UNIT-I

8-BIT EMBEDDED PROCESSOR

It is an Integrated chip designed under VLSI technique.

8-BIT MICRO CONTROLLER :-

↳ consists of processor with peripherals.

Introduction :-

→ To make complete microcomputer system, only μprocessor is not sufficient.

→ Add some peripherals like RAM, ROM, I/O devices, etc

→ But μcontroller incorporates all the features of μprocessor.

Adv of μcontroller :-

→ Smaller access time hence speed is high.

→ Reduce hardware due to single chip

→ Reduce PCB size

→ Increase Realiability.

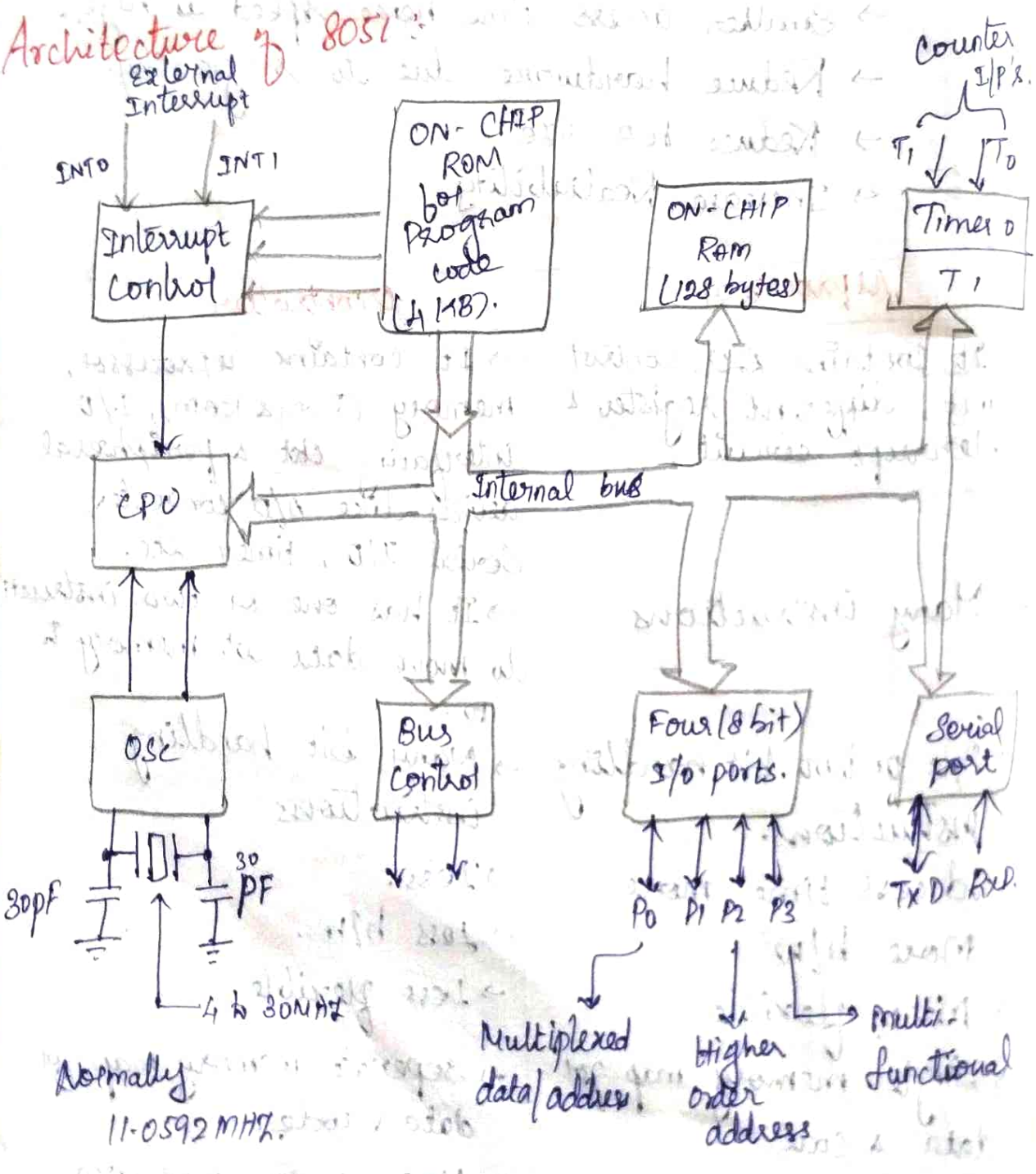
μprocessor	μcontroller
→ It contains ALU, control unit, different register & interrupt circuit.	→ It contains μprocessor, memory (ROM & RAM), I/O interfacing ckt & peripheral devices like A/D converter, Serial I/O, timer etc.
→ Many instructions	→ It has one or two instructi to move data bt memory & CPU.
→ One or two bit handling instructions.	→ Many bit handling instructions
→ Access time more	→ Less.
→ More h/w	→ Less h/w.
→ More flexible	→ Less flexible.
→ Single memory map for data & code	→ Seperate memory map for data & code.
→ Less no of pins are multifunctioned	→ More no. of pins are multifunctioned.

8 byte = 1 word
8 word = 1 frame

Features of 8051 microcontroller :

- → Program memory - 4096 bytes
- → Data memory - 128 bytes. 1 kB = 1024 bytes
- → 4 register banks.
- → 32 bidirectional I/O lines.
- → 16 bit timers/counters
- → 64 kB for each program & external RAM addressability.

Architecture of 8051 :



Normally 11.0592 MHz.

→ It consists of:

1) CPU,
2) Memory < Data memory (RAM)
Program memory (ROM / EPROM).
3) I/O ports.
4) Special function Registers
5) Control logic < Timer - Counts internal clk pulses to measure time
Counter - Counter counts external pulse to count events.
6) Interrupt functions. (8-bit bus).

→ Communicates through Internal data bus.

CPU :- →It's heart of recom.
→ Arithmetic & logic units.
→ Registers A, B, PSW, SP & 16 bit PC & DPTR (Data Pointer).
→ A Reg:
* 8-bit
* Accumulator
* It holds source operand
Δ Receive result of the arithmetic instructions.

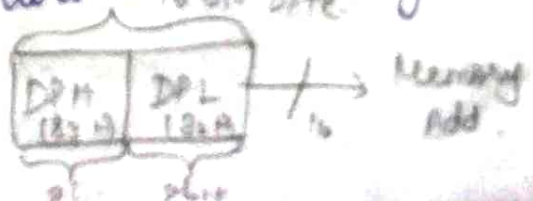
opcode
It is a single instruction that can be executed by the CPU
(operation code)

operand → data on address
It's single piece of data that can be operated by the op-code

Ex: Add a, b.
↓ ↓
opcode operands

B Reg :
→ In addition to accumulator, an 8-bit B-registers is available as a general purpose register.
→ Multiply / divide operation.

Data pointer :- < DPH
DPL
used to point to memory locations.
Access data memory (RAM) & Program memory (ROM) / EPROM
→ Used in LUT
hold → 16-bit address
→ 16 bit Reg or two 8 bit Reg.
→ DPTR does not have single internal address.



PC :-

→ 16-bit

→ Hold the address of memory location from which the next instruction is to be fetched.

8051 Flag Bits & PSW Register :- → stores the status of the CPU after operations

→ PSW also known as Flag register.

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
CY	AC	F0	RS1	RS0	OV	-	P

Functions :

1) Indicates result of arithmetic/logic operation
2) Select reg bank
3) stores Carry, overflow, parity information

CY- flag :-

→ This flag is set if there is an overflow out of bit 7.

→ It servers borrow flag for subtraction

AC- Auxiliary Carry Flag :- Used in BCD arithmetic

→ This flag is set if there is an overflow out of bit 3. i.e) carry from lower nibble to higher nibble (D_3 bit to D_4 bit).

F0 :- Available for user for general purpose → user defined flag

Register Bank Select (RS1 & RS0).

→ Select working Reg bank.

A - 10
B - 11
C - 12
D - 13
E - 14
F - 15

Addition.

```
  9B H   1001 1011
+ 75 H   0111 0101
-------  ---------
Carry 1  10 H   1  0001 0000
```

Subtraction

```
  89 H   1000 1001   1's comp → 0101 0110
  AB H   1010 1011   Add 1    → 0000 0001
-------  ---------              ---------
Borrow 1  DE H   1  1101 1110    0101 0111 → 2's comp
```

Because high value no sub in small value no

```
  1000 1001
  0101 0101
-------------
1 1101 1110
```

RS1	RS0	Bank selection
0	0	00H - 07H Bank 0.
0	1	08H - 0FH B1
1	0	10H - 17H B2
1	1	18H - 1FH B3.

Ex:-

```
  1 1 0 1          1 1 0 1
+ 1 0 1 1        + 1 0 1 1
-------          -------
1 0 0 0          1 0 0 0
```

```
  1 1 0 1          1 1 0 1
+ 1 0 0 1        + 1 0 0 1
-------          -------
1 0 1 1 0        1 0 1 1 0
```

```
  1 1 0 1          1 1 0 1
+ 1 0 0 1        + 1 0 0 1
-------          -------
1 0 1 1 0        1 0 1 1 0
+ 1 0 0 1        + 1 0 0 1
-------          -------
1 1 0 1 1        1 1 0 1 1
```

```
  1 1 0 1          1 1 0 1
+ 1 0 0 1        + 1 0 0 1
-------          -------
1 0 1 1 0        1 0 1 1 0
+ 1 0 0 1        + 1 0 0 1
-------          -------
1 1 0 1 1        1 1 0 1 1
+ 1 0 0 1        + 1 0 0 1
-------          -------
[1] 1 0 1 0      1 0 1 0 0
```

Over flow flag :- 8-bit signed no range -127 to 128.
8-bit unsigned " " 0 to 255.

Signed means - positive & -ve.
Unsigned " - positive values or zero.

→ OV flag is set - Result of signed no operation is too large. Set 1 - OF occured
cleared - Not OF

Ex:-
2 signed no. 100 & 50.

```
  0 1 1 0 0 1 0 0   (100)
+ 0 0 1 1 0 0 1 0   (50)
-------------------
  1 0 0 1 0 1 1 0   (150)
```

But 100 is outside the valid range. So overflow flag set.

P- Parity flag :- No of ones present in the accumulator.
P=0, if the no of ones are even
P=1, " " " " odd.

Ex: 98H & 65H addition Status of CY, AC, P flags.

```
        1 1 1 1 1 1 1
98H     1 0 0 1 1 0 1 1      Carry = 1
65H     0 1 1 0 0 1 0 1      AC = 1
      ---------------------
    1 | 0 0 0 0 0 0 0 0      P = 0
      ---------------------
      <-------------------
          Accumulator
```

MOV A, #05H = 0000 0101
two 1's
P=0 (even)
#07H

SFR [Special Function Register of 8051]

→ The group of registers, implemented to perform special functions & are located immediately above the 128 bytes of RAM are called SFR.

→ SFR control table used for running & monitoring the operation of microcontroller.

→ All access I/O ports, CPU registers, Interrupt control registers, the timer/counter, UART & power control are performed through registers bt 80H & FFH.

Pin Description of 8051:

→ 8051 - 40 pins.

→ 32 - I/O pin [4 -> 8 bit parallel ports P0, P1, P2, P3]
↓
Each pin configured as I/P (or) O/P (or) both bidirectional.

→ Each port consist of a latch, O/P driver & I/P buffer.

Port 0: (Pin 32-39)

↳ I/O pins.

→ O/P drives & I/P buffers of P0 are used to access external memory.

→ P0 O/P's low order byte of external memory address, time multiplexed wt data written or read.

→ P0 can be used as a multiplexed address/data bus

P1 [pins 1-8] → Used only as I/O pins.

P2 [Pins 21-28] → O/P drivers of P2 are used to access external memory.

→ P2 O/P's high order byte of external memory address when the address is 16 bits wide. otherwise, P2 is used as an I/O port.

P3 [Pins 10-17] → All pins of P3 are multifunctional

→ Each pin of P3 can be programmed to use as I/O (or) as alternate function.

Symbol	Position	Alternate Use
$\overline{RD}$	P3.7	External memory Read sig
$\overline{WR}$	P3.6	" " Write "
T1	P3.5	" timer 1 I/P.
T0	P3.4	" " 0 I/P.
$\overline{INT1}$	P3.3	" Interrupt 1 I/P.
$\overline{INT0}$	P3.2	" " 0 "
TXD	P3.1	serial data O/P.
RXD	P3.0	" " I/P.

Power Supply Pins V_{cc} (Pin 40) & V_{ss} (Pin 20):

→ 8051 operates on DC power supply +5V

→ V_{cc} - +ve Current - 125 mA

V_{ss} - Ground

Oscillator Pins XTAL2 [Pin 18) & XTAL1 (Pin 19):

→ Generating clock signal

ALE [Address Latch Enable, Pin 30]:

→ AD_0 to AD_7 lines are multiplexed.

RST (Reset, Pin 9): → Used to Reset

$\overline{PSEN}$ (Program Store Enable, Pin 29):

→ Used to activate, enable signal of external ROM/EPROM

→ It activate every six oscillator periods while reading the external memory.

$\overline{EA}$ [External Access, Pin 31] :-

→ $\overline{EA}$ pin high (Vcc) - program fetches addresses 0000H thru 0FFFH (internal ROM) & Program fetches 1000H through FFFFH (External ROM / EPROM)

$\overline{EA}$ pin low (Ground) - all addresses 0000H to FFFFH fetched by program from external ROM / EPROM

Internal & External Memories :-

→ 2 types of memory
1) Program memory - 64k
2) Data memory - 64k.

→ 8051 has program memory - 4k bytes. } Internal
data " - 256 bytes } memory

(diagram see books)

Internal RAM Organization :-

→ 8051 has 128-byte Internal RAM.

→ 3 areas
1) Register bank → 32 bytes address 00H to 1FH ↓ 4 RB
2) Bit addressable / Byte addressable ↓ 16 byte addressable area. RB0 B1 B2 B3.
3) General Purpose ↓
→ 30H to 7FH
→ addressable by byte
20H to 2FH
128 bits

ROM Space in the 8051 :-

→ 8051 has 4k byte - internal ROM
→ 0000H to 0FFFH.
→ Can't be erased or altered after fabrication.
→ Used to store final version of the program

Stack & Stack Pointer :

Stack - to store & retrieve data quickly.

top of stack - hold an internal RAM address.

8 bit

PUSH & CALL } increased before data is stored

POP & RET } decremented after data is restored

Instruction Set & Programming

8051 Addressing Mode :

Sources or destination addresses are specified in the instruction mnemonic for moving the data, is called addressing Mode. (R0-R7)

1) Register Addressing → 8 working Reg; instructions select one of the reg.
2) Direct byte Addressing → can access on-chip variable RAM & SFR.
3) Register indirect addressing → access data in dynamic manner → Ex: MOV A, @R0. ADD A, @R1.
4) Immediate Add → Source operand is constant, → indicates the sign # → Ex: MOV A, #52.
5) Register Specific
6) Index.
7) Stack addressing Mode.

Specific Reg such as Accumulator or DPTR.

Ex: SWAP A

Only Program memory can be accessed.

→ It's sub type of direct addressing Mode.
→ Stack instructions are used PUSH & POP.

Classification of Instruction set of 8051 :

1) Data Transfer Instructions → Tx data bt Internal RAM & SFR location without going through the accumulator
2) Byte Level Logical
3) Bit " "
4) Arithmetic ins
5) Jump & CALL Instru

Data Transfer Instructions :

→ Immediate
→ direct & Indirect
→ Register

} addressing modes are used in different MOVE instructions

Ex:- MOV A, Rn => MOV A, R0.

MOV A, direct.

MOV A, 30H.

Instructions to Access External Data Memory :

MOVX A, @Ri. → Copy the content of external address in Ri to A.

→ All External data moves with external RAM.

→ Address 8th byte & DPTR - 64kbytes

Instructions to Access External Program Memory :

→ access External ROM memory.

MOVC A, @A+DPTR.

Data Transfer with Stack (PUSH & POP) Instructions :

PUSH - Incremented by 1 (RB. B0 to B3).

POP - decremented by 1.

Data Exchange Instructions :

→ Move data from source to destination (or) vice versa.

XCH A, Rn.

Byte level Logical Instructions :-

→ ANL
→ ORL
→ XRL.

} there are using clearing (ANL) setting (ORL) complement (XRL)

→ Byte level operations use all four addressing modes for the source of data byte.

→ Directly addressed modes are used.

ANL A, Rn.

ORL A, Rn.

XRL A, Rn.

CLR A - clear Accumulator.

CPL A - Complement Accumulator

Arithmetic Instructions :

→ Increment - INC A

→ Decrement - DEC A.

→ Addition - ADD A, Rn.

→ Subtraction - SUBB A, Rn.

→ Multiplication - MUL AB
  - Multiply unsigned integers in A & B.
  - low order byte - A
  - Higher " " - B.
  - If the Product is greater than 255 (FFH), the ov flag is set, otherwise cleared
  - carry flag is always cleared

→ Division

→ Decimal operations

DIV AB
- A - receives quotient
- B - " remainder

DA A Decimal Adjust Accumulator for addition.

→ Adjust the Acc value from the earlier addition to produced packed BCD result

Bit Level Logical Instructions :

→ It's necessary to set or reset a particular bit in the internal RAM or SFRs.

→ Internal RAM addresses from 20H through 2FH is both byte & bit addressable.

→ Byte & bit addresses are different

CLR C - clear carry flag

CLR bit - clear direct bit

SETB C - Set carry flag.

SETB bit - Set direct bit.

ANL C, bit - Logical AND of bit variable and carry flag.

ANL C, /bit - AND direct bit to carry flag.

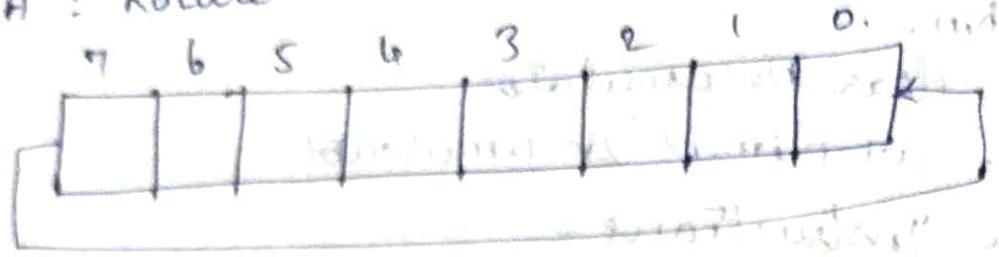
ANL C

ORL C, bit.

ANL C, bit

Rotate & Swap instructions:-

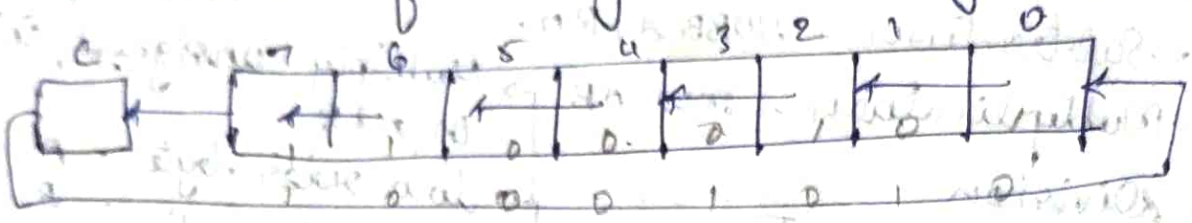
RL A : Rotate Accumulator left.



Ex: 1 1 0 0 0 1 0 1

1 0 0 0 1 0 1 1

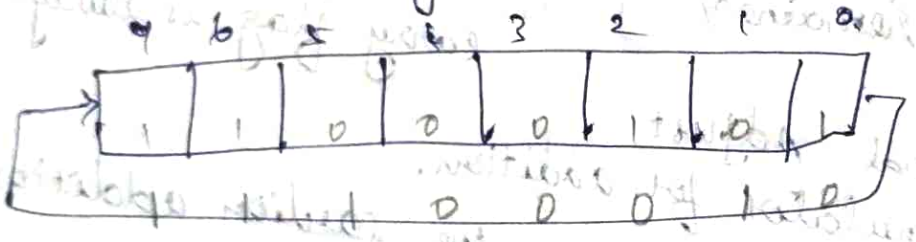
RLC A : Rotate Left through the carry flag.



Ex: 1 1 0 0 0 1 0 1

1 0 0 0 1 0 1 0 - carry flag is set.

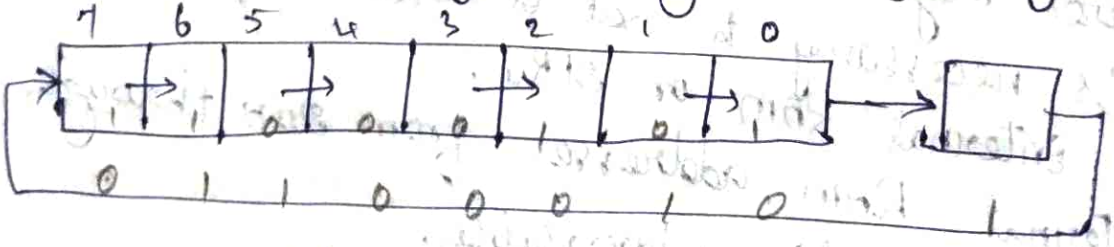
RRA : Rotate Acc Right.



1 1 0 0 0 1 0 1

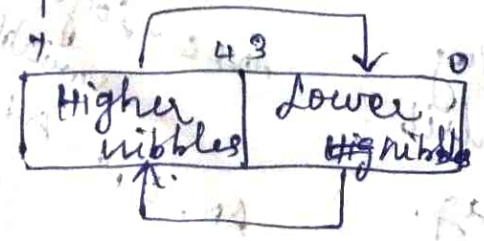
1 1 1 0 0 0 1 0

RRC A : Rotate A Right through carry flag.



Carry set.

SWAP A : Swap nibbles within the Accumulator.



HO LO

1 1 0 0 0 1 0 1

0 1 0 1 1 1 0 0

Jump and CALL Instructions :

→ change the flow of the program by changing the contents of PC.

→ Jump - permanently changes the program flow

call - Temporarily " "

→ CJNE (compare & Jump if not equal).

Jump & call Program Range :

→ Replace the content of PC wt new address.

→ The new address can be specified either by specifying the difference bt the new address & the current program counter contents

→ The difference of new address from the address in the PC is called the range of the jump or call.

Ex: Jump is located at Program address 0200H & jump causes the PC to become 0280H, then the range of jump is 80H bytes.

3 Ranges :-

Relative (short) range : +127 to -128 (7FH to -80H).

Absolute " : 0000H to 07FFH

Long " : 0000H to FFFFH.

Call & Subroutines :-

A Subroutine - call instructions.

LCALL ACALL

→ Each increments the PC to the 1st byte of the following instruction, then pushes it onto the Stack (low byte first).

→ SP increment by 2.

→ RET (Return) instruction pop the high & lower order bytes of the PC from stack.

→ Decrement by 2.

& ACALL addr11 : Absolute call.

→ 1st increments the PC twice.

→ then pushes to the stack. (lower order byte 1st).

LCALL addr16 : Long call.

RET : Return

RETI : Return Interrupt

Time Delay for 8051 :

→ Many times, it's necessary to generate time delays

→ Square wave generated time delay - T/2 period

O/P - 1 for T/2 period

O/P - 0 for T/2 period.

→ 8051 operating frequency : 1/12 of the crystal freq

→ One machine cycle lasts for 12 oscillator periods

$$\therefore \text{MC period} = \frac{12}{\text{Osc frequency}}$$

Crystal fre = 11.0592 MHz.

$$\text{MC period} = \frac{12}{11.0592 \times 10^6} = 1.085\ \mu s$$

Instruction	MC	Time to execute
MOV R2, #40	1	1 x 1.085 μs =
DJNZ R1, SHIFT	2	2 x 1.085 μs = 2.17
MOV AB	4	4 x 1.085 μs = 4.34 μs

Parallel Ports, Timers, Serial Port & Interrupts

Programming parallel ports:

8051 I/O Ports structure : (see book).

- → 32 I/O pins
- → Four 8 bit parallel ports (P0, P1, P2, & P3).
- → All ports are bidirectional
- → Each port consist of latch, an o/p driver & I/P buffer

P0 : (Pin 32 - 39) :

- → P0 pins can be used as I/O pins.
- → O/P drives & I/P buffers - access the external memory
- → P0 - used as multiplexed addr/data bus.

Programming I/O Ports using 8051 C

- → To include the file reg51.h.
- → This file contains all definitions of 8051 reg.
- → C compiler produces hex. file that can be downloaded into the ROM of the μcontroller.

reg51.h → file.

↓

C compiler produce O/P.

↓

download to ROM of μcontroller.

hex. file - produced by assembly language is small as compare to C compiler.

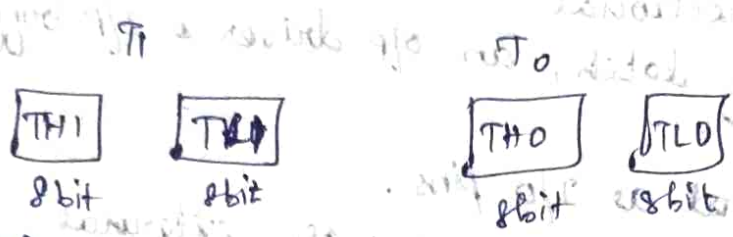
Why use C :-

- → It's easy compare to A.L.
- → C is more flexible ⇒ modify & update.
- → Available in fun libraries.
- → No modifications in μcontroller
- → It used in various data type like bit, float, signed char, unsigned char, SFR, Logical functions.

Operation	Assembly	In C	Example in C
NOT	CPL A	~	A = ~A;
AND	ANL A, #DATA	&	A = A & DATA;

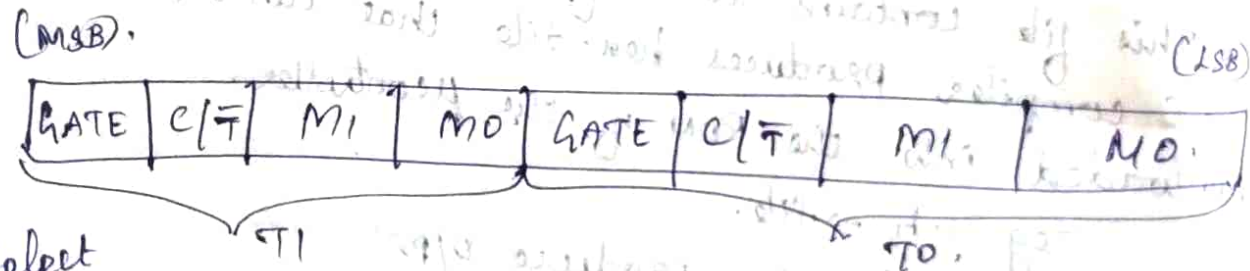
8051 Timers:

→ 2 timer
→ T0 & T1.
→ T0 & T1 are 16-bit reg
→ 16 bit ⇒ TH & TL



Timer registers :-
TCON - Timer control reg
TMOD - Timer mode reg.

TMOD Register :- Select mode ie in which timer should operate

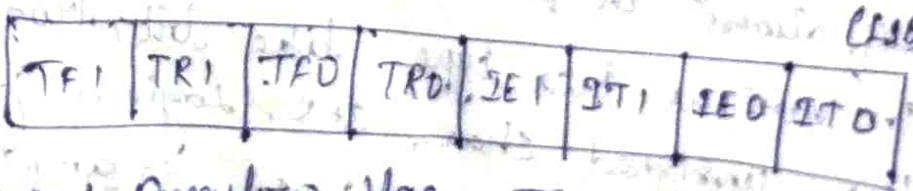


Select
↳ Mode - Timer / counter
Select T0 & T1 - act as timer / counter.
4 modes - M0, M1, M2, & M3.
C/T̄ ⇒ Bits clear (or) Set.

Bits clear - C/T̄ = 0 ⇒ Select Time operation
Bits Set = C/T̄ = 1 ⇒ Select Counter operation

M1, M0 - Select the timer mode
these bits are used

TCON Register :-



TF1 - Timer 1 Overflow flag TR1 - Timer 1 Run control bit
TF0 - " 0 " TR0 - Timer 0 " "

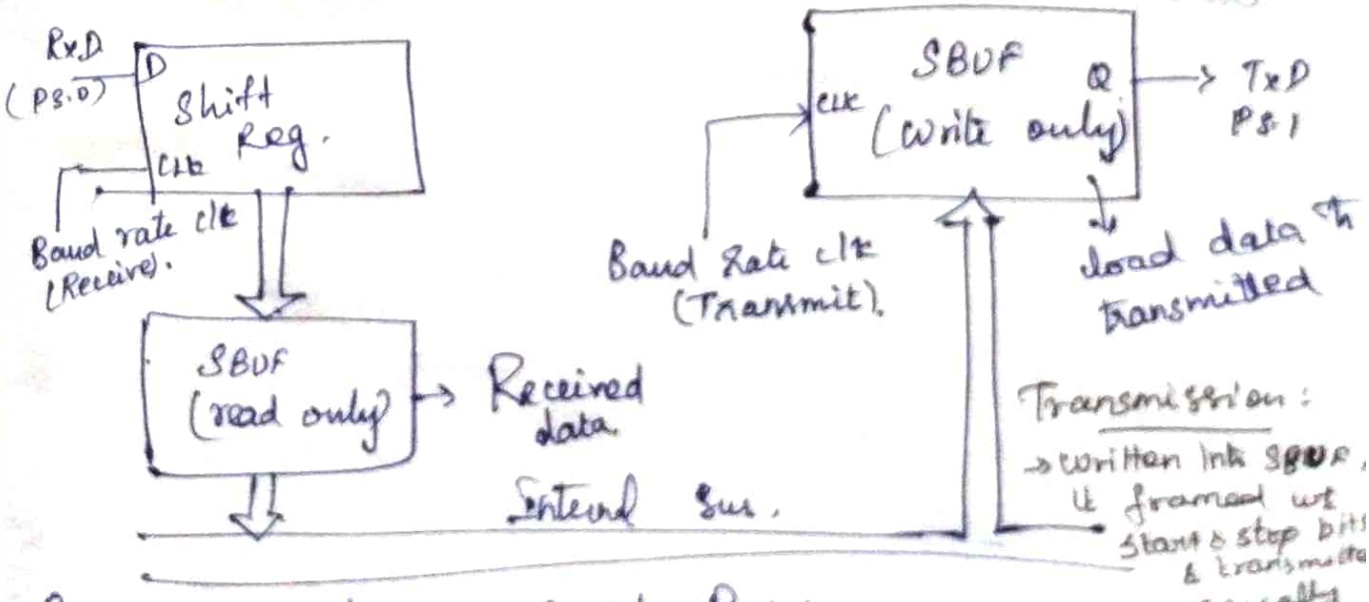
IE1 - Interrupt 1 Edge flag.
IE0 - " 0 " "
IT1 - " 1 Type control bit
IT0 - " 0 " " "

Operations :

→ Start & stop timer 0 & 1
→ Provides status of T/C overflows.
→ " " " External interrupts
→ Configure external interrupt as either low level triggered or falling edge triggered.

8051 Serial Port :

→ Full duplex.
→ Use Registers
SBUF - Hold data - 8bit reg.
SCON - Control data comm.
PCON - Control data rates
TXD - P3.1 } data transfer
RxD - P3.0 }



Operating Modes for Serial Port :-

Mode 0 :- enter & exits through RxD/TxD.
8 bits transmit / Received
Baud rate = 1/12 the oscillator fre

Mode 1:-

→ 10 bits transmit / receive.
→ Start bit & Stop bit – 8 bits (1SB)
→ Baud Rate variable.
→ Stop bit goes into RB8 in SCON.

Mode 2:-

→ 11 bits
→ start & stop bit, a programmable 9th data bit, & data bits.
→ Transmit 9th data bit in TB8 can be assigned value of 0 or 1.
→ Eg:- Parity bit moved to TB8, on receiving 9th data bit goes to RB8, while stop bit is ignored.
→ Baud Rate is either 1/32 or 1/64 oscillator fr

Mode 3:-

Same as Mode 2. Baud Rate is variable.

UNIT - II

Programming Embedded Systems in C :

→ Embedded C programming language is widely used for development of embedded sys.

→ It's a extension of C language.

→ Same syntax as C language
like main function
declaration of data types
defining variables
loop
functions
Statements etc.

→ It include h/w addressing, fixed pt arithmetic operations, etc.

→ Features of Embedded C :
1) Extension of C language.
2) Has source code format - kind of Mc or Mp.
3) High level optimization
4) Mc (or) Mp applications :
5) Embedded C pre defined programs can run
6) Ex of Embedded C app's - digital camera & digital TV etc
7) Adv : Coding speed & size is very simple. easy to understand.

Keywords in Embedded C :-

→ It's a special word
→ Special meaning to the compiler.
→ Ex. Keil compiler :

1) Bit
2) Sbit
3) Sfr
4) Small
5) large

} Keywords (or)

Data types in Embedded C :-

→ Declaring variables in the program.
→ Many data types :

1) Signed int.
2) Unsigned int
3) Signed char.
4) Unsigned char.
5) float
6) double, etc.

→ A variable is an addressable storage location to information. Variable - indicate size, type of info to be stored, name to be used to refer
→ Automatic or static.
→ Space for variables allocated in Reg, Ram or Rom / Flash.
→ Variable array :

Need for RTOS :-

→ Real time sy is inform processing sy
→ Respond to externally generated I/P stimuli within a finite & specific period.
→ Time constraints is key parameter
↓
satellite
Robots
traffic control

→ Determinism :
→ Produce same o/p for known I/P.
Non deterministic sy => o/p random variable.

→ Deadline :-
→ It's finite window of time (certain task must be completed)

→ QoS :-
Overall peryomance of nt
BW, throughput, availability, lattency, etc.

→ Real time fail - does not give result.
good - produce result.

→ Real time systems - Overall correctness of the sy
functional correctness &
timing correctness.

Two types :-
1) Hard R.T → Critical task completed in a time
↓
does not support by OS
for hard real time sy
2) Soft R.T.
↓
Time can't guarantee meet deadline under all conditions.
Ex:- digital telephone
digital audio

R.T can't waiting for longer time without allocating kernel.

→ RTOS use priority scheduling algorithm

Sn:

Transportation - Air traffic control & Traffic light sy

Communication - digital telephone.

Process control - Petroleum & paper m&u.

Detection - Burglar sy & Radar sy

Flight simulator - Auto pilot shuttle mission simulator.

QoS:- (Good RTOS).

Performance - fast & high throughput

Reliability - RTOS should not fail.

Compactness - Size of OS small.

Scalability - Add & Remove

Characteristics of RTOS :-

1) Determinism
2) Responsiveness
3) User control
4) Reliability
5) Fail-Soft operation.

Choose RTOS :-

→ Functional - Processor support.

→ Non-Functional - cost

Multiple Task & Process :-

Task :- Sequential code implementing the sy actions executed

thread.

↓

Smallest unit of execution

Process - Sequential program execution

EDF :-

→ Best algorithm in real time processing

→ It's an optimal dynamic algo

→ Dynamic algorithms - priority of task can be change during its execution

↓

Produce valid Schedule whenever one exist.

→ EDF - preemptive scheduling.

→ Task with a shorter deadline has a higher priority.

→ Whenever new task arrive, → end of the period, it give highest priority

→ Two task arrives same deadline.

↓

choose one of the two at random

↓

priority is dynamic. since it changes for different jobs of the same task.

T_3 start t_0 & lock semaphore S at t_1

t_2 - T1 arrives & preempts T_3 inside critical section.

T_1 - request to use resources, but it gets blocked. as is currently using it.

↓

t_3 - continuously execute inside the C.S

Next, T_2 arrive at t_4, } highest priority.
it preempts T_3.

T_2 - execution time increases &
blocked the time of T1