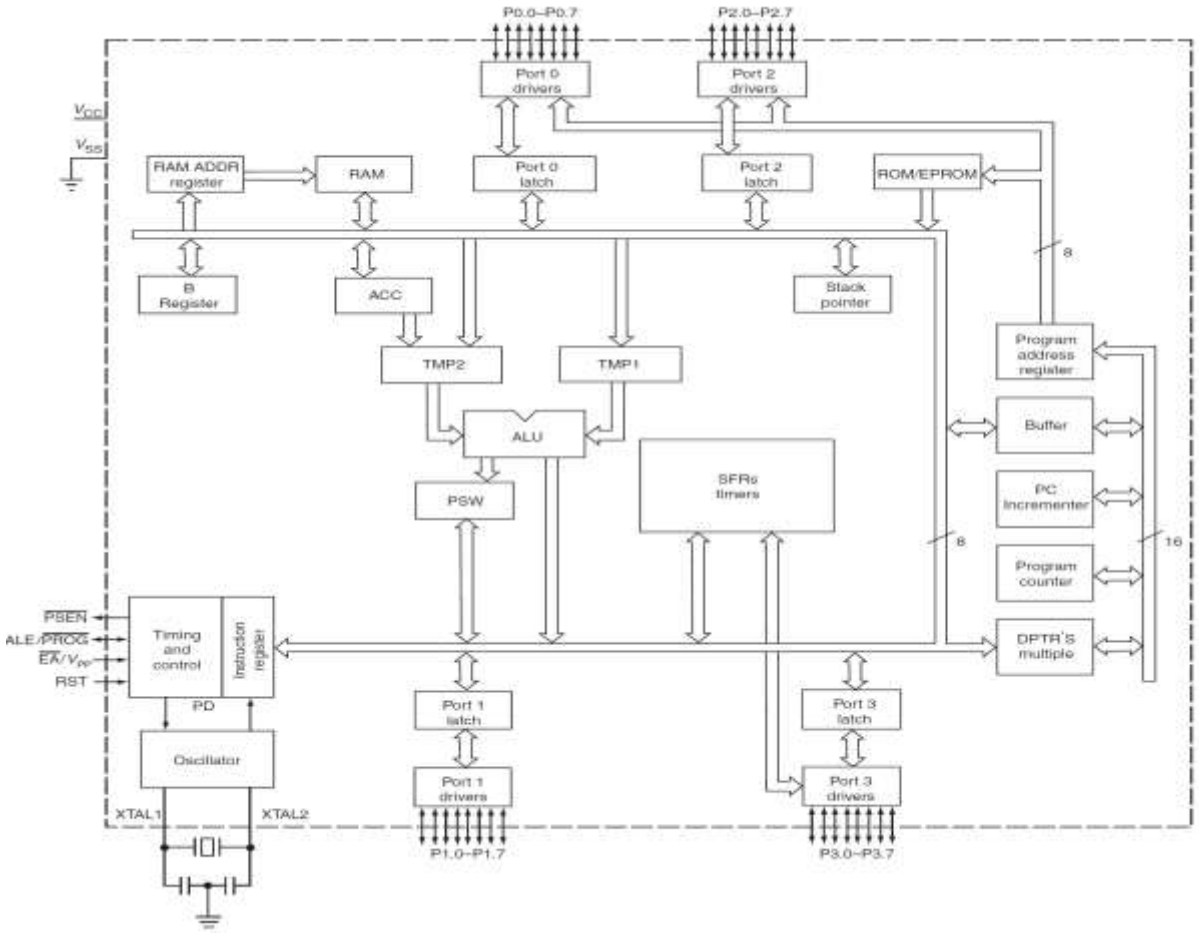


Pin Description Summary

PIN	TYPE	NAME AND FUNCTION
Vss	I	**Ground:** 0 V reference.
Vcc	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0 - P0.7	I/O	**Port 0:** Port 0 is an open-drain, bi-directional I/O port. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory.
P1.0 - P1.7	I/O	**Port 1:** Port I is an 8-bit bi-directional I/O port.
P2.0 - P2.7	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O. Port 2 emits the high order address byte during fetches from external program memory and during accesses to external data memory that use 16 bit addresses.
P3.0 - P3.7	I/O	**Port 3:** Port 3 is an 8 bit bidirectional I/O port. Port 3 also serves special features as explained.

PIN	TYPE	NAME AND FUNCTION
RST	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device.
ALE	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory.
PSEN*	O	**Program Store Enable:** The read strobe to external program memory. When executing code from the external program memory, PSEN* is activated twice each machine cycle, except that two PSEN* activations are skipped during each access to external data memory.
EA*/VPP	I	**External Access Enable/Programming Supply Voltage:** EA* must be externally held low to enable the device to fetch code from external program memory locations. If EA* Is held high, the device executes from internal program memory. This pin also receives the programming supply voltage Vpp during Flash programming. (applies for 89c5x MCU's)





8051 addressing Modes

1) The CPU can access data in various ways, which are called addressing modes

a) **Immediate** **b) Register** **c) Direct** d) **Register indirect** e) **External Direct**

Immediate Addressing Mode

- The source operand is a **constant.**
- The immediate data must be preceded by the pound sign, **"#"**
- Can load information into **any registers**, including 16-bit DPTR register
  - DPTR can also be accessed as two 8-bit registers, the high byte DPH and low byte DPL

```
MOV A,#25H      ;load 25H into A
MOV R4,#62      ;load 62 into R4
MOV B,#40H      ;load 40H into B
MOV DPTR,#4521H ;DPTR=4512H
MOV DPL,#21H    ;This is the same
MOV DPH,#45H    ;as above

;illegal!! Value > 65535 (FFFFH)
MOV DPTR,#68975
```

Register Addressing Mode

- Use registers to hold the data to be manipulated.

```
MOV A,R0    ;copy contents of R0 into A
MOV R2,A    ;copy contents of A into R2
ADD A,R5    ;add contents of R5 to A
ADD A,R7    ;add contents of R7 to A
MOV R6,A    ;save accumulator in R6
```

- The source and destination registers must match in size.
  MOV DPTR,A will give an error

```
MOV DPTR,#25F5H
MOV R7,DPL
MOV R6,DPH
```

- The movement of data between Rn registers is not allowed
  MOV R4,R7 is invalid

Direct Addressing Mode

- It is most often used the direct addressing mode to access RAM locations 30 – 7FH.
- The entire 128 bytes of RAM can be accessed.
- Contrast this with immediate addressing mode, there is no "#" sign in the operand.

```
MOV R0,40H  ;save content of 40H in R0
MOV 56H,A   ;save content of A in 56H
```

SFR Registers & their Addresses

```
MOV   0E0H, #55H   ;is the same as
MOV   A, #55H      ;which means load 55H into A (A=55H)
MOV   0F0H, #25H   ;is the same as
MOV   B, #25H      ;which means load 25H into B (B=25H)
MOV   0E0H, R2     ;is the same as
MOV   A, R2        ;which means copy R2 into A
MOV   0F0H, R0     ;is the same as
MOV   B, R0        ;which means copy R0 into B
```

Example

Example

Example 5-1

Write code to send 55H to ports P1 and P2, using (a) their names (b) their addresses.

Solution:

```
(a)   MOV  A,#55H      ;A=55H
      MOV  P1,A        ;P1=55H
      MOV  P2,A        ;P2=55H

(b)   From Table 5-1, P1 address = 80H; P2 address = A0H
      MOV  A,#55H      ;A=55H
      MOV  80H,A       ;P1=55H
      MOV  0A0H,A      ;P2=55H
```


Stack and Direct Addressing Mode

Show the code to push R5 and A onto the stack and then pop them back them into R2 and B, where B = A and R2 = R5

Solution:

```
PUSH 05        ;push R5 onto stack
PUSH 0E0H      ;push register A onto stack
POP  0F0H      ;pop top of stack into B
               ;now register B = register A
POP  02        ;pop top of stack into R2
               ;now R2=R6
```

Register Indirect Addressing Mode

- Write a program to copy the value 55H into RAM memory locations 40H to 41H using **(a) direct addressing mode**, **(b) register indirect addressing mode without a loop**, and **(c) with a loop.**

Solution:

```
(a)
    MOV A,#55H   ;load A with value 55H
    MOV 40H,A    ;copy A to RAM location 40H
    MOV 41H.A    ;copy A to RAM location 41H
(b)
    MOV A,#55H   ;load A with value 55H
    MOV R0,#40H  ;load the pointer. R0=40H
    MOV @R0,A    ;copy A to RAM R0 points to
    INC R0       ;increment pointer. Now R0=41h
    MOV @R0,A    ;copy A to RAM R0 points to
(c)
        MOV A,#55H     ;A=55H
        MOV R0,#40H    ;load pointer.R0=40H,
        MOV R2,#02     ;load counter, R2=3
AGAIN:  MOV @R0,A      ;copy 55 to RAM R0 points to
        INC R0         ;increment R0 pointer
        DJNZ R2,AGAIN  ;loop until counter = zero
```


MOV Instruction

- **MOV destination, source** ; copy source to destination.

- MOV A,#55H ;load value 55H into reg. A
  MOV R0,A ;copy contents of A into R0
  ;(now A=R0=55H)
  MOV R1,A ;copy contents of A into R1
  ;(now A=R0=R1=55H)
  MOV R2,A ;copy contents of A into R2
  ;(now A=R0=R1=R2=55H)
  MOV R3,#95H ;load value 95H into R3
  ;(now R3=95H)
  MOV A,R3 ;copy contents of R3 into A
  ;now A=R3=95H

ADD Instruction

- **ADD A, source** ;ADD the source operand to the accumulator

- MOV A, #25H ;load 25H into A
  MOV R2,#34H ;load 34H into R2
  ADD A,R2 ;add R2 to accumulator
  ;(A = A + R2)

Structure of Assembly Language

```
        ORG  0H          ;start (origin) at location 0
        MOV  R5,#25H     ;load 25H into R5
        MOV  R7,#34H     ;load 34H into R7
        MOV  A,#0        ;load 0 into A
        ADD  A,R5        ;add contents of R5 to A
                         ;now A = A + R5
        ADD  A,R7        ;add contents of R7 to A
                         ;now A = A + R7
        ADD  A,#12H      ;add to A value 12H
                         ;now A = A + 12H
HERE:   SJMP HERE        ;stay in this loop
        END              ;end of asm source file
```

Conditional Jump Example

Example 3-5

Find the sum of the values 79H, F5H, and E2H. Put the sum in registers R0 (low byte) and R5 (high byte).

Solution:

```
        MOV  A,#0       ;clear A(A=0)
        MOV  R5,A       ;clear R5
        ADD  A,#79H     ;A=0+79H=79H
        JNC  N_1        ;if no carry, add next number
        INC  R5         ;if CY=1, increment R5
N_1:    ADD  A,#0F5H    ;A=79+F5=6E and CY=1
        JNC  N_2        ;jump if CY=0
        INC  R5         ;If CY=1 then increment R5(R5=1)
N_2:    ADD  A,#0E2H    ;A=6E+E2=50 and CY=1
        JNC  OVER       ;jump if CY=0
        INC  R5         ;if CY=1, increment 5
OVER:   MOV  R0,A       ;Now R0=50H, and R5=02
```