Karnatak Law Society's

Vishwanathrao Deshpande Institute of Technology

Haliyal - 581329

VTU MODEL QUESTION PAPER SCHEME & SOLUTION

Subject : Microcontrollers

code : 18EC46

Semester : 04

Prepared By : Prof. Prasann D. Kulkarni
Asst. Prof. ECE Dept.
KLS VDIT, Haliyal

22.07.2021

Head of the Department
Dept. of Electronic & Communication Engg.
KLS V.D.I.T., HALIYAL (U.K.)

Model Question Paper-1 with effect from 2019-20 (CBCS Scheme)

USN □□□□□□□□□□

Fourth Semester B.E. Degree Examination
Subject Title: MICROCONTROLLER

TIME: 03 Hours **Max. Marks: 100**

Note: Answer any **FIVE** full questions, choosing at least **ONE** question from each **MODULE**.

		Module -1	***Bloom's Taxonomy Level**	**Marks**
Q.01	a	With neat block diagram explain features of microcontroller 8051.	L1	8
	b	Write a note on Embedded microcontrollers.	L1	4
	c	Write an interfacing diagram 8051 microcontroller interfaced to 8k bytes of ROM and 8k bytes of RAM.	L2	8
		OR		
Q.02	a	With neat diagram explain the internal memory structure and programming model of 8051 microcontroller.	L1	8
	b	Write a short not criteria for choosing a microcontroller.	L1	4
	c	Write an interfacing diagram 8051 microcontroller interfaced to 8k bytes of ROM and 16k bytes of RAM.	L2	8
		Module-2		
Q. 03	a	With neat diagram explain the bit contents of PWS.	L1	4
	b	Write a note on branching instructions defining their range.	L2	8
	c	Write an assembly language program to add two 16 bit numbers loaded in R1R0 and R3R2. Store the result in R6, R5 and R4 from MSB to LSB.	L3	8
		OR		
Q.04	a	Write a note on bit manipulation instructions.	L1	4
	b	Explain how the instructions work: 1. JMP @A+DPTR 2. XCHD A, @Ri 3. JBC bit, rel8 4. MOVC A, @A+PC	L2	8
	c	Write an assembly language program to multiply a 16 bit number loaded in R1R0 (multiplicand) with an 8-bit number loaded in R2 (multiplier). Store the resultant product in R6, R5 and R4 from MSB to LSB.	L3	8
		Module-3		
Q. 05	a	Explain PUSH and POP instructions with a help of example program.	L2	4
	b	3 eight bit numbers X, NUM1 and NUM2 are stored in internal data RAM locations 20h, 21h and 22H respectively. Write an assembly language program to compute the following: IF X=0; then NUM1 (AND) NUM2,	L3	8

		IF X=1; then NUM1 (OR) NUM2,
		IF X=2; then NUM1 (XOR) NUM2,
		ELSE RES =00, RES is 23H RAM location.		

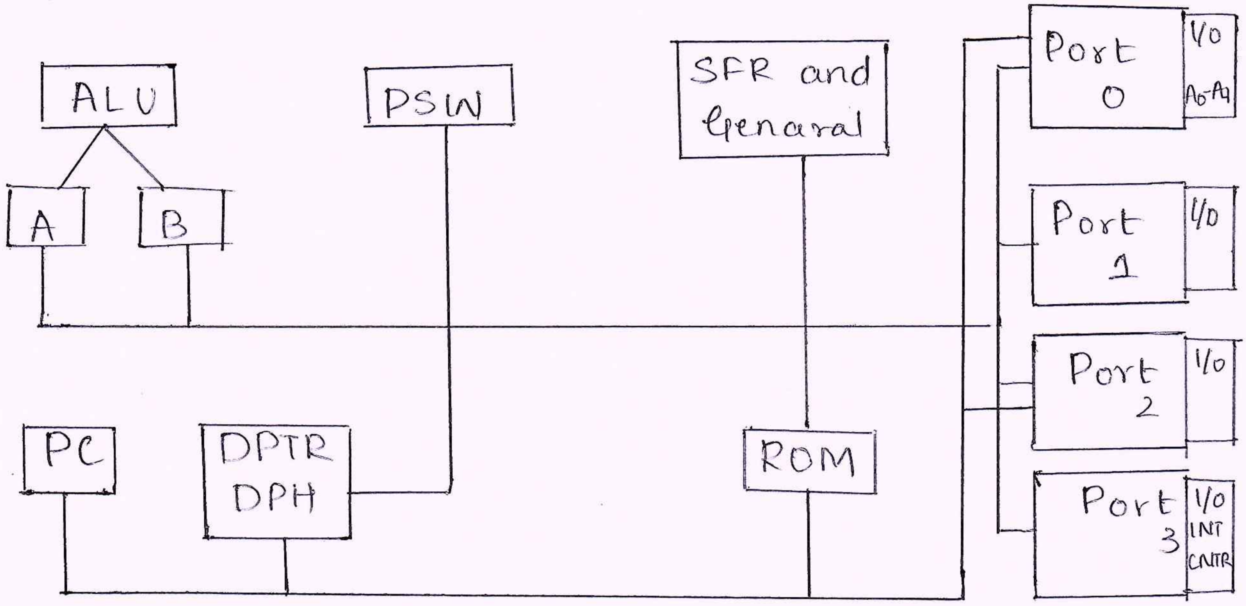
	c	Write a assemble language program to toggle all the bits of Port 2 for every 200ms. Assume crystal is 11.0592MHz. Show all the calculations needed.	L3	8
		OR		
Q. 06	a	Explain why pull-up resistors are connected to Port 0.	L2	4
	b	Write an assembly language program to find the factorial of a number. Use Subroutine programming.	L3	8
	c	Write an assembly language program to find the average of 10 students marks stored in external RAM memory address 8000H. Load the average value in internal RAM memory 30H.	L3	8
		Module-4		
Q. 07	a	Explain RS232 standard and 9 pin DB connector.	L1	4
	b	Explain the mode 2 operation of timers and mention the steps involved in programming timers in mode 2.	L2	8
	c	Write a C program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously.	L3	8
		OR		
Q. 08	a	Explain the importance of MAX232 IC with its pin details.	L1	4
	b	Explain how timers are used as counters, explain the counters operation using a code snippet.	L2	8
	c	Assume XTAL = 11.0592 MHz, write a assembly language program to generate a square wave of 50 kHz frequency on pin P2.3.	L3	8
		Module-5		
Q. 09	a	Explain the Interrupt Vector Table of 8051 microcontroller.	L1	5
	b	Explain how multiple interrupts are handled in 8051 microcontroller.	L2	5
	c	With neat diagram write an assembly language program to interface LCD to 8051 microcontroller.	L3	10
		OR		
Q. 10	a	List the steps involved in executing interrupts in 8051 microcontroller.	L1	5
	b	Explain how interrupt programming is done using C programming in 8051 microcontroller.	L2	5
	c	With neat diagram write an assembly language program to interface Stepper motor to 8051 microcontroller.	L3	10

*Bloom's Taxonomy Level: Indicate as L1, L2, L3, L4, etc. It is also desirable to indicate the COs and POs to be attained by every bit of questions.

MODULE - 1

1(a) With neat block diagram explain features of microcontroller 8051 08M

⟹ BLOCK DIAGRAM OF 8051 MICROCONTROLLER



-03M

Features of 8051 Microcontroller.

* 8 bit CPU with registers A (accumulator) and B for arithemetic and logical operations.

* 16 bit Program Counter (PC) and Data Pointer (DPTR)

-03M

* 8 bit Program Status Word (PSW)

* 8 bit Stack Pointer (SP)

* 4K Code Memory

* Internal memory of 128 bytes.

* Four, 8 bits I/O ports P_0 to P_3 through which the microcontroller interfaces to external devices.

* Two, 16 bit timers / counters T_0 and T_1 for clocking events counting events and generating high frequency signals.

* Full Duplex serial Data receiver / Transmitter.

-02M

* Control Registers: TCON, TMOD, SCON, PCON, IP & IE.

* Two external and internal interrupt sources.

POK

1(b) Write a note on Embedded Microcontrollers. 04M

⟹ A Microcontroller is a programmable digital processor with necessary peripherals. Both microcontrollers and microprocessors are complex sequential digital circuits meant to carry out job according to the program / instructions. Sometimes analog input / output interface makes a part of microcontrollers circuit as both analog and digital mode in nature.

-02M

A Microcontroller incorporated multiple functions in the same integrated circuits. The design incorporates all the features found in microprocessor CPU :: ALU, PC, SP and registers. It also has other features needed to make a complete computer ROM, RAM, Parallel I/O, serial I/O, Counters and clock circuits. A microcontroller is a general purpose device but one that is meant to read data, perform limited calculations on that data and control environment based on those calculations. The prime use of microcontroller is to control the operation of a machine using a fixed program that is scored in ROM and that does not change over the life time of the system.

-02M

⟸ VIMP

CPU	RAM	ROM
I/O Port	Timer	Serial Port

← A single chip.

1(c) Write an interfacing diagram 8051 Microcontroller. interfaced on 8k bytes of ROM and 8k bytes of RAM. 8M

⇒ Memory size ROM : 8K.

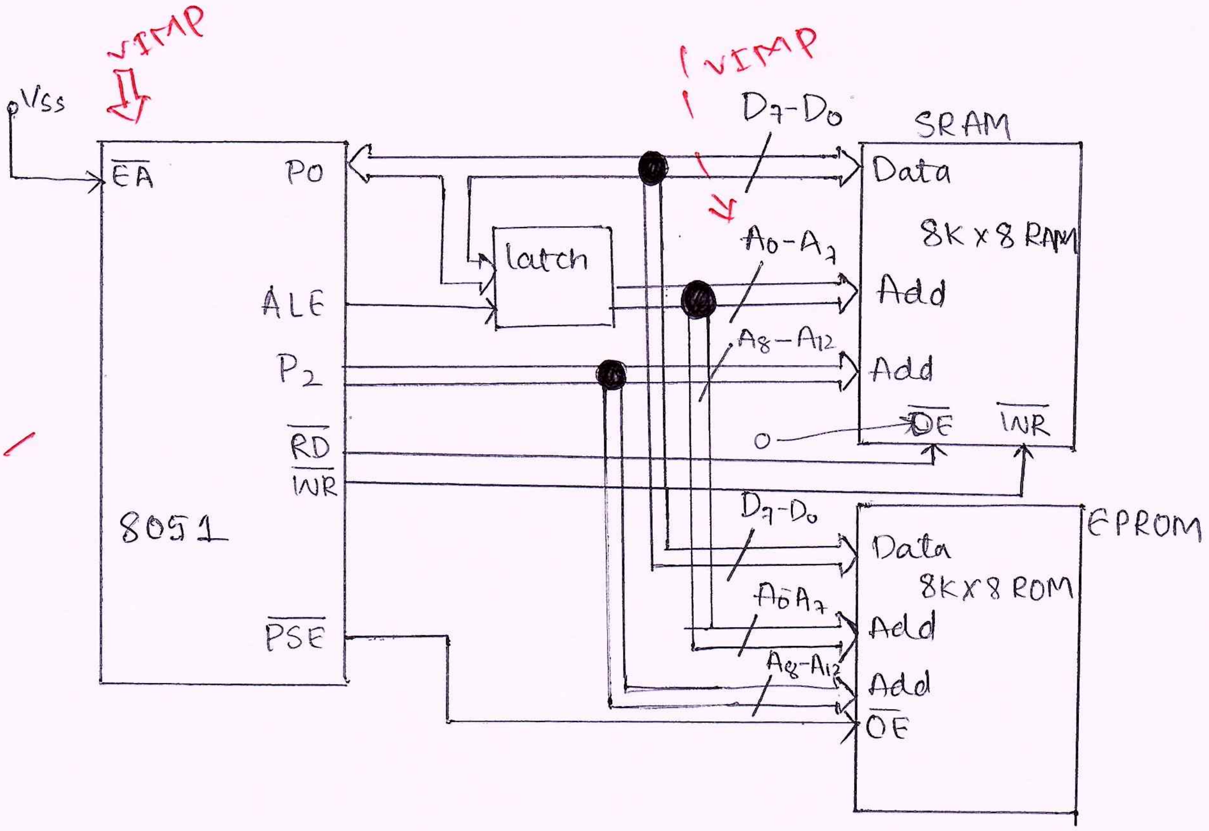
$2^n = 8K$ ⇒ we require n address lines.

04M here $n=13$ ⇒ $A_0 - A_{12}$ address lines are required.

Memory size RAM = 8K.

$2^n = 8K$ ⇒ we require n address lines.

here $n=13$ ⇒ $A_0 - A_{12}$ address lines are required.

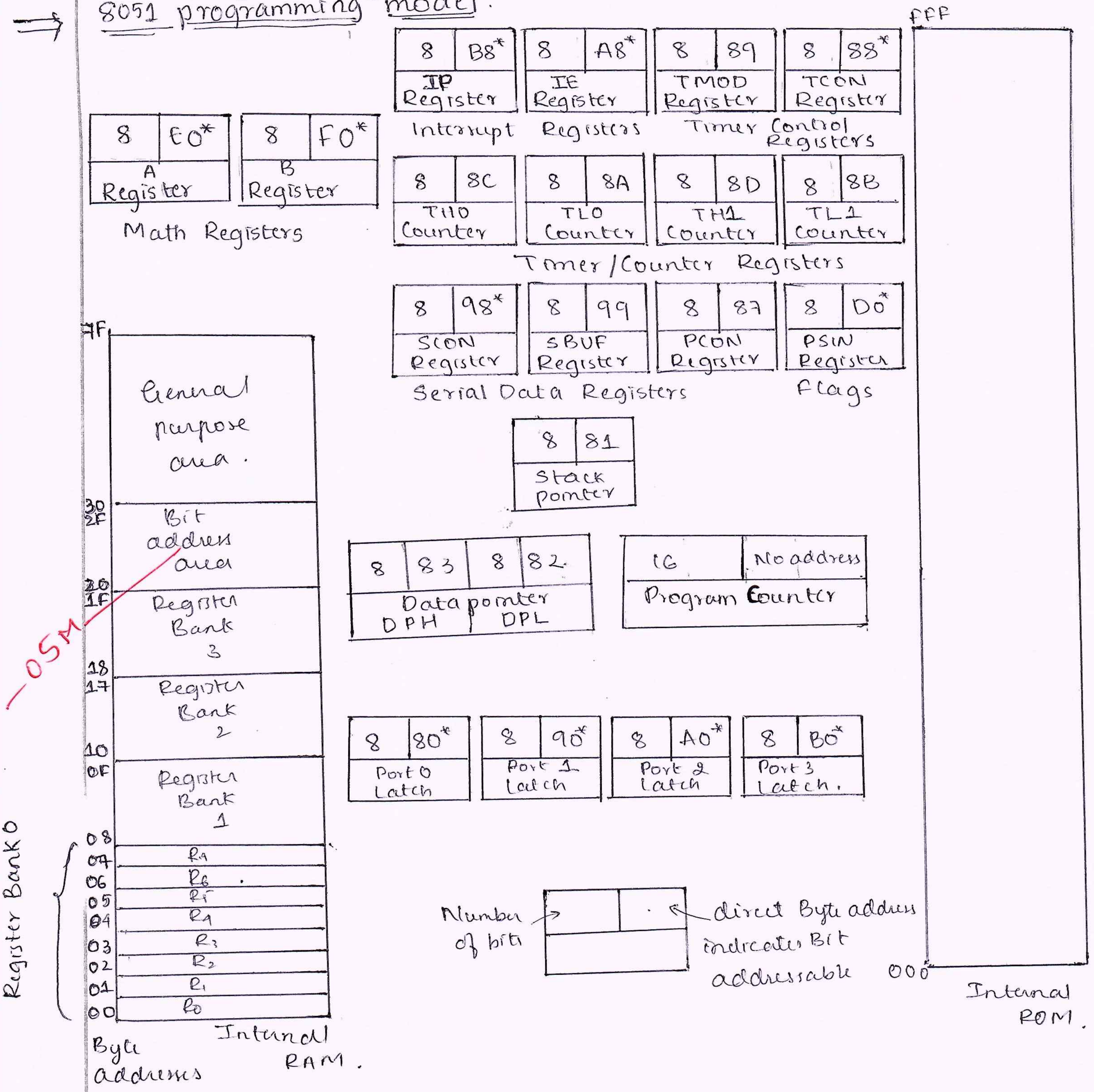


04M

PDK

2 (a) With neat diagram explain the internal memory structure and programming model of 8051 microcontroller: -08M-

⇒ 8051 programming model:



Programming Model of 8051 Microcontroller shows the usual CPU components: ALU, working registers and clock circuits consists of these specific features:

- * 8 bit CPU with registers A (the accumulator) and B .
- * 16-bit program Counter (PC) and Data Pointer (DPTR) .
- * 8-bit Program Status Word (PSW) .
- * 8-bit Stack Pointer (SP) .
- * Internal ROM of 4K .
- * Internal RAM of 128 bytes:

NIMP

4 registers banks, each containing 8 registers.
16 bytes, which may be addressed at the bit level
8 bytes, of general purpose data memory.

—03M

- * 32 input/output pins arranged as four 8 bit ports - P_0-P_3
- * Two 16 bit timers/counters: T_0 and T_1.
- * Full duplex serial data receiver/transmitter: SBUF.
- * Control Registers: TCON, TMOD, SCON, PCON, IP and IE.
- * Two external and three internal interrupt sources.
- * Oscillators and clock circuits.

04M

2(b) Write a short note on criteria for choosing a microcontroller.
⇒ We use more number of microcontrollers compare to microprocessors. Because, Microprocessor consists of only CPU, whereas microcontroller contains a CPU, Memory, I/O ports all integrated into one chip. Microprocessor is used in Personal computers whereas Microcontroller is used in an embedded system. Microcontroller uses an internal controlling bus. and Microcontroller is inexpensive and straightforward with fewer instructions to process. Applications of microcontroller are numerous starting from domestic applications such as washing machines, TVs, air conditioners. Microcontrollers are also used in automobiles, process control

—02M

—02M

industries, cell phones, electrical drives, robotics and in space applications.

2(c) Write an interfacing diagram 8051 microcontroller interfaced to 8k bytes of ROM and 16k bytes of RAM. 08M

⟹ Memory size ROM : 8k.

$2^n = 8k \Rightarrow$ we require n address lines.

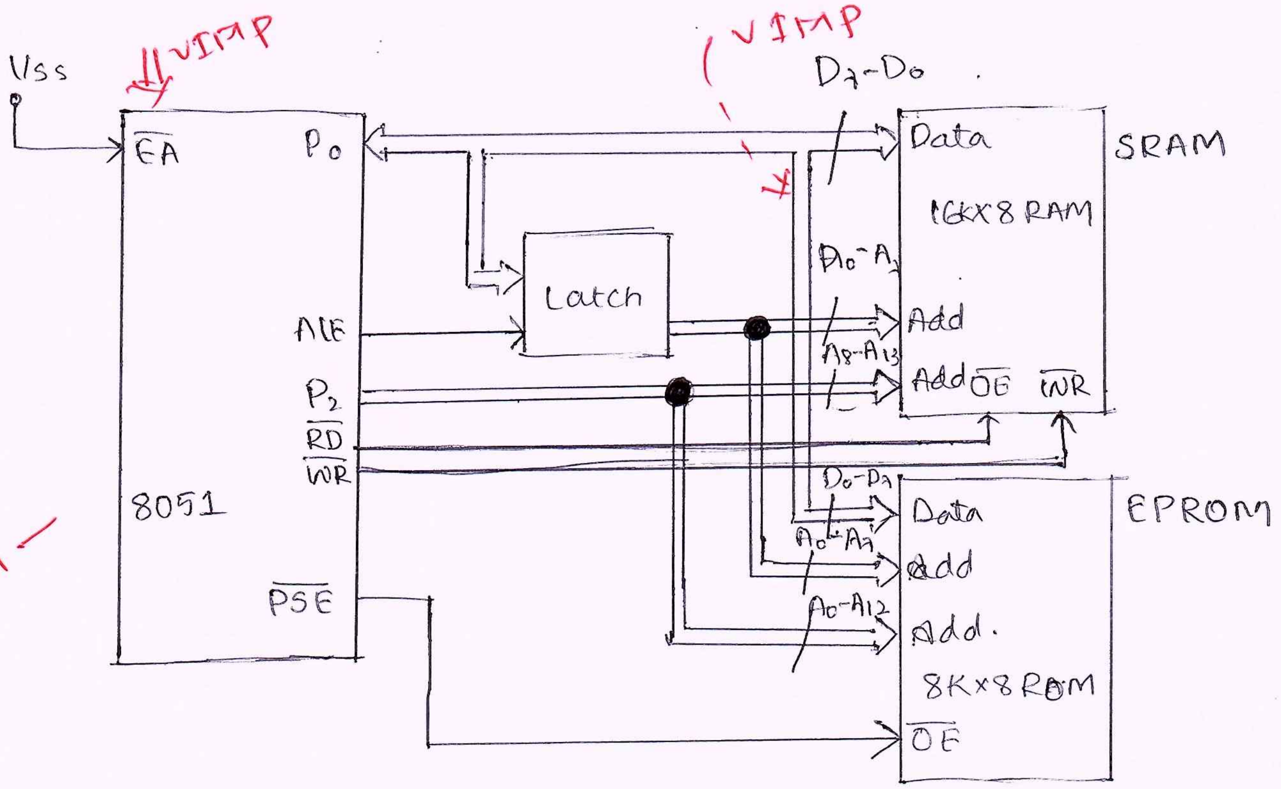
here $n=13 \Rightarrow A_0 - A_{12}$ address lines are required

—04M—

Memory size RAM : 16K

$2^n = 16K \Rightarrow$ we require n address lines.

here $n = 14 \Rightarrow A_0 - A_{13}$ address lines are required.



—04M—

MODULE-2

04M

3(a) With a neat diagram explain the bit contents of PSW

⇒ PSW - Program Status Word Register.

-01M

7	6	5	4	3	2	1	0
CY	AC	FO	RS1	RS0	OV	—	P

The program status word (PSW) register is an 8bit register

Bit	Symbol	Function.
7	CY	Carry flag; used in arithmetic, JUMP, ROTATE, and BOOLEAN instructions.
6	AC	Auxilliary carry flag; used for BCD arithmetic
5	FO	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0 RS1 RS0 0 0 select register bank 0 0 1 select register bank 1 1 0 select register bank 2 1 1 select register bank 3
2	OV	Overflow flag; used in arithmetic instructi
1	—	Reserved for future use.
0	P	Parity flag; shows parity of register A: 1 = odd Parity

-03M

PDIC B

08M

3(b) Write a note on branching instructions defining their range.

⇒ JUMP OPERATION:

LJMP — Long jump: The LJMP causes the program to branch to a destination address defined by the 16-bit operand in the jump instruction. Because 16 bit address is used, the instruction can cause a jump to any location within the 64Byte program space. 02M

SJMP - Short jump: SJMP uses a single byte address. This address is a signed 8bit number and allows the program to branch to a distance (0 - 127) range} and (-1 to -128) range 02M

AJUMP - Absolute jump: AJMP uses an 8bit destination address, based on relative addressing within the range of -128 to +127 bytes.

CALL OPERATION:

02M LCALL - Long Call: LCALL instruction is used to call a subroutine at a specifed address. The address is 16 bit long so the call can be made to any location within the 64KByte memory space.

ACALL - Absolute Call: ACALL instruction is used to call a subroutine located at the specified address. Calls a subroutine in the maximum adress range of 2KByte range. 02M

RET - Return: The Return from subroutine is achieved using the RET instruction.

08M.

3(c) Write an assembly language program to add two 16 bit numbers loaded in R1 R0 and R3 R2. Store the result in R6 R5 and R4 from MSB to LSB

⇒

```
ORG 0h
MOV    R1 , #20h
MOV    R0 , #30h
MOV    A , @R1          04M
ADD    A , @R0
MOV    R6 , A
inc R1
inc R0
MOV    A , @R1
ADDC   A , @R0          04M
MOV    R5 , A
MOV    A , #0h
ADD C  A , #0h
MOV    R4 , A
SJMP   START
END.
```

PDK

04M

4 (a) Write a note on bit manipulation instructions.

⇒ 8051 has 128 bit addressable memory. Bit addressable SFR bit. The carry flag (C) in the PSW special-function register is the destination for most of the opcodes because flag can be tested and the program flow changed using instructions. There is no-flag affected accept Carry flag, unless the flag is an addressed bit.

The Mnemonics corresponding to the Bit Manipulation instructions are: CLR, SETB, MOV, JC, JNC, JB, JNB, JBC, ANL, ORL, CPL. These instructions can perform set, clear, and, or, complement etc. at bit level.

Mnemonic	Operations
ANL C, b	- ANDC and the addressed bit; put the result in C.
ANL C, /b	- ANDC and complement of the addressed bit; put the result in C; the addressed bit is not altered.
ORL C, b	- ORC and the addressed bit; put the result in C
ORL C, /b	- ORC and the complement of the addressed bit; put the result in C; the addressed bit is not altered.
CPL C	- Complement the C flag
CPL b	- Complement the addressed bit.
CLR C	- Clear the C flag to zero.
CLR b	- Clear the addressed bit to ~~the C flag~~ zero
MOV C, b	- Copy the addressed bit the ~~the~~ C flag
MOV b, C	- Copy the C flag to the addressed bit
SETB C	- Set the flag to one
SETB b	- Set the addressed bit to one.

02M

02M

4(b) Explain how the instructions work. 08M

2x4=08M

1. JMP @A+DPTR
2. XCHD A,@Ri
3. JBC bit , rel8
4. MOV C A, @A+PC

⇒ 1. JMP @A + DPTR: The jump instruction adds the eight bit unsigned contents of the Accumulator with the sixteen bit data pointer. and load the resulting sum to the program Counter. Neither the accumulator nor the Data Pointer is altered. No flags are affected. 02M

2. XCHD A, @Ri : XCHD exchanges the low order nibble of the ACC (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specific register. The high order nibbles (bits 7-4) of each register are not affected. No flags are affected. 02M

3. JBC bit, rel8: If the indicated bit is a one, branch to the address indicated; otherwise proceed with the next instruction. The bit will not be cleared if it is already a zero. The branch destination is compared by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected. 02M

PDK

4. MOVC A, @A+PC: MOVC moves a byte from Code Memory into accumulator. The code Memory address from which the byte will be moved is calculated by summing the value of the Accumulator with the Program Counter (PC). In case of Program counter, PC is first incremented by 1 before being summed with the Accumulator.

02M

08M

4(c) Write an assembly language program to multiply a 16 bit number loaded in R1 R0 (multiplicand) with an 8 bit number loaded in R2 (multiplier). Store the resultant product in R6, R5 and R4 from MSB to LSB.

```
ORG 0000H
MOV  DPTR, #1234H
MOV  B,    #56H
MOV    R2, B
MOV    A, DPL          04M
MUL    AB
MOV    R4, A
MOV    R1, B
MOV    A, DPH
MOV    B, R2           02M
MUL    AB
ADD    A, R1
MOV    R5, A
MOV    A, B            02M
ADDC   A, #00H
MOV    R6, A
END
```

MODULE-3.

5(a) Explain PUSH and POP instructions with a help of example program. —04M

⇒ PUSH Instruction: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. No flags are affected.

Example: On entering an interrupt routine, the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The following instruction sequence. PUSH DPL PUSH DPH leaves the Stack Pointer set to OBH stores 23H and 01H in internal RAM. locations OAH and OBH, respectively. —02M

POP Instruction: The contents of the internal RAM location addressed by the Stack Pointer. is read, and the Stack Pointer is decremented by one. The value read is then transfered to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The following instruction sequence, POP DPH POP DPL leave the Stack Pointer equal to the value 30H and sets the Data Pointer to 0123H. —02M

PDK

08M

5(b) 3 - 8bit numbers X, NUM1 and NUM2 are stored in internal data RAM locatrons 20h, 21h, 22H respectively. Write an assembly language program to compute the following:
IF X=0 ; then NUM1 (AND) NUM2,
IF X=1 ; then NUM1 (OR) NUM2,
IF X=2 ; then NUM1 (XOR) NUM2,
ELSE RES=00, RES is 23H RAM locatrons.

⇒
```
ORG  00H
SJMP  30H        02M
ORG   30H
MOV   R0, 20H
CJNE   R0, #0
 MOV    A, 21H   02M
 ANL    A, 22H
 SJMP   LAST.
 CJNE   R0, #1
  MOV    A, 21H  02M
  ORL    A, 22H
  SJMP   LAST.
  CJNE   R0, #2
  MOV  A,  21H   02M
  XRL   A, 22H
  MOV   23H, A
  END.
```

08M

5(c) Write an assembly language program to toggle all the bits of port 2 for every 200ms. Assume crystal is 11.0592 MHz. Show all the calculations needed.

⇒

```
#include <reg 51.h>
void T1M1 Delay (void);          01M
void main (void).
{
  unsigned char i;
   P2 = 0X56
   while (1)                      03M
   {
      P2 = ~P2;
      for (i=0; i<20; i++)
       T1M1 Delay ();
   }
}
 void T1M1 Delay (void)
{
   TMOD = 0X10;                   04M
    TL1  = 0XDB;
    TH1  = 0XFF;   IMP
     TR1  = 1;
     while (TF1==0);
      TR1 = 0;
      TF1 = 0;
}
```

PDK

04M

6(a) Explain why pull-up resistors are connected to Port 0

⇒ Since 8051 microcontroller does not have internal pullup resistor on port 0. External pull up resistor on Port 0 of 8051 microcontroller. We use resistor of 10K connected to pins and Vcc for getting high voltage output that can drive the device. diagram -02M

-02M

08M

6(b) Write an assembly language program to find the factorial of a number. Use subroutine programming.

```
⇒  FACTO   : MOV R3,A        } 04M
             DEC  A
             MOV R2,A
   DO MUL  :  CLR A
   DO ADD  :  ADD A,R2
              DJNZ R3, DO ADD
              MOV R3,A                } 04M
              DJNZ R2, DO MUL
              RET
```

08M

6(c) Write an assembly language program to find the average of 10 students marks stored in external RAM memory address 8000H. Load the average value in internal RAM memory 30H.

⇒

```
MOV   DPTR , #2200H
MOV   A    , @DPTR          02M
MOV   R0 , A
INC   DPTR
MOV   R1, #00               02M
MOVX  A, @DPTR
ADD   A, R1
MOV   R1 , A                02M
INC   DPTR
DJNZ  R0  , BACK.
MOV   DPTR , #2300H         02M
MOV   A    , R1
MOVX  @DPTR , A
```

DDK

MODULE-4.

04M

7(a) Explain RS232 standard and 9 pin DB connector

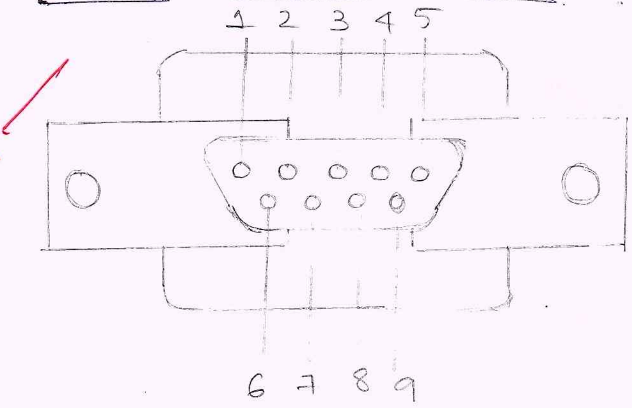
⇒ An interfacing standard RS232 was set by the Electronics Industries Association (EIA) in 1960. The standard was set long before the advent of the TTL logic family, its input and output voltage level are not TTL compatible. In RS232, a 1 is represented by -3 ~ -25V, while a 0 bit is +3 ~ +25V, making -3 to +3 undefined.

01M

Since not all pins are used in PC cables, IBM introduced the DB9 version of the serial I/O standard.

02M

RS232 Connector DB-9



RS232 DB-9 Pins.

Pin	Description
1.	Data carrier Detect (-DCD)
2.	Received Data (RxD)
3.	Transmitted Data (TxD)
4.	Data terminal ready (DTR)
5.	Signal ground (GND)
6.	Data Set Ready (-DSR)
7.	Request to send (-RTS)
8.	Clear to send (-CTS)
9.	Ring indicator (RI)

01M

DCD (Data carrier detect): The modem asserts signal DCD to inform the DTE that a valid carrier has been detected and that contact between it and the other modem is established

DTR (Data terminal Ready): When terminal is turned on, it sends out signal DTR to indicate that it is ready for communication.

DSR (Data Set ready): When DCE is turned on and has gone through the self test, it assert DSR to indicate that it is ready to communicate.

RTS (Request to Send): When the DTE device has byte to transmit, it asserts RTS to signal to modem that it has a byte of data to transmit.

CTS (Clear to Send): When the mode m has room for storing the data it is to receive, it sends out signal CTS to DTE to indicate that it can receive the data now.

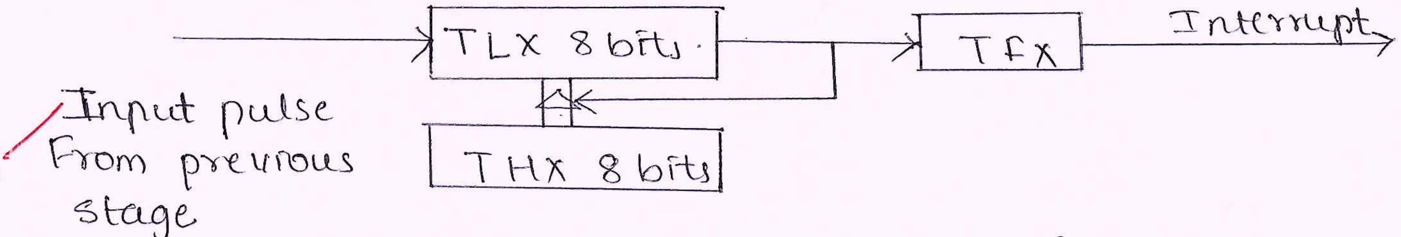
RI (Ring indicator): An output form the mode m and an input to the PC indicates that the telephone is ringing. In goes on and off in synchronous with the ringing sound.

08M

7(b) Explain the mode 2 operation of timers and mention the steps involved in programming timer in mode 2.

⇒ Mode-2 operation of timers: (Auto Reload Mode):
This is a 8 bit timer operation. Counting in performed in TLX while THX stores a constant value. In this mode when the timer overflows i.e. TLX becomes FFH, it is fed with the value stored in THX

-02M



-02M

PDK

Steps for programming timers - Mode-2 in 8051.

1. Load the TMOD value register indicating which timer (0 or 1) is to be used; select timer mode 2.
2. Load TH register with the initial count value. As it is an 8 bit timer, the valid range is from 00 to FFH.
3. Start the timer.
4. Keep monitoring the timer flag (TFX) with the "JNB TFx, target" instruction to see if it raised. Get out of the loop when TFx goes high.
5. Clear the TFx flag.
6. Go back to step 4, since mode 2 is auto-reload.

04M

08M

7(c) Write a C program for the 8051 to transfer "YES" serially at 9600 baud, 8 bit data, 1 stop bit, do this continously.

⇒

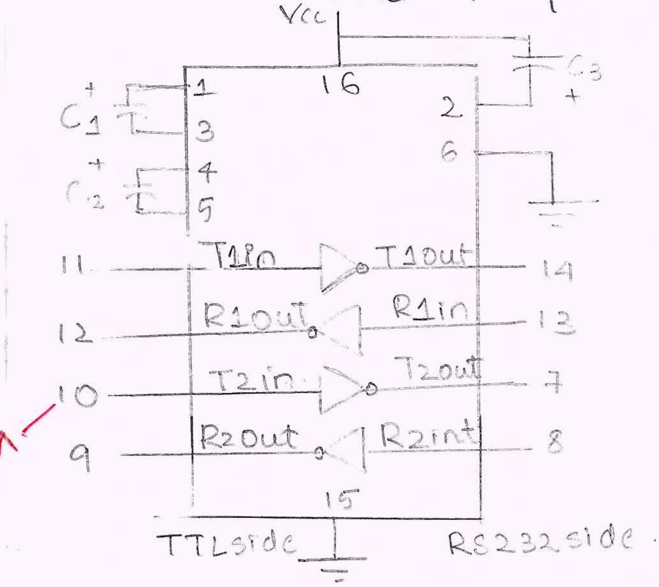
```
#include <reg51.h>      } 02M
void SerTx ;
void main
{
  TMOD = 0X20 ;        // use Timer 1, mode 2
  TH1 = 0XFD;          // 9600 baud rate        } 04M
  SCON = 0X50 ;
  TR1 = 1 ;              // start timer
  while (1)
{
    SerTx ('Y');
    SerTx ('E');         } 02M
    SerTx ('S');
 }
}
Void SerTx
{
```

```
SBUF = x ;          // place value in buffer
while (T1==0) ;       // wait until transmitted.
T1=0 ;
}
```

8(a) Explain the importance of MAX232 IC with its pin details. OUM

⇒ The MAX232 chip is required to convert RS232 voltage level to TTL levels, and vise versa. 8051 has two pins that are used specifically for transferring and receiving data serially. These two pins are called TxD and RxD and are part of the port3 group (P3.0 and P3.1). These pins are TTL compatible; therefore, they require a line driver to make thin RS232 compatible. We need a line driver to convert the RS232' signals to TTL. voltage levels that will be acceptable to 8051's TxD and RxD pins.

02M



02M

PDK

08M

8(b) Explain how timers are used as counters, explain the counters operation using a code snippet.

⇒ Timer in 8051 is used as timer, counter and baud rate generator. Timer always counts up irrespective of whether it is used as timer, counter, or baud rate generator. Timer is always incremented by the micro controller. The time taken to count one digit ~~on master clock~~ up is based on master clock frequency.

04M

TIMER1				TIMER 0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE: when TRx (in TCON) is set and GATE=1 TIMER/COUNTER x will run only while INTx pin is high (h/w control) when GATE=0, TIMER/COUNTER x will run only while TRx=1 (S/w control).

02M

C/T : Bit selection for counter/timer

M1 : } mode selector Bits
M0

Timers/counters are working in 4 various modes of operation the work of the counter is to count the number of events happening and the job of it has unequal delay for counting the sequences. Both up and down counting of sequence is possible. and even they work for external frequency.

02M

MODULE-5.

05M

9(a) Explain interrupt vector table of 8051 microcontroller.

⇒ INTERRUPT VECTOR TABLE:

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2(12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3(13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

03M

* Reset - power up reset.
* Two interrupts are set for the timers. One for timer0 and one for timer1.
* Two interrupts are set for hardware external interrupts P3.2 & P3.3 are for the External hardware interrupts INT0 (or EX1) and INT1 (or EX2).
* Serial communication has a single interrupt that belongs to both receive and transfer.

02M

05M

9(b) Explain how multiple interrupts are handled in 8051 microcontroller.

⇒ When the 8051 is powered up, the priorities are assigned. Each interrupt can have either high priority or low priority. Priority level of interrupt can be set to high by setting corresponding bit in interrupt priority (IP) register or it can be set to low by clearing corresponding bit in IP register.

02M

If multiple interrupts of different priority level are received simultaneously then high priority interrupt is

PDF 8

serviced first. If requests of same priority level are received simultaneously, an internal polling sequence determines which request is to be seriveed, Thus within each priority level is a second priority structure determined by the polling sequence.

03M

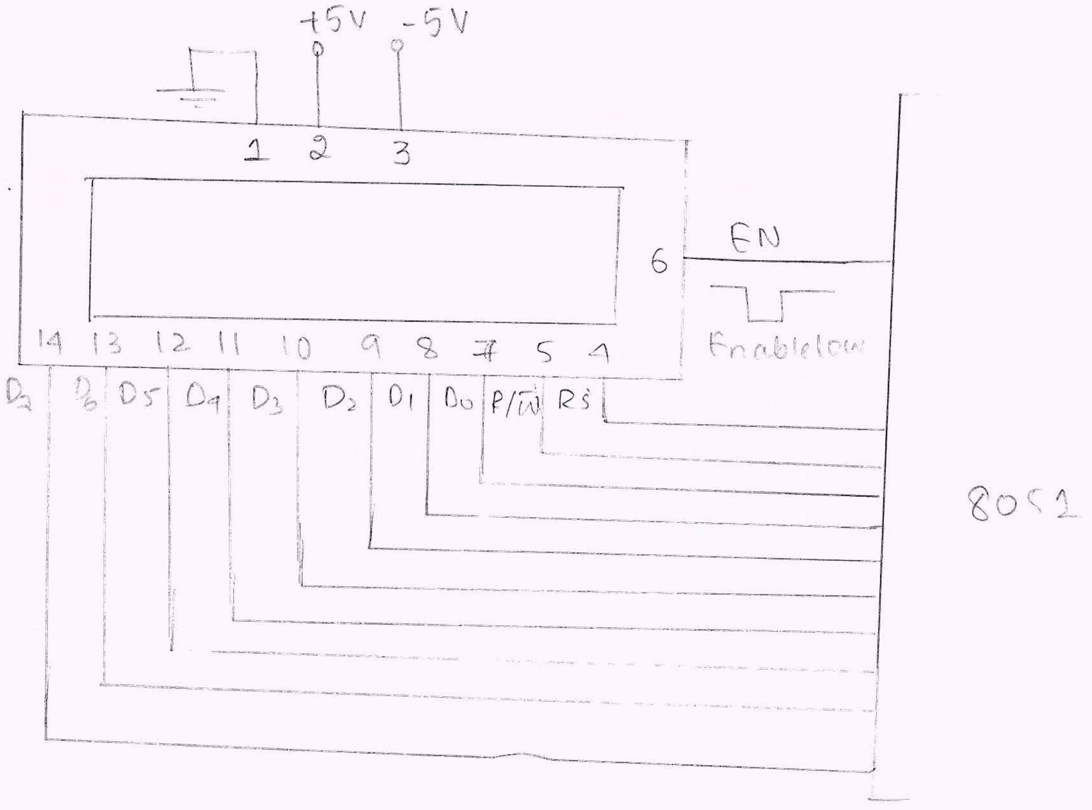
Combination of IP registers and polling sequence gives unique priorities to all 5 interrupts in 8051 microcontrolle If all bits in IP register are cleared then external interrupt INTO will have highest priority, timer0 will be next and serial communication interrupt will have lowest. priority.

If multiple interrupts are triggered at same time, then the interrupts are serviced according to priority.

10M

9(c) With neat diagram write an assembly language program to interface LCD to 8051 microcontroller.

05M



Program:

```
MOV A, #38H
ACALL CMND
MOVA ,#0FH
ACALL CMND
MOVA ,#01H
ACALL CMND
MOV A , #06H
ACALL CMND
MOVA , #82H
ACALL CMND.
MOVA , #3CH
ACALL CMND
MOV A, #49D
ACALL ~~CMND~~ DISP
MOVA, #54D
ACALL DISP
MOV A, #88D
ACALL DISP
MOV A, #50D
ACALL DISP
MOVA, #32D
ACALL DISP
MOVA, #76D
ACALL DISP
MOVA, #67D
A CALL DISP
MOVA ,#68D
ACALL DISP.

MOVA , #0C1H
ACALL CMND
```

05M

PDK
8

```
MOV A ,#67D
 ACALL  DISP
MOVA,#73D
 ACALL  DISP
MOVA,#82D
 ACALL  DISP
MOVA ,#67D
ACALL  DISP
MOV A,#85D
 ACALL  DISP
MOVA,#73D
ACALL DISP
MOVA,#84D
ACALL  DISP
MOVA,#83D
ACALL ·DISP
MOVA,#84D
ACALL  DISP
MOVA,#79D
ACALL DISP
MOVA,#68D
ACALL DISP.
MOVA,#65D
ACALL  DISP
MOVA,#89D
ACALL DISP.
 HERE: SJMP HERE.
 CMND: MOV P1,A
 CLR  P3.5
CLR  P3.4
SETB P3.3
CLR    P3.3
ACALL   DELY
RET
```

```
DISP : MOV P1, A
SETB   P3.5
CLR    P3.4
SETB   P3.3
CLR    P3.3
ACALL    DELY
RET
DELY : CLR P3.3
CLR  P3.5
SETB   P3.4 .
MOV   P1 , #0FFh
SETB   P3.3
MOVA , P1
JB  ACC.7, DELY

CLR   P3.3
CLR   P3.4
RET
END.
```

PDK

05M

10(a) List the steps involved in executing interrupts in 8051 microcontroller.

⇒ Steps involved in executing interrupts are as follows:

1. ~~8051~~finish the instruction it is executing and saves the address of the next instruction on the stack.

2. It also saves the current status of all the interrupt internally

1×5=05M

3. It jumps to a fixed location in memory called the interrupt vector table that holds the address of the interrupt service routine.

4. The microcontroller gets the address of the ISR from the interrupt vector and jumps to it. It starts to execute the interrupt service subroutine until it. reaches the last instruction of the subroutine.

5. Upon executing the RETI instruction, the microcontrolle returns to the Place where it was interrupt.

10(b) Explain how interrupt programming is done using C programming in 8051 microcontroller. 05M

⟹ In 8051 Timer0 and Timer1 interrupts are generated by the timer register bits TF0 and TF1. These interrupts programming C code involves.

1×5=05M

* Selecting the timer by configuring TMOD register and its mode of operation.

* Choosing and loading the initial values of TLx and THx for appropriate modes.

* Enabling the IE registers and corresponding timer bit init.

* Setting the timer run bit to start the timer.

* Writing the subroutine for the timer for time required and clear timer value TRx at the end of subroutine.

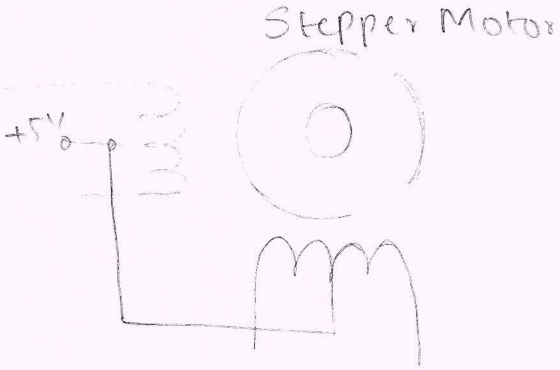
10M

10(c) With neat diagram write an assembly language program to interface stepper motor to 8051 microcontroller.

⇒

```
ORG 00H
MOV TMOD,#01H
MAIN:                       02M
MOV P2,#0CH
ACALL DELAY.
MOV P2,#06H
ACALL DELAY.
MOV P2,#03H                 03M
ACALL DELAY.
MOV P2,#09H
ACALL DELAY
SJMP MAIN.

DELAY : MOV RO,#200
BACK. : MOV THO,#OFCH                05M
        MOV TLO,#018H
        SETB TRO
WAIT  : JNB TFO,wait
        CLR TRO
        CLR TFO
        DJNZ RO,BACK
        RET
        END.
```



PDK