SESHASAYEE INSTITUTE OF TECHNOLOGY(AUTONOMOUS)
ARIYAMANGALAM,TRICHY – 10

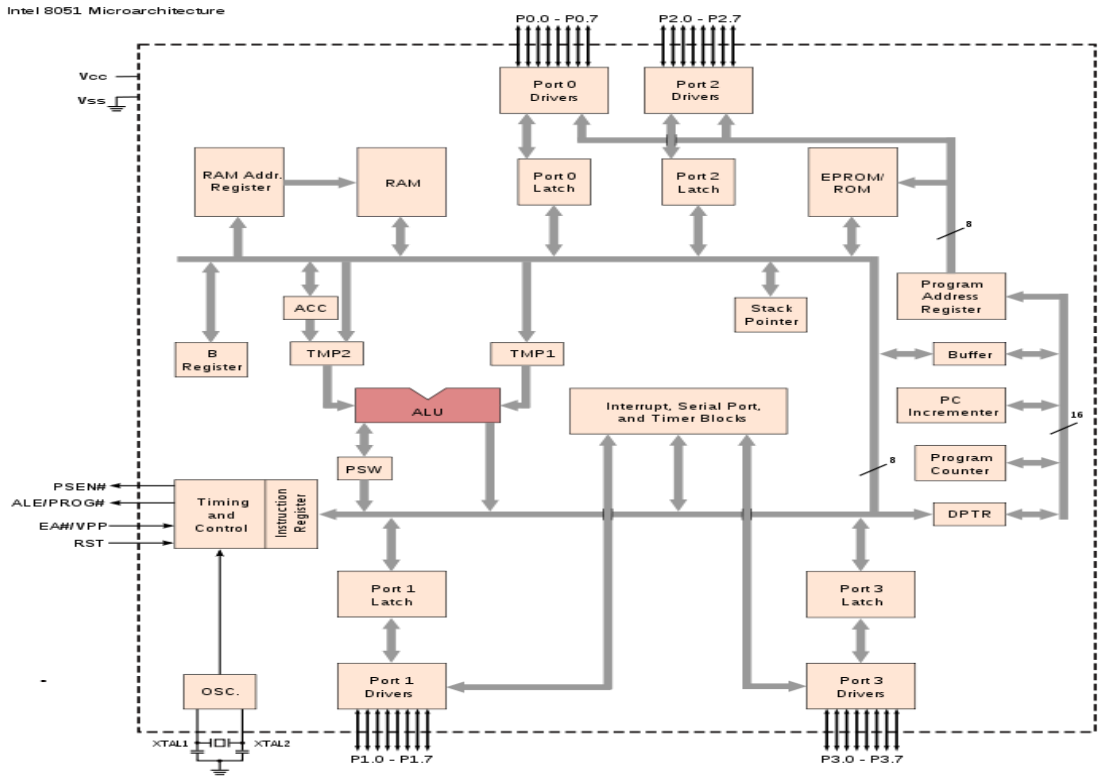
DIPLOMA IN ELECTRICAL & ELECTRONICS ENGINEERING

EAR/SEMESTER : III/V

COURSE CODE : 2E5402.1 COURSE TITLE : MICROCONTROLLER

UNIT – I

a) Draw and Explain the Architecture of 8051. (08)



Functional Description Of Each Block :

a) **Accumulator: (Acc)** Acc is an 8 bit special function register. It acts an operand register. Result is temporarily stored in this register. It is used in parallel I/O transfer.

b) **B Register :** B register is 8 bit SFR. It is used during multiply and divide operations.
For other operations , it can be used as a scratchpad register.

c) **Program Status Word (PSW) :** PSW register is 8 bit SFR. It contains program status information. It is also used to select any one of the required register bank.

d) **Stack Pointer (SP):** It is 8 bit register. It is used to point the stack memory. The stack may reside in anywhere in on-chip memory. It is incremented before data is stored during PUSH & CALL instructions. After reset SP is initialized to 07h. This causes the stack begin at location 08h.

e) **Data Pointer (DPTR) :** It is 16 bit register. It may be manipulated as a 16 bit register or as two independent 8 bit registers. Its function is to hold a 16 bit address. This register is used for external reference.

f) **Port 0 to Port 3 :** Each port contains separate address. Using this address,User can communicate with these ports. Each port contains latch,output driver & input buffer.

g) **Serial Data Buffer :** Serial data buffer contains two independent registers of a transmit buffer register and a receiver buffer register.
- Transmit buffer is a parallel in and Serial out register.
- Receiver buffer is a Serial in and parallel out register.
- When data is moved to SBUF, it goes to transmit buffer .
- When data is moved from SBUF, it comes from the receive buffer.

h) **Timer Registers :** Register pairs (TH0,TL0) & (TH1,TL1) are the two 16 bit counting registers for Timer/Counter 0 and 1 respectively.

i) **Control Registers :** The special function registers IP,IE , TMOD, TCON SCON and PCON contain control and ststus information for interrupts, timer/counters and serial port.

j) **Timing & Control Unit :** This unit derives an necessary timing and control signals required for the internal operations of the circuit. It derives control signals required for contolling the external system bus. The interrupt, serial port and timer circuits are controlled by the control signals generated by timing & control unit.

k) **Oscillator :** This circuit generates the basic timing clock signal for the operation of the circuit using crystal oscillator.

l) **Instruction Register :** This register decodes the opcode of an instruction to be executed and gives information to the timing & control unit, and to generate necessary signals for the execution of the instruction.

m) **EPROM & Program Address Register :** These blocks provide an onchip EPROM and a Mechanism to internally address it.

n) **RAM & RAM address register :** These blocks provide an onchip RAM and a mechanism to internally address it.

o) **ALU :** The **Arithmetic And Logic Unit** performs 8 bit arithmetic and logical operations over the operands held by the temporary registers TMP 1 and TMP 2. User cannot access these temporary registers

p) **SFR Register Banks :** It is a set of registers, which can be addressed using their respective addresses which lie in the range 80h to FFh.

Draw the pin Diagram of 8051 and explain the function of each pin (08)

Pin Description of 8051 micro controller **Vcc &V**ss : Power supply Signals

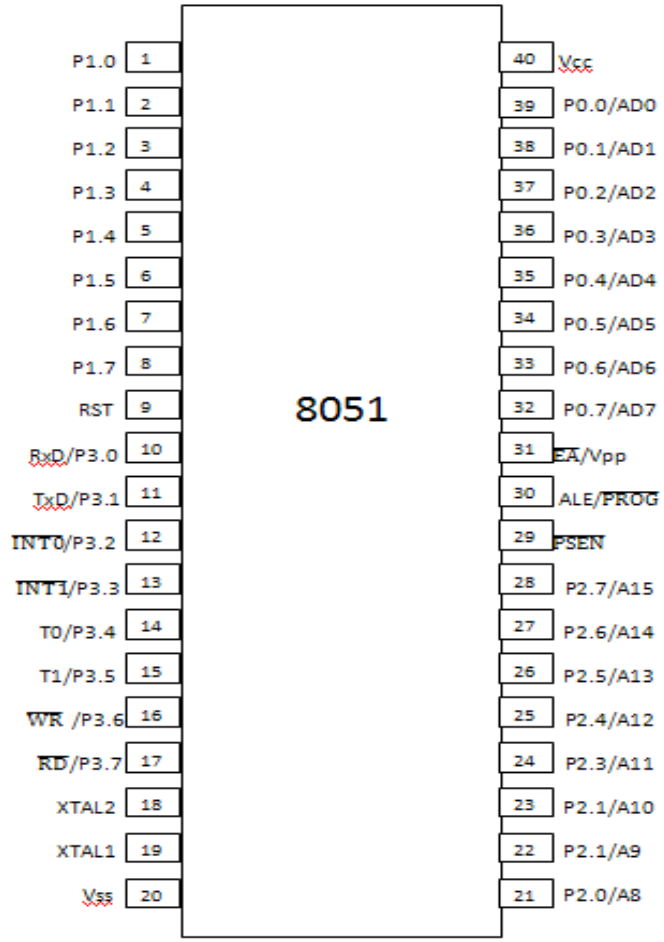
Port 0(p0.0 to p0.7) : It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access, it functions as multiplexed data and low-order address bus AD0-AD7

Port 1(p1.0 to p1.7) : It is 8-bit bi-directional I/O port. It is bit/ byte addressable. When logic 1 is written into port latch then it work as input mode. It functions as simply I/O port and it does not have any alternative function.

Port 2(p2.0 to p2.7) : It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access it functions as higher order address bus (A8-A15).

Port 3(p3.0 to p 3.7) : It is 8-bit I/O port. In an alternating function each pins can be used as a special function I/O pin.

XTAL 1 and XTAL 2 : These are two I/P line for on-chip oscillator and clock generator circuit.



(EA)'…/ VPP : External Access : It is an active low I/P to 8051 µc. when (EA)'= 0, then 8051 µc access from external program memory (ROM) only.

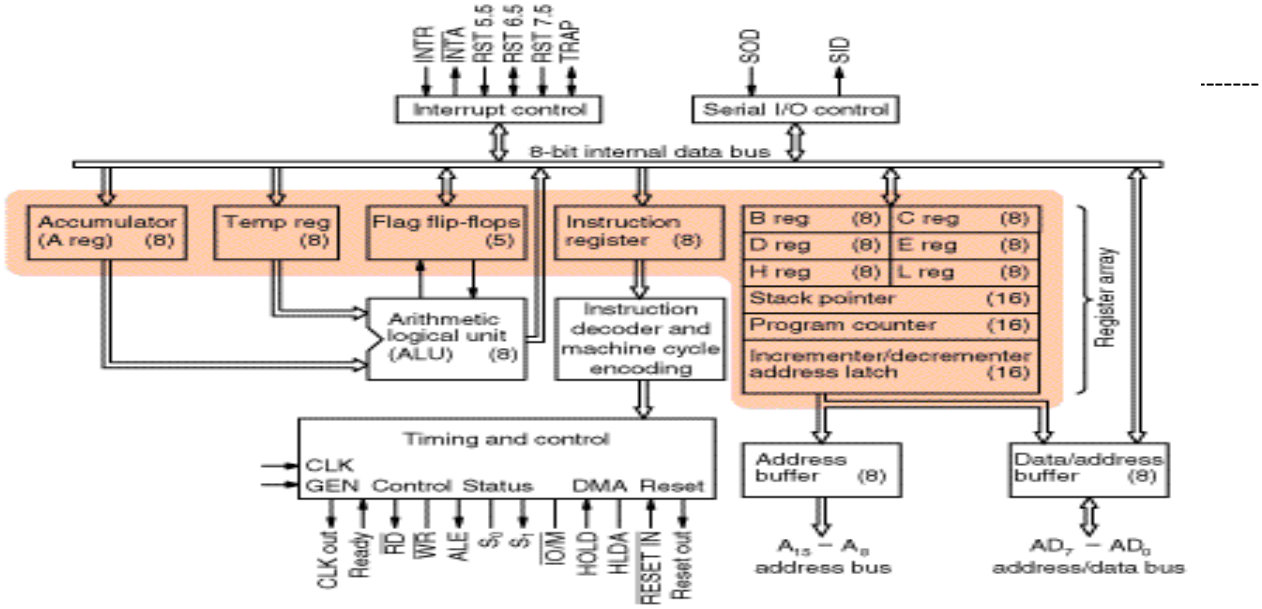
When (EA)'= 1, then it access internal and external program memories (ROMS).

(PSEN)' : Program Store Enable : It is active low O/P signal. It is used to enable external program memory (ROM). When (PSEN)'=0, then external program memory becomes enable and µc read content of external memory location. Therefore it is connected to (OE)' of external ROM. It is activated twice every external ROM memory cycle.

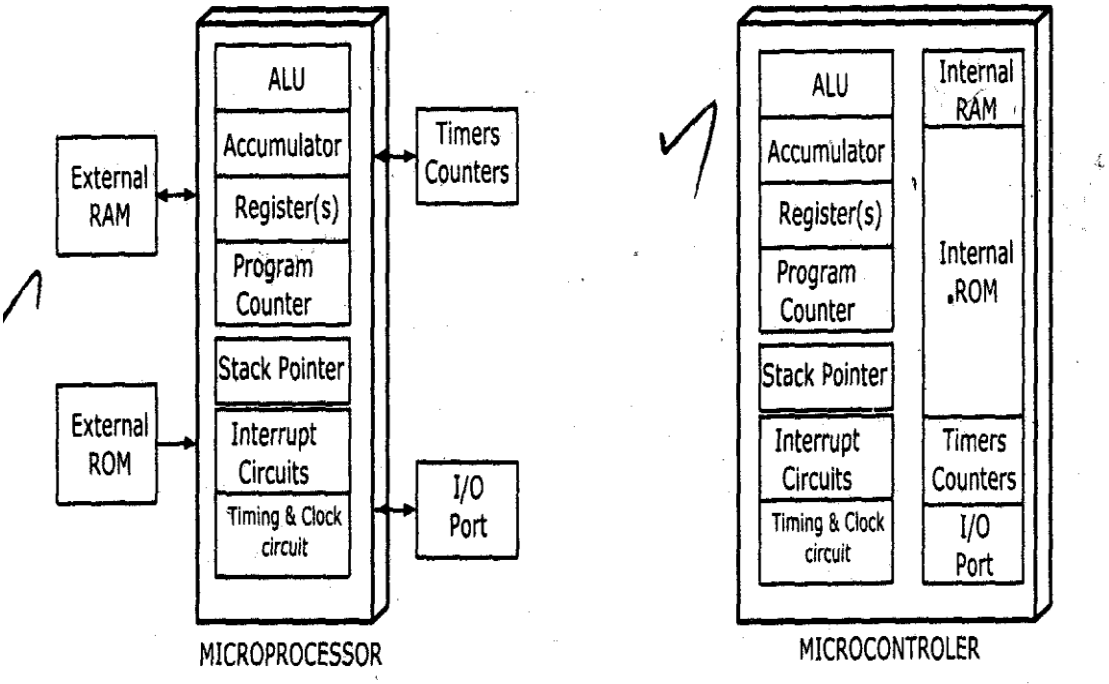
ALE-Address latch enable: It is active high O/P signal. When it goes high, external address latch becomes enabling and lower address of external memory (RAM or ROM) latched into it. Thus it separates A0-A7 address from AD0- AD7. It provides properly timed signal to latch lower byte address. The ALE is activated twice in every machine cycle.

RESET: It is active high I/P signal. It should be maintained high for at least two machine cycle while oscillator is running then 8051 µc resets, i.e. It clears the following registers: A,B,PSW

Draw the architecture of 8085 and briefly explain each block.



COMPARE MICROPROCESSOR & MICROCONTROLLER



COMPARING MICROPROCESSOR AND MICROCONTROLLER

Sl.No.	Microprocessor (mP)	Microcontroller (mC)
1.	It is intended tobe general - purpose digital computers	It is intended to be special purpose digital controllers.
2.	It contains CPU memory addressing circuits and interrupt, handling circuit	It contains CPU as well as timers, parallel and serial I/O and internal RAM and ROM
3.	Microprocessor have many operational codes (opcodes) for moving data from external memory to the CPU.	It may have many bit handling instructions.
4.	It has one or two types of bit handling instructions.	It has many bit handling instructions.
5.	Microprocessor is concerned with rapid movement of code and data from external address to the chip	Microcontroller is concerned with the movement of bits within the chip.
6.	Microprocessor must have many additional parts peripherals to function as a computer	Microcontroller can function as a computer with the addition of no external digital parts.

APPLICATIONS

Application of Microcontroller	
Industrial Control Devices: • Industrial instrumentation devices • Process control devices	**Day to Day Life Devices:** • Light sensing & controlling devices • Temperature sensing and controlling devices • Fire detection & safety devices • Security Alarms • In Computers-> Modems and Keyboard Controllers
Metering & Measurement Devices: • Volt Meter • Measuring revolving objects • Current meter • Hand-held metering systems	**electronic equipments :** • Mobile Phones • Auto Mobiles • CD/DVD Players • Washing Machines • Cameras • Microwave Oven.

List down the applications of microcontroller.

Keyboard	MP3 music players	Electronic toys
Printer	Data switches	Washing machines
video game	Embedded memories to keep	Stepper motor control
Mobile phone	Domestic (home) appliances	Traffic control
Automotive	Small business system	DC motor speed control
Office	Xerox copier	Inverters

The use of microcontroller in various fields such as automobile, aeronautics, space, robotics, electronics, defense application, mobile communications, rail transport, industrial processing, and medical applications is rapidly increasing.

There are numerous applications of 8051 microcontroller in science and technology and a few more of them can be listed as below

- Prepaid energy meter using microcontroller and GSM modem
- Transformer/Generator health monitoring remotely over internet
- Power theft detection and intimation to control room using GSM
- Controlling remote industrial plant using SCADA
- Avoiding excessive electric bills for industries and commercial establishments
- Induction motor speed control using Arduino
- Density based traffic signal system controlled using a smartphone
- Virtual display of messages using propeller drive LEDs
- Digital sensor based temperature control
- High power dual converter using a pair of SCR bridge
- LAN like setup of multiple microcontrollers
- Positioning of dish antenna using the remote
- I2C protocol based auto dialing on burglary detection

- **Energy Management:** Competent measuring device systems aid in calculating energy consumption in domestic and industrialized applications. These meter systems are prepared competent by integrating microcontrollers.
- **Touch screens:** A high degree of microcontroller suppliers integrate touch sensing abilities in their designs. Transportable devices such as media players, gaming devices & cell phones are some illustrations of micro-controller integrated with touch sensing screens.
- **Automobiles:** The microcontroller 8051 discovers broad recognition in supplying automobile solutions. They are extensively utilized in hybrid motor vehicles to control engine variations. In addition, works such as cruise power and anti-brake mechanism has created it more capable with the amalgamation of micro-controllers.
- **Medical Devices:** Handy medicinal gadgets such as glucose & blood pressure monitors bring into play micro-controllers, to put on view the measurements, as a result, offering higher dependability in giving correct medical results.

Explain the memory organization of 8051 microcontroller . (08)

MEMORY ORGANIZATION : The 8051 architecture provides both on chip memory as well as off chip memory expansion capabilities. It supports several distinctive 'physical' address spaces, functionally EA separated at the hardware level by different addressing mechanisms, read and write controls signals or both. The 8051 has on chip memory of 4 Kbytes of internal Program Memory and 128 bytes of internal Data memory. It can access upto 64 K program memory and 64Kdata memory.

The 3 different address spaces of 8051

i) 64 Kbyte program memory ii) 64 Kbyte external data memory iii) 256 byte internal data memory.

Internal Data Memory (256bytes) is divided into two physically separate and distinct blocks.

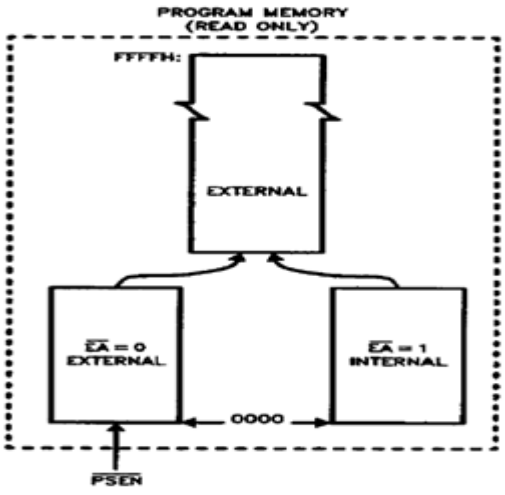
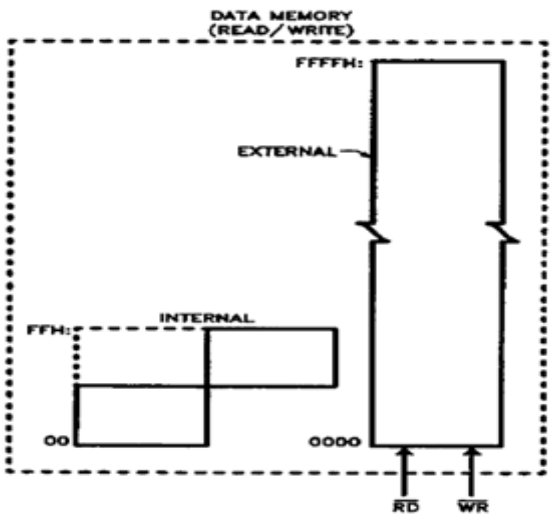
i) **128 Bytes Internal RAM Area**. This area is further divided into three.

a) Four register banks [each bank has eight 8 bit registers] **(address range 00 – 1Fh)**

b) 16 bytes bit addressable locations **(address range 20h – 2Fh)**

c) 80 bytes byte addressable locations **(address range 30h – 7Fh)**

ii) **128 Bytes SFR Area (address range 80h – 0FFh**)

Program memory(Read Only)	Data Memory(Read/Write)
	

Connecting External Program Memory(ROM)

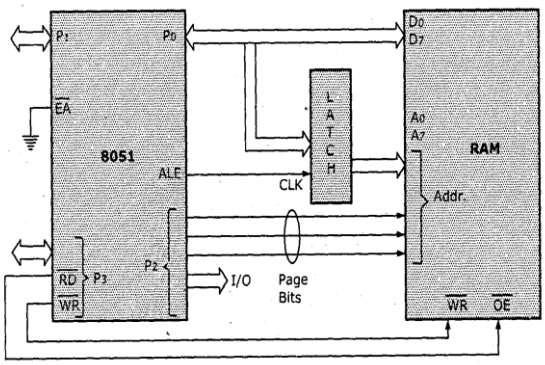


Fig 1.19 Accessing External Data Memory

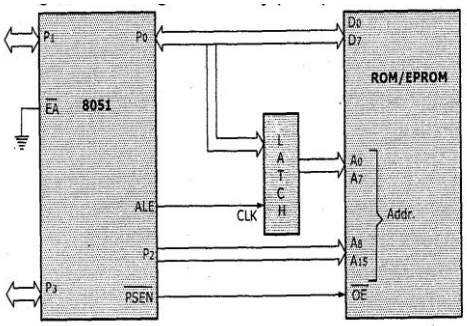


Fig 1.18 Accessing External Program Memory

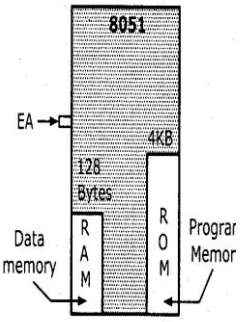


Fig 1.13

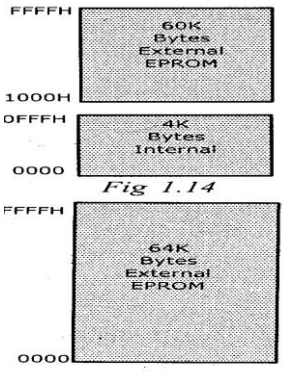


Fig 1.14

Fig 1.15

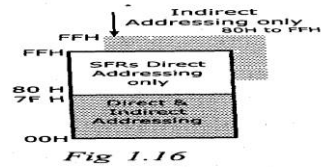


Fig 1.16

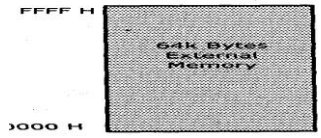


Fig 1.17

Explain internal memory Organization of 8051 with neat diagram

Internal data memory (256bytes) is divided into two physically separate and distinct blocks.

i) 128 bytes internal RAM area. This area is further divided into three.

a) Four register banks [each bank has eight 8 bit registers] **(address range 00 – 1Fh)**

b) 16 bytes bit addressable locations **(address range 20h – 2Fh)**

c) 80 bytes byte addressable locations **(address range 30h – 7Fh)**

ii) 128 bytes SFR area **(address range 80h – 0FFh**)

Bit Addressable RAM

RAM

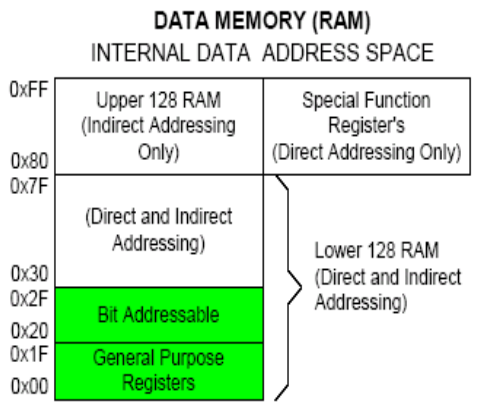
Byte address	Bit address							
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F	Bank 3							
18								
17	Bank 2							
10								
0F	Bank 1							
08								
07	Default register bank for R0–R7							
00								

Bit-addressable locations

Byte address	Bit address							
7F	General purpose RAM							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40

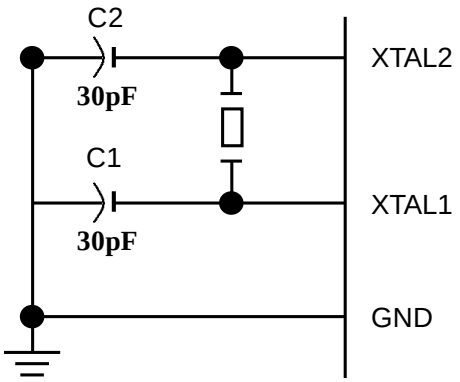
Bit-addressable locations

Bit Addressable RAM



Byte address	Bit address									Byte address	Bit address								
98	9F	9E	9D	9C	9B	9A	99	98	SCON	FF									
										F0	F7	F6	F5	F4	F3	F2	F1	F0	B
90	97	96	95	94	93	92	91	90	P1										
										E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
8D	not bit addressable								TH1										
8C	not bit addressable								TH0	D0	D7	D6	D5	D4	D3	D2	–	D0	PSW
8B	not bit addressable								TL1										
8A	not bit addressable								TL0	B8	–	–	–	BC	BB	BA	B9	B8	IP
89	not bit addressable								TMOD										
88	8F	8E	8D	8C	8B	8A	89	88	TCON	B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
87	not bit addressable								PCON										
										A8	AF	–	–	AC	AB	AA	A9	A8	IE
83	not bit addressable								DPH										
82	not bit addressable								DPL	A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
81	not bit addressable								SP										
80	87	86	85	84	83	82	81	80	P0	99	not bit addressable								SBUF

XTAL Connection to 8051



Find the machine cycle for

- (a) XTAL = 11.0592 MHz
- (b) XTAL = 16 MHz.

Solution:

(a) 11.0592 MHz / 12 = 921.6 kHz;

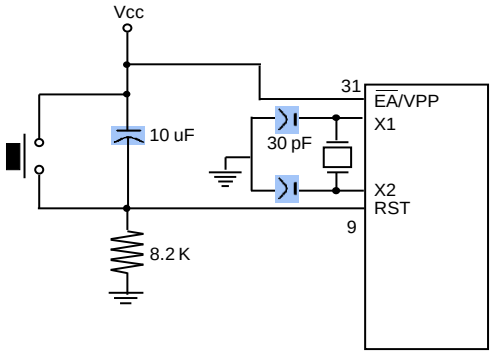
machine cycle = 1 / 921.6 kHz = 1.085 μs

(b) 16 MHz / 12 = 1.333 MHz;

machine cycle = 1 / 1.333 MHz = 0.75 μs

Power-On RESET

- RST（pin 9）：reset



(i) input pin and active high（normally low）.

The high pulse must be high at least 2 machine cycles.

(ii)power-on reset. Upon applying a high pulse to RST, the microcontroller will reset and all values in registers will be lost.

Reset values of some 8051 registers

(A) ← 00 ;(SP) ←07

(iii)power-on reset circuit -as shown in figure

List down SFRs with their byte and bit addresses. (08)

SPECIAL FUNCTION REGISTERS (SFRS) : Special Function Registers (SFRs) are a sort of control table used for running and monitoring the operation of the microcontroller. Each of these registers as well as each bit they include, has its name, address in the scope of RAM and precisely defined purpose

such as timer control, interrupt control, serial communication control etc.. All types of 8051 microcontrollers, has only 21 such registers..

SN	SFR	NAME	BYTE ADDRESS IN HEXA	SN	SFR	NAME	BYTE ADDRESS IN HEXA
01	ACC*	Accumulator	0E0	12	IE*	InterruptEnable	0A8
02	B*	B register	0F0	13	TMOD	Timer mode control	89
03	PSW*	Program status word	0D0	14	TCON*	Timer control	88
04	SP	Stack pointer	81	15	TH0	Timer/counter0 high	8C
05	DPL	Low byte	82	16	TL0	Timer/counter0 low	8A
06	DPH	High byte	83	17	TH1	Timer/counter1 high	8D
07	P0*	Port 0	80	18	TL1	Timer/counter1 low	8B
08	P1*	Port 1	90	19	SCON*	Serial control	98
09	P2*	Port 2	0A0	20	SBUF	SerialData buffer	99
10	P3*	Port 3	0B0	21	PCON	Power control	87
11	IP*	InterruptPriority	0B8	22	PC	Program Counter (No address)	

*- Both byte and bit ADDRESSABLE :

All SFRs such as (ACC, B, PCON, TMOD, PSW, P0 ,P1,P2,P3, …) are accessible by name and direct address. But both of them Must be coded as direct address.

SN	SFR	NAME	BYTE ADDRESS IN HEXA	BIT ADDRESS IN HEXA
01	ACC*	Accumulator	0E0	E7 – E0
02	B*	B register	0F0	F7 – F0
03	PSW*	Program status word	0D0	D7 – D0
04	P0*	Port 0	80	87 – 80
05	P1*	Port 1	90	97 – 90
06	P2*	Port 2	0A0	A7 – A0
07	P3*	Port 3	0B0	B7 – B0
08	IP*	InterruptPriority	0B8	BF – B8
09	IE*	InterruptEnable	0A8	AF – A8
10	TCON*	Timer control	88	8F – 88
11	SCON*	Serial control	98	9F - 98

UNIT II ,unit iii & UNIT IV

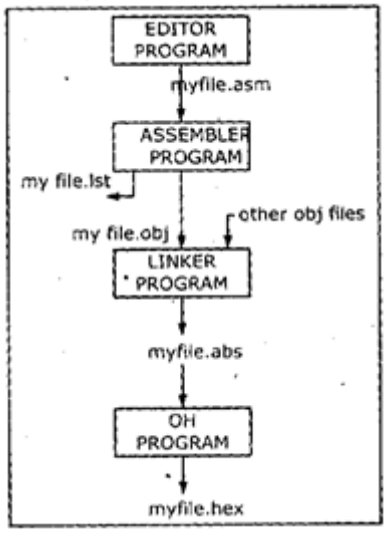
01) a) Explain how to assemble and run an 8051 program using assembler . (08)

Assembling And Running an 8051 program using ASM-51

The steps to create an executable Assembly language program are outlined as follows.

(1) We use an editor to type in a program similar to program many excellent editors or word processors are available that can be used to create and/or edit the program. A widely used editor is the MS-DOS EDIT program (or Notepad in Windows).which comes with all Microsoft operating systems.The editor must be able to produce an ASCII file. For many assemblers, the file names follow the usual DOS conventions, but the source file has the extension “asm” or “src” depending on which assembler you are using.

The "asm" extension for the source file is used by an assembler in the next step.



(2) The "asm" source file containing the program code created in step 1 is fed to an 8051 assembler. The assembler converts the instructions into machine code. The assembler will produce an object file and a list file. The extension for the object file is "obj" while the extension for the list file is "1"

(3)Assemblers require a third step called linking. The link program takes one or more object files and produces an absolute object file with the extension "abs". This abs file is used by 8051 trainers that have a monitor program.

(4) Next the "abs" file is fed into a program called "OH" (object to hex converter)which creates a file with extension "hex" that is ready to bum into ROM. This program comes with all 8051 assemblers. Recent Windows-based assemblers combine steps 2 through 4 into one step.

Explain different addressing modes of 8051 with examples. (08)

SLNO	ADDRESSING	EXAMPLE 1	EXAMPLE 2
1	Immediate	MOV A,#05	MOV R4,#7AH
2	Register	MOV A,B	MOV R5,A
3	Direct	ADD A,40H	MOV 40H,R0
4	Register indirect	MOV A,@R1	ADDC A,@R0
5	Indexed	MOVC A,@A+DPTR	MOVC A,@A+PC

Immediate Mode – specify data by its value

- **mov A, #0** ;put 0 in the accumulator ;(A) = 00000000
- **mov R4, #11h** ;put 11hex in the R4 register ;(R4) = 00010001
- **mov B, #11** ;put 11 decimal in b register ; (B) = 00001011
- **mov DPTR,#7521h** ;put 7521 hex in DPTR ;(DPTR) = 0111010100100001

Register Addressing – either source or destination is one of CPU register

(i) **MOV R0,A (ii) MOV A,R7 (iii) ADD A,R4 (iv) ADD A,R7 (v) MOV DPTR,#25F5H**

(vi) MOV R5,DPL (vii) MOV R,DPH Note that MOV R4,R7 is incorrect

Direct Mode – specify data by its 8-bit address . Usually for 30h-7Fh of RAM

- **Mov a, 70h ; copy contents of RAM at 70h to a**
- **Mov R0,40h ; copy contents of RAM at 70h to a**
- **Mov 56h,a ; put contents of a at 56h to a**
- **Mov 0D0h,a ; put contents of a into PSW**

Register Indirect – The address of the source or destination is specified in registers . Uses registers R0 or R1 for 8-bit address:

- **mov psw, #0 ; use register bank 0**
- **mov r0, #0x3C**
- **mov @r0, #3 ; memory at 3C gets #3 ; M[3C] ← 3**
- Uses DPTR register for 16-bit addresses:
- **mov dptr, #0x9000 ; dptr ← 9000h**
- **movx a, @dptr ; a ← M[9000]** Note that 9000 is an address in external memory

Register Indexed Mode – source or destination address is the sum of the base address & the accumulator(Index)

- Base address can be DPTR or PC

  MOV DPTR, #4000H
  MOV A, #5
  MOVC A, @A + DPTR ;A ← M[4005]
- BASE ADDRESS CAN BE DPTR OR PC

  ORG 1000H
  1000 MOV A, #5
  MOVC A, @A + PC ;A ← M[1008]
  NOP
- MOVC only can read internal code memory

Explain any ten Data Transfer instructions with examples

DATA TRANSFER INSTRUCTIONS

SN	Description	MNEMONICS		LENGTH in bytes
		OPCODE	OPERANDS	
1	Data transfer instructions move data from source operand to destination operand. source operand and destination operand may be placed in Accumulator,internal RAM or any one of special function registers	MOV	A,#Data	2
2		MOV	Data	3
3		MOV	@Ri,#Data	2
4		MOV	Rn,#Data	2
5		MOV	DPTR,#DATA	3
6		MOV	Data addr	3
7		MOV	Data addr	2
8		MOV	Data addr ,Rn	2
9		MOV	Data addr,A	2
10		MOV	@Ri,Data	2
11		MOV	Rn,Data addr	2
12		MOV	A, Data addr	2
13		MOV	@Ri,A	1
14		MOV	Rn,A	1
15		MOV	A,@Ri	1
16		MOV	A,Rn	1
17	Data transfer between Accumulator & external memory	MOVX	@DPTR,A	1
18		MOVX	@Ri,A	1
19		MOVX	A,@DPTR	1
20		MOVX	A,@Ri	1
21	Data transfer between CY and bit address	MOV	C,BIT ADDR	2
22		MOV	BIT ADDR,C	2
23	Move code byte to accumulator	MOVC	A,@A+DPTR	1
24		MOVC	A,@A+PC	1
25	Retrieve from stack	POP	Data addr	2
26	Push into stack	PUSH	Data addr	2
27	Exchange Byte with Accumulator.Byte may be in any one of register or internal RAM	XCH	A, Data addr	2
28		XCH	A,@Ri	1
29		XCH	A,Rn	1
30	Exchange low order bits of Byte with low order bits of Accumulator. Byte may be in any one of register or	XCHD	A,@Ri	1

	internal RAM			

Explain the instructions "ROTATE & S WAP" with example . (08)

SWAP:-

INSTRUCTION FORMAT :- SWAP A

FUNCTION :- Exchange Low Order 4 Bits With Upper 4 Bits Of Register A.

LENGTH :- 1B

DESCRIPTION : This instruction interchanges the lower 4 bits (lower nibble) with the higher 4bits (higher nibble) of A register. This instruction is equivalent to four times rotate (either left or Right).

EXAMPLE : MOV A,#58H ; (A) ← 58H
SWAP A ; (A) ← 85H

ROTATE:- This instruction rotates the bits of Accumulator either left or Right. The bits rotated out from the Content is rotated back or front into the opposite end of the content of the accumulator.

Example: CLR C ; CY ← 0
MOV A, #55H; (A) ← 01010101

SLNO	MNEMONICS	LENGTH	FLAGS AFFECTED	EXAMPLE	AFTER EXECUTING ROTATE INSTRUCTION		
					CY	(A) - BINARY 01010101	(A) – HEXA (A) ← 55H
1	RR A	1B	NONE	CLR C MOV A,#55H RR A	0	00101010	2A
2	RRC A	1B	CARRY	CLR C MOV A,#55H RRC A	1	00101010	AA
3	RL A	1B	NONE	CLR C MOV A,#55H RL A	0	10101010	AA
4	RLC A	1B	CARRY	SETB C MOV A,#55H RLC A	0	10101011	AB

Explain Various arithmetic instructions used in 8051 with suitable example. (08)

SN	MNEMONICS	ADDRESSING MODE			
		REGISTER	DIRECT	INDIRECT	IMMEDIATE
1	ADD →	ADD A,Rn	ADD A,DIRECT	ADD A,@Ri	ADD A,#8BIT DATA
2	ADDC →	ADDC A,Rn	ADDC A,DIRECT	ADDC A,@Ri	ADDC A,#8BIT
3	SUBB→	SUBB A,Rn	SUBB A,DIRECT	SUBB A,@Ri	SUBB A,#8BIT DATA
4	INC A	INC Rn	INC DIRECT	INC @Ri	
5	DEC A	DEC Rn	DEC DIRECT	DEC @Ri	
6	INC DPTR				
7	MUL AB				
8	DIV AB				
9	DA A				

Arithmetic Instructions : Source is a byte. Byte may be an any one of immediate operand(#data), register operand (Rn) or Memory content(that may be accessed by direct or indirect addressing).

Mnemonic	Description	Explanation	Example	Result
ADD A, byte	Add (A) to byte	(A) ← (A)+byte	MOV A,#07 ADD A,#05	(A)←0Ch
ADDC A, byte	Add with carry	(A) ← (A)+byte+CY	MOV A,#56H ADDC A,#0FH	(A)←65h
SUBB A, byte	Subtract with borrow	(A) ← (A)-byte-CY	MOV A,#07 SUBB A,#04	(A)←03
DEC A	Decrement (A) by 1	(A) ← (A)-1	MOV A,#07 DEC A	(A)←0Ch
DEC byte	Decrement byte	(byte) ← (byte)-1	MOV R2,#A0H DEC R2	(R2)←9Fh
INC A	Increment (A) by 1	(A) ← (A)+1	MOV A,#0FH INC A	(A)←10h
INC byte	Increment byte by 1	(byte) ← (byte)+1	MOV R3,#3FH INC R3	(R3)←40h
INC DPTR	Increment data pointer	(DPTR)←(DPTR)+1	MOV DPTR,#422FH INC DPTR	(DPTR) ← 4230h

MUL AB	Multiply (A) by (B)	(A) * (B),Product is stored both registers A & B.
		(A) ←LOB of product
		(B) ←HOB of product
			MOV A, #07
			MOV B,#05
			MUL AB
				(A)←23h
				(B)←00

DIV AB	Divide (A) by (B)	(A) / (B), Result is in
		(A) ←Quotient
		(B) ←Remainder
			MOV A, #07
			MOV B,#05
			DIV AB
				(A)←01
				(B)←02

DA A	Decimal adjust the accumulator	Result is in BCD	MOV A, #29H
			ADD A,#63H
			DA A
				(A)←92 (BCD)

Write an assembly language program to perform Multibyte addition (08)

INPUT				**INPUT**			
Address	**DATA1**		**COMMENTS**	**Address**	**DATA2**		**COMMENTS**
	S1	**S2**			**S1**	**S2**	
9200	**45**	**23**	**LOB -- B1**	**9250**	**63**	**02**	**LOB -- B1**
9201	**14**	**56**	**↓ -- B2**	**9251**	**A1**	**07**	**↓ -- B2**
9201	**25**	**01**	**HOB --B3**	**9252**	**35**	**FF**	**HOB --B3**

OUTPUT					
Address	**DATA**		**COMMENTS**	**Set 1**	**Set 2**
	S1	**S2**			
9000	**A8**	**25**	**LOB --- B1**	**Data 1 : 251445**	**Data 1 : 015623**
9001	**B5**	**5D**	**↓ --- B2**	**Data2 : 35A163+**	**Data2 : FF0702+**

9002	**5A**	**00**	**---- B3**	**----------**	**----------**
9003	**00**	**01**	**HOB--- B4**	**Sum : 5AB5A8**	**Sum : 01005D25**

MULTIBYTE ADDITION PROGRAM:

LABEL	MNEMONICS	Description
	MOV DPTR,#9200H	Initialise Data Pointer with data1 address
	MOV R0, #35H	Initialise R0 with internal RAM address 35h
	MOV R5, #03H	Store number of bytes in a data into R5.
TOP1:	MOVX A, @DPTR	Data from external memory starting at address 9200H is copied into internal RAM starting at address 35H
	MOV @R0, A	
	INC R0	
	INC DPTR	
	DJNZ R5, TOP1	
	MOV DPTR, #9250H	Initialise Data Pointer with data2 address
	MOV R0, #45H	Initialise R0 with internal RAM address 45h
	MOV R5, #03H	Store number of bytes in a data into R5.
TOP2:	MOVX A, @DPTR	Data from external memory starting at address 9250H is copied into internal RAM starting at address 45H
	MOV @R0,A	
	INC DPTR	
	INC R0	
	DJNZ R5, TOP2	
	MOV DPTR, #9000H	Initialise Data Pointer with Result address
	MOV R0, #35H	Initialise R0 with internal RAM address 35h
	MOV R1, #45H	Initialise R1 with internal RAM address 45h
	MOV R5, #03H	Store number of bytes in a data into R5.
	CLR C	Clear A register
TOP3:	MOV A, @R0	Multibyte data1 starting at address 9200H is added with multibyte data2 starting at address 9250H,Result is stored starting at address 9000H
	ADDC A, @R1 /SUBB A,@R1	
	MOVX @DPTR,A	
	INC DPTR	
	INC R1	
	INC R0	
	DJNZ R5, TOP3	
	MOV A, #00H	Check the status of CY and store the status of CY.
	JNC NXT	
	INC A	
NXT:	MOVX @DPTR,A	
HLT:	SJMP HLT	

List Down The Various Bit Handling Instruction Available In 8051?

sl	MNEMONICS	length	DESCRIPTION
1.	CLR C	1B	Clear the carry bit
2.	CLR BIT	2B	Clear th e indicated bit.
3.	SET C	1B	Set the carry flag.

4.	SET BIT	2B	Set the indicated bit.
5.	CPL C	1B	Complement the carry flag.
6.	CPL BIT	2B	Complement the indicated bit.
7.	MOV C, BIT	2B	The value of the indicated bit is transfer to carry flag.
8.	MOV BIT,C	2B	The value of the carry flag is transfer to indicated bit.
9.	ANL C,BIT	2B	Logical ANDing the carry flag with the indicated bit value. Result is placed in carry.
10.	ANL C,/BIT	2B	Logical ANDING carry flag with the complement of the indicated bit value. Result is placed in carry.
11.	ORL C,BIT	2B	Logical ORING carry flag with the indicated bit . Result is placed in carry.
12.	ORL C,/BIT	2B	Logical ORING the carry flag with the complement of the indicated bit. Result is placed in carry.
13.	JC REL	2B	Jump to relative address when carry=1
14.	JNC BIT,REL	2B	Jump to relative address when carry=0
15.	JB BIT,REL	3B	Jump to relative address if the indicated bit = 1
16.	JNB BIT,REL	3B	Jump to relative address if the indicated bit = 0
17.	JBC BIT,REL	3B	Jump to relative address if the indicated bit = 1 and clear the bit.

Explain the following instruction CJNE Instruction.

Instruction Format : CJNE dest-byte, source-byte,target

Description : The magnitudes of the source byte and destination byte are compared. If they are not equal, it jumps to the target address. Notice that the target address can be no more than 128 bytes backward or 127 bytes forward, since it is a 2-byte instruction.

CJNE jumps only for the not-equal value. To find out if it is greater less after the comparison, we must check the CY flag. After CJNE instruct affects the CY' flag only and after the jump to target address the carry flag indicates which value is greater, as shown here.

(i)Dest < Source CY =1 (ii) Dest> Source CY = 0

This instruction supports four addressing modes. In two of them, A is the destination

1. Immediate: CJNE A, # data, target address

Example: CJNE A, #96, NEXT ; JUMP IF A is not 96

2. Direct : CJNE A, direct, target address

Example: CJNE A, 40H, NEXT ; JUMP IF A NOT= With the Value Held By RAM LOC.

3. Register: CJNE Rn, #data, target

Example: CJNE R5, #70, NEXT ; JUMP if R5 is not 70

4. Indirect: CNJE @Ri, # data, target

Example: CJNE @R1, #80, Next ; Jump if RAM Location whose address is held by R1 is not equal to 80

Explain DJNZ Instruction

Instruction Format : DJNZ Byte,Target

Meaning : Byte is decremented, and if the result is not zero it will jump to target address.

Example : DJNZ R2, BACK ; repeat if R2 not = zero

The following two fomats are supported by this instruction.

1. Register : DJNZ Rn, target (where n=0 to 7)

Ex: DNJZ R3, HERE ; repeat if R3 not = zero

2. Direct: DJNZ direct, target

Use of DJNZ (Looping Program) : Repeating a sequence of instructions a certain number of times is called a loop. **The loop is one of most widely used actions that any microprocessor performs. In the 8051, the loop action is performed by the instruction 'DJNZ reg, Label"**.In this instruction, the register is decremented if it is not zero, it jumps to the target address referred to by the label. Prior to the start of the loop the regsiter is loaded with the counter for the number of repetitions.

Explain the logical instructions with example.

BYTE LEVEL LOGICAL OPERATION(Byte operands) The following are the byte level operations. They use all four addressing modes for the source of a data byte. The A register or direct address in internal RAM is the destination of logical operation result. Note that no flags are affected by the byte level logical operations unless the direct RAM address is the PSW .

SN	MNEMONICS	RESULT STORED IN	ADDRESSING MODE			
			REGISTER	DIRECT	INDIRECT	IMMEDIATE
1	ANL ➔	ACCUMULATOR	ANL A,Rn	ANL A,DIRECT	ANL A,@Ri	ANL A, #8BIT DATA
		INTERNAL RAM	----	ANL DIRECT,A	----	ANL DIR, #8BIT DATA
2	ORL ➔	ACCUMULATOR	ORL A,Rn	ORL A,DIRECT	ORL A,@Ri	ORL A, #8BIT DATA
		INTERNAL RAM	----	ORL DIRECT,A	----	ORL DIR, #8BIT DATA
3	XRL ➔	ACCUMULATOR	XRL A,Rn	XRL A,DIRECT	XRL A,@Ri	XRL A, #8BIT DATA
		INTERNAL RAM	----	XRL DIRECT,A	----	XRL DIR, #8BIT DATA
4	CPL A	ACCUMULATOR	COMPLEMENT ACCUMULATOR			
5	CLR A	ACCUMULATOR	CLEAR ACCUMULATOR			

Logical Instructions : Source is a byte. Byte may be an any one of immediate operand(#data), register operand (Rn) or Memory content(that may be accessed by direct or indirect addressing).

Mnemonic	**Description**	**Explanation**	**Example**	**Result**
ANL A, byte	Logical AND (A) with byte	(A) ← (A)^byte	MOV A,#07 ANL A,#05	(A)←05h
ANL DIR,BYTE	Logical AND content of IRAM with BYTE	(IRAM) ← (IRAM)^BYTE	MOV 45h,#56H ANL 45h,#0FH	((45))←06h
ORL A, byte	Logical OR (A) with byte	(A) ← (A) v (byte)	MOV A,#07 ORL A.#05	(A)←07h
ORL DIR,BYTE	Logical ORing content of IRAM with BYTE	(IRAM) ← (IRAM) v (BYTE)	MOV 45h,#56H ORL 45h,#0FH	((45))←5Fh
XRL A, byte	Logical XOR (A) with byte	(A) ← (A)⊻(byte)	MOV A,#07 XRL A,#05	(A)←02h
XRL DIR,BYTE	Logical XORing content of IRAM with BYTE	(IRAM) ← (IRAM) ⊻ (BYTE)	MOV 45h,#56H XRL 45h,#0FH	((45))←59h
CPL A	COMPLEMENT (A)	(A) ← COMPLEMENT OF (A)	MOV A,#3FH CPL A	(A)←C0h
CLR A	CLEAR (A)	(A)← $(00000000)_2$	MOV A,#42H CLR A	(A) ← 00h

SNO	MNEMONICS	DESCRIPTION
1	ANL A,#N	AND each bit of A with the same bit of immediate number 'N'. Put the result in A
2	ANL A, DIRECT	AND each bit of A with the same bit of the direct RAM address. Put the result in A
3	ANL A, Rr	AND each bit of A with the same bit of Register Rr. Put the result in A
4	ANL A,@RP	AND each bit of A with the same bit of the content of RAM address contained in RP. Put the result in A
5	ANL DIRECT,A	AND each bit of A with the same bit of the direct RAM address. Put the result in direct RAM address.
6	ANL DIRECT,#N	AND each bit of the direct RAM address with the same bit of immediate number 'N'. Put the result in the direct RAM address .
7	ORL A,#N	OR each bit of A with the same bit of immediate number 'N'. Put the result in A
8	ORL A, DIRECT	OR each bit of A with the same bit of the direct RAM address. Put the result in A
9	ORL A, Rr	OR each bit of A with the same bit of Register Rr. Put the result in A
10	ORL A,@RP	OR each bit of A with the same bit of the content of RAM address contained in RP. Put the result in A
11	ORL DIRECT,A	OR each bit of A with the same bit of the direct RAM address. Put the result in direct RAM address.
12	ORL DIRECT,#N	OR each bit of the direct RAM address with the same bit of immediate number 'N'. Put the result in the direct RAM address .
13	XRL A,#N	XOR each bit of A with the same bit of immediate number 'N'. Put the result in A
14	XRL A, DIRECT	XOR each bit of A with the same bit of the direct RAM address. Put the result in A
15	XRL A, Rr	XOR each bit of A with the same bit of Register Rr. Put the result in A
16	XRL A,@RP	XOR each bit of A with the same bit of the content of RAM address contained in RP. Put the result in A
17	XRL DIRECT,A	XOR each bit of A with the same bit of the direct RAM address. Put the result in direct RAM address.
18	XRL DIRECT,#N	XOR each bit of the direct RAM address with the same bit of immediate number 'N'. Put the result in the direct RAM address .
19	CPL A	COMPLEMENT (A). Every one becomes zero. Every zero becomes one.
20	CLR A	CLEAR CONTENT OF ACCUMULATOR.

Explain I/O programming with example.

- In 8051, I/O devices should be treated as memory.
- So memory mapped I/O is only possible for accessing I/O devices.
- MOVX instructions are used for accessing both memory and IO devices.
- By using bit manipulation instructions, we can perform various functions like clear,set,complement,AND,OR ,move and control jump in bit level.
- Example : Write an ALP for receiving data in a port pin P2.0 and store it in memory location 2FH

LABEL	Mnemonics	Description
	SETB P2.0	Set the port pin P2.0 ,make P2.0 as input pin
	MOV C,P2.0	Move the bit data inP2.0 to Carry
	MOV 2FH,C	Store the data in bit location 2FH

HLT:	SJMP HLT	

Explain Time delay routines with examples.

Delay routines are subroutines used for maintaining the timing of various operations in 8051.

- In some applications,certain operations are repeated after a specified time interval.
- In Control applications,certain equipment is needed to be ON/OFF after a specified time delay.
- In such cases, Delay routines can be used to maintain the timing of operations.

Example : Delay program with one loop:

Mnemonics		No of m/c cycles	No of times executed	Total No of machine cycles
Label	Commands			
Delay:	MOV RO,#0FFH	1	1	1
LOOP:	DJNZ R0,LOOP	2	$(255)_{10}$	$(500)_{10}$
	RET	2	1	2
			TOTAL	$(503)_{10}$

- (assume XTAL frequency = 11.0592MHz)
- Total time period produced by this subroutine = 503 * 1.085microsecond.
- This delay subroutine can produce a maximum delay of 545 microsecond.

What is the difference between the flags CY and OV.

This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations.

Write an ALP to convert a hexa decimal to its equivalent BCD number. (08)

LABEL	MNEMONICS	COMMENT
	MOV DPTR, #4150h	INITIALIZE DPTR WITH INPUT ADDRESS 4150H.
	MOVX A, @DPTR	(A) ← INPUT DATA AT ADDRESS 4150H.
	MOV B, #64H	TO FIND OUT NUMBER OF 100'S IN A GIVEN DATA, IT IS DIVIDED BY 100(64H).
	DIV AB	
	MOV DPTR, #4250H	QUOTIENT IS STORED AT ADDRESS 4250H
	MOVX @DPTR, A	
	MOV A, B	REMAINDER IS MOVED TO A REGISTER
	MOV B, #0AH	TO FIND OUT NUMBER OF 100'S IN A GIVEN DATA, IT IS DIVIDED BY 10(0AH).
	DIV AB	
	INC DPTR	QUOTIENT IS STORED AT ADDRESS 4251H
	MOVX @DPTR, A	
	INC DPTR	REMAINDER IS MOVED TO A REGISTER AND STORED AT ADDRESS 4252H
	MOV A, B	
	MOVX @DPTR, A	

LABEL	MNEMONICS	COMMENT
HLT:	SJMP HLT	

INPUT				OUTPUT			
ADDRESS	DATA1		COMMENTS	ADDRESS	DATA2		COMMENTS
	S1	S2			S1	S2	RESULT BCD
4150H	FF	25	DATA IN HEXA	4250	02	00	100'S
				4251	05	03	10'S
				4252	05	07	1'S

Explain the format of IE & IP register. (08)

Format of IE register.

a) 8 bit register. Register Address **0A8h**

b) interrupt enable register.(byte/bit addressable register)

c) To enable/disable All of the interrupts in 8051,we make high/low in the bit **EA of** the IE register .

d)To enable/disable any one of the interrupts,we make the corresponding bit in the IE register high/low.

←----------8bit IE----------							
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
EA	**---**	**---**	**ES**	**ET1**	**Ex1**	**ET0**	**EX0**

Bit	Bit	function	Bit	Bit	Function
7	EA	Enable all interrupts	3	ET1	Timer 1 interrupt enable
6		-----	2	EX1	External interrupt 1 interrupt
5		-----	1	ET0	Timer 0 interrupt enable
4	ES	Serial port interrupt	0	EX0	External interrupt01 interrupt

Format of IP register : a) 8 bit register.
b) interrupt priority register.
(byte/bit addressable register)
c) Upon reset , IP register contains all 0's.
d) With the use of IP register,user may change the priority. d) To give a higher priority to any of the interrupts,we make the corresponding bit in the IP register high.

←----------8bit IP----------→							
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
---	**---**	**---**	**PS**	**PT1**	**Px1**	**PT0**	**PX0**

Bit no	Bit name	Function	Bit no	Bit	Function
7		-----	3	PT1	Timer 1 interrupt priority
6		-----	2	PX1	External interrupt 1 interrupt priority
5		-----	1	PT0	Timer 0 interrupt priority
4	PS	Serial port	0	PX0	External interrupt0 interrupt priority

---

Explain External interrupts with suitable program . [08]

PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

The 8051 has two external hardware interrupts. Piri 12 (P3.2) and pin 13 (P3.3) of the 8051, designated as INTO and INT1, are used as external hardware interrupts. Upon activation of these pins, the 8051 gets interrupted in whatever it is doing and jumps to the vector table to perform the interrupt service routine. In this section we study these two external hardware interrupts of the 8051 with some examples.

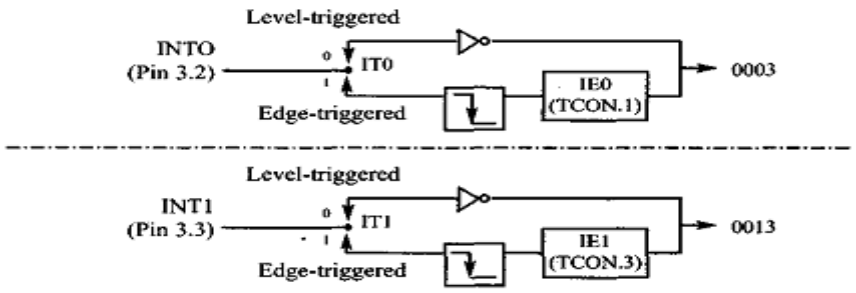


Figure A. Activation of INTO and INT1 External interrupts INTO and INT1

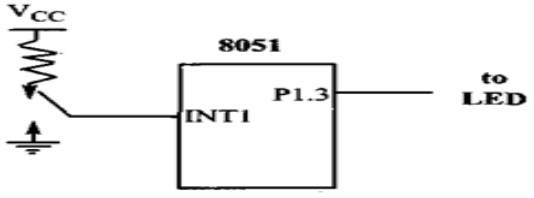
There are only two external hardware interrupts in the 8051: INTO and INT1. They are located on pins P3.2 and P3.3 of port 3, respectively. The interrupt vector table locations 0003H and 0013H are set aside for INTO and INT1, respectively. They are enabled and disabled using the IE register. There are two types of activation for the external hardware interrupts: (1) level triggered, and (2) edge triggered.

Level-triggered interrupt

In the level-triggered mode, INTO and INT1 pins are normally high (just like all I/O port pins) and if a low-level signal is applied to them, it triggers the interrupt. Then the microcontroller stops whatever it is doing and jumps to the interrupt vector table to service that interrupt. This is called a *level-triggered* or *level-activated interrupt* and is the default mode upon reset of the 8051. The low-level signal at the INT pin must be removed before the execution of the last instruction of the interrupt service routine, RETI; otherwise, another interrupt will be generated. In other words, if the low-level interrupt signal is not removed before the ISR is finished it is interpreted as another interrupt and the 8051 jumps to the vector table to execute the ISR again. Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to PI .3 and is normally off. When it is turned on it should stay on for a fraction of a second. As long as the switch is pressed low, the LED should stay on.

```
            ORG   0000H
            LJMP  MAIN              ;bypass interrupt vector table
;--ISR for hardware interrupt INT1 to turn on the LED
            ORG   0013H             ;INT1 ISR
            SETB  P1.3              ;turn on LED
            MOV   R3,#255           ;load counter
BACK:       DJNZ  R3,BACK           ;keep LED on for a while
            CLR   P1.3              ;turn off the LED
            RETI                    ;return from ISR
;--MAIN program for initialization
            ORG   30H
MAIN:       MOV   IE,#10000100B     ;enable external INT1
HERE:       SJMP  HERE              ;stay here until interrupted
            END
```

Pressing the switch will turn the LED on. If it is kept activated, the LED stays on.



Write an assembly language program to generate a square wave of 5 KHz on port pin 1.5 using timer mode 2. Assume crystal frequency to be 11.0592 MHz. (08)

Crystal frequency = 11.0592 MHz : Frequency to timer = 11.0592 MHz /12 = 921.6KHz

Clock period to timer = (1/921.6KHz) = **1.085 microsecond**

a) in Hex T = Time delay = (FF - XX +1) x 1.085 microsecond

where XX are TL initial value.Notice that value XX is in hex.

b) in decimal: Convert XX value of the TL register to decimal to get a NNN decimal number, then T = Time delay = (256 - NNN) x 1.085 microsecond

Programming Steps to generate a time delay using the timer's mode 2

- T = 1/5 KHz = 0.2ms
- 1/2 of it for the high and low portions of the pulse = 0.1ms
- 0.1 ms / 1.085 μs = 92.1659
- 256 – 92.17 = 163.84 = 164(approximately) in decimal = 0A4H
- TL1 = TH1 = 0A4H

In programming the 8051 to generate a time delay using the timer's mode 2,

the following steps must be taken.

1) The TMOD register is loaded with the value indicating which timer(timer 1or timer 0) and which mode is selected.(mode 2: **8 –bit Auto Reload timer/counter**) .
2) The THx and TLx is loaded with the initial value for the desired time delay. (assuming XTAL=11.0592 MHz)

MOV TH0,#XX	Initialise TH0 with initial value.
MOV TH1,#XX	Initialise TH1 with initial value.

3) Start the timer. **SETB TR0** : Start timer 0. & **SETB TR1** : Start timer 1.
4) The TFx flag bit is monitored with the use of the following instruction to see if TIMER overflow is occurred. **"wait : JNB TF0, wait"** (for Timer 0)
and **"wait : JNB TF1, wait"** (for Timer 1)
5) Clear the timer overflow flag. **CLR TF0** ; To Clear timer overflow flag 0.
CLR TF1 ; To Clear timer overflow flag 1.
6) Stop the timer. **CLR TR0** : To Stop timer 0. And **CLR TR0** : To Stop timer 0.
7) go to step 3 **since it is auto reload(for mode 2, no need to initilaise the value)**.

PROGRAM: To generate a square wave at P1.5 using timer 0 in mode 2.

Step	Label	Mnemonics	Comments
1		MOV TMOD,#02H	; timer 0, mode 2 (8 BIT auto reload)
2		MOV TH0,#0A4H	; Initialise TH0 with initial value for timer 0.
3	Again:	CPL P1.5	Complement P1.5
4		SETB TR0	; start timer 0
5	HERE:	JNB TF0,HERE	;monitor timer 0 overflow flag.
6		CLR TR0	;stop the timer 0
7		CLR TF0	;TF0 is cleared for next round
8		SJMP Again	;repeat the process

PROGRAM: To generate a square wave at P1.5 using timer 1 in mode 2.

Step	Label	Mnemonics	Comments
1		MOV TMOD,#20H	; timer 1, mode 2 (8 BIT auto reload)
2		MOV TH1,#A4H	; Initialise TH1 with initial value for timer 1.
3	Again:	CPL P1.5	Complement P1.5
4		SETB TR1	; start timer 1
5	HERE:	JNB TF1,HERE	;monitor timer 1overflow flag.
6		CLR TR1	;stop the timer 1
7		CLR TF1	;TF1 is cleared for next round
8		SJMP Again	;repeat the process [**since it is auto reload(for mode 2, no need to**

PROGRAM (USING TIMER INTERRUPTS):

Program to generate a square wave at P1.5 using timer 0 in mode 2.

Step	Label	Mnemonics	Comments	
1		MOV TMOD,#02H	; timer 0, mode 2 (8 BIT auto reload)	
2		MOV IE,#10001010B	; B- BINARY. Enable timer interrupts.(both TF0 & TF1)	
3		MOV TH0,#0A4H	; Initialise TH0 with initial value for timer 0.	
4	Again:	CPL P1.5	Complement P1.5	
		SETB TR1	; start timer 1	
6	HERE:	JNB TF0,HERE	;monitor timer 0 overflow flag.	
7		SJMP Again	;repeat the process.[**since it is auto reload(for mode 2, no need to initilaise the value)**.	
ISR	000Bh	CLR TR0	;stop the timer 0	Vector address is 000Bh
1		CLR TF0	;TF0 is cleared for next round	
2		RETI	Return from ISR **[Interrupt Service Routine]**	

PROGRAM (USING TIMER INTERRUPTS):

Program to generate a square wave at P1.5 using timer 1 in mode 2.

Step	Label	Mnemonics	Comments	
1		MOV TMOD,#20H	; timer 1, mode 2 (8 BIT auto reload)	
2		MOV IE,#10001010B	; B- BINARY. Enable timer interrupts.(both TF0 & TF1)	
3		MOV TH1,#A4H	; Initialise TH1 with initial value for timer 1.	
4	Again:	CPL P1.5	Complement P1.5	
5		SETB TR1	; start timer 1	
6	HERE:	JNB TF1,HERE	;monitor timer 1overflow flag.	
7		SJMP Again	;repeat the process [**since it is auto reload(for mode 2, no**	
ISR	001Bh	CLR TR1	;stop the timer 1	Vector Address is 001Bh
1		CLR TF1	;TF1 is cleared for next round	
2		RETI	Return from ISR **[Interrupt Service Routine]**	

We can develop a formula for delay calculations using **MODE 1 (16-BIT) OF THE TIMER** for a crystal frequency of XTAL= 11.0592 MHz. This is given in calculation.

Crystal frequency = 11.0592 MHz : Frequency to timer = 11.0592 MHz /12 = 921.6KHz

Clock period to timer = (1/921.6KHz) = **1.085 microsecond**

a) in Hex

T = Time delay = (FFFF - YYXX +1) x 1.085 microsecond

where YYXX are TH, TL initial values respectively. Notice that values YYXX are in hex.

b) in decimal: Convert YYXX values of the TH, TL register to decimal to get a NNNNN decimal number, then T = Time delay = (65536 - NNNNN) x 1.085 microsecond

Programming Steps to generate a time delay using the timer's mode 1

- T = 1/50 Hz = 20 ms
- 1/2 of it for the high and low portions of the pulse = 10 ms
- 10 ms / 1.085 us = 9216
- 65536 - 9216 = 56320 in decimal = DC00H
- TL = 00 and TH = DCH
- The calculation for 12MHz crystal uses the same steps

In programming the 8051 to generate a time delay using the timer's mode 1,

the following steps must be taken.

1) The TMOD register is loaded with the value indicating which timer(timer 1or timer 0) and which mode is selected.(mode 1:16-bit timer/counter) .
2) The THx and TLx is loaded with the initial value for the desired time delay.
(assuming XTAL=11.0592 MHz)

MOV TH1,#yy	Initialise TH1 & TL1
MOV TL1,#xx	with initial value.

MOV TH0,#yy	Initialise TH0 & TL0 with
MOV TL0,#xx	initial value.

3) Start the timer. **SETB TR1** : Start timer 1. And **SETB TR0** : Start timer 0.
4) The TF1/TF0 flag bit is monitored with the use of the instruction **"xx : JNB TF1, xx"** OR **"xx : JNB TF0,xx"** to see if TIMER overflow is occurred.
5) Clear the timer overflow flag.
CLR TR1 : To clear timer overflow flag 1. & **CLR TR0** ; To Clear timer overflow flag 0.
6) Stop the timer. **CLR TR1** : To Stop timer 1. & **CLR TR0** : To Stop timer 0.
7) go to step 2 **since it is not auto reload(for mode 1, need to initilaise the value)**.

Example: Write a program for the 8051 to generate the max.time delay using timer 1 in mode 1.
For maximum time delay, initial value should be 0000h.

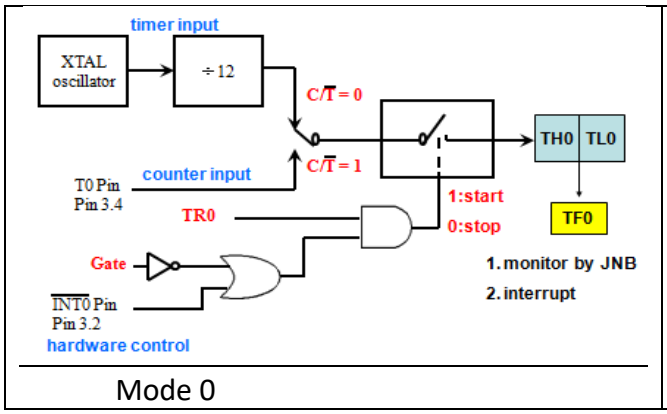
Label	Mnemonics	Comments
	MOV TMOD,#10H	; timer 1, mode 1 (16 BIT)
	MOV IE , #82H	; Enable Timer interrupts
Again:	MOV TH1,#0DCH	; Initialise TH1 with initial value.
	MOV TL1,#00	; Initialise TL1 with initial value.
	CPL P.5	;Generate a square wave of frequency 50Hz
	SETB TR1	; start timer 1
HERE:	JNB TF1,HERE	;monitor timer overflow flag.
	SJMP again	;repeat the process
001Bh	CLR TR1	;stop the timer
	CLR TF1	;TF1 is cleared for next round
	RETI	;return from interrupt service routine.

Explain the various modes of timer operation with diagram(s). (08)

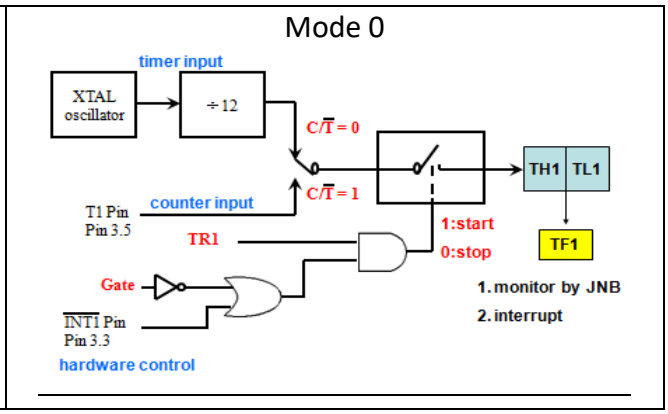
- The **timer** is used as a time delay generator. The clock source is the **internal** crystal frequency of the 8051.
- 8051 timers use 1/12 of XTAL frequency as the input of timers, regardless of machine cycle.
- Because the input of timer is a regular, fixed-periodic square wave, we can count the number of pulses and calculate the time delay.

M1	M0	MODE	DESCRIPTION
0	0	MODE 0	13 bit Timer/counter
0	1	MODE 1	16 bit Timer/counter
1	0	MODE 2	8-bit Auto Reload Timer/counter
1	1	MODE 3	(two 8 – bit) split Timer/counter

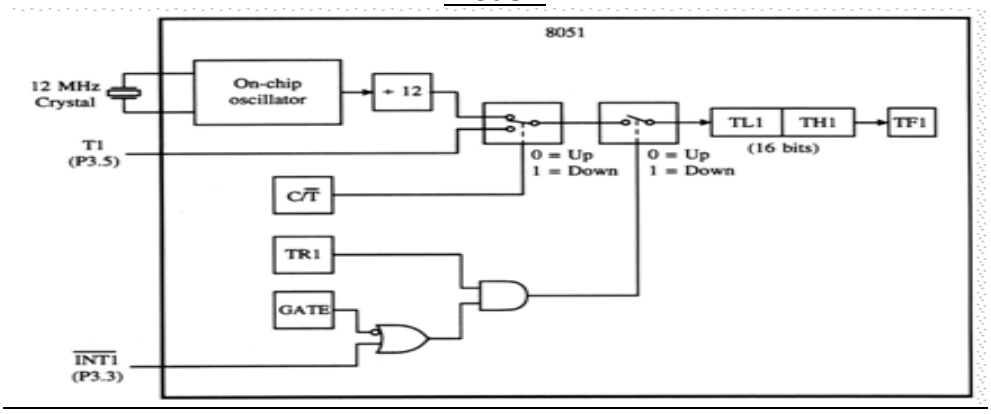
Timer/Counter Selection:



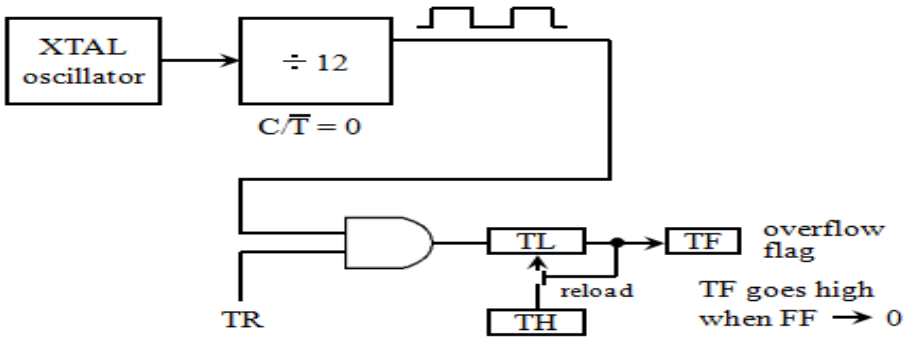
Mode 0



Mode 1



Mode 2



Mode 3 : For Timer/Counter 0

- As a Timer: TH0 and TL0 can be two **8-bit timers**.
- As a Counter: TL0 can be an **8-bit counter**. For Timer/Counter 1 : Not available

---

Explain various modes of serial communication in detail. (08)

Serial Data Transmission Modes:

Mode-0: shift register

- In this mode, the serial port works like a shift register.
- The data transmission works synchronously with a clock frequency of f_{osc} /12.
- Serial data is received and transmitted through RXD.
- 8 bits are transmitted/ received at a time.
- Pin TXD outputs the shift clock pulses of frequency f_{osc} /12, which is connected to the external circuitry for synchronization.
- The shift frequency or baud rate is always 1/12 of the oscillator frequency.

Mode-1 (standard UART mode) :

- Serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode.
- 10 bits are transmitted through TXD or received through RXD.
- The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1').
- Once received, the stop bit goes into RB8 in the special function register SCON.
- The baud rate is variable.

Serial Data Mode-2 - Multiprocessor Mode : In this mode 11 bits are transmitted through TXD or received through RXD.

- The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th (TB8 or RB8)bit and a stop bit (usually '1').
- While transmitting, the 9 th data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8.
- On reception of the data, the 9 th bit goes into RB8 in 'SCON', while the stop bit is ignored.
- The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency.
  f baud = $(2^{SMOD}/64)$ fosc.

Mode-3 - Multi processor mode with variable baud rate :

- In this mode 11 bits are transmitted through TXD or received through RXD.
- The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th bit and a stop bit (usually '1').
- Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1).
- f baud = $(2^{SMOD}/32) * (fosc / 12 (256-TH1))$.
- This baud rate holds when Timer-1 is programmed in Mode-2.

SMO	SM1	MODE	DESCRIPTION
0	0	Mode 0	Shift register, baud =1/12
0	1	Mode 1	8-bit UART, baud = variable
1	0	Mode 2	9-bit UART, baud = 1/32 or 1/64
1	1	Mode 3	9 - bit UART, baud = variable

Explain the steps to receive a character serially with example.

Programming the 8051 to Receive Data Serially, the following steps must be taken.

1) The TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8 bit auto reload) to set the baud rate
2) TH1 is loaded with one of the values in table to set the baud rate (assuming XTAL = 11.0592 MHz).
3) The SCON register is loaded with the value 50H, indicating serial mode 1, where 8-bit data is framed with Start and stop bits.
4) TR1 is set to 1 to start timer 1
5) RI is cleared with the “CLR RI” instruction.
6) The RI flag bit is monitored with the use of the instruction “JNB RI, xx” to see if an entire character has Been received yet.
7) When RI is raised, SBUF has the byte, Its contents are moved into a safe place.
8) To receive the next character, go to step 5.

Example : Program the 8051 to receive bytes of data serially, and put them in P1, Set the baud rate at 4800, 8-bit data, and 1 stop bit.

Label	Mnemonics	Comments
	MOV TMOD,#20H	timer 1, mode 2 (auto reload)
	MOV THI ,#-6	Set 4800 baud rate

	MOV SCON,#50H	Initialise SCON with 8-bit, 1 stop bit, REN enabled
	SETB TR1	Start timer 1
HERE:	JNB RI,HERE	Wait for character to come in
	MOV A,SBUF	Save incoming byte in A
	MOV P1,A	Send to port1
	CLR RI	Get ready to receive next byte
	SJMP HERE	Keep getting data.

Explain the steps to **Transfer** a character serially with example.

Programming the 8051 to Transfer Data Serially, the following steps must be taken.

1) The TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8-bit auto- reload) to set the baud rate.
2) The TH1 is loaded with one of the values in table to set the baud rate for serial data ' transfer (assuming XTAL=11.0592 MHz)
3) The SCON register is loaded with the value 50H, indicating serial model, where an 8- bit data is Framed with start and stop bits.
4) TR1 is set to 1 to start timer 1.
5) TI is cleared by the "CLR TI" instruction.
6) The character byte to be transferred serially is written into the SBUF register.
7) The TI flag bit is monitored with the use of the instruction "JNB TI, xx" to see if the character had been transferred completely.
8) To transfer the next character, go to step 5.

Example : Program the 8051 to transfer bytes of data serially, Set the baud rate at 4800, 8-bit data, and 1 stop bit.

Label	Mnemonics	Comments
	MOV TMOD,#20H	timer 1, mode 2 (auto reload)
	MOV THI ,#-6	Set 4800 baud rate
	MOV SCON,#50H	Initialise SCON with 8-bit, 1 stop bit, REN enabled
	SETB TR1	Start timer 1
Step5:	CLR TI	Get ready to transfer next byte
	MOV SBUF, #'A'	character byte to be transferred serially is written into the SBUF
HERE:	JNB TI,HERE	Wait for the character had been transferred completely
	SJMP step5	Go to step 5

Describe 8051 serial communication programming.

Programming the 8051 to Transfer Data Serially :

In programming thç 8051 to transfer character bytes serially, the following steps must be taken.

1) The TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8-bit auto- reload) to set the baud rate.

2) The TH1 is loaded with one of the values in table to set the baud rate for serial data ' transfer Aassuming XTAL=11.0592 MHz
3) The SCON register is loaded with the value 50H, indicating serial model, where an 8- bit data is framed with start and stop bits.
4) TR1 is set to ito start timer 1.
5) TI is cleared by the "CLR TI" instruction.
6) The character byte to be transferred serially is written into the SBUF register.
7) The TI flag bit is monitored with the use of the instruction "JNB TI, xx" to see if the character had been transferred completely.
8) To transfer the next character, go to step 5.

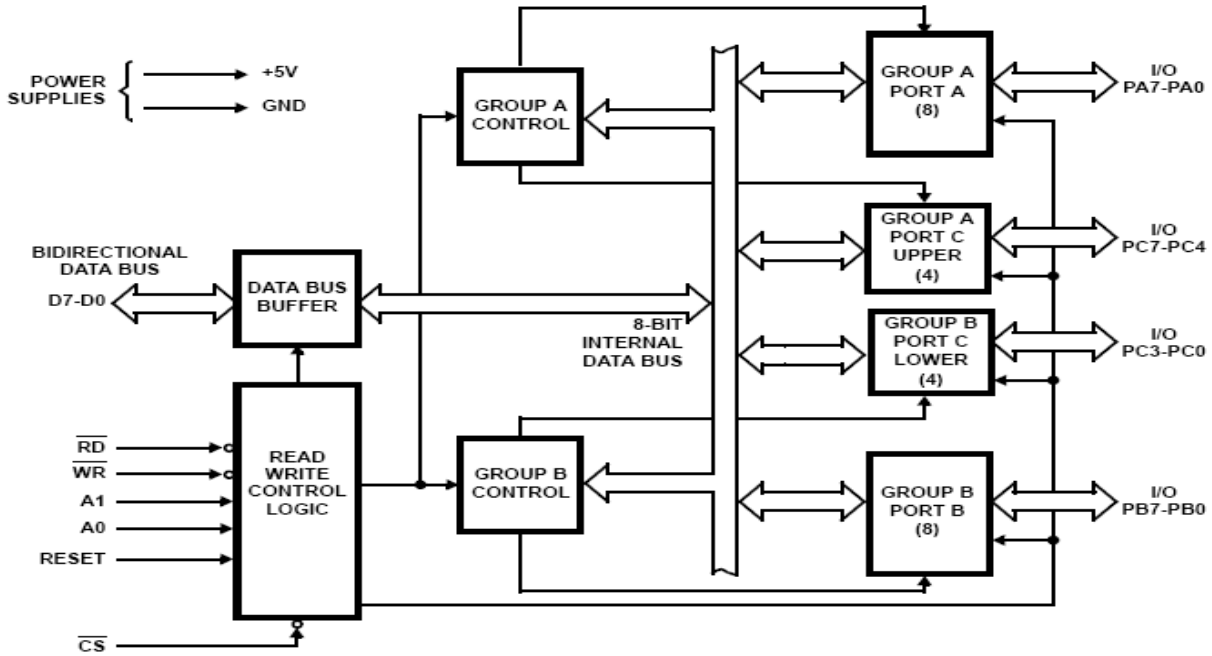
```
        MOV   TMOD,#20H  ;Timer 1, mode 2(auto-reload)
        MOV   TH1,#-6    ;4800 baud rate
        MOV   SCON,#50H  ;8-bit, 1 stop, REN enabled
        SETB  TR1        ;start Timer 1
AGAIN:  MOV   SBUF,#"A"  ;letter "A" to be transferred
HERE:   JNB   TI,HERE    ;wait for the last bit
        CLR   TI         ;clear TI for next char
        SJMP  AGAIN      ;keep sending A
```

Programming the 8051 to Receive Data Serially In the programming of the 8051 to received character bytes serially, the following steps must be taken.

1) The TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode 2 (8 bit auto reload) to set the baud rate
2) TH1 is loaded with one of the values in table to set the baud rate (assuming XTAL = 11.0592 MHz).
3) The SCON register is loaded with the value 50H, indicating serial mode 1, where 8-bit data is framed with Start and stop bits.
4) TR1 is set to 1 to start timer 1
5) RI is cleared with the "CLR RI" instruction.
6) The RI flag bit is monitored with the use of the instruction "JNB Rh xx" to see if an entire character has Been received yet.
7) When RI is raised, SBUF has the byte, Its contents are moved into a safe place.
8) To receive the next character, go to step 5.

UNIT V

Draw and Explain the block diagram of 8255. (08)



Features:

- It is a programmable device.
- It has 24 I/O programmable pins like PA,PB,PC (3-8 pins).
- T T L compatible.
- Improved dc driving capability

Function of pins:

A1	A0	Select
0	0	PA
0	1	PB
1	0	PC
1	1	Control register

- Data bus(D_0-D_7):These are 8-bit bi-directional buses, connected to 8085 data bus for transferring data.
- CS: This is Active Low signal. When it is low, then data is transfer from 8085.
- Read: This is Active Low signal, when it is Low read operation will be start.
- Write: This is Active Low signal, when it is Low Write operation will be start.
- Address (A_0-A_1):This is used to select the ports. like this →
- RESET: This is used to reset the device. That means clear control registers.
- PA_0-PA_7:It is the 8-bit bi-directional I/O pins used to send the data to peripheral or to receive the data from peripheral.
- PB_0-PB_7:Similar to PA
- PC_0-PC_7:This is also 8-bit bidirectional I/O pins. These lines are divided into two groups.
  - PC_0 to PC_3(Lower Groups)
  - PC_4 to PC_7 (Higher groups)
  - These two groups working in separately using 4 data's.
- **Data Bus buffer**:
- It is a 8-bit bidirectional Data bus.
- Used to interface between 8255 data bus with system bus.
- The internal data bus and Outer pins D_0-D_7 pins are connected in internally.
- The direction of data buffer is decided by Read/Control Logic.
- **Read/Write Control Logic:**

- **This is getting the input signals from control bus and Address bus**
- **Control signal are RD and WR.**
- **Address signals are A0,A1,and CS.**
- **8255 operation is enabled or disabled by CS.**
- **Group A and Group B control:**
  - Group A and B get the Control Signal from CPU & send the command to the individual control blocks.
  - Group A send the control signal to port A and Port C (Upper) PC7-PC4.
  - Group B send the control signal to port B and Port C (Lower) PC3-PC0.

PORT A:

- This is a 8-bit buffered I/O latch.
- It can be programmed by mode 0 , mode 1, mode 2 .

PORT B:

- This is a 8-bit buffer I/O latch.
- It can be programmed by mode 0 and mode 1.

PORT C:

- This is a 8-bit Unlatched buffer Input and an Output latch.
- It is splitted into two parts.
- It can be programmed by bit set/reset operation.

Mention the control word of BSR mode of 8255.

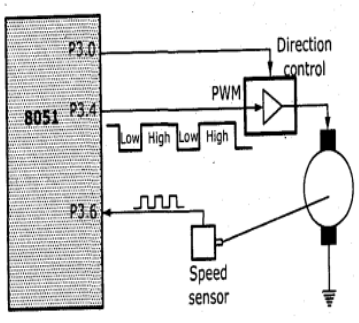
BSR mode : **B**it **S**et **R**eset mode : Setting/Resetting the bits of Port C in 8255.

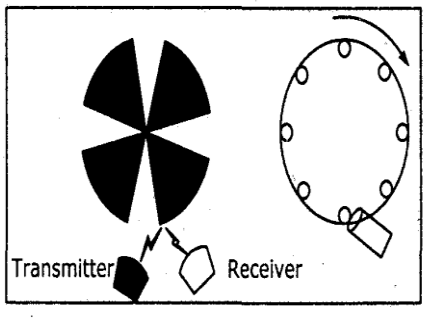
←----------8bit Control word register----------→							
Bit7 (D7)	**Bit6 (D6)**	**Bit5 (D5)**	**Bit4 (D4)**	**Bit3 (D3)**	**Bit2 (D2)**	**Bit1 (D1)**	**Bit0 (D0)**
BSR = 0	**←---Not Used----→**			**←--Bit select---→**			**S/R (S = 1; R=0)**

←----------Bit Select (D3 D2 D1) in PORT C----------→							
111	**110**	**101**	**100**	**011**	**010**	**001**	**000**
Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**

Draw the connection diagram of 8051 with DC motor interfacing with 8051. (08)

DC Motor interface Diagram.





Programmable peripheral interface 8255

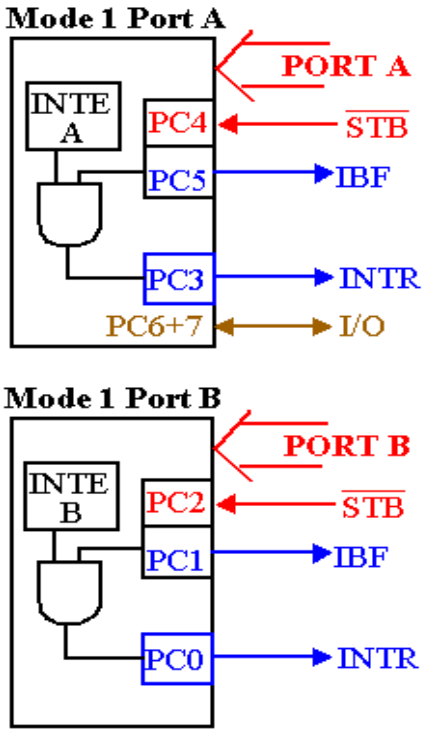
Mode 1 strobed Input

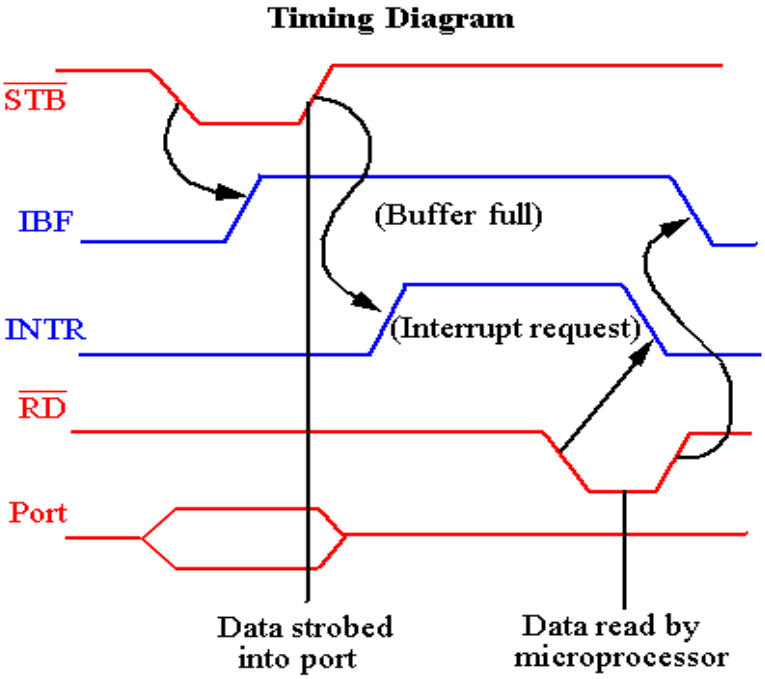
Port A and/or port B function as latching input devices. External data is stored in the ports until the microprocessor is ready.

Port C used for control or handshaking signals (cannot be used for data).

Signal definitions for Mode 1 Strobed Input

$\overline{STB}$	The strobe input loads data into the port latch on a 0-to-1 transition
IFB	**Input buffer full** is an output indicating that the input latch contain information
INTR	**Interrupt request** is an output that requests an interrupt
INTE	The **interrupt enable signal** is neither an input nor an output; it is an internal bit programmed via the PC4(port A) or PC2(port B) bits.
PC7,PC6	The port C pins 7 and 6 are general-purpose I/O pins that are available for any purpose.





MODE 1 STROBED OUTPUT

Similar to Mode 0 output operation, except that handshaking signals are provided using port C.

Signal Definitions for Mode 1 Strobed Output

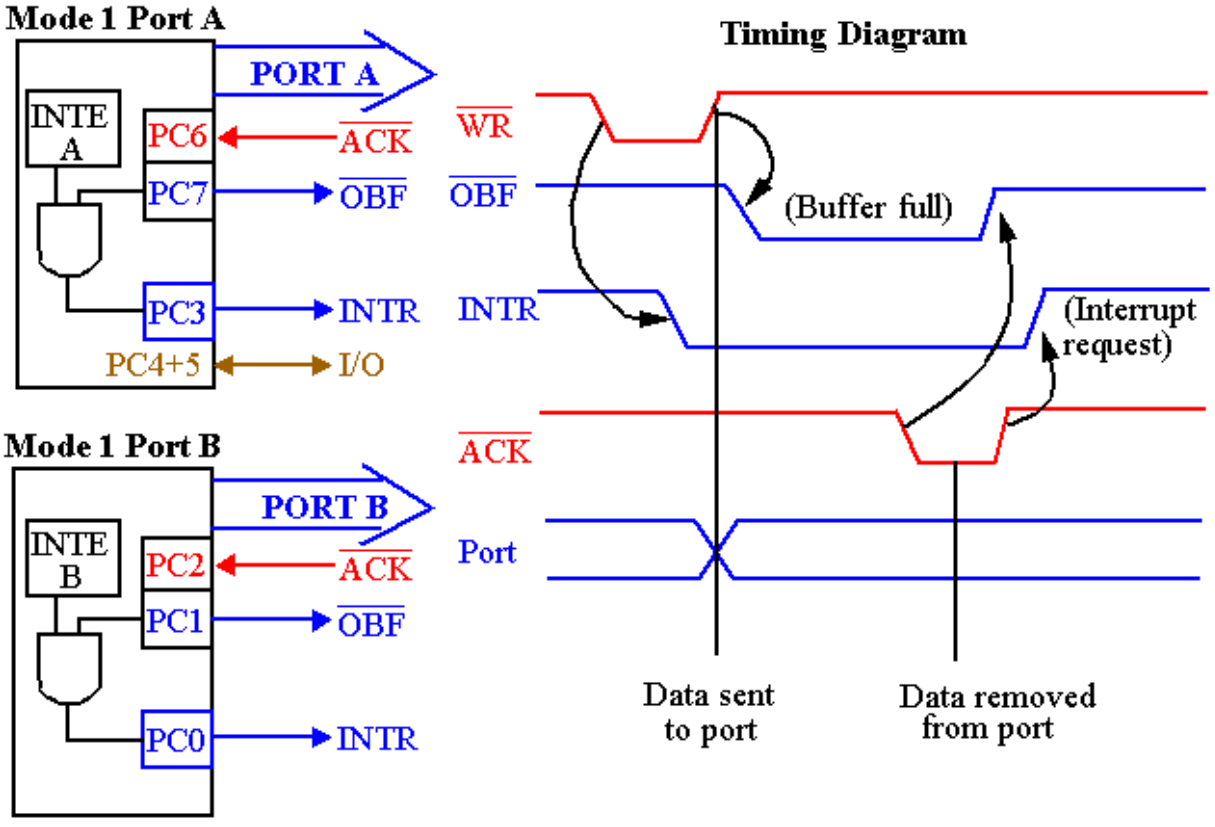
$\overline{OBF}$ **Output buffer full** is an output that goes low when data is latched in either port A or port B. Goes low on $\overline{ACK}$.

$\overline{ACK}$ The **acknowledge** signal causes the $\overline{OBF}$ pin to return to 0. This is a response from an external device.

INTR **Interrupt request** is an output that requests an interrupt

INTE The **interrupt enable signal** is neither an input nor an output; it is an internal bit programmed via the PC6(port A) or PC2(port B) bits.

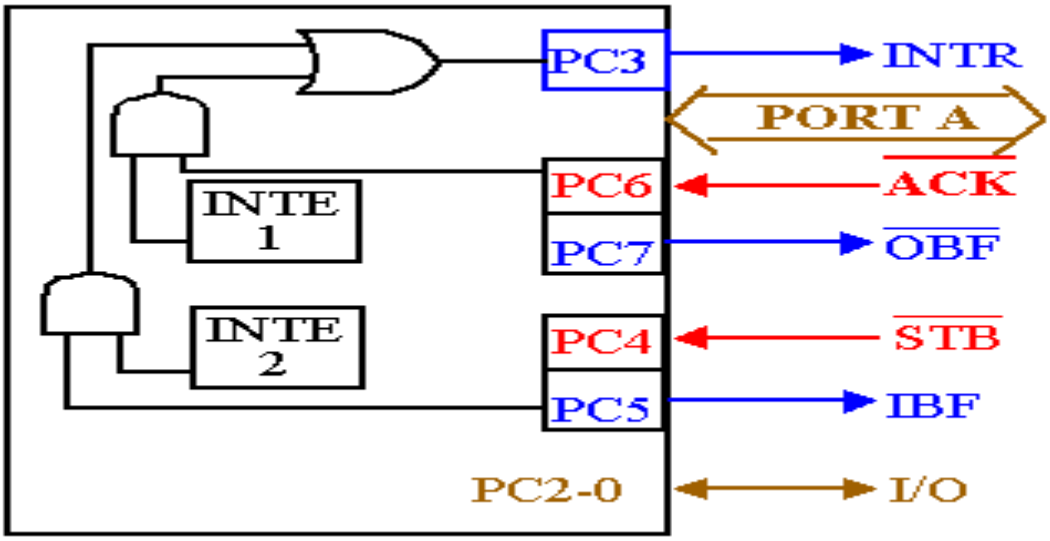
PC5,PC4 The port C pins 5 and 4 are general-purpose I/O pins that are available for any purpose.



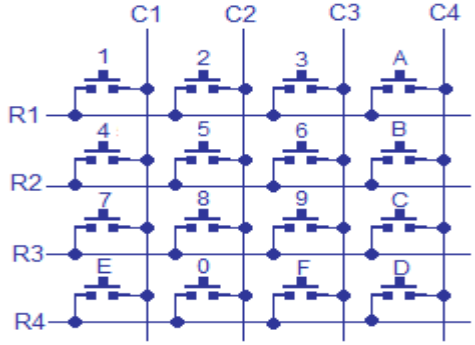
MODE 2 BIDIRECTIONAL OPERATION

Only allowed with port A. Bi-directional bused data used for interfacing two computers, GPIB interface etc.

INTR	**Interrupt request** is an output that requests an interrupt
$\overline{OBF}$	**Output buffer full** is an output indicating that the output buffe: contains data for the bi-directional bus
$\overline{ACK}$	**Acknowledge** is an input that enables tri-state buffers which are otherwise in their high-impedance state
$\overline{STB}$	The strobe input loads data into the port A latch
IFB	**Input buffer full** is an output indicating that the input latch contains information for the external bi-directional bus
INTE	**Interrupt enable** are internal bits that enable the INTR pin. Bit PC6(INTE1) and PC4(INTE2)
PC2,PC1 and PC0	Theses port C pins are general-purpose I/O pins that are available for any purpose.



Explain the interfacing diagram of matrix keyboard with 8051. (08)



Hex key pad is essentially a collection of 16 keys arranged in the form of a 4×4 matrix. Hex key pad usually have keys representing numerics 0 to 9 and characters A to F. The simplified diagram of a typical hex key pad is shown in the figure below.

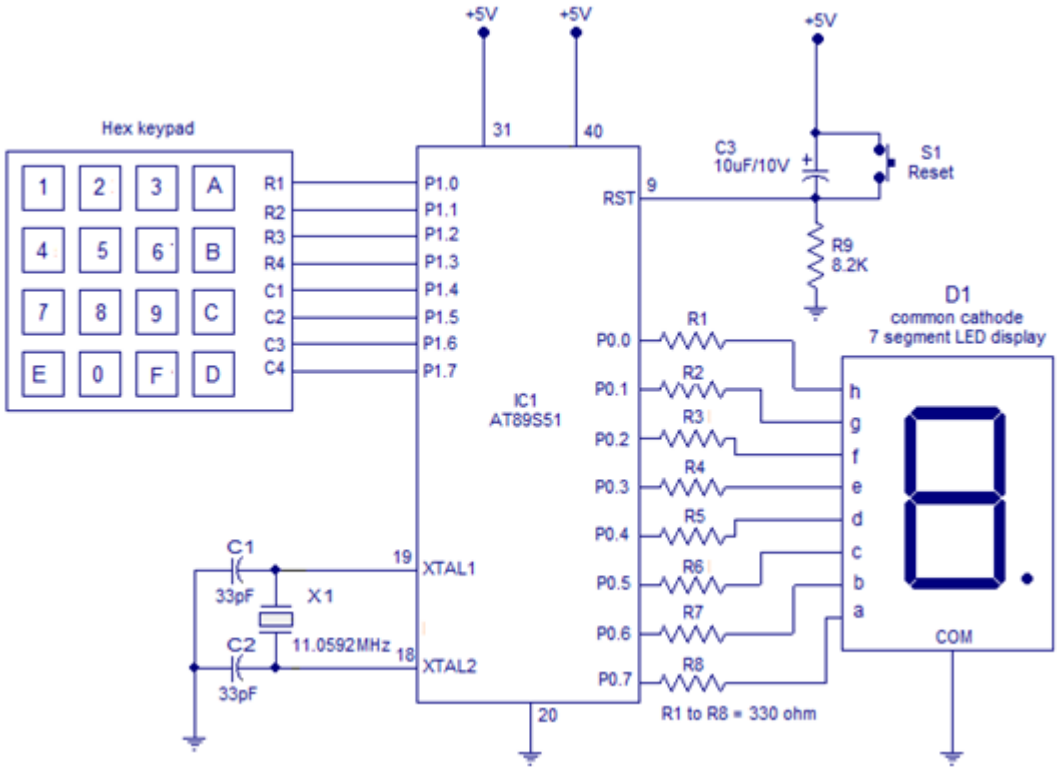
Hex keypad

The hex keypad has 8 communication lines namely R1, R2, R3, R4, C1, C2, C3 and C4. R1 to R4 represents the four rows and C1 to C4 represents the four columns. When a particular key is pressed

the corresponding row and column to which the terminals of the key are connected gets shorted. For example if key 1 is pressed row R1 and column C1 gets shorted and so on. The program identifies which key is pressed by a method known as column scanning. In this method a particular row is kept low (other rows are kept high) and the columns are checked for low. If a particular column is found low then that means that the key connected between that column and the corresponding row (the row that is kept low) is been pressed. For example if row R1 is initially kept low and column C1 is found low during scanning, that means key 1 is pressed.

Interfacing hex keypad to 8051.

The circuit diagram for demonstrating interfacing hex keypad to 8051 is shown below.Like previous 8051 projects, AT89S51 is the microcontroller used here. The circuit will display the character/numeric pressed on a seven segment LED display. The circuit is very simple and it uses only two ports of the microcontroller, one for the hex keypad and the other for the seven segment LED display.



Interfacing hex keypad to 8051

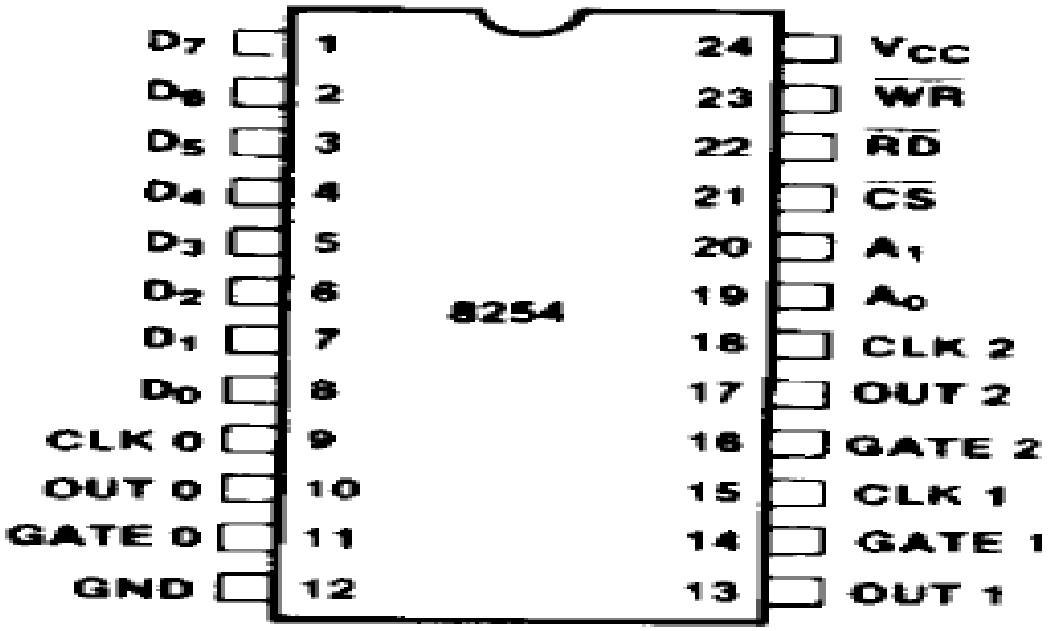
The hex keypad is interfaced to port 1 and seven segment LED display is interfaced to port 0 of the microcontroller. Resistors R1 to R8 limits the current through the corresponding segments of the LED display. Capacitors C1, C2 and crystal X1 completes the clock circuitry for the microcontroller. Capacitor C3, resistor R9 and push button switch S1 forms a debouncing reset mechanism.

- The 5V DC power supply must be well regulated and filtered.
- Column scanning is not the only method to identify the key press. You can use row scanning also. In row scanning a particular column is kept low (other columns are kept high) and the rows are tested for low using a suitable branching instruction. If a particular row is observed low then that means that the key connected between that row and the corresponding column (the column that is kept low) is been pressed. For example if column C1 is initially kept low and row R1 is observed low during scanning, that means key 1 is pressed.

Programmable interval timer 8254:

Features:

Three Independent 16-Bit Counters,	Clock input upto 10 MHz,	Status Read-Back Command,
Six Programmable Counter Modes,	Binary or BCD Counting,	Single +5V Supply,



SIX MODES OF TIMER	APPLICATIONS OF TIMER
○ Mode 0 : Interrupt On Terminal Count ○ Mode 1 : Hardware Retriggerable One Shot ○ Mode 2 : Rate Generator ○ Mode 3 : Square Wave Mode ○ Mode 4 : Software Triggered Strobe ○ Mode 5 : Hardware Triggered Strobe(Retriggerable)	• Real time clock • Event Counter • Digital One Shot • Programmable rate generator • Square Wave Generator • Binary rate multiplier • Complex waveform generator • Complex motor controller

01) What is the purpose of register ACCUMULATOR
02) What are the flags used in 8051 and 8085.
03) What is the purpose of register STACK POINTER in 8051.
04) Explain the logical AND operation of 8051.
05) What is the purpose of the instruction "MOV A,@R1"
06) What is the purpose of the instruction "JNZ"
07) What is the difference between the flags CY and OV.
08) Write an instruction to store the content of port 1 into Accumulator.
09) Explain the instruction " MUL AB " in 8051.
10) Mention the timers of 8051.
11) What is the purpose of C/T' bit in TMOD register
12) What is interrupt signal.
13) Mention the ports placed in 8255.
14) What is relay.
15) Name the six different modes of external timers.
16) List down the timer/counter modes.
17) List down the serial communication modes.
18) List down the 8255 PPI operating modes
19) which register used as memory pointer in 8085
20) which register used as memory pointer in 8051
21) List down the special function registers used for serial communication
22) List down the special function registers used for timer/counter
23) List down the special function registers used for interrupts.
24) Name the interrupts used in 8085
25).Name the interrupts used in 8051 and their vector address
26) Draw the circuit diagram for port 0 I/O configuration.
27) Draw the circuit diagram for port 1 I/O configuration.
28) Draw the circuit for interfacing opto isolator with 8051.
29) Write the format of CWR of 8051.
30) Define 'Assembler Directives.'
31) Define "CPU & ALU".
32)Explain the status signals of 8085.
33) Draw the structure of internal RAM.
34) List down the various features of 8051.
35) What is the value to be loaded in timer registers to generate a square wave of frequency 10KHz. [Assume frequency of crystal is 12 MHz.]
36) Explain the format of IE register.
37) Draw the interfacing diagram of 4 digit seven segment LED display with 8051.
38) Write an ALP for receiving an 8 bit data placed in 4450h and store it in R5 register.
39) Explain is the use of the instruction 'SUBB' with example.
40) Write and explain the format of TMOD register.
41) How interrupts are handled.
42) Draw the interfacing diagram of ADC with 8051
43) Write byte addresses for I/O ports available in 8051.
44) What are the flags available in 8051?
45) Specify the three I/O modes of 8255.
46) What is the purpose of SBUF register.

8 mark questions:

- Draw the Architecture of 8085 and explain.
- Draw and explain the architecture of 8051.
- Explain the bit manipulation instructions used in 8051.
- Write an ALP for receiving an 8 bit data placed in 4450h and store it in R5 register.
- Mention the SFRs used in timer operation.
- What is the value to be loaded in timer registers to generate a square wave of
    - frequency 10KHz. [Assume frequency of crystal is 12 MHz.]
- Explain the steps to program the timer 1 in mode 1.
- Explain the various modes of timer operation with diagram.
- Specify the different modes of serial communication.
- Explain the format of IE register.
- Write the steps involved in programming 8051 to transfer data serially.
- Explain briefly about serial interface RS 232.?
- Draw the interfacing diagram of 4 digit seven segment LED display with 8051.
- Draw the interfacing diagram of stepper motor with 8051.
- What is the value to be loaded in timer registers to generate a square wave of
    - frequency 10KHz. [Assume frequency of crystal is 12 MHz.]
- What is the use of program counter in 8085.
- List down the applications of the microprocessor.
- Draw the pin out diagram of 8085 and Explain.
- Briefly explain about memory organisation in 8051 with neat diagram(s).
- List down the differences between microprocessor and microcontroller.
- Explain bit addresses for RAM.
- Write and explain the format of TMOD register.
- What is the value to be loaded in timer registers to get a time delay of 10ms
  [Assume frequency of crystal is 11.0592 MHz.]
- Explain the steps to program the timer 0 in mode 2.
- Explain the steps to program the timer 1 in mode 1.
- Write an ALP for alternatively switch ON / OFF the load connected at port pin P1.5 when the counter overflows by using counter 1 in mode 2.
- Write the use of PCON register.
- How interrupts are handled.
- Draw the block diagram of 8254.
- Write the steps involved in programming 8051 to receive data serially.
- Draw the pin diagram of microcontroller 8051
- List down any 8 SFRs with their byte and bit addresses.
- With the diagram, Explain the I/O ports of 8051.
- Draw and Explain the Architecture of 8051
- Explain how to assemble and run an 8051 program using assembler.
- Explain the logical OR and XOR operation of 8051.
- Explain the instructions “CJNE & DJNZ” with example .
- Explain I/O programming with example.
- Write an ALP for converting a hexadecimal number placed in memory location 4450h to its equivalent BCD number. Store the result from location 4560h.

- Write an ALP using timer 0 in mode 1 to generate a square wave signal of frequency 5 MHz with 50 % duty cycle of port pin P1.2.
- Explain different modes of timer with suitable diagram.
- Explain External interrupts with suitable program .

- Write the format of SCON and PCON registers and explain
- Write an ALP for sending a character placed in register A in mode 1 by using timer1 in mode 2 for baud rate.
- Draw the interfacing diagram of LCD with 8051 and explain its operation.
- Draw the interfacing diagram of stepper motor with 8051 and explain its operation.
- Draw the functional block diagram of 8255 and explain each block.

Chapter	Name Of The Topic	Hours	Marks
UNIT I	**Architecture of 8051 & their Pin details** **1.1 Introduction to microprocessor & microcontroller :** Architecture of 8085 -Functions of each block. Comparison of Microprocessor and Microcontroller -Features of microcontroller -Advantages of microcontroller - Applications Of microcontroller -Manufactures of microcontroller. **1.2 Architecture of 8051 :** Block diagram of Microcontroller –Functions of each block. Pin details of 8051 -Oscillator and Clock -Clock Cycle -State - Machine Cycle - Instruction cycle –Reset - Power on Reset - Special function registers **:** Program Counter -PSW register -Stack - I/O Ports . **1.3 Memory Organisation & I/O port configuration:** ROM - RAM - Memory Organization of 8051,Interfacing external memory to 8051	**14**	**20**
UNIT II	**Overview of 8051 Instruction Sets:** **2.1 Instruction Sets :** Instruction Format, Different addressing modes of 8051, Assembling and running an 8051 program –Structure of Assembly Language - Assembler directives Classification of 8051 Instructions(based on Length) - Classification of 8051 Instructions(based on Function) **2.2 Data Handling instructions :** Data transfer instructions — Arithmetic Instructions - Logical instructions(byte Operands) - Format of instructions and examples **2.3 Bit addresses for I/O , RAM & control instructions:** **-** I/O programming - I/O bit manipulation programming **-** Bit Manipulation Instructions - -Branching instructions	**14**	**20**
UNIT III	**Interrupts and Programming Examples:** 3.1 **8051 Interrupts** - Interrupts available in 8051,their vector addresses, Interrupt priority in 8051- **Interrupt related SFRs:** interrupt enable register (IE) Interrupt priority register (IP) **3.2 Interrupt handling** - Programming Timer Interrupts –Programming external hardware interrupts -Programming the serial communication interrupt **3.3 Programs** - Multibyte Addition - 8 Bit Multiplication and Division - Biggest Number / Smallest Number -Ascending order / Descending order –Conversion Programs -HEX to BCD, BCD to HEX , HEX to ASCII & ASCII to Binary -Time delay routines	**14**	**20**

UNIT IV	**Timer/Counter and Serial Communication:** **4.1 Special function registers :** -Timer 0 and Timer 1 registers- TCON register - TCON register -SCON register -SBUF register - PCON register . **4.2 Programming 8051 Timers/ Counter programming :** Different modes of Timer -Programming Modes - Mode 0,Mode 1, Mode2 & Mode 3. Different modes of Counter - Programming Modes - Mode 0,Mode 1, Mode2 & Mode 3. **4.3 Basics of Serial programming :** RS 232 Standards -8051 -connection to RS 232 - 8051 Serial Communication Programming -Programming the 8051 to transfer data serially -Programming the 8051 to Receive data serially	**14**	**20**
UNIT V	**Interfacing Techniques :** **5.1 Programmable interface ICs:** IC 8255 -Block Diagram -Modes of 8255 -CWR format - 8051 interfacing with the 8255 - IC 8254 -Block Diagram - Modes of 8254 . **5.2 Interfacing circuits :** Relays and opto isolators –Sensor interfacing -ADC interfacing -DAC interfacing - Keyboard interfacing - Seven segment LED Display Interfacing - LCD display interfacing - **5.3 Microcontroller based Application :** Stepper Motor interfacing -DC motor interfacing -PWM.	**12**	**20**
	Revision	**05**	
	Cycle Tests and Model Examination	**07**	
	Total	**80**	**100**