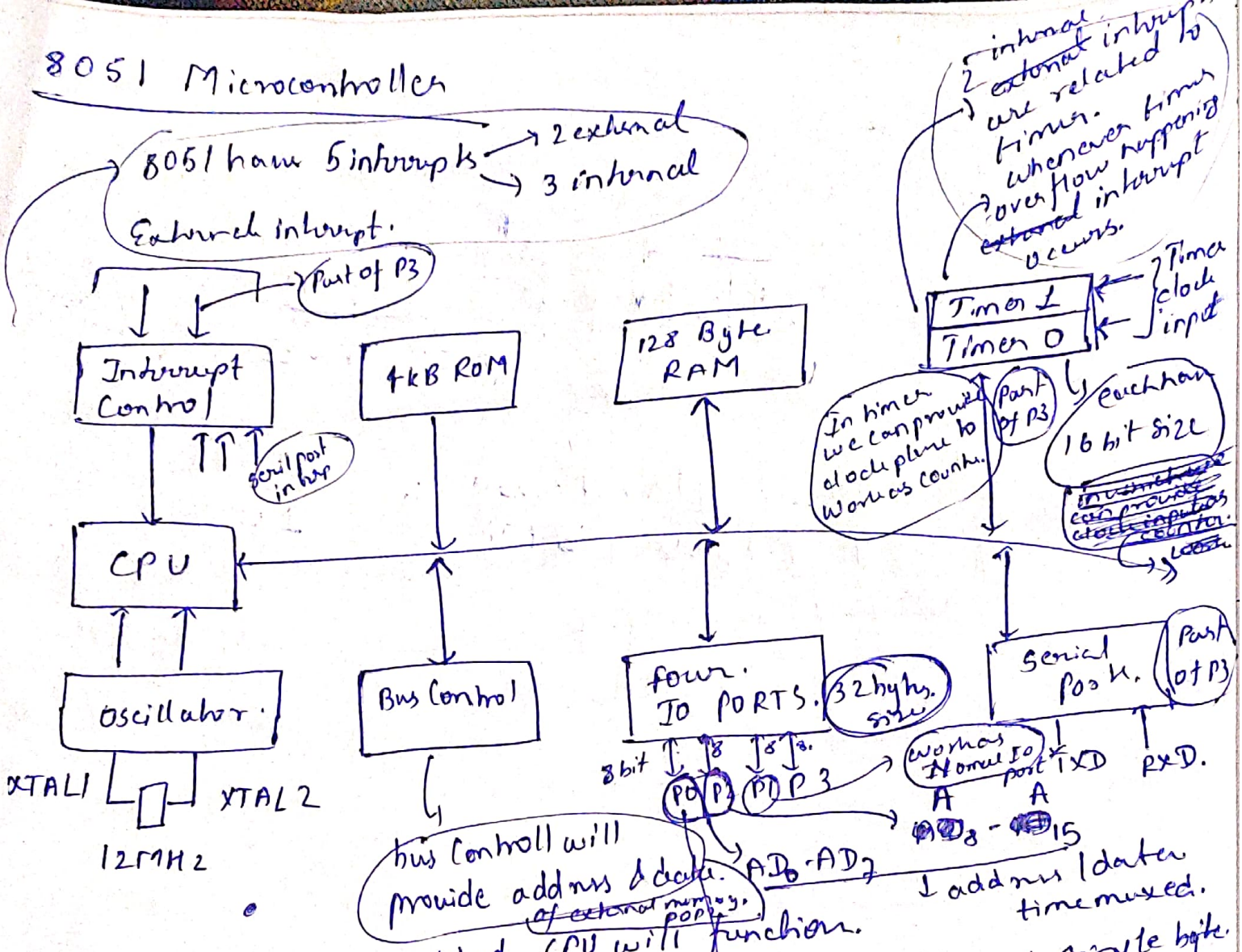
8051 Microcontroller



1) once 12MHz is provided CPU will function.

2 Interrupts → for timer
1 Interrupts → for serial ports.

Interrupt Control: handle / tells which interrupts will execute first by the processor / CPU. (Total 5 interrupts are there).

4kB ROM: program is stored in ROM. → Address range. (0000H to 0FFFH).

128kB RAM: data is stored in RAM.

8051 follows Harvard architecture as separate memory is there for. RAM & ROM.

→ with the help of P0 And P2 the bus Controll will Controll external memory address / data.

Port-3 have Multiple functionality.

1) It is working as IO port.
2) It is used for Serial communication. (RxD, TxD)
3) It is used for timer clock input. (Timer 1, Timer 0)
4) It is used for external hardware interrupts.

P3 - 8 bit (8 pins)

4 $\overline{RD}$ } ~~external memory Read/write~~
4 $\overline{WR}$ } for External RAM

* Bus Controll will provide address/data from external memory through P0.

* RAM Memory Space Allocation 8051 {00 - 7FH} 128 bytes.

00H - 1FH → 32 bytes → Register banks. → 32 bytes of RAM

128 → 16 bytes → 20H - 2FH → bit address area in RAM

→ 80 bytes

R0-R7	Bank 3 (Default Bank)	18H - 1FH
R0-R7	Bank 2	10H - 17H
R0-R7	Bank 1	08H - 0FH
R0-R7	Bank 0	00H - 07H

8 registers each bank.

MOV A, R0
MOV A, 08H → address without #

00H - 1FH
10H - 2FH
20H - 7FH

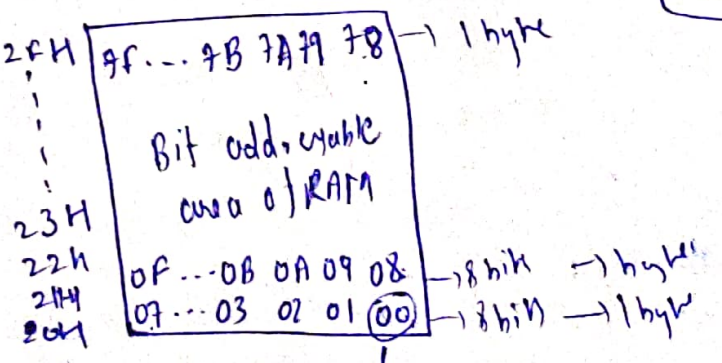
80 bytes of RAM for general purpose (Scratchpad RAM) 30H - 7FH.

16 bytes of RAM for bit wise addressing.

~~Stack pointer reside inside Scratchpad RAM only. responsible for push/pop operation~~

→ push/pop operation can be used by programmer.

→ only decided by programmer how he use it that area.



each bit can be addressable.

SETB 7FH : set MSB of 2FH Ram
CLR 08H : clear LSB of 21H Ram location

★ Stack of 8051 μC (8 bits) register.

→ Stack is memory area operated with SP. (Stack pointer).

→ SP ~~associated~~ associated Internal RAM. (128 Bytes) (00H – 7FH) (0 – 127)

→ SP can point any of the byte in (00H – 7FH) range.

→ Stack used to store & return address during ISR and subroutines.

→ default address SP. during Reset is 07H.

→ programmer can change SP location as per their requirement by using:

MOV SP, #20H ; SP holds 20H address of RAM.

SP → data point karta h islye # lage hua. but vo. data ek address h.

→ push/pop operation performed are.

```
1)  MOV R1, #11H   ; R1 ← 11H
2)  MOV R2 #22H    ; R2 ← 22H
3)  MOV SP, #2FH   ; SP ← 2FH   Is by default address of SP is 07
4)  PUSH R1        ; push R1 on stack
5)  PUSH R2        ; push R2 on stack.
6)  POP R3         ; pop R3 from stack.
7)  POP R4         ; pop R4 from stack.
```

Address	Data
09H	
08H	
SP → 07H	XX

Default location of SP.

Address	Data
31H	
30H	
SP → 2FH	XX.

After 3rd instruction.

Addr	Data
31H	
→ 30H	11H
2FH	XX

After 4th.

Address	Data
→ 31H	22H
30H	11H
2FH	XX.

After 5th instruction

Address.	Data.
~~→~~ 31H	22H
→ 30H	11H
2FH	XX

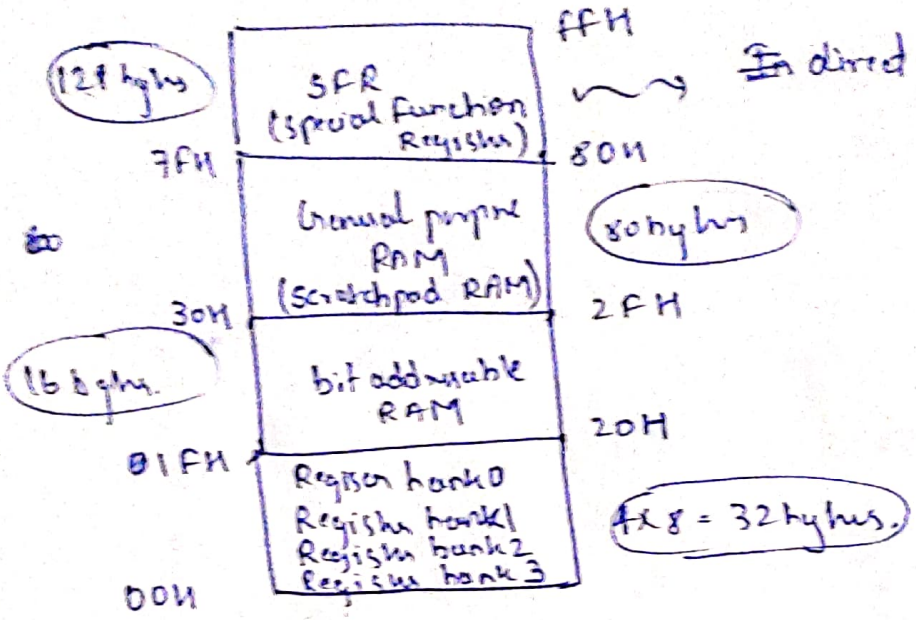
R3 = 22H
decrement by 1.

A	D.
31H	XX
~~→~~ 30H	11H
→ 2FH	XX

R4 = 11H
decrement SP.

...saved. become address of ISR ~~interrupt~~ received from ...

* Memory Structure of 8051



* Special function Registers. (SFR) (128 Bytes).

→ we have 21 Special function Registers. (8bit each) 1bytes.

→ Timer, Counters, IO, Serial, communication, Interrupt.

→ There all SFR along with address to reduce the No. of OP codes.

→ Internal Ram is from (00H to 07H) & SFR addressing are used in (80H to FFH)

→ It supports Byte & bit wise operations.

Eg SETB P0.0 ; SETB has opcode & P0.0 has address. i.e. 80H

→ 8051 supports → bit wise Special function
→ byte wise Special function

→ If addressing not done with SFR then there will be many opcodes that will ↑↑ the Complexity of instruction decode.

8051 Instructions.

1) MOV A, #55H
2) MOV DPTR, #55H
3) MOV P1, #55H

② Exchange Instruction

→ XCH A, 30H ; A ↔ M[30H]
→ XCH A, R0 ; A ↔ R0
→ XCH A, @R0 ; A ↔ M[R0]

Logic Instructions

① ANL
② ORL
③ XRL
③ CPL → only work with Port No A.

Q. Find 2's complement of value 85H.

```
MOV A, #85H
CPL A
ADD A, #01
```

③ Arithmetic Instructions.

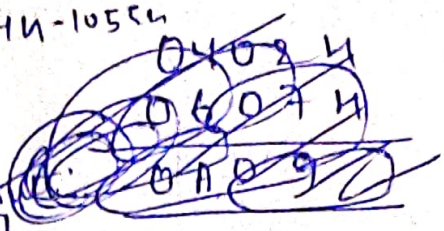
→ ADD A, byte.
→ ADDC A, byte
→ SUBB A, byte
→ INC A
→ INC Byte
→ INC DPTR
→ DEC A
→ DEC ~~DPTR~~ Byte
→ MUL AB
→ DIV AB
→ DA A → use to facilitate BCD addition
only work after add Instruction
Not work after INC instruction

Eg
```
MOV A, #23H
MOV B, #29H
ADD A, B    A ← A+B = 4CH
DA A.       A ← A+6 = 52
```

23
29
53 → in decimal but ans is 4CH

★ Addition of 16 bit Numbers.

1st No: 1050 - 1051 H
2nd No: 1052 H - 1053 H
Store at 1054 H - 1055 H



```
LHLD 1050H      H=[1051] L=[1050]
XCHG            D=[1051] E=[1050]
LHLD 1052H      H=[1052] L=[1053]

MOV A, L
ADD E
MOV L, A

MOV A, D
ADD H
MOV H, A

SHLD 1054H
```

★ Multiplication of two Numbers. (8 bit)

1st No => 1050H
2nd No => 1051H.

B C
04 03

```
MOV A, 00H
MOV D, 00H      ---> for MSB

LXI H, 1050H
MOV B, M
INX H
MOV C, M

loop: ADD B
      JNC exit
      INR D      ~~> increment carry generated.
exit: DCR C
      JNZ loop

MOV H, D;
MOV L, A;
SHLD 1052 H
```

SIA ~~> for same as above.

* Division of two 8 bits Numbers

```
MOV D, 00H        (D → Quotient)
LXI H, 1050H
MOV B, M          B = 2
INX H
MOV C, M          C = 10
loop: SUB B       A - B (10 - 2)
INR D
CMP B             (A > B) Cy = 0, Z = 0
JNC loop
STA 1052H         → Remainder
MOV A, D          → Quotient
STA 1053H
HLT
```

divisor 2 / 10 dividend

① 1050H — 2 — (B)

② 1051H — 10 — (C)

* Find Max of 5 Numbers stored at 1050H to 1051H.

```
LXI H, 1050H
MOV A, M
MOV C, 04H
loop: INX H       H = 1051H
CMP M             (A > M) C = 0
JNC exit          (A < M) C = 1
MOV A, M
exit: DCR C
JNZ loop
```

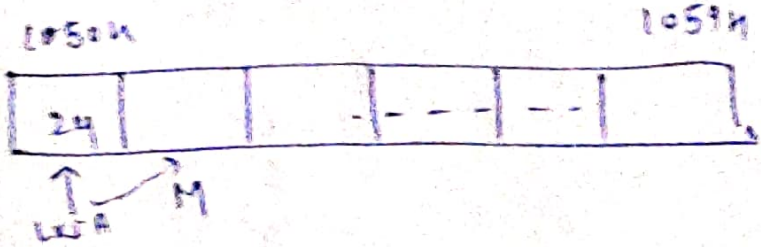
(A) → contain greatest/max value.

* Store Number In Reverse Order 1050H - 1054H and 1060H - 1064H.

(↑H at 1050H, ↑D at 1064H)

```
LXI H, 1050H
~~MOV A, M;~~
LXI D, 1064H
MOV C, 05H
loop: MOV A, M
INX H
STAX D
DCX D JNZ loop:
```

Addition of Number array.




```
LXI H,1050H
MOV A,M
```
A ← [1050] = 51

```
MOV C,09H
MOV B,00H
```
For carry

```
loop: INX M
      ADD B    A ← [A+M]
      JNC skip
      INR B
skip: DCR C
      JNZ loop
```

For storing back number

```
      STA 1060H   → lower bit of sum
      MOV A,B
      STA 1061H
      HLT
```

Ascending order.

```
      LXI H, 2050H
      MOV B, 04H
loop2: MOV C, 04H
loop1: INX H
       MOV A,M
       INX H
       MOV D,M
       CMP D      (A<D) ✓ skip (Y=1
       JC skip:
       MOV M, A
       DCR H
       MOV M,D
       INX H
skip: DCR C
      JNZ loop1   (in)
      DCR B
      JNZ loop2   (out)
      HLT
```