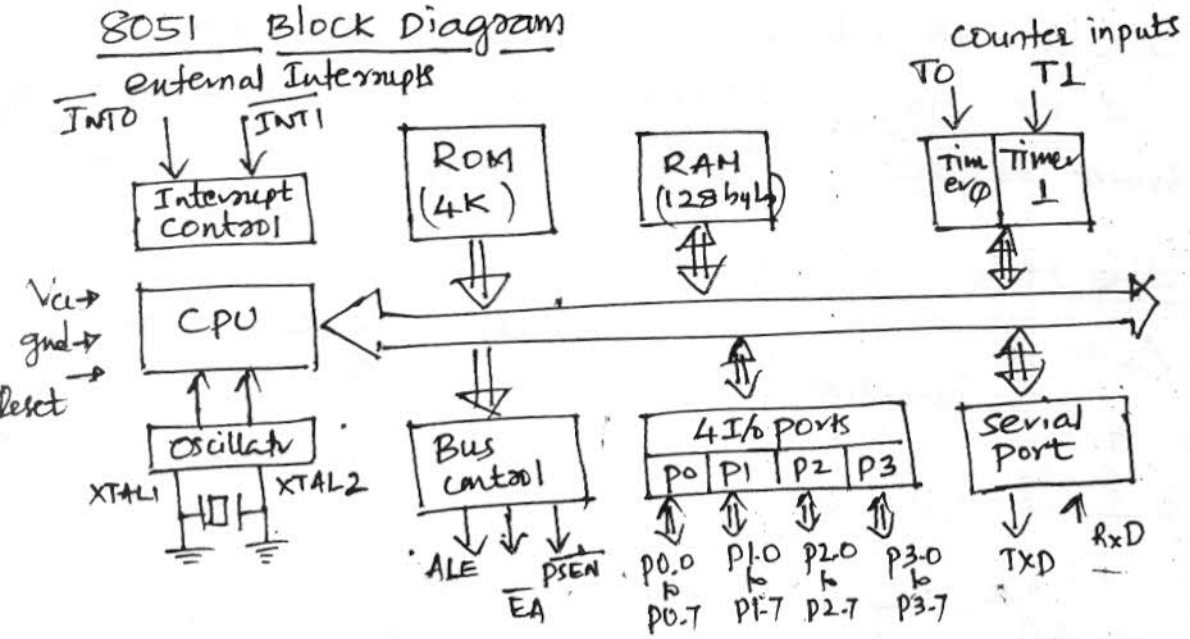
8051 Block Diagram



8051 is a 8 bit Microcontroller from Intel, which has inbuilt

* program/code Memory, used to store instns & constant data
  (4K - 8051, 0K - 8031, 4K EPROM - 8751
  ROM latest versions like 89C51, 89V51.. has upto 64K flash memory)
* Data Memory / RAM
  (128 bytes, in 8052 and other variants it has 256 bytes)
  used to store variables, stack & to represent registers
* programmable Input/output bit addressable ports
  (4 ports - P0, P1, P2, P3, total 4×8 = 32 I/o lines)
* 2 programmable 16 bit timers/counters, used for generating time related signals/waveforms, for counting of events, without causing any overhead to ~~xx~~ Microcontroller
* uC can communicate to pc, using serial communication, (full duplex) for uploading/downloading of data using the ~~xx~~ inbuilt serial port, so that lot of programming burden of uC is reduced.

Also uC-8051 supports, two external interrupts, INT0 & INT1 which can be connected to external devices to support interrupt driven data transfer.

8051 supports 64K external code memory and 64K data memory, ALE, $\overline{EA}$ & $\overline{PSEN}$ are used to support this interfacing.

P0 & P2 has dual functionalities (as ports and Addr/data bus),

P3 shares with TxD, RxD, $\overline{RD}$, $\overline{WR}$, T0, T1, $\overline{INT0}$ & $\overline{INT1}$ lines.
(port 3 lines can be used for any one functionality).

[8051 supports total five interrupt's - 2 external H/w ints, 2 for timers, one for serial port]

Addressing Modes

8051 can have its operands, as part of the instn, in Regs (A, B, R0-R7, DPTR), in Memory (internal memr (RAM) / external mem (RAM/ROM)). Based on the source of operands, following adding modes are available.

Immediate Adding Mode

ex: MOV A, #12 ← immediate data

destination can be reg or memory or SFR register

A ← 12

Register Adding Mode

ex: MOV A, R1 ← Register operand

A ← (R1)

Note: MOV R_m, R_n (X) not allowed; m, n – 0 to 7

Memory Related:

Direct Addressing Mode

ex: MOV A, 20H

A ← (20H) ↑ refers to memory (RAM) location 20H

MOV 30H, 20H ; (30H) ← (20H)

MOV 30H, #10H (30H) ← #10

Mem adds – 00-FFH is allowed, 00-7FH – refers to RAM

80-FFH – refers to SFR adds space.

Indirect Adding Mode

ex: MOV A, @R0 , R0 & R1 can be used to hold adds of mem locations

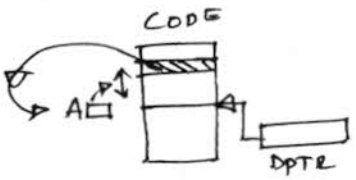
A ← ((R0))

(R0) – adds of mem locatn

((R0)) – contents of that mem locatn

Indexed Adding Mode

ex: MOVC A, @A+DPTR



A ← ((A) + (DPTR))

↳ refers to memory location in code memory, (EA decides internal OR external memory)

– commonly used to access look up tables, strings etc

Related to Branch Instructions, we have 3 adding modes are associated:

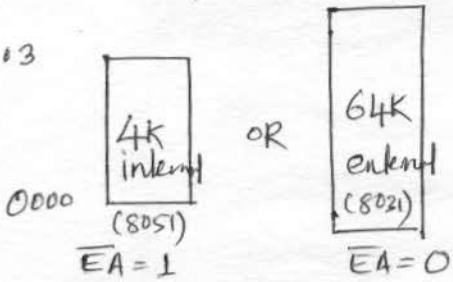
Relative Adding Mode — SJMP LOOP1, relative to PC, in the range of −128 to +127

Absolute Adding Mode — AJMP LBL, anywhere within 2K

Long Adding Mode — LJMP Begin, anywhere within 64K

Memory organisation of 8051

8051 supports two address spaces (i.e two types of memory spaces) ① program memory or code memory – upto total 64K (either located internally (8051) or externally (8031))



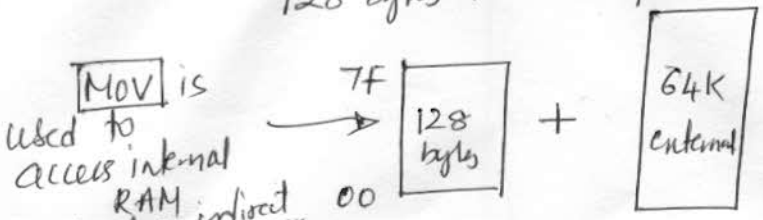
– The pin $\overline{EA}$ (external Access) indicates, whether the code memory is located inside or outside

($\overline{PSEN}$ (program storbe enable) signal is used as read storbe for interfacing external ROM)

– MOVC A, @A+DPTR } are used to access data stored as part of code ex: look up tables, strings.
MOVC A, @A+PC }

② Data memory or RAM memory –

128 bytes internal plus upto total 64K external RAM possible

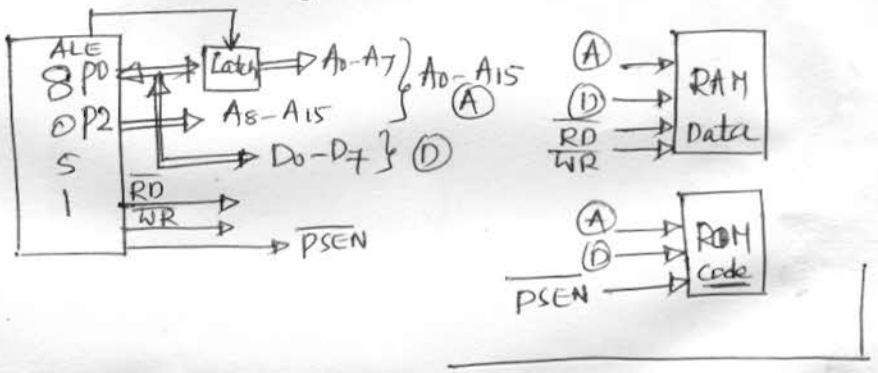


– MOVX instruction used to access external data/RAM memory
(MOVX A, @Ri or MOVX A, @DPTR)
MOVX @Ri, A or MOVX @DPTR, A
i=0,1

($\overline{RD}$ (i.e P3.7) & $\overline{WR}$ (P3.6) are used for interfacing external RAM)

NOTE: (extra information) Interfacing external RAM & ROM.

- P0 & P2 generates 16 bit address bus & 8 bit Data bus
- $\overline{RD}$ & $\overline{WR}$ are for RAM read & write
- $\overline{PSEN}$ is for ROM, read operation.



ex: program to read one byte from code memory and store it in internal RAM location.

```
MOV DPTR, #STR ,  MOV A, #00H
MOVC A, @A+DPTR ,  MOV 50H, A ; let 50H be the addr of RAM location
SJMP $
STR: DB 'A' ;  Part of code memory
```

(ASCII code of 'A' is stored in code memory, which is pointed by DPTR register, and then moved to RAM location 50H)

ex: program to store the contents of RAM location 70H to external RAM location 2000H

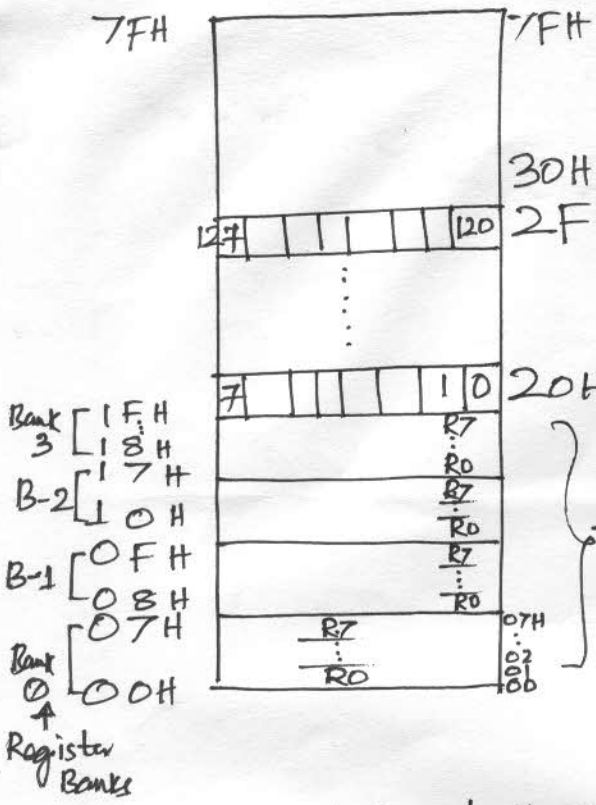
```
MOV A, 70H
MOV DPTR, #2000H
MOVX @DPTR, A
```

Internal RAM organisation

8051 has 128 bytes of internal RAM, (00H to 7FH) which can be accessed directly (MOV A, 10H) or indirectly (MOV R0, #10H ; MOV A, @R0) using MOV instruction.

Internal RAM is used for,

- Register Banks, (4 Reg Banks, each having eight 8 bit reg's - R0 to R7)
- Bit addressable Memory, (128 bits, i.e 16×8 = 128)
- Stack implementation, as stack is always implemented in RAM area
- Scratch pad memory, for storing temperary data



- Selection of Banks is through PSW.4 & PSW.3 bits

PSW.4	PSW.3	
0	0	- Bank0
0	1	- Bank1
1	0	- Bank2
1	1	- Bank3

On reset Bank0 is default selection, hence R0 to R7 are refered to adds 00 to 07H.

- Write a program to switch to Bank1

```
SETB PSW.3
CLR  PSW.4
```

(: MOV R0, #25 is same as MOV 8H, #25)

- Write a program to set bit adds 7 and output it to P0.7

```
SETB 7
MOV C, 7 ;   Move the value of bit(whose adds is 7)
             to CY flag
MOV P0.7, C
```

Delay Calculation : using software, (through instructions)

(a) write a delay program for 10 msec? Given crystal freq = 11.059 MHz
(Instructions execution time is indicated by no of machine cycles, in typical 8051, one machine cycle consumes 12 clock cycles.)

1 MC = 12 × T, T-refers to time period of 1 clock cycle
= $12 \times \frac{1}{f} = 12 \times \frac{1}{11.059M} = \underline{1.085\ \mu sec}$

```
delayprog: MOV R1, #count   — 1
  Outer:   MOV R0, #250     — 1
  Inner:   NOP              — 1
           NOP              — 1
           DJNZ R0, Inner   — 2
           DJNZ R1, outer   — 2
           RET              — 2
                              ↑ No. of M/c cycles
```

(Inner loop comprises of 4 MC's - 2+1+1, executed 250 times, which in turn executed count times)

T_D = Count × Time by Inner loop

Total Delay = Count × [4 × 250 × MC]

T_D = Count × 1000 × MC

$Count = \frac{T_D}{1000\ MC} = \frac{T_D}{1000 \times 1.085\mu}$

if T_D = 10 msec

$Count = \frac{10M}{1000 \times 1.085\mu} = \frac{9.21}{10} \simeq 9$ or

(b) Find the size of delay produced by following program?

```
DELAY: MOV R3, #200   — 1
here:  DJNZ R3, here  — 2
       RET            — 2
                        ↑ MC's
```

$T_D = (2 \times 200)MC + (1+2)MC \simeq (2 \times 200)MC = 400 \times 1.085\ \mu sec$
$= 432\ \mu sec.$

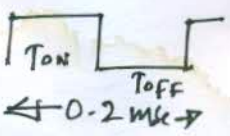
(c) write a program to generate pulse of 10 msec; on the port pin P0.0

```
SETB P0.0 ; configure as o/p
CLR P0.0, SETB P0.0 ;
LCall Delay-10ms    ;
SETB CLR P0.0       ;
SJMP $.
Delay-10ms: ; (refer prev. Question)
RET
```

(d) write a program to generate square waveform of 5 KHz, on P1.0

```
cont: SETB P1.0
      LCall Delay-
      CLR P1.0
      LCall Delay
      SJMP cont
Delay: ;
      RET
```

$f = 5$ KHz, $T = \frac{1}{5\ KHz} = 0.2$ msec.



$T_{ON} = T_{OFF} = 0.1$ msec

(Delay program can be implemented using single loop)