Page No. 1
Date

Unit 1: Basics of Microcontroller (μC) and intel 8051 architecture

Microcontroller is a single chips controller which control the operation of the all units

An embedded system uses either μC or μP to perform a single task. Mouse, printer, remote control of TV are example of embedded system

A μP doesnot have any ROM or RAM, and no any I/O ports on the chip itself

However, a microcontroller has many of these feature i.e memory, ports, etc. on the chip itself
It is a computer by itself. So μC is also called a micro computer.

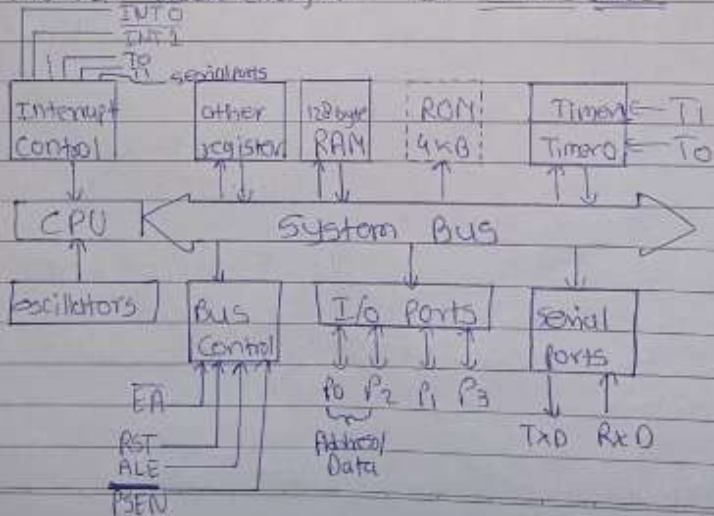
Difference bet^n Micro-processor & Micro-controller

Micro processor (μP)	Microcontroller (controller (μC)
① Micro processor are widely used in computer system	② Microcontroller is widely used in embedded System
② It has only CPU embedded into it	② It has CPU, a fixed amount of RAM, ROM and other peripherals all embedded on it
③ In the case of μP we have to connect all the component externally so the circuit became to complex	③ As all the components are Internally connected in μC so the circuit size is small
④ It consume more power	④ It consume less power than μP

① µp has less no of registe	① µc has more no of register
② µp has a high speed	② µc ~~has~~ speed depends on the architecture
③ complex & Expensive	③ simple & affordable

Features of 8051 µc
i) RAM - 128 Byte
ii) ROM: 4 KB on chip ROM
iii) I/O ports - P0, P1, P2, P3
iv) Oscillator clk circuit with crystal frequency 11.059 MHz
v) Timers :- T0 & T1 (Each timer is of 16 bits)
vi) Special function Registers :- PCON, TCON, TMOD IP, IE, etc
vii) serial ports - 1

Internal block diagram of 8051 µc.



1) CPU (central processing unit) :-
- 8051 uc has on chip CPU with units like Memory unit, control unit, ALU (Arithematic & logical units)
- 8051 has preprocessor which prcess on data & produces the result i.e O/P.

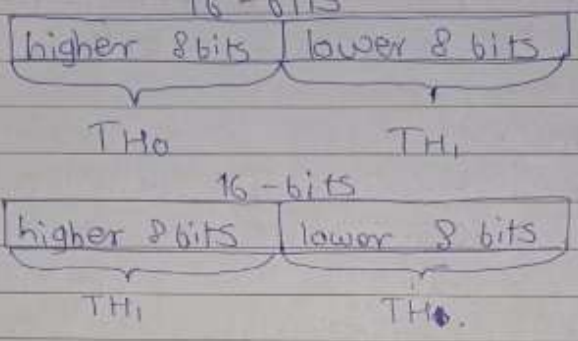
2) oscillator's :-
- 8051 uc has on chip oscillator with crystal frequeny 11.0592 MHz ≈ 12 MHz
- The pins XTAL-1 & XTAL-2 are used to connect the crystal externally

3) Interupts control :-
- 8051 uc has consist of in all total 6 interupts such as $\overline{INT\,0}$, $\overline{INT\,1}$, T0, T1, serial ports, RST (Reset)

4) RAM :-
- In all total 256 Bytes RAM is available
- The upper 128 Bytes are reserved for SFR's and the lower 128 are reserved for User's
- On chip RAM is divided into three part's namely Register Banks, bit addressable area, general purpose area.

5) ROM :-
- 8051 uc has consist of onchip 4 kB ROM.

6) Timers (T0 & T1) :-
- 8051 uc consist of two timers namely T0 & T1
- Each timer is of 16 - bits.
- Timer can be divided into two parts such as



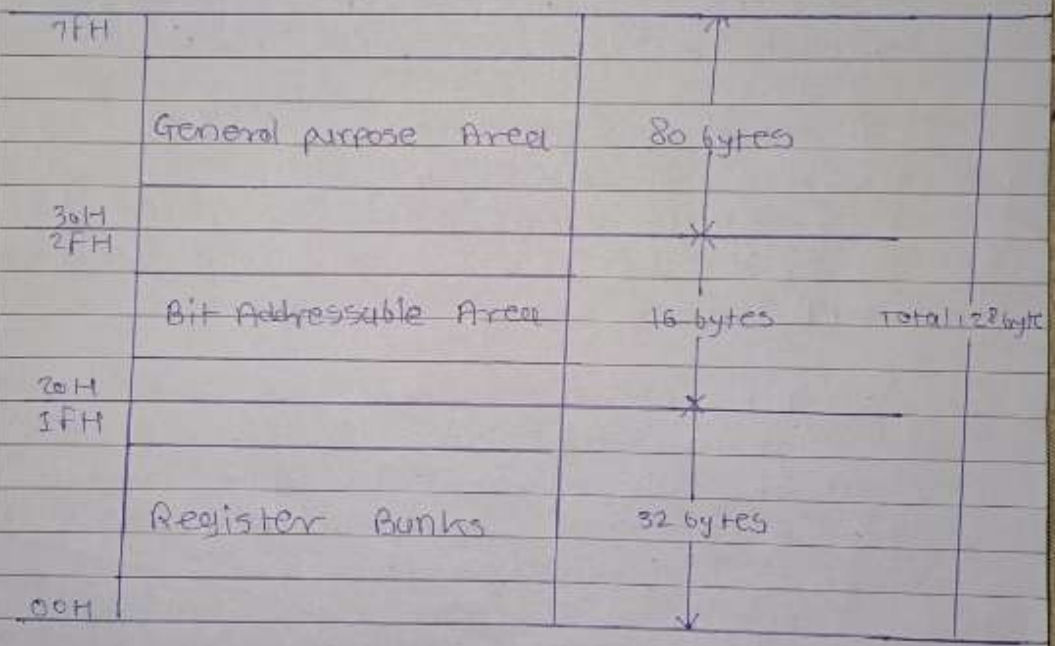
7) I/O ports :-
- 8051 uc has consist of 4 I/O ports namely P0, P1, P2, P3
- each port is of 8 bits
- All the ports are bidirectional ports
- All the ports except P1 port shows dual function.
- All the ports are bit aswell as byte addressable.

~~8) 8051 uc has serial port~~

8) serial bus unit
- 8051 uc has consist of 1 serial port with 2 lines Namely TxD, RxD.
- This serial ports is use in case of external memory interfacing

4) Bus control unit:-
RST : Reset pin
ALE : Address latch enable pin
PSEN : Program store enable pin
EA : External Access pin.

Internal RAM organisation:-
- 8051 uC has consist of 128 bytes internal RAM which is get divide into the three parts such as Register banks, Bit Addresable Area, General purpose area, which is shown in the figure.



① Register Bank
- 8051 uc contains 128 bytes internal RAM.
- In that 32 bytes Register Banks are available

There are 4 Register banks Namely RB_0, RB_1, RB_2, RB_3
- Each Register bank consist of 8 GPR. Namely $R_0 \rightarrow R_7$
- Each Register in each register Bank has its own unique address

07H	R7	0FH	R7	17H	R7	1FH	R7
06H	R6	0EH		16H		1EH	
05H		0DH		15H		1DH	
04H		0CH		14H		1CH	
03H		0BH		13H		1BH	
02H		0AH		12H		1AH	
01H	↓	09H	↓	11H	↓	19H	↓
00H	R1	08H	R1	10H	R1	18H	R1

3) General purpose Area
- It is of 80 bytes
- It starts from 30H to 7FH.
- One can use stack memory area of this GPA, if needed.

2) Bit address area:-
- Internal RAM of 8051 etc consist of 16 byte bit addressable area.
- It can access single bit rather than byte in this bit addressable area.

~~3) General purpose area.~~

Page No. 2
Date

Pin discription of 8051 uc.
- It is of 40 Pin Ic
- Out of 40 pins two pins are GND & +Vcc pins
- Out of 38 pins 32 pins are assigned for ports
- The remain 6 pin are special function pins
- Pin specification.
1) Pin 20 - Ground Pin
2) Pin 40 - + Vcc Pin.
3) Pin 9 - Rest Pin
It resets / or clear the uc
~~4) Pin 18 XTAL 2 & XTAL 1~~
4) Pin 18 & 19 - XTAL 2 & XTAL 1 resp. onchip crystal oscillators is get connected to XTAL 2 whereas external crystal oscillator will be connected to XTAL 1 Pin.
5) Pin 31 - $\overline{EA}$ - External Access pin
- It is Active low ~~pin~~ Input pin
- If it is connected to +Vcc the external Memory can be access
- function of $\overline{EA}$ pin :- when low $\overline{EA} = 0$, 8051 accesses external program / ROM. $\overline{EA} = 1$, on chip memory is access.
6) Pin ~~30~~ 29 :- $\overline{PSEN}$ = It is program store enable pin. which is Active low.
This has to be connected to $\overline{OE}$ pin of ROM in case of external ROM access
- function of $\overline{PSEN}$ pin :- It is used to read signal from the external

Page No.
Date

① Pin - 30 - ALE Pin :- It is Address latch enable Pin. It is an active high output pin.
Function of ALE Pin :- ALE signal is used for demultiplexing & multiplexed Address / Databus of Port P0 during external Memory interfacing

② Pin - 9 - RST :- It is Reset pin. It is an active high input. When the signal on this pin is activated the microcontroller will terminate all activities & reset itself.

③ Pin - 39 → 32 are assign for port P0 as P0.0 → P0.7 respectively
- Port P0 is Bidirectional
- It is Dual port
- It is I/O port.
- It is Bit as well ass byte addressable.
- It is use in external memory interfacing to hold the address as well as to transfer data betn internal & external Memory.

④ Pin - 21 - 28 are assing for port P2 as P2.0 → P2.7
- Port P2 is also Bidirectional port.
- It is Dual por.
- It is I/O port
- It is bit ass well as byte addressable
- It is use in external memory interfacing to hold the higher byte address of external memory.

11) Pin 1 -> 8 is assign for port P1 as P1.0 -> P1.7
- It is used as a simple I/O port.
- It doesnot handel dual function.

special function Register.
- 8051 has 128 byte of special function RAM
- The address of RAM. The address of RAM varies from 80 H to FFH.

Page No. 1
Date 6

Unit 2: Programming model of 8051

Instruction function.

There are many types of instruction function.
Data transfer instruction
Arithematic —11—
logical —11—
Bit manipulation —11—
Program control —11—
Interrupt related —11—
Machine control —11—

Every instruction execute in a different way

Classification base on instruction size.

Depending on how much memory space an instruction uses for its storage it can be classified into

- One byte instruction
- Two byte instruction
- Three byte instruction.

Instruction set of 8051

Arithematic instruction of 8051

1) ADD Dest, source

Adds content of source with destination & result is stored in destination.

Here destination is always accumulator A, source can be immidiate data, register, direct or indirect addres of 2nd operand.

e.g
```
ADD   A, #data
ADD   A, R5
ADD   A, Address
ADD   A, @Ri   Ri store address of operand
```

② ADDC Dest, source :- Add contents of source with destination also considering carry generated

ADDC A ← A + Address + CY
ADDC A ← A + RS + CY

Syntax

Lable: opcode	destn operand	source operand.
	• can be register (A, B, C, D / R0 → R7) • direct Memory locn • Indirect ⊥ • It can be any port • but it can't be imidiate data	• Can be any register (A, B, C, D / R0 → R7) • direct Memory locn • Indirect ⊥ • It can be any Port • but it can be imidiate data.

e.g i) ADD A, R0
These instruction will add the content of with the content of R0 & store the result in A.
e.gn (A) ← (A) + (R0)

e.g Let us assume that
~~(A) ← (A) + (R0)~~
Let (A) = 45 H
(B) = 23 H
(A) = 68 H

$$\begin{array}{r} 0100\ 0101 \\ 0010\ 0011 \\ \hline 0110\ 1000 \\ 6\quad 8 \end{array} \Rightarrow \begin{array}{l} B \\ A \end{array}$$

After executing above ~~eg~~ instruction we have (A) = 68 H

ii) ADD A, <direct Memory location>
ADD A, 35 H

Statement] The instruction will Add the (A) with (35H) & store the result in A.

eq^n] [A] ← (A) + (35H)

e.g] Let us assume that

(A) = 36 H	0011 0110	B
(35H) = 1F H	0001 1111	
(A) = 55 H	0101 0101	
	↳5	↳5

~~After~~ executing above instruction we have (A) =

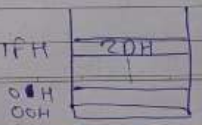
iii) ADD A, <indirect Memory location>
ADD A, @ Ri where i = 0, 1
ADD A, @ R0.

Here @ indicate indirect Memory location & the pointer are R0 & R1 only.

Statement] These instruction will Add (A) with indirect memory locat^n pointed by Ri and then store the result in (A)

eq^n) (A) ← (A) + ((R0))

eg] Let us assume that
(A) = 39 H
(R0) = 1F H
((R0)) = 20 H



(A) = 39 H
((R0)) = 2D H

$$\begin{array}{r} 0011\ 1001 \\ 0010\ 1101 \\ \hline 0110\ 0110 \\ \hline 6 \quad\ 6 \end{array}$$

After executing above instruction we have (A) = 66 H & AC sets.

iv) ADD A, <PORT>

ADD A, P_3

Statement] These instruction will add (A) with content of port P_3 and store the result in A.

eqn] (A) ← (A) + (P_3)

Eg.) Let us assume that,
(A) = 10 H
(P_3) = 78 H

$$\begin{array}{r} (A) = 10\ H \Rightarrow 0001\ 0000 \\ (P_3) = 78\ H \Rightarrow 0111\ 1000 \\ \hline 1000\ 1000 \\ \hline 8 \quad\ 8 \end{array}$$

After executing above instruction we have (A) = 88 H & AC remain clear.

v) ADD A, immidiate data

Here, an immidiate data is precedded by # sign

ADD A, # 36 H.

Statement These instruction will Add (A) with imidiate data 36 H & then stored the result in (A).

eqn] (A) ← (A) + immidiate data (#36)

e.g Let us consider that,
(A) = 5AH ⇒ 0101 1010
(#36H) = 36H 0011 0110
1001 0000
9 0

Afterexecuting above instruction we have (A) = 90 H & AC sets.

# ADDC A, (imidiate data)
ADDC A, #0ACH

put '0' in betn # sign & first Hexadigit has character.

Statement] These instruction will ADD (A) with immediate data ACH with carry & store result in A.

eq] (A) ← (A) + immidiate data + (Cy)

e.g] Let us assume that,
(A) = F3 H & immidiate data = ACH.
(Cy) = 1

A + immediate data + (Cy)
F3H + ACH + 1
⇒ 1111 0011 + 1010 1000 + 1
1 1010 0000
A 0

with imidiate data in (A).

After executing above instruction (A) become A0 & Cy clears.

ADDC <dest operand> <source operand>

<dest operand> can be	<source operand> can be
• registers	Ports.
• direct memloration	
• Indirect memlocation	
• ports	
• can't be imidiate data	

INC (Destination operand)

can be
- register
- direct
- Indirect
- Ports
- can't be immediate data.

i) INC A

statement) These instruction will increament the content of A by 1.

eqn) (A) ← (A) + 1

e.g) Let us assume that,
(A) = FF H.

(A) = FFH
+ 1 = + 1
00 H

1 1111 1111
1 0000 0000
carry 0
0

After executing above instruction (A) = becomes 00H & AC, Cy, ov sets.

ii) INC DPTR.

Statement] These instruction will incre. content of immediate data DPTR by 1

eqn) (DPTR) ← DPTR + 1

e.g) Let,
(DPL) = FFH.
(DPH) = 00H.

DPTR
DPH 00H | DPL FFH.

(DPTR) = DPH 00 | DPL FF

DPH
0000 0000
+ 1
0000 0001
0 1 01H

DPL
1111 1111
+ 1
1 0000 0000 = 00H
0 0

After executing above instruction (A) = DPH. become 1H & DPL become 00H, AC sets

iii) INC @ Ri where i = 0, 1
INC @ R1

Statement) These instruction will increment the content of indirect Memory location pointed by R1

eqn] (CRi)) ← ((CRi)) + 1

eg) (R1) = ffH
((R1)) = FEH
((R1)) ⇒ FEH ⇒ 1111 1111 1111 1110
+1 1 1
((R1)) FFH 0000 0000 1111 1111
F F

After executing instruction ((R1)) Become ffH

DEC (destination operand)
can be
→ register
→ direct
→ indirect
→ ports
↳ can't be immidiate data

1) DEC A
statement) these instruction will decrement the content of A by 1

eg let (A) = 00H.
(A) = 00H ⇒ 0000 0000
1 = 1 1
1111 1111

After executing above instruction (A) become FFH & Cy sets.

★ SUBB <dest^n operand> <source operand>

dest^n operand can be	source operand can be
→ register	→
→ direct	→
→ indirect	→
→ ports	→
→ can't be an imm. data	→

i) SUBB A, <source operand>

SUBB A, B

Statement] These instruction will substract (B) with carry from (A) the store the result in A.

eq^n] (A) ← (A) - (B) - (CY)

eg) Let (A)= 34 H → 0011 0100
(B)= 3F H 0011 1111
(CY)= 1 1

1 1111 0000
F 4

After executing above instruction (A) becomes F4 H & CY sets.

ii) SUBB A, @ R_i

SUBB A, @R_i

Statement] These instruction will substract content of indirect memory location pointed by R_i & carry from (A) & store the result in A.

eg"] (A) ← (A) − ((R1)) − C4

A = C9 H
((R1)) = A0 H
C4 = 0 H

1100 1001
1010 0000
0
0010 1001
2 9

After executing above instruction (A) becomes 29 H.

MUL AB

Statement] These instruction will multiple (A) with (B) & lower - 8 bit register in A where higher higher 8 bit ~~register~~ register in B.

RULE] i) The two operand must be A & B
ii) Their should be comma in bet" two operand.
iii) ~~Their should be~~ Don't change the post" of A&B

e.g let
(A) = 34 H
(B) = 52 H

3×16¹ + 4×16° = 52 D
5×16¹ + 2×16° = 82 D.

(A) = 52 D
(B) × 82 D
4264

16	4264	→ 1	8
16	266	10	A
16	16	0	0
	16	1	1

quotient
divisor) divident
remainde

After executing above instruction (A) = A8H & (B) = 10H.
ie (A) = A8H & (B) = 10H.

DIV AB

Statement These instruction will divide the A & B. & the quotient resides in A where remainder reside in B.

eg) let, (A) = 50 D
(B) = 10 D

16	50	→ 2
16	3	→ 3
	0	

16	10	→ A
16	0	→ 0
	0	

∴ (A) = 32 H & (B) = 0A H.

SWAP A

These instruction will swap lower nibble with higher nibble.

higher nibble ⟷ lower nibble.

(A) = CB H
∴ (A) = (1100 1011)
SWAP A gives
(A) = 1011 1100
B C
∴ (A) = BC H.

Page No. 3
Date

Logical Instruction

The logic gate which perform basic logic operation are used here

* ANL (dest^n operand) (source operand)

i) ANL <30H> , P3.

Statement These ~~operation~~ instruction will perform logical AND operation bet^n the direct memory location 30H & content of port P3 & store the result in direct memory location 30H.

eq^n] (30H) ← (30H) logical AND (P3)

e.g Let us assume that

(30H) = 3CH

(P3) = FFH.

(30H)	3CH	0011 1100
AND	⇒ AND	⇒ AND
(P3)	FFH	1111 1111
(30H)	(3CH)	0011 1100
		3 C

ORL (dest^n operand) (source operand)

i) ORL 33H , P0

These instruction will perform logical OR operat^n bet^n (33H) & (P0) & store result in 33H.

eq^n (33H) ← (33H) logical OR (P0)

e.g let (33H) = 73H

(P0) = EFH.

(33H) OR (P0) 3H ⇒ 73H OR FFH 73H ⇒ 01110011 OR 11111111 11110001 11111111 F F

after executing above instruction (33H) become FFH.

XRL destⁿ operand, (source operand)

XRL A, R3

Statement] These instruction will perform XOR operatⁿ betⁿ (A) with (R3) and store the result in A.

eqⁿ] (A) ← (A) + (R3)

Eg) Let
(A) = 13H.
(R3) = 89H.

(A) XOR (R3) ⇒ 13H XOR 89H ⇒ 00010011 XOR 10001001 10011010 9 A

After executing above instruction (A) become 9AH.

CLR A

Statement] clear the content of A.
It is 1 byte instruction & the operand must be accumulator)

e.g) Let (A) = 1FH.
CLR A
After executing above instruction (A) = 00H

CPL A

Statement] complement content of A (take complement of A.)

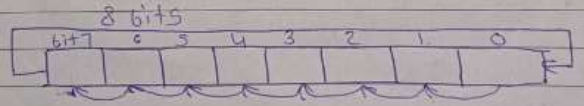
eqn] $(\bar{A}) \leftarrow (A)$

e.g Let (A) = 55H.
(A) = 0 1 0 1 0 1 0 1
$(\bar{A})$ = 1 0 1 0 1 0 1 0 — 1's complement
A A

After executing above instruction (A) = AAH.

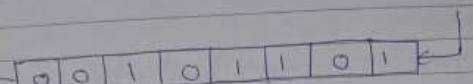
RL A

Statement] Rotate the content of A to left bit by bit



= bit 1 ← bit 0, bit 2 ← bit 1 ... bit 7 ← bit 6.

e.g Let (A) = 2DH.
= 0010 1101

∴ (A) = 

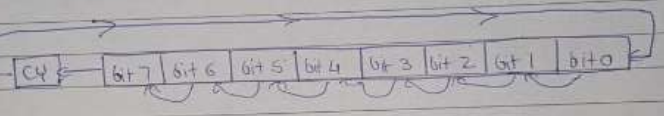
After executing RL A.

A = 0 1 0 1 1 0 1 0
5 A

∴ (A) = 5AH.

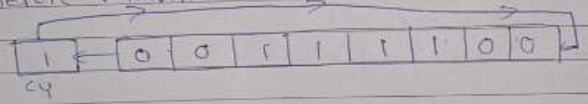
RLC A:

Statement) Rotate the content of A to left to carry.

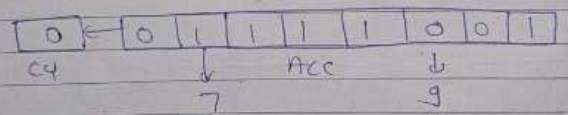
(A) = 

eg) Let (A) = 3CH & CY = 1
= 0011 1100

Before rotation,



After Rotation,

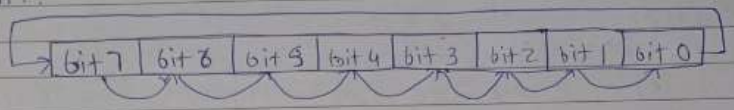


∴ A = 79 H.

After executing above instruction A become 79 H.

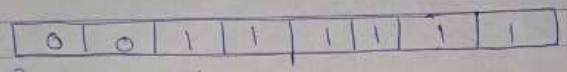
RR A

Statement] Rotate the content of A to the Right, bit by bit.

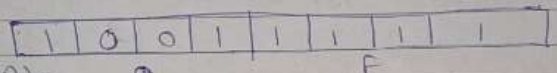


e.g] Let (A) = 3FH.
= 0011 1111

Before executing,



After executing



(A) = 9 F

After executing above instruction A become 9FH.

RR C A

statement] Rotate the content of A to the Right through carry (same like RLC.)

Data transfer instruction.

MOV dest^n operand <source operand>

can be
- → registers
- → direct Mem loc^n
- → Indirect Mem loc^n
- → Ports
- → can't be immediate data

i) MOV <direct> , <Indirect>
MOV 30H , @R1

e.g Let (R1) = 31 H.
((R1)) = 20 H.
After executing above instruction (90) H become (20 H)

ii) MOV DPTR , # 1234 H.
1234 H.
DPH DPL
After executing above instruction (DPH) becomes 12 H. & DPL become 34 H

MOV x A @Ri , i = 0, 1
Statement] These instruction will move the content of external memory location whose address is pointed by register Ri to accumulator (where i = 0, 1)

eg] Let (Ri) = 40 H.
((Ri)) = AB H.
∴ After execution above instruction
(A) = AB H.

MOV x @Ri , A
Statement] These instruction will move the content of the accumulator to the external memory location whose address pointed by Ri (where i = 0, 1)

MOV x @R0 , A.
let (R0) = 50 H.
(A) = 25H
After executing above instⁿ ((R0)) = 25H OR (50)H = 25 H.

PUSH Direct.

Statement

XCH destination operand, source

Statement] This instruction will exchange the content of dest^n oper^n & source oper^n

i) XCH A, R0

Let (A) = 53 H.
(R0) = 26 H.

∴ After executing above instruction (A) = 26H. & (R0) = 53H.

MOV A, @A+DPTR.

Statement: These instruction will move the code byte to accumulator whose address is obtained by A & DPTR.

eg Let (DPTR) = 15 00 H
(A) = 52 H.
∴ The address generated as
(A)+(DPTR) = 1500 + 52 H
= 1552 H

Now, code byte of an address 1552 is then move to A.

Bit Manipulation / Boolean variable Instruction.

i) CLR Acc 0
These instruction will clear the 0th bit of Acc

ii) CLR P3.2
These instruction will clear the 2nd bit of P3

iii) CLR 20 H.
These instruction will clear bit addressable RAM 20H

iv) CLR C / SET C
These instruction will clear / sets the carry flag C

v) SETB P3.1
These instruction will set the 1'st bit of port P3

e.g) (P3) = 00H bit
0000 0000 - bit 0
0000 0010 ∴ After execution,
0 2 (P3) = 02 H.

in 8051 uc

Write a c program ∧ to display on Port P1, array of integers. size of any is 5.

→
```
#include <reg51.h>
void main()
{
    unsigned int Ar[] = 2,3,4,5,6;
    unsigned int n;
    for (n=0 ; n<=4 ; n++)
      P1 = Ar[n];
}
```

Write a c program in 8051 uc to do sum of all element in an array. & display on Port P3.

→
```
#include <reg51.h>
void main()
{
    unsigned int Ar[] = 1,2,3,4,5;
    unsigned int i;
    unsigned int sum = 0;
    for (i=0 , i<=4 ; i++)
      sum = sum + Ar[i];
    P3 = sum;
}.
```

Jc rel address

Jump if carry flag is set & the control is transferred to specified address Here the real address is presetl as it is or inlable

JNC rel address
Jump if carry flag is not set & the control is transferred to the specified address. o.w. (cy=1) contol will continue with net instruction execution.

~~JNB~~ JB Bit, address
Jump if indicated bit is set and the controlled is transfered to specified address.

JNB Bit, address.
Jump if the indicated bit is not set then the fontroll will transfereded to the specifide address lable

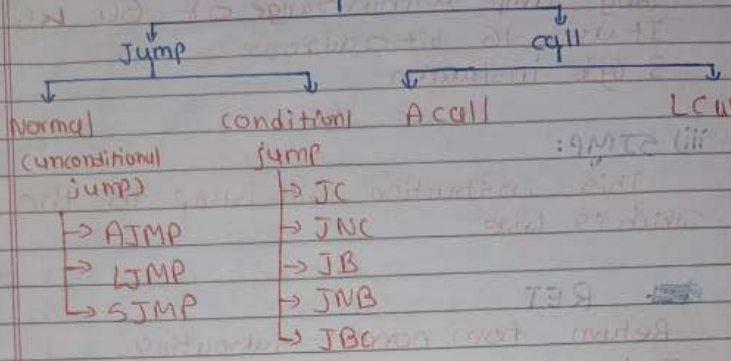
e.g
Back: JNB P3.1, BACK
Here uc will continue monitor first bit of port P3 until it become 1.
if it is become 1, then uc get out of the loop & start the program execution form the next instruction.

JBC Bit, address
Here, check the indicated bit is set, if it is set then jump to specified address & then compliment the indicated bit.

e.g
JBC P3.2, 300H ; Here P3.2 = 1
As P3.2 = 1 ∴ uc will jump to address 300H & also compliment the indicated bit
∴ P3.2 = 0.

Page No.
Date

Branching Instruction.



① call

i) ACALL:

- ACALL 11 bit Adress.

This instruction will call the 11 bit address & the program will start from that address. 11 bit address is present at 2kB memory location.

ii) LCALL:

LCALL 16 Bit address.

This instruction will call the 16 bit address. Here the address is a 16 bit & the range is available 63 kb.

Jump

i) AJMP 11-bit address.

Absolute jump to any where within 2kB range of program memory.

It use 11 bit address.

2byte instruction

ii) LJMP:
long jump within range of 64 kb
It uses 16 bit address
3 byte instruction

iii) SJMP:
This instruction will jump to the address which is lable

RET
Return from normal subrouting.
It is does not require any

RETI
retur from ISR (Interval service routine

JMP @ A+ DPTR.
This instruction will jump to the specified address generated by A+DPTR.

JZ address
Jump if equal to zero o.w continue with next instruction

JNZ address
Jump if not equal to zero o.w continue with next instruction.

CJNE desti, source, address
Compare & jump is content of destination & source operand is not equal to zero.
Here destination & source operand cannot be an immidiate data.

V.IMP
4 mark

e.g

DJNZ source, address
Decrement & jump if source content is not zero to specified address.
e.g
~~Back~~ BACK : DJNZ Rs, BACK
These instruction will continously decrement & monitor & register RS until bit becomes zero.

NOP (No operation)
It pause the microcontroller from the function.

V.IMP
4mark # Assembly Directives.
① ORG - (originate) counter from the
- It originate locath address.
(It is the starting point of program
e.g ORG 00H
It doesnot require any machine cycle

② END
It will stop the program execution or terminate the program.
The ~~e~~ instruction after END directive well not be executed.

③ DB (Define byte)
X DB, 82H.
Here x variable is a byte type variable whose value is 8 bit.
④ DW (Defined word)
It refered word type variable.

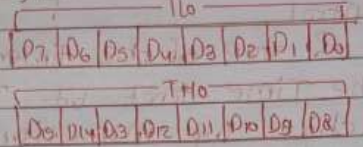
Page No.
Date

e.g X DW, 1234H

Here X variable is a word type variable whose value is 1234H.

Timer 0 Register:-

The lower byte of timer 0 is called TL0 & upper byte register is reffered as TH0.



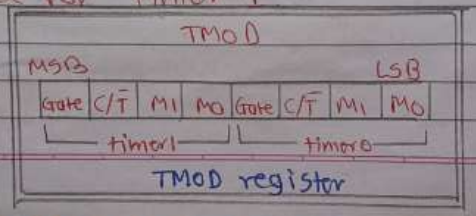
- These instruction can be accessed like any other register such as A, B, R0, R1, R2, etc.
- Instruction 'MOV TL0, #4FH' move the value 4FH into TL0
- Instruction 'MOV R0, TH0' move value of TH0 into R0.

Timer 1 Register:-

The Timer 1 register is also split into two bytes, TL1 & TH1.
These registers are accessible in the same way as register of T0.

Draw a bit format of TMOD register & explain function of each.

:- Timer has 4 operating modes. Both the timer use the register called TMOD, to set the various timer operating modes.
TMOD is an 8-bit register in which the lower 4 bit are set aside for timer 0 & upper 4 bit are set aside for timer 1.



① Gate:-
The purpose of this bit is to start & stop the time It can be done by s/w or by the hardware control
The start & stop of the timer are controlled by s/w by the Time start bit TR0 & TR1
This is achieve by instruction "setB TR1" and "CLR TR1" for timer 1 & "set B TR0" and "CLRTR0" for timer 0.
The setB instruction start it & CLR instruction stop it.

②C/T
This bit in the TMOD register is used to decide whether the timer is used as a timer or as an event counter
If C/T̄=0 then timer operation is activated.
If C/T̄=1, then event counter operation is enable

③ M1/M0: Timer operates in 4 different modes
This selection is doen by M1/M0

M1	M0	mode	
0	0	0	mode 0 → 13 bit timer mode.
0	1	1	mode 1 → 16 bit timer mode.
1	0	2	mode 2 → 8 bit timer mode.
1	1	3	mode 3 → split timer mode.

Indicate mode & timer selected for each of following.

MOV TMOD, #01H
MOV TMOD, #20H
MOV TMOD, #12H.

We convert above value in hex to binary comparing with TMOD register structure we get

a) TMOD = 0000 0001
Timer 1 Timer 0.

Here mode 1 of timer 0 is selected.

b) TMOD = 0010 0000.
Here mode 2 of timer 1 is selected

c) TMOD = 0001 0010
Here mode 1 of timer 1 & mode 2 of time 0 is get selected.

TCON Register :-
Timer / counter control register (TCON) consist of control bits & flag. Only 4 of all 8 bits of this register are used for timer control & the other are used for time control through external interrupt.

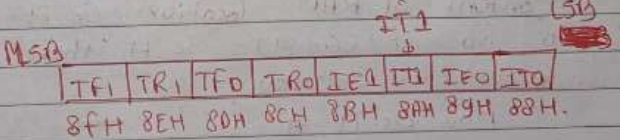


fig :- TCON Register

① TF1/0 :- Timer 1/0 overflow flag : set by hardware on timer / counter overflow

② TR1/0 Run Control bit :- set/clear by s/w to turn timer/counter ON/OFF.

③ IE1/0 edge flag :- set by hardware when external interrupt edge is detected

④ IT1/0 control bit :- set/cleare by software to specify falling edge/low level triggered external interrupt.

Addressing Modes in 8051 etc.

① Register ③ Indirect ⑤ Relative
② Direct ④ Immediate ⑥ Asolute
⑦ Indexed.

① Direct Addressing Mode :-
Direct addressing can access any on chip variable or hardware register.
- There are 128 byte of RAM in 8051
- this mode most often used to access RAM location 30 to 7f H.
- Its also possible both source & destination ~~Operand~~ address can be specified.

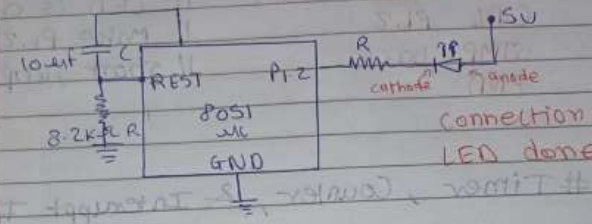
e.g) ① Mov R0, 60H
content of RAM location 60 H is copied into R0 if (60 H) = 30 H then after that instruction is executed

② Mov 40H, A.
content of A are copied into RAM location whose address is 40 H.

Explain how LED can be interfaced to Port P1.2 of 8051 uc with neat diagram write a program to blink/ the LED.

toggle

→



connection for a single LED done directly.

Fig shows LED can be connected to 8051 A driver may require if many LED's are to be interfaced.

- Here LED is connected to Pin P1.2
- R is a current limiting register
- The anode is connected to 5V
- The cathod is get connected to Pin P1.2 through R.
- LED conducts when it is in forward biase so, when voltage at P1.2=0, The LED is forward biased & current flow from anode to cathode into the port Pin P1.2
- On the other hand when P2.1 = 1 (+5V) the LED is not conducting as its anode & cathode are in same potential.
- To operate these software we need to cause LED become on & off
- The hardware is run when there is a software support to it.

Page No.
Date

Pgm

#Program:-

```
ORG 000H
SETB P1.2          // make P1.2 = 1
back: ACALL DELAY  // LED is ON/Off.
CPL P1.2           // Make P1.2 = 0
SJMP back          // short jump to back
END.                  lable.
```

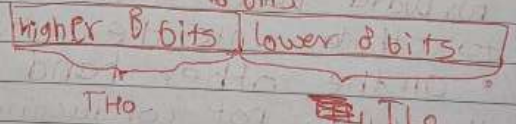
Timer, Counter, & Interuppt

list the no. of timer used in 8051 uc.
→ timer 0,
timer 1.

- 8051 has two counter / timer.

1) timer is used for to generate a time delay
2) Counter is used to count event happening outside the microcontroller.
3) Timer can be divided into two parts

T0 =



T1 =