MICRO CONTROLLER

UNIT-I :

Introduction, comparison of microprocessor and micro controller, Evaluation of micro controllers from 4 bit to 32-bit, Development tools for micro controllers, Assembler-compiler-Simulator/Debugger.

UNIT-II : Microcontroller Architecture :

Overview and block diagram of 8051, program counter and memory organization, Data types and directives, psw register, Register banks and stack, pin diagram of 8051, interrupts and timers.

UNIT-III : Addressing modes, instruction set of 8051:

Addressing modes and accessing memory using various addressing modes, instruction set : Arithmetic, logical, simple bit, jump, loop and call instructions.

UNIT-IV : Assemble language programming

Addition, multiplication, subtraction, divison, arranging a given set of numbers in largest/smallest order, Timer/counter programming.

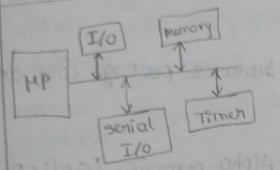
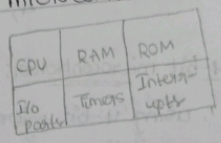
UNIT-V : Interfacing and Application of microcontroller:

Interfacing of ppi 8255, interfacing seven segment displays, displaying information on a LCD, control of a stepper motor (uni-polar).

Microcontroller:-

microcontroller is a programmable IC. It contains input and output ports, cpu, memory inbuilt. And it is mainly used for special purpose applications.

Difference blw microprocessor and microcontroller:-

Microprocessor	microcontroller
	
→ microprocessor is called off chip.	→ microcontroller is called on chip.
→ It's designing and hardware cost is high because microprocessor is used for multiple tasks.	→ It's designing and hardware cost is low because microcontroller is used for single task.
→ It is not easy to replace.	→ It is easy to replace.
→ Large number of instructions with flexible Addressing modes.	→ limited no. of instructions with few Addressing modes.
→ less no. of pins are multifunctional.	→ More no. of pins are multifunctional.
→ Access time for memory and I/O is more.	→ less access time for built in memory and I/O.
Eg:- Industrial applications.	Eg:- ATM, washing machine etc.
→ ROM for programming code. → RAM for data.	

Evaluation of microcontroller from 4 bit to 32 bit :-

Microcontroller can be classified into various catagories based on bits Architecture, based on memory locations, based on instructions set.

→ Based on bits, microcontrollers are classified into 4 catagories :- 1, 4 bit microcontroller
2, 8 bit "
3, 16 bit "
4, 32 bit "

4-bit microcontroller :-

1, using 4-bit microcontroller reduces package cost and pin count to a minimum.
2, these microcontrollers used in Alpha-numeric-LED/LCD displays portable battery changes.
3, 4-bit microcontrollers from different manufactures are :-

Model	I/O	pins	RAM	ROM	counters	Extra features
COP400 (National)	23	28	64	1k	1	Serial bit I/O
HMCS40 (Hitachi)	10	28	32	512	—	10 bit ROM
TMS1000 (Texas)	23	28	64	1k	—	LED Displays.

8-bit microcontroller :-

1, 8-bit microcontroller used in Arithmetic and logical operations.
2, 8-bit microcontrollers are more popular microcontrollers in used today.

3, capable of 256 decimal values, of 1 byte data word.

Model	I/O	pins	RAM	ROM	counters	Extra features
8048 (Intel)	27	40	64	1K	1	8K external memory
8051 (Intel)	32	40	128	4K	2	128K External memory, serial port
COP800 family (national)	24	28	64	1K	1	Serial bit I/O, 8-channel A/D converter
6805	20	28	64	1K	1	PLL frequency synthesizer
68HC11 (motorola)	48	52	256	8K	2	A/D, PWM
TMS370 (TEXAS)	55	68	256	4K	2	watchdog timer, serial port, A/D (8bit, 8channel)
PIC (microchip)	12	18	25	1K	0	Small pin count, very slow power consumption

16-bit microcontroller:-

1, 16-bit microcontroller works like a 8-bit microcontroller, but with greatly increased memory speed and size.

2, these microcontrollers use high level languages.

Applications:- Calculation and data sensitive and include Disk drives, modems, printers, scanners, pattern recognization, Automotive and sonometer control.

Model	I/O	pins	RAM	ROM	counters	Extra features
80C196 (Intel)	40	68	232x8K	2		PWM generator, watchdog timer.
HPC family (national)	52	68	512 x 16K	4		PWM generator, watchdog timer, 8-channel I A/D, serial port.

32-bit microcontroller:

1, these microcontrollers are used high end applications like Automotive control, communication network, Robotics, $cellphones, GPRS etc.

2, PIC 32, ARM7, ARM9, SHARP, LH79520, ATMEL 32 (AVR), Texas instruments, TMS 320F2802x/2803 etc. are some of the 32 bit microcontrollers.

Model	pins	RAM	UARTS	counters	I/O pins
sharp: LH79520	176	32K	3	4	64

Development tools for microcontroller:-

The basic development tools used for microcontroller based systems are :-1, software development tools
2, Hardware development tools

→ The software development tools are :, Assemblers 2, editors 3, compilers 4, simulators 5, debuggers 6, IDE (integrated development environment) → Final output

Editor:-

→ It is the first tool.

→ It is a program that allows the programmer to enter and edit our programs.

→ It helps the user to create a file that contains the Assembly language statement or high level language..

→ the job of the Editor is to store the ASCII codes for the letter and numbers in the successful RAM locations and this file is called source file.

Assembler :-

→ Each Assembly level instruction has a mnemonic for example:-
MOV A,C.

→ For every mnemonic manufacturers designs op-codes, the op-codes are written in hexadecimal numbers

→ An Assembler is a program which translate the Assembly language mnemonic into corresponding binary codes.

→ the Assembler reads the source file more than once.

Assembler operations :-

→ The Assembler first reads the source file of the program. then it determines the displacement of data items, offset of labels and puts this information into a symbolic table. Then it produces the binary codes for each Assembly language instructions.

Compiler :-

→ compiler is a program that translate the high level language source program into a machine level language program. that is source code to object code.

→ A compiler translates the source program into relocatable object modules.

→ the object modules are linked by the linker. locator loads the complete program in memory where it can be executed.

Debugger :-

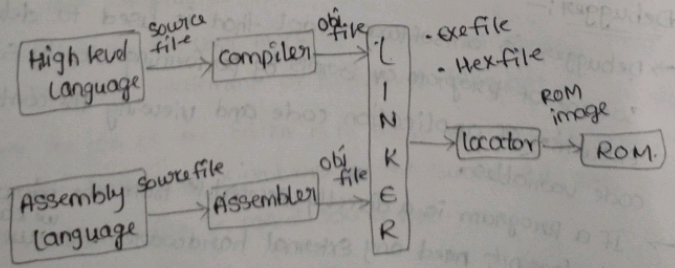
→ Debugger is a software tool that is used to detect the source of program or errors by performing step by step execution of application code and viewing the content of code variables.

→ If a program is a directly accessible from the microcomputer and doesn't need any external hardware then we can debugger to run and test the program

→ Debugger is basically a program which permits the user to load object code program into the system memory and execute the program and debug it.
→ the debugger also permits the change in register content memory locations, and re run the program.
→ we can use the debugger to check and correct the program till all the errors are corrected.

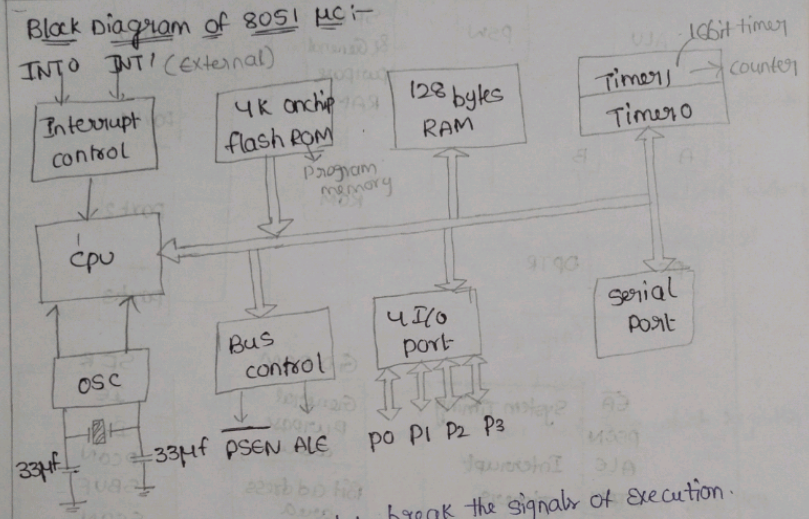
Simulators:-

→ It will give idea about how output comes.
→ A simulator is often used to execute a program that has to run on some convinient type of computer.
→ It helps to see how the code will work in real life.
→ Simulator is a software package that functions like hardware without acquiring hardware. the 8051 microcontroller simulator gives the use of an 8051 environment on the P.C.
→ It also performs simulation of different peripherals that are used with microprocessor or microcontroller in any application. It provides the facilities that help the user to find logical errors and learn about the initialization of different peripherals.
→ It shows all the internal registers, entire memory and peripherals on the monitor.



Microcontroller Architecture

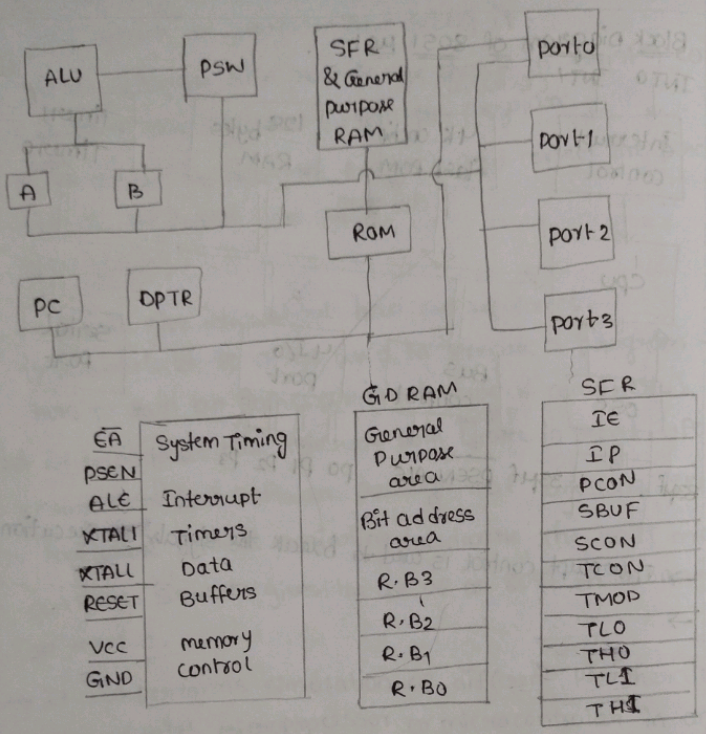
Block Diagram of 8051 MC :-



→ Interrupt control is used to break the signals of execution.

→

8051 Architecture:-



→ 8051 supports multiplication and divison also.

→ psw (programmable status word) which contains flag registers.

→ 'B' is a register used for performing only multiplication and division. (MUL) (DIV)

→ 'PC' (program counter) which is used to store the address of next instruction to be executed.

→ DPTR (Data pointer) which stores address. 1, DPH
2, DPL.

→ SFR (special function register).

→ Input and output ports are :- port 0, 1, 2, 3 contains 8 pins.

→ 32 pins used for input and output.

→ These ports multipurpose ports.

→ ALE seperate the Address and data in 8085 and 8051.
→ If ALE is high Address is identified and ALE is low data is identified.
→ And ports contains both Address and data bits (AD_0-AD_7).
→ General purpose area contains 80 byte used to store data temporarily.
→ Bit Address area contains 16 byte used to store bits and Address.
→ $R.B_3$, $R.B_2$, $R.B_1$, $R.B_0$ are called Register banks (collection of registers).
→ R.B contains R_0 to R_7 register (8 bit length).
→ R.B contains 32 byte (8 bytes × 4 = 32 bytes).
→ psw used to identify which registers are used from which register bank.
→ psw contains 8 bit out of 2 bits are used to select register bank.

0 0	RB_0
0 1	RB_1
1 0	RB_2
1 1	RB_3

It contains flags :- 1, zero flag
2, carry flag
3, Sign flag
4, overflow flag

→ TCON (Timer control) used to find timer either run or not.
→ TMOD (Timer modes) used to find the timer how it acts 8, 16, 32 ...
→ To contains TH0 (higher), TL0 (lower).
→ T1 contains TH1 (higher), TL1 (lower).
→ SCON (Serial control) used to control transmit and receive data.
→ SBuf (Serial Buffers) used to store transmitting and receiving data.
→ PCON (power control) used to control power.
→ IE (Interrupt Enable) used to find either it enable or disable.
→ IP (Interrupt priority) used to find the Interrupt out of 5.

→ RESET is used to Resets the Address of program counter.

→ XTAL1, XTAL2 used to connect to crystal oscillator then it generate clock frequency. (11.0592 M)

→ PSEN (Program store Enable) used to read data from External memory.

→ EA (External Access) is used to connect memory externally called secondary memory.

ALU:-

→ It performs Arithmetic and logical operations are Addition, Subtraction, multiplication, Division

→ logical operations like AND, OR, NOT, complement, EXOR.

Bus control:-

8051 contains mainly two types of buses :- 1, Data 2, Address.

→ Data bus is used to transfer 8 bit data i.e input and output data.

→ Address bus is used to store the 16 bit memory location or Address location.

I/O ports:-

→ There are mainly 4 ports. each ports length is 8 bit.

→ port 0, port 2 act as general purpose I/O ports, if External memory present then it act as multiplex Address and data bus.

→ port 1 is only used as I/O port, it does not perform any dual function.

→ port 3 behaves as a dedicated I/O port that means each pin in this port has specific function.

Timers:-

→ 8051 contains two 16 bit timers, it also acts as counters.

→ Internal operation can be synchronised using clock circuit.

Serial port:-

→ serial port performs serial communication, it contains SBuf and SCON special function registers.

1, SBuf holds the transmit and receiving data.
2, SCON manages data communication.

Interrupt control logic:-

→ An interrupt can be come from internally or externally.
→ there are 2 ways of giving interrupt to the μc :-
  1, By sending software interrupt
  2, By sending hardware interrupt
→ There are 5 interrupts in 8051 :- 1, INT0, INT1, serial port interrupt, Timer flag zero, Timer flag 1.

Program counter:-

→ It is the 16 bit register, it contains Address of next instruction to be executed.
→ on reset condition pc will set to 0000H, after fetching every instruction program counter will increment by '1'.

Data pointer:-

→ It is used for storing 16 bit values of Address information for internal and external program memory and for external data memory.

Special function register:-

→ These are used to control timers, serial port, input and output ports and interrupts.

Accumulator:-

→ Accumulator performs the arithmetic and logical operations and stores the final result.
→ It is also used to transfer data b/w External memory.

B register:-

→ It is the temporary register, it is used along with A for multiplication and division. eg:- MUL AB, DIV AB

RAM:-

→ It supports the 128 bytes of internal RAM.

→ The 128 bytes of RAM is divided into 80 bytes of general purpose area, 16 bytes of bit Addressable Area and 32 bytes of register banks RB0, RB1, RB2, RB3

ROM:-

→ 8051 supports 4 KB memory.

→ It can address program memory as well as 64 KB data memory

PSW:-

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	-	P

→ Program status word is a bit Addressable 8 bit register that has all the flags which contains the Arithmetic status of ALU and bank select bits of register banks.

carry flag:-

→ This flag is set whenever there is a carry from 8 bit Addition otherwise Reset.

Auxillary carry flag:-

→ If there is a flag carry from Nibble number of bits during addition or subtraction operation this bit is set otherwise Reset.

Flag '0':-

→ It is available to the user for general purpose.

RS1, RS0:- (Register select)

→ These are used to change the bank registers, this bits set by the software to determine which register bank is used.

RS1	RS0	selected bank
0	0	RB0
0	1	RB1
1	0	RB2
1	1	RB3

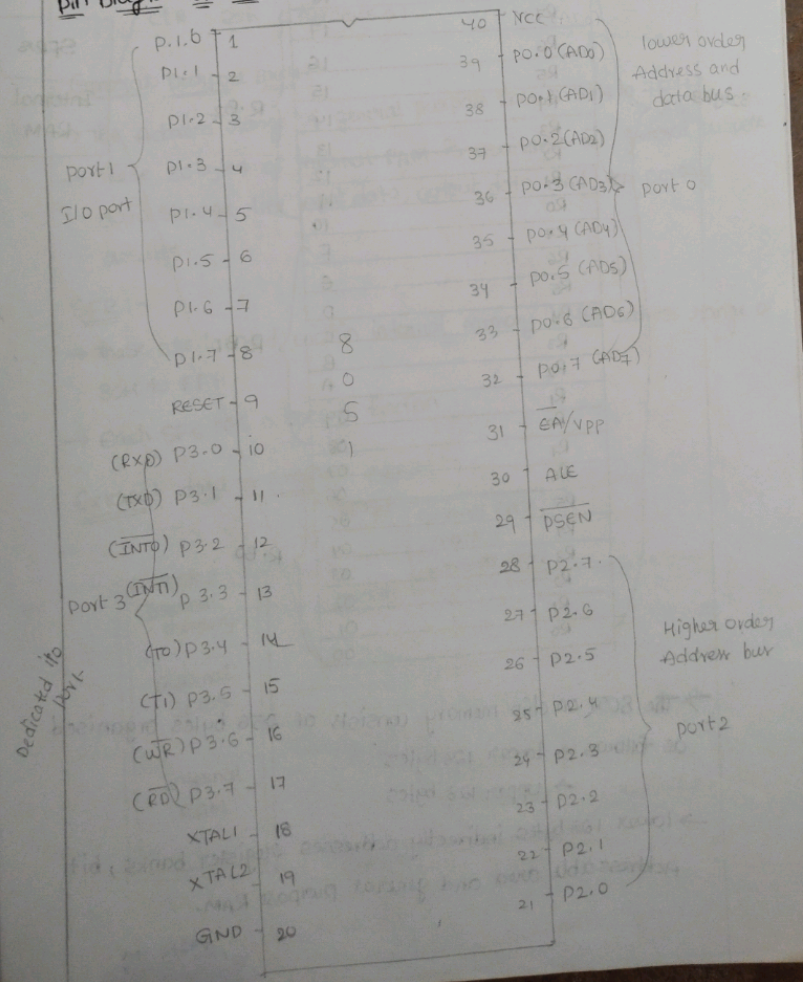
overflow:-

→ this flag is set whenever the result of a signed number operation is too large causing the higher order bit to overflow into sign bit.

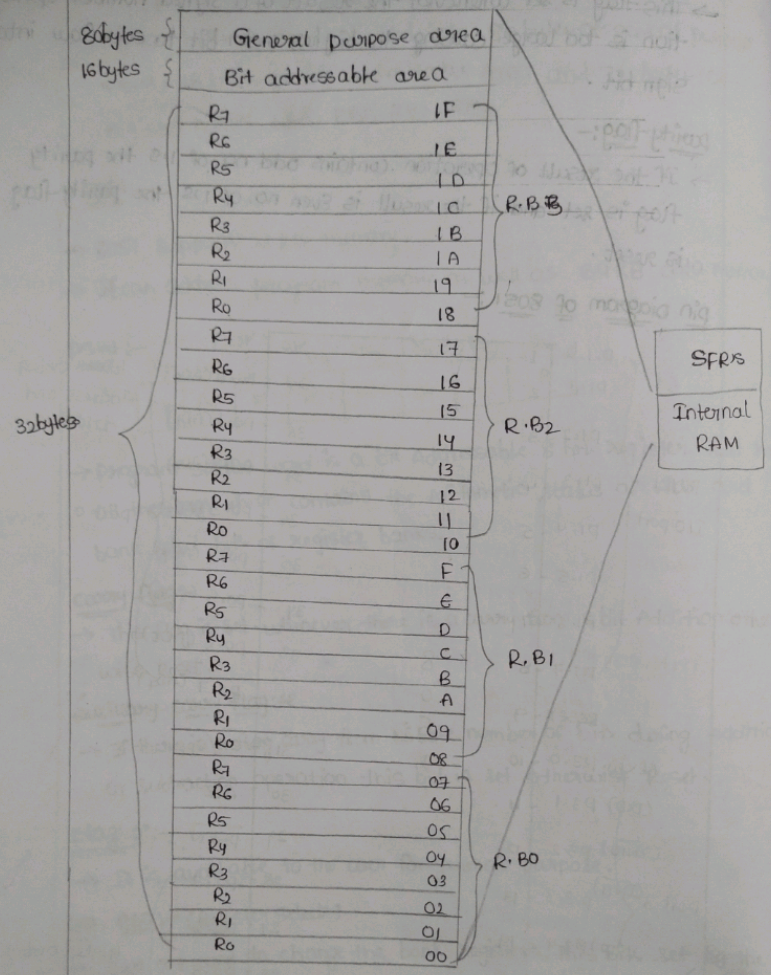
parity flag:-

→ If the result of operation contains odd no. of 1's the parity flag is set and if the result is even no. of 1's the parity flag is reset.

pin diagram of 8051 :-



Memory Organisation :- (Internal memory)



→ The 8051 on chip memory consists of 256 bytes organised as follows → lower 128 bytes
→ upper 128 bytes

→ lower 128 bytes indirectly addresses register banks, bit Addressable area and general purpose RAM.

→ Upper 128 bytes directly addresses special function registers.

Bit Addressable Area :-

→ The 8051 supports a special feature which allows access to bit variable i.e. the individual memory bits in internal RAM can be set or cleared.

→ A bit variable can be set with a command such as SETB, and cleared with a command such as CLR.

Eg :- SETB 25h (Becomes 1) ⇒ set the bit of 25h.
CLR 25h (Becomes 0)

General purpose area :-

→ The Address range of general purpose RAM is 30H to 7FH.

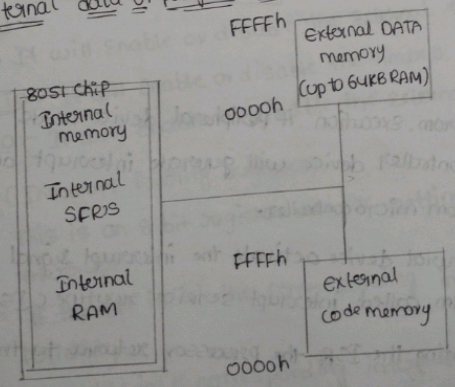
→ These 80 bytes of internal RAM is available for general purpose data storage like input data, output data and temporary results.

SFR :-

→ These are located within internal memory in the address range of 80H to FFH.

→ Each SFR has a specific function

External data or program memory :-



SFRs
Internal RAM

organised

nks, bit

External code memory :-

→ the executable program code is stored in this code memory this code memory size is limited to 64KB.

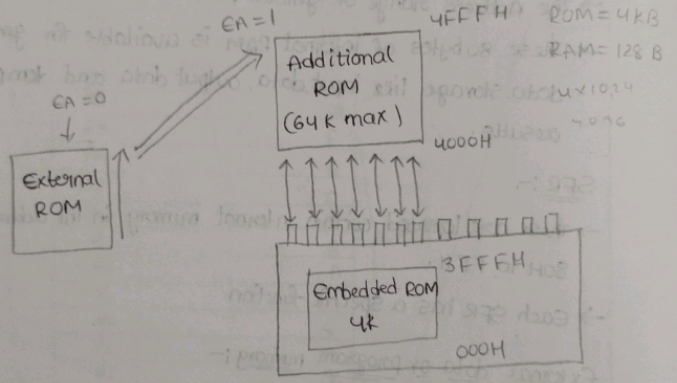
→ the code memory is read only memory in normal operation and is programmed under special conditions.

External RAM data memory :-

→ this is available for storage of data upto 64KB of external RAM data memory is supported.

→ this RAM is accessed by using the DPTR & movX registers.

→ Microcontroller can handle external memory depends upon the pin 'EA' logical state



Interrupts :-

→ During program execution if peripheral device needs service from microcontroller device will generate interrupt and gets the service from microcontroller.

→ when peripheral device activate the interrupt signal branches to a program called interrupt service routine (ISR).

→ After executing the ISR the processor returns to the main-program.

8051 interrupt structure :-

→ 8051 have 5 interrupts out of these 5, 2 are external interrupts and 3 are internal interrupts and the interrupts are external Int 0, external int 1, internal timer 0 interrupt, internal timer 1 interrupt, internal serial interrupt.

IE (Interrupt enable, bit addressable) If EA=1, remaining are either enable or disable. EA=0, all are disable.

IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
EA	-	-	ES	ET1	EX1	ET0	EX0

→ this is an 8-bit register used for Enabling or disabling the interrupts

EA :-

→ If it is '0', it disables the all interrupts, if EA=1, each interrupt is individually enable or disable by setting or clearing its enable bit.

IE6, IE5 :-

→ these 2 bits are reserved for future purpose.

ES :-

→ It will enable or disable the serial port interrupt.

ET1 :-

→ It will enable or disable the timer 1 interrupt.

EX1 :-

→ It will enable or disable the external interrupt 1.

ET0 :- It will enable or disable the timer 0 interrupt.

EX0 :- It will enable or disable the external interrupt 0.

IP (Interrupt Priority register) :-

→ this is an 8 bit register used for setting the priority of interrupt.

→ If the bit is '0' the corresponding interrupt has a lowest priority

→ If the bit is '1' corresponding interrupt has a highest priority.

IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
-	-	-	PS	PT1	PX1	PT0	PX0

only one bit is set & rest remaining all are reset.

→ PS → It define the serial port priority interrupt.

→ these are reserved for future purpose. (IP7, IP6, IP5)

IP4: PS:- defines the serial port interrupt priority level.

IP3: PT1:- defines timer1 interrupt priority level.

IP2: PX1:- defines external interrupt 1 priority level.

IP1: PT0:- defines the interrupt level of timer 0.

IP0: PX0:- defines the external interrupt 0 priority level.

Timers:-

→ Timers are generally used for:-

1, Time reference
2, creating delay
3, waveforms, properties, measurements
4, periodic interrupt generation
5, waveform generation.

→ 8051 has 2 timers :- timer 0, timer 1.

→ Timer in 8051 is used as timer, counter and baudrate generator.

→ Timer is always incremented by (1) and the time taken to count one digit up is based on master clock frequency. (No. of instruction execute in 1 sec)

for eg:- master clock frequency = 12MHz, timer clock frequency =

$$\text{Timer clock frequency} = \frac{\text{master clock frequency}}{12}$$

If mcf = 12,

$$TCF = \frac{12}{12} = 1\,MHz$$

$$\text{Time period} = \frac{1}{\text{frequency}}$$

$$= \frac{1}{1\times10^{6}}$$

$$= 10^{-6}\,s$$

∴ Time period = 1 μs.

→ this will
→ the 2 tim
controls
TL.
TMOD regis
Ga
Gate:-
→ Whe
will run
→ when
(soft
C/T
→ this
inte
→ se
M1, M
→ th

→ This will indicate 1 increment in count will take 1 μs.

→ The 2 timers in 8051 share 2 SFRs (TMOD, TCON) which controls the timers and each timer has 2 SFRs like TH and TL.

TMOD registers:- (Timer/counter mode control)

Gate	C/T	M1	M0	Gate	C/T	M1	M0
← Timer 1 →				← Timer 0 →			

Gate:- (Timer run)

→ When TRx(0,1) (in TCON) is set and gate=1, timer or counter will run only while Intx(0,1) pin is high (Hardware control).

→ when gate=0, timer or counter will run only while TRx(0,1)=1. (Software control).

→ ~~set~~

C/T̄ :-

→ This timer/counter selector. cleared for timer operation (input from internal system clock).

→ set for counter operation.

M1, M0 :-

→ These are mode selection bits

M1	M0	Mode	operating mode
0	0	0	13-bit timer (Actually 16 bit timer)
0	1	1	16-bit Timer/counter
1	0	2	8-bit Auto-Reload timer
1	1	3	(Timer 0) TL0 is an 8 bit timer or counter controlled by the standard timer 0 control bits. TH0 is an 8 bit timer and is controlled by timer 1 control bits.

TCON register:- (Timer/counter control)
(2 internal interrupts)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1; TF0:-

→ Timer 1,0 overflowed flag set by the hardware when timer/counter overflows ~~exact~~ cleared by the hardware as processor vectors to the ISR (interrupt service routine).

→ TR1, TR0:-

→ Timer 1,0 run controlled bit set or cleared by the software to turn timer or counter on and off.

IE1, IE0:-

→ these are external interrupt 1,0 edge flag set when External interrupt edge is detected, cleared when interrupt is processor. (Execute)

IT1, IT0:- (used to detect whether the IE1, IE0 are set or not).

→ Interrupt 1,0 type control bit, set or cleared by the software to specify low level trigger external interrupt

Addressing modes of 8051 :-

→ 8051 provides a total of 5 addressing modes they are :-

1, Immediate addressing mode.
2, Direct addressing mode.
3, Register indirect A.M.
4, Register A.M
5, Index A.M.

Immediate A.M :-

In this A.M the source operand is constant. Immediate data must proceeded by the sign '#'. this can be used to load the information into any of the registers.

Eg :- MOV A, #25H
MOV R4, #62H.

MOV A, #62 (without # it is Address.)
↓
Data

Direct Addressing mode :-

In 8051, RAM storage in 128 bytes this mode mostly used to Access RAM location.

Eg :- MOV R0, 40H
MOV 56H, A.

Register Indirect A.M :-

In this, register is used as pointer to the data, only reg. R0, R1 used in this purpose.

Eg :- MOV A, @R1
MOV A, @R0.

Register A.M :-

In this, register is used as holding the data.

Eg :- MOV A, Ri (i = 0 to 7)
MOV Ri, A
ADD A, Ri

Indexed A.M:-

this mode is used in accessing data elements in program code ROM at 8051 in this program counter or DPTR is used to hold base Address, Accumulator is used to hold offset Address, adding these 2 values forms the effective Address, this Addressing mode is used with JMP, MOVC.

Eg:- MOVC A, @A+DPTR.

Here MOVC is the MOVe code which moves the data from External code memory space.

Instruction set:-

there are 5 types of instructions set:-

1. Data transfer
2. Arithmetic
3. logical
4. Branch
5. Bit manipulation

Data transfer:-

In this group, the instructions perform data transfer operations of the following types:-

a, Move the contents of a register Rn to A. where n=0 to 7.

Eg:- MOV A, R_2
MOV A, R_3

b, Move the contents of a register A to Rn where n=0 to 7.

Eg:- MOV R_2, A
MOV R_3, A.

c, Move an immediate 8 bit to register A or Rn to a memory location.

d, Eg:- MOV A, #45H
MOV R_5, #54H.
MOV 30H, #44H.
MOV DPTR, 0F52 H.

d) Move the content of a memory location to Rn or Rn to a memory location using Direct Addressing mode.

Eg:- MOV R3, 65H
MOV 45H, R2.

e) Move the contents of memory location to another memory location using direct and indirect A.M.

Eg:- MOV 45H, 54H.
MOV 45H, @R0.

f) Move the contents of External memory to Accumulator or Accumulator to external memory.

Eg:- MOVX A, @R1
MOVX @R0, A
MOVX A, @DPTR
MOVX @DPTR, A.

g) Move the contents of program memory to the Accumulator.

Eg:- MOVC A, @A+PC
MOVC A, @A+DPTR.

Arithmetic instructions:-

→ the 8051 can perform addition, subtraction, multiplication, Devision, increment and decrement operations on 8-bit number.

Addition:-

→ In these we have instructions to add the contents of A with immediate data or with register with or without carry.
→ using direct and indirect addressing

Eg:- ADD A, #45H
ADDC A, #54H
ADD A, R2
ADD A, 51H
ADD A, @R1
ADDC A, @R0

Subtraction:-

→ In this group, we have instructions to subtract the contents of A with immediate data or register with or without carry using direct and indirect A.M.

Eg:- SUBB A, #45H
SUBB A, #54H
SUBB A, R2
SUBB A, 51H
SUBB A, @R1
SUBB A, @R0.

Multiplication:- (MUL AB)

This instruction multiplies 2 8-bit unsigned numbers which are stored in A and B registers. After multiplication the lower byte of the result will be stored in Accumulator and higher byte of the result will be stored in B register.

Eg:- MUL A, B

Division:- (DIV AB)

This instruction divides 8 bit unsigned number which is stored in A by the 8-bit unsigned number which is stored in B register. Quotient is stored in A and remainder stored in B.

Eg:- DIV AB

Increment/dec:-

This instruction increments/decrements the register or Address by 1.

Eg:- Inc A
Inc Rn (n=0 to 7)
Inc @Ri [i=0, 1]
Inc DPTR

Eg:- DEC A
DEC Rn
DEC @Ri
DEC DPTR

Logical Instructions:-

Logical AND:-

Mnemonic : ANL Destination, source.

The ANL does a bit wise AND operation between source and Destination leaving the result value in destination.

Eg:- ANL A, # data
ANL A, Rn
ANL A, @ Ri
ANL Direct address, # data.

Logical OR :-

Mnemonic : ORL Destination, source;

ORL does a bitwise OR operation between source & destination leaving the value in destination.

Eg:- ORL A, #, # data
ORL A, Rn
ORL A, @Ri
ORL Direct Address, # data.

Logical XOR :-

Mnemonic : XRL Destination, source.

XRL does a bitwise XOR operation between source & destination leaving the value in destination.

Eg:- XRL A, # data
XRL A, Rn
XRL A, @Ri
XRL Direct Address, # data.

logical complement (NOT):-

Memonic : CPL operand

It complements the operand leaving the result in operand.

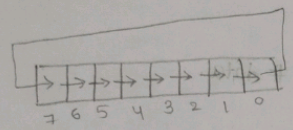
If operand is a single bit then state of the bit will be reversed, if operand is accumulator then all the bits in accumulator

will be reversed.

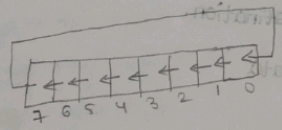
Eg:- CPL A

Rotate instructions:-

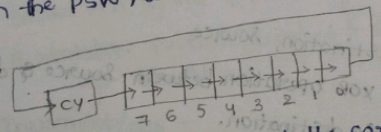
RRA :- Rotate right accumulator. this instruction is rotate right the accumulator its operation is like below each bit is shifted one location to the right with bit '0' to '7'.



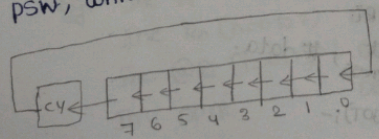
RLA :- Rotate left accumulator. this instruction is rotate left the accumulator its operation is like below each bit is shifted one location to the left with bit '7' to '0'.



RRCA :- Rotate right through the carry. Each bit is shifted one location to the right with bit '0' going into carry position in the PSW, while the carry goes into bit '7'



RLCA :- Rotate left through the carry. Each bit is shifted one location to the left carry position is going into bit '0' in the PSW, while the bit '7' goes into carry.



Bit manipulator

→ this loop in

→ carry flag i

instructions.

Mne

· ANL C, b

ANL C, (

ORL C, b

ORL C, (

CLR C

SETB C

CPL C

CLR b

SETB

Branch inst

→ These ins

Either con

return ins

→ Depending

instruction

→ The cond

below.

the 1st b

holds the

Bit manipulation or boolean instructions:-

→ this loop implements boolean bit operations.

⇒ carry flag is only allowed destination operand for 2 operands instructions.

Mnemonic	operations
ANL C, bit	The bit is ANDed with 'C' and result is stored in carry flag.
ANL C, /bit	complement of the bit is ANDed with C and stored in 'C' flag.
ORL C, bit	The bit is ORed with 'C'.
ORL C, /bit	Complement of the bit is ORed with 'C'.
CLR C	clears the carry flag.
SETB C	Set the bit of carry flag.
CPL C	Complement the carry flag.
CLR bit	clears the bit of A bit addressable SFR or RAM.
SETB bit	Sets the bit of bit addressable SFR or RAM.

Branch instructions:-

→ These instructions transfer the control of the sequence of execution either conditionally or unconditionally. All the Jump, call and return instructions come under this group of instructions.

→ Depending upon address space b/w first byte of the Jump or call instruction under specified Jump or target address.

→ The conditional instructions can be divided into 3 catagories as below.

1, short Jump (This Jump instruction is a 2 byte instruction, the 1st byte represents an op-code byte, whereas the 2nd byte holds the 8bit relative address.

(SJmp 8 bit address)

→ All the conditional Jumps are short Jumps.

2, Absolute Jump :- (AJMP 11 bit Address)

- this Jump instruction is mainly for a Jump within a memory space of 2KB. this is unconditional instruction.

3, long Jump :- (LJMP 16 bit Address)

- the long Jump instruction is 3 byte instruction, The 1st byte is op-code, the remaining 2 bytes point to the sub routine or location to which the control to be transfer.

→ LJMP is unconditional instruction.

1, unconditional instructions :-

Mnemonic	operation
JMP address	Jumps to the directly specified address 8 bit if SJMP, 11 bits if AJMP, 16 bits if LJMP.
JMP @A+DPTR	Jumps to the address indirectly specified by the Addition of DPTR and A this is 16 bit Address
CALL address	It calls a subroutine at specified address 11 bit if ACALL, 16 bits if LCALL.
RET	Return from subroutine.

2, conditional Jump instructions :-

mnemonic	operation
Jc - 8 bit Address	Jumps to the specified address if c=1.
JNC - 8 bit Address	Jumps to the specified address if c=0.
JB bit, 8 bit address	Jumps to the Address only specified bit=1.
JNB - Bit, 8 bit address	Jumps to the Address only specified bit=0.
JZ 8 bit address	Jumps to specified address only if zero.
JNZ 8bit address	Jumps to the Address if it is not zero

DJNZ SRC, Address	Decrement the specified operand and Jump to the specified address only if the content of the specified operand is not zero. SRC can be B, R0-R7, SFR.
CJNE A, SRC, Address	This performs a subtraction A-SRC and jumps to the specified address only if the result of subtraction is not zero.
CJNE SRC, #8bit data, Address	

8-bit Addition:-

```
ORG 0000H
MOV A, #12H
MOV R0, #15H
ADD A, R0
MOV 20H, A
END
```

A = 12H
R0 = 15H
A + R0 → A
12 + 15 → 27H.
20H ← 27H.

8 bit Subtraction:-

```
ORG 0000H
MOV A, #12H
MOV R0, #06H
SUBB A, R0
MOV 20H, A
END
```

A = 12H
B = 06H
A - R0 → A
12 - 06 → 06H.
20H ← 06H.

12 → F → 15
10 → 10
11 → 17
12 → 18

18 / 6 / 12

0001 1000
0000 0110
0001 1010
1D

8-bit multiplication:-

```
ORG 0000H
MOV A, #02H
MOV B, #03H
MUL AB
MOV 20H, A
MOV 21H, B
END
```

A : 06H
B = 00H.

8bit Division:-

```
ORG 0000H
MOV A, #0004H
MOV B, #02H
DIV AB
MOV 20H, A
MOV 21H, B
END
```

20H : 02H (Quotient)
21H : 00H (Remainder)

16-bit Addi

16 bit Su

16-bit Addition:-

```
ORG 0000H
MOV R0, #34H
MOV R1, #12H
MOV R2, #78H
MOV R3, #36H
CLR C
MOV A, R0
ADD A, R2        } A+R2=A
MOV 22H, A
MOV A, R1
ADDC A, R3       } A+R3=A
MOV 21H, A
END
```

lower byte of no1
Higher byte of no1
lower byte of no2
higher byte of no2.

```
 1234 H
 3678 H
-------
 48AC
```

22H = AC
21H = 48

16 bit subtraction:-

```
ORG 0000H
MOV R0, #78H
MOV R1, #36H
MOV R2, #34H
MOV R3, #12H
CLR C
MOV A, R0
SUBB A, R2
MOV 22H, A
MOV A, R1
SUBB A, R3
MOV 21H, A
END.
```

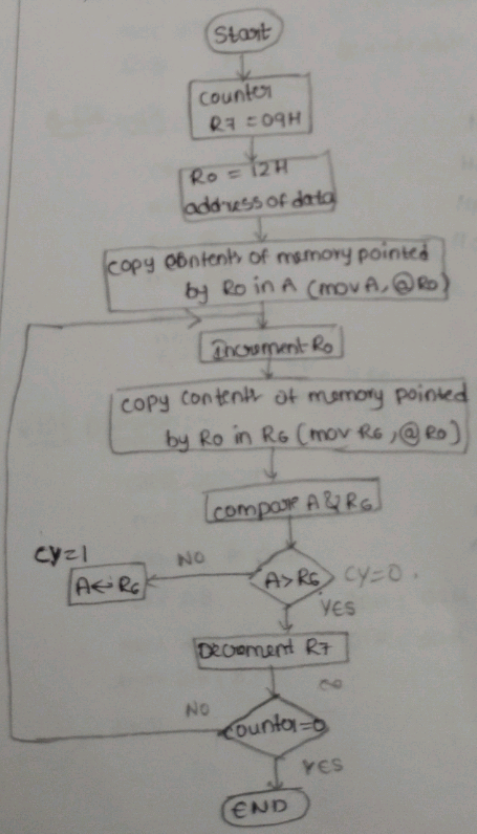
```
 3678
 1234
------
 2444
```

22H = 44
21H = 24

Find out largest number in the series of numbers:-

```
        mov R7, #09H
        mov R0, 12H
        mov A, @R0
Back :- INC R0
        MOV R6, @R0
        CJNE A, R6, L1
        SJMP L2
L1 : JNC L2
        mov A, R6
L2 : DJNZ R7, Back
        END.
```

Flowchart:-



Find out smallest number in the series of numbers :-

```
        MOV R7, #09H
        MOV R0, 12H
        MOV A, @R0
Back:   INC R0
        MOV R6, @R0
        CJNE A, R6, L1
        SJMP L2
L1:     JC L2
        MOV A, R6
L2:     DJNZ R7, Back
        END.
```

Timer or counter programming :-

Timer modes :-

Mode 1 :-

→ It is a 16 bit timer.

→ values from 0000 to FFFF can be loaded into the timers, registers TL and TH.

→ Timer must be started after loaded TH and TL with 16 bit initial value.

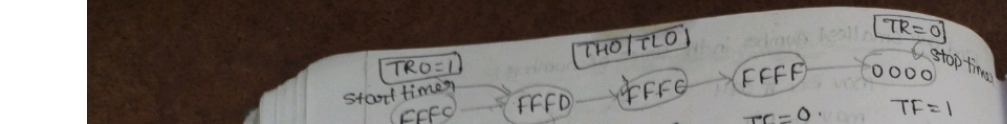
→ we can give initial values by using "SETB TR0" for timer '0' and "SETBTR1" for timer '1'.

→ when timer is started, it's starts counting until it reaches its limit of FFFFH.

→ After this limit i.e. 0000H it sets a TF bit, TF0 for timer '0', TF1 for timer '1'.

→ Now stop the timer with the instruction CLR TR0 for timer '0' or CLRTR1 for timer '1'.

→ To repeat the process, the original values must be reloaded in TH and TL registers and timer flag must be reset to '0'



TF goes high when FFFF → 0.

Mode 0 :-

→ It is a 13 bit timer.

→ It is same as mode 1 but with only difference of size.

→ mode 0 register can hold values upto 1FFFFH in TH and TL.

→ when timer reaches its maximum of 1FFFFH, it rolls over to 0000 and TF is raised to high.

→ Not used generally.

Mode 2 :-

→ It is an 8 bit timer.

→ This timer first loads the values upto FFH in TH register.

→ when TH is loaded, 8051 copies the same bit value to TL.

→ The timer must be started by SETBTR0 and SETBTR1.

→ when its starts, it continues to increment the TL register unfill it reaches FFH, after that it rolls over to 00. and raises the TF.

→ After that TL is loaded automatically with its original value, which is reserved by the register TH.

→ The main feature of this mode is that it repeats the process, we just have to clear TF register and it will load original value to the register.

→ while in mode 1 programmer reloads the values to TH and TL.

TR=0
stop timer
0000
TF=1

difference of size.
IFFFFH in TH and
FFH, it rolls over

FFH in TH register.
bit value to TL.
TR0 and SETB TR1.
ent the TL register
olls over to 00. and

with its original value,

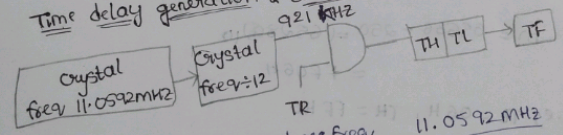
at it repeats the
register and it will

the values to TH and

MODE 3:-

→ It is also known as a split timer mode.
→ It splits timer '0' into 2-8bit timers.
→ these 8 bit timers can count from 00H to FFH.
→ this mode is using in the applications where we require an additional 8 bit timer or counter.
→ we can program timer '0' and timer '1' independently in mode 0, 1, 2. of similar mode for other timers, but mode 3 can't work independently.
→ If we choose mode 3 for timer 0 and we place timer '1' in mode 3 it will cause the timer to stop counting, then timer '0' will use TR1 (Timer 1) and TF1 (Timer 1 Flag) i.e timer 0' has to be depend on timer '1'.

Time delay generation & calculation:-



$$\text{Timer freq} = \frac{\text{crystal osc freq}}{12} = \frac{11.0592\,\text{MHz}}{12}$$

$$= 921\,\text{KHz}$$

$$\text{Time period} = \frac{1}{921\,\text{KHz}} = 1.085\,\mu\text{s}$$ (Takes to execute one instruction)

Eg: For the delay of 10 milliseconds.

1, first divide the desired time delay value by the timer clock period.

$$N = \frac{10\text{ms}}{1.085\mu\text{s}} = 9216$$

2, Subtract the value of N from maximum number of counts possible for 16 bit timer i.e $2^{16} = 65,536$.

$$M = 65,536 - 9216$$
$$= 56320$$

3, convert this value into Hexadecimal and write in TH and TL registers.

71 = 16 | 56320		16) 56320 (3520
16 | 3520 - 0		48
16 | 220 - 8		83
13 - 12		80
		32
		32

H = DC00H.
TH TL

Write a program to continously generate a square wave of 2KHz on pin P1.5 using timer 1. Let (port 1.6S)
Let crystal frequency = 12MHz
the period of the square wave is $T = \frac{1}{2KHz} = 500\mu s$

Each half pulse = 250μs
the value of N for 250μs is $\frac{250\mu s}{1\mu s\left(\frac{12}{12}\right)} = 250$

$m = 65536 - 250 = (65280)_{10}$
$= FF06H$

i.e; TL = 06H, TH = FFH.

Program:-

```
        MOV TMOD, #10; Timer 1, mode 1
AGAIN:- MOV TL1, #06H ; TL1 = 06H
        MOV TH1, #FFH; TH1 = FFH
        SETB TR1      ; start timer 1
BACK:   JNB TF1, BACK ; stay until timer rolls on
        CLR TR1       ; stop timer 1
        CPL P1.5      ; complement P1.5
        CLR TF1       ; clear timer flag
        SJMP AGAIN    ; Reload timer
        END.
```