CBCS SCHEME

USN | 1 | C | R | 1 | 8 | E | E | 0 | 4 | 2 |

18EE52

Fifth Semester B.E. Degree Examination, Jan./Feb. 2021
Microcontroller

Time: 3 hrs. Max. Marks: 100

Note: ***Answer any FIVE full questions, choosing ONE full question from each module.***

VTU-22-02-2021 02:02pm CR - CR - CR - CR

appeal to evaluator and /or equations written eg. 42+8 = 50, will be treated as malpractice.

Module-1

1 a. Draw and explain the architecture of 8051 microcontroller. **(08 Marks)**

b. Compare the microprocessor and microcontroller. **(06 Marks)**

c. Explain with the help of diagram, how to interface external code memory to 8051 microcontroller. **(06 Marks)**

OR

2 a. Describe the functions of various pins of 8051 microcontroller with pin diagram. **(08 Marks)**

b. Explain the various addressing modes of 8051 microcontroller examples. **(08 Marks)**

c. List and explain the criteria for choosing a microcontroller. **(04 Marks)**

Module-2

3 a. Define assembler directives. Explain the assembler directives of 8051 micronctroller with examples. **(08 Marks)**

b. Write a program to load the accumulator with the value 55H and complement the ACC 700 times. **(06 Marks)**

c. Write a program to count positive and negative numbers in a given array. **(06 Marks)**

OR

4 a. Explain the operation performed by the following instructions with examples.
i) DJNZ R1, rel ii) DA A iii) MOVX A, @ DPTR iv) SWAP A. **(08 Marks)**

b. Write a program to find factorial of a number. **(06 Marks)**

c. Write an assembly language program to toggle the bits of port P1. **(06 Marks)**

Module-3

5 a. Write 8051 program to generate square wave with t_{ON} = 3ms and t_{OFF} = 10ms an all pins of port 0. **(08 Marks)**

b. Explain the bit structure of TMOD register. **(06 Marks)**

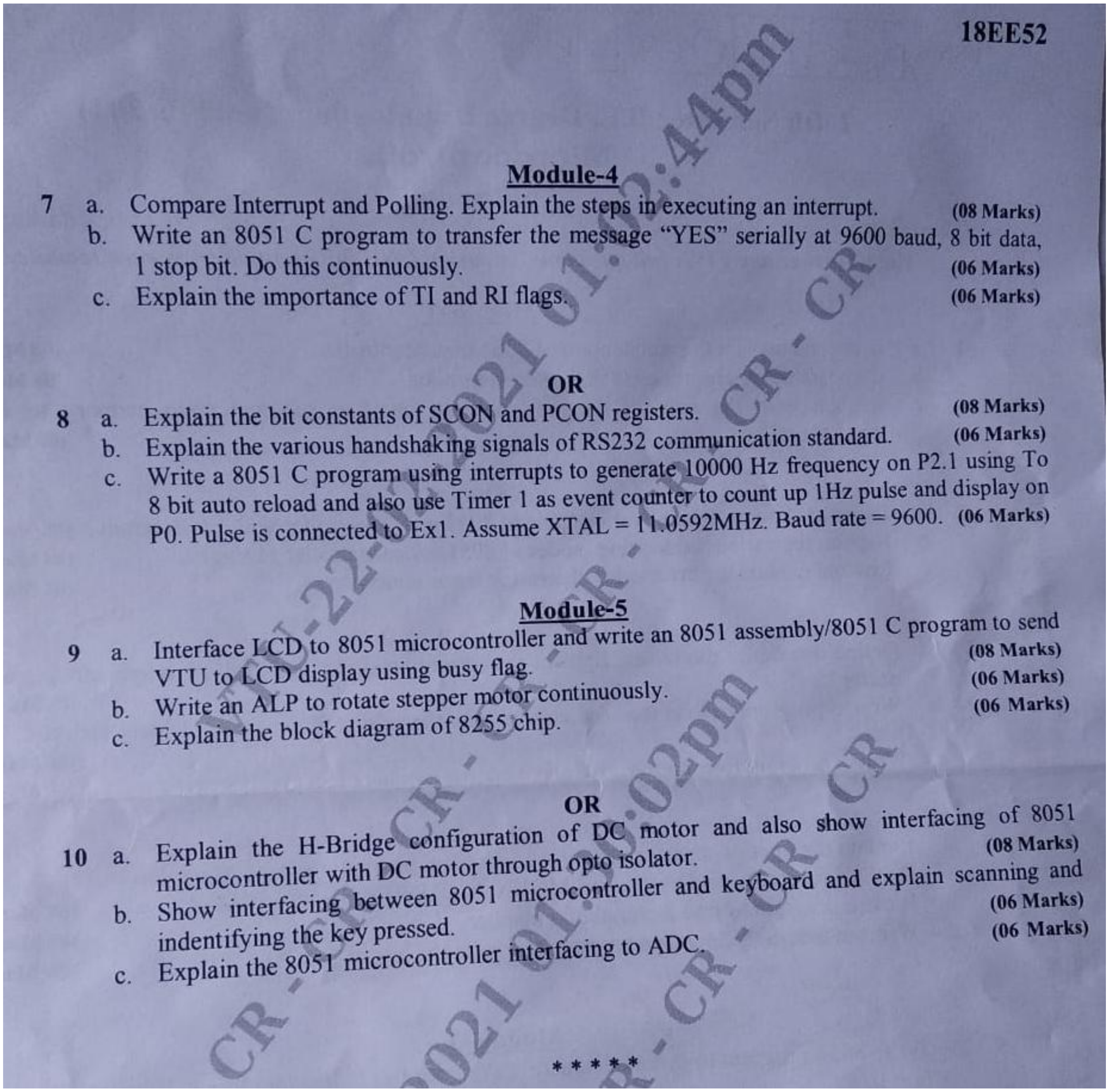
c. Write an 8051 C program to convert FD hex to decimal and display the digits on P0, P1 and P2. **(06 Marks)**

OR

6 a. Explain Mode – 2 programming of 8051 timer. Describe the different steps to program in Mod 2. **(08 Marks)**

b. Write a 8051 C program to bring in a byte of data serially one bit at a time Via P2.0. The LSB should come in first. **(06 Marks)**

c. Write a 8051 C program to toggle all the bites of port P2 continuously with some delay in between. Use Timer 0, 16 bit mode to generate the delay. **(06 Marks)**



18EE52

Module-4

7 a. Compare Interrupt and Polling. Explain the steps in executing an interrupt. (08 Marks)

b. Write an 8051 C program to transfer the message "YES" serially at 9600 baud, 8 bit data, 1 stop bit. Do this continuously. (06 Marks)

c. Explain the importance of TI and RI flags. (06 Marks)

OR

8 a. Explain the bit constants of SCON and PCON registers. (08 Marks)

b. Explain the various handshaking signals of RS232 communication standard. (06 Marks)

c. Write a 8051 C program using interrupts to generate 10000 Hz frequency on P2.1 using To 8 bit auto reload and also use Timer 1 as event counter to count up 1Hz pulse and display on P0. Pulse is connected to Ex1. Assume XTAL = 11.0592MHz. Baud rate = 9600. (06 Marks)

Module-5

9 a. Interface LCD to 8051 microcontroller and write an 8051 assembly/8051 C program to send VTU to LCD display using busy flag. (08 Marks)

b. Write an ALP to rotate stepper motor continuously. (06 Marks)

c. Explain the block diagram of 8255 chip. (06 Marks)

OR

10 a. Explain the H-Bridge configuration of DC motor and also show interfacing of 8051 microcontroller with DC motor through opto isolator. (08 Marks)

b. Show interfacing between 8051 microcontroller and keyboard and explain scanning and indentifying the key pressed. (06 Marks)

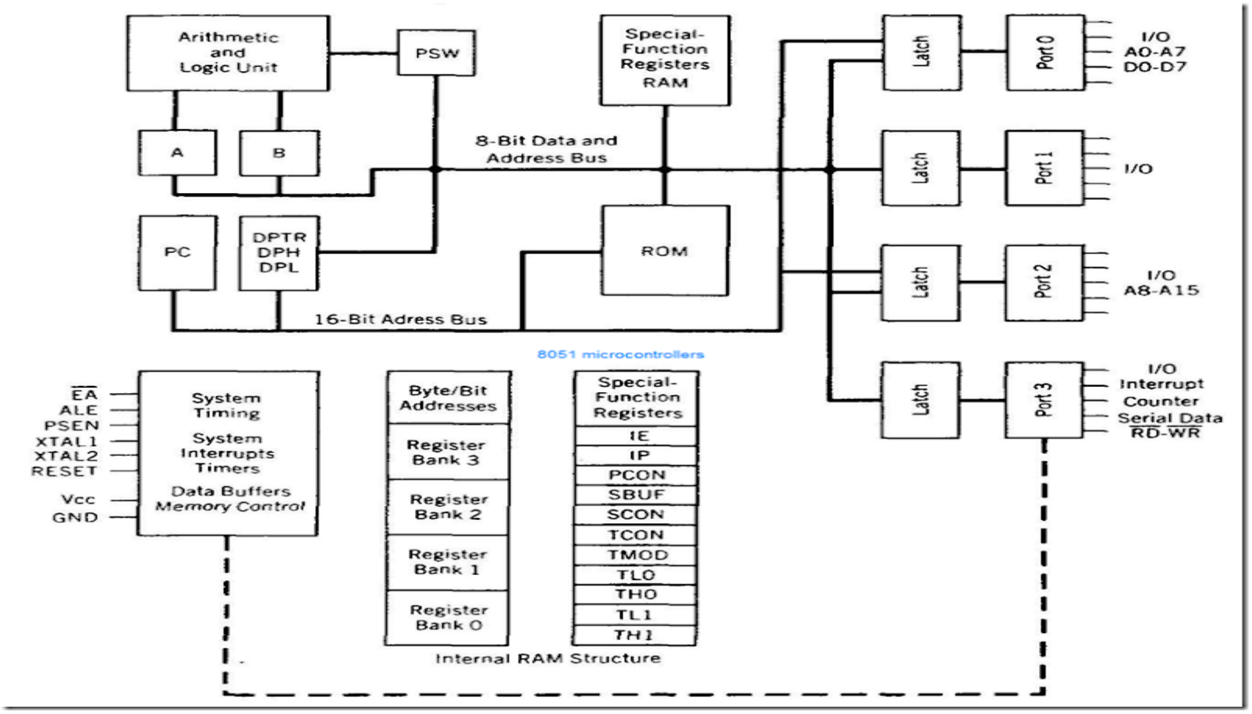
c. Explain the 8051 microcontroller interfacing to ADC. (06 Marks)

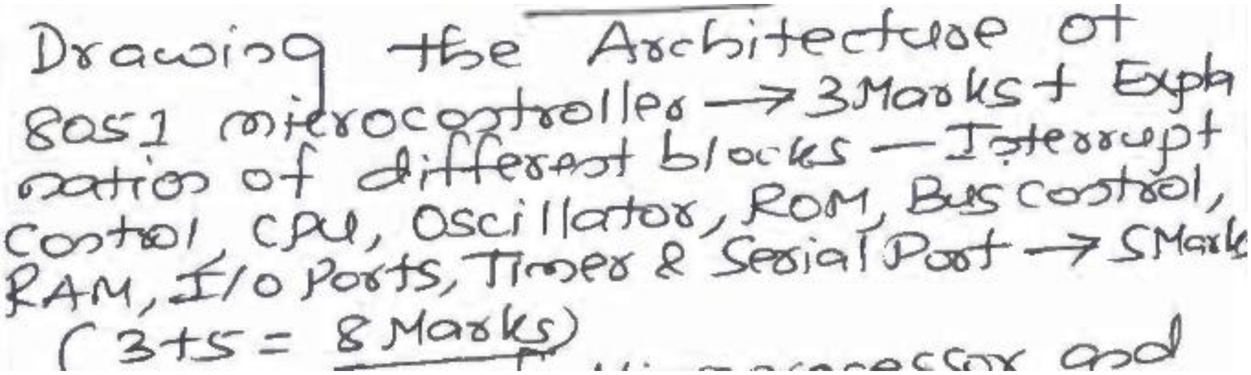
* * * * *

VTU-22-01-2021 01:02:44pm CR - CR - CR

Solution of model QP

1. Explain With the neat diagram, the programming model of 8051 Microcontrollers.



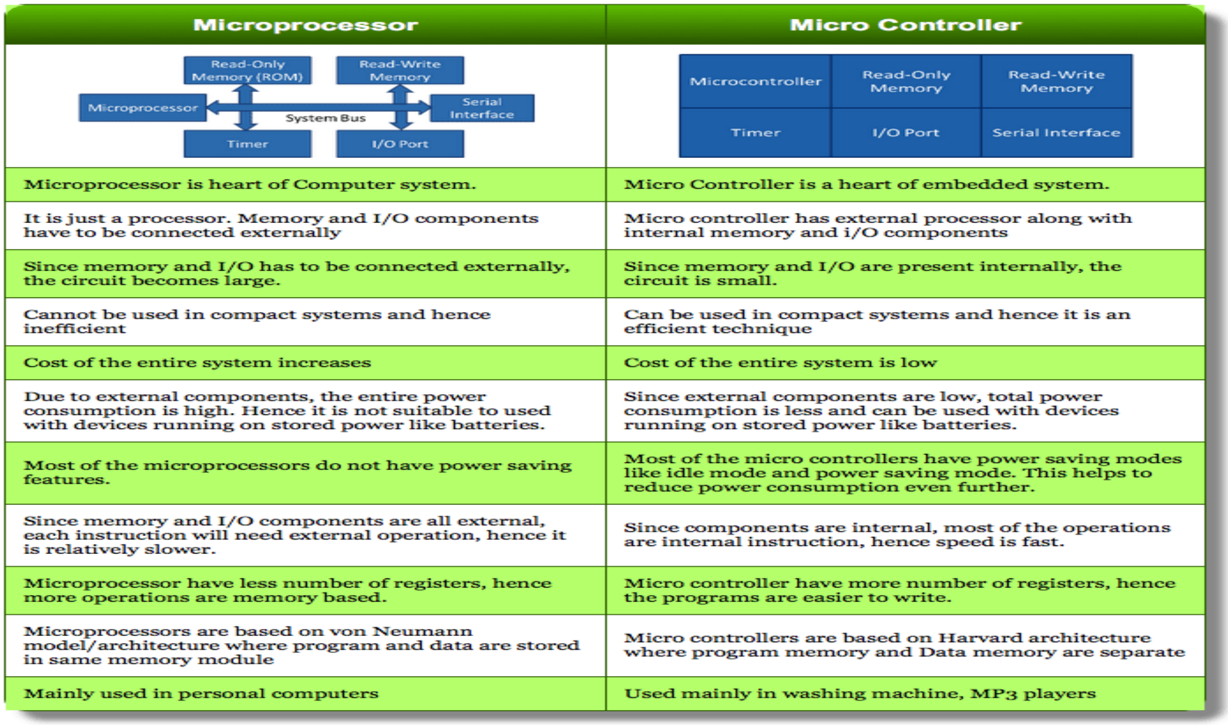


Drawing the Architecture of 8051 microcontroller → 3Marks + Explanation of different blocks — Interrupt control, CPU, Oscillator, ROM, Bus control, RAM, I/O Ports, Timer & Serial Port → 5Mark
(3+5 = 8 Marks)

Microprocessor and

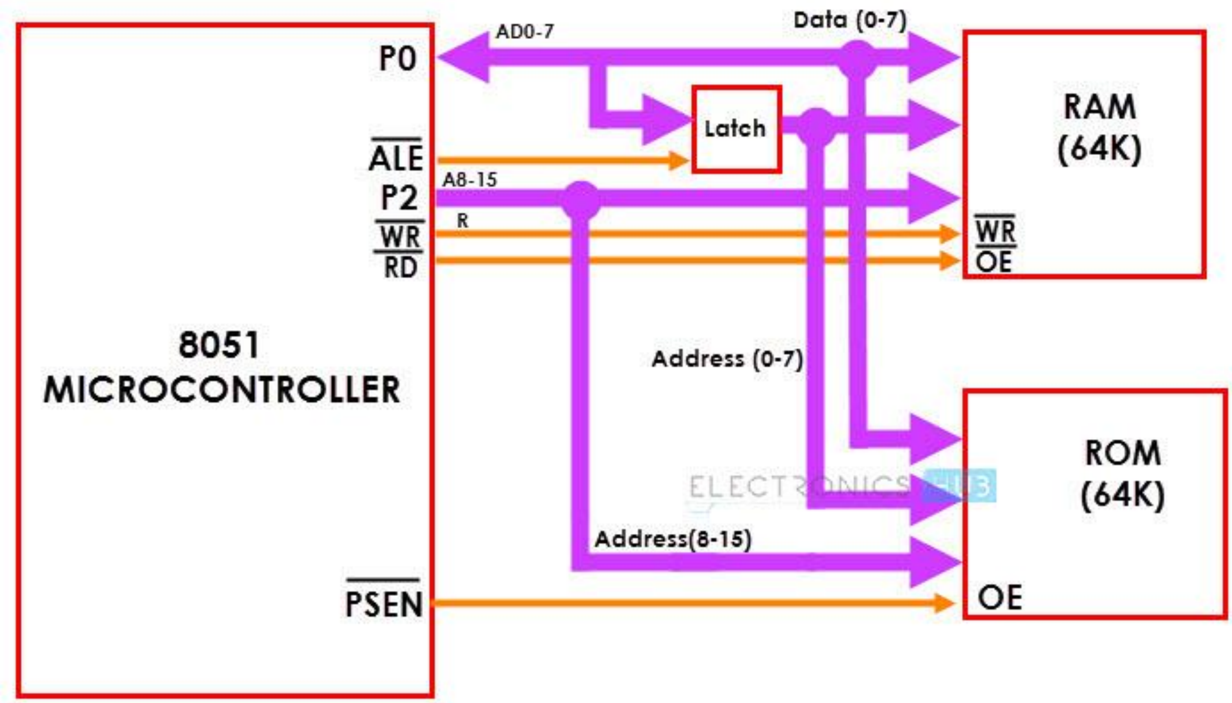
2.

2. Compare the Microprocessor and Microcontroller.



Microprocessor	Micro Controller
Read-Only Memory (ROM) Read-Write Memory Microprocessor System Bus Serial Interface Timer I/O Port	Microcontroller Read-Only Memory Read-Write Memory Timer I/O Port Serial Interface
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

3. Interface 4k bytes RAM and 8k bytes ROM to 8051 microcontroller in such a way that starting address of RAM is 1000H and ROM is C000H.



General procedure is given as:

If external program/data memory is to be interfaced, they are interfaced in the following way. External program memory is fetched if either of the following two conditions is satisfied.

- * $\overline{EA}$ (Enable Address) is low. The microcontroller by default starts searching for program from external program memory.
- * Content of program counter register (**PC**) is higher than 0FFFH for 8051.

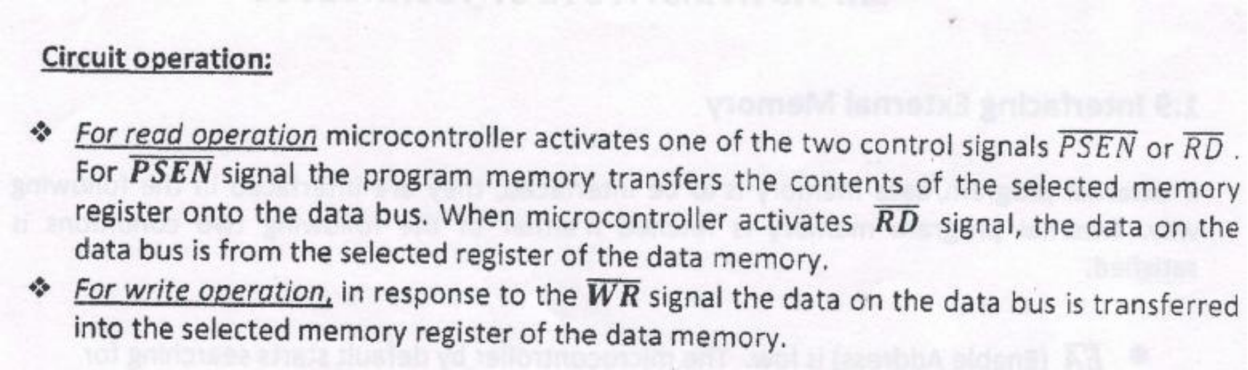
$\overline{PSEN}$, an output pin, tells the outside world whether the external memory is accessed for fetching the program or data memory ($\overline{EA}$ is user configurable. $\overline{PSEN}$ is processor controlled.)

The ports **P0** and **P2** are used as address bus.

- ■ The port **P0** is also used as data bus. Therefore the port **P0** is said to be multiplexed (contains both address and data).
- ■ To de-multiplex the port **P0** a latch **74LS373** is used. The latch is enabled by the ALE signal activated by the microcontroller.
- ■ The output of the latch and the port **P2** forms the complete 16-bit address bus.
- ■ After the port **P0** is de-multiplexed, the same can be used for data.
- ■ The address bus is connected to both data and program memory. The address on the address bus selects one of the registers of the memory.

OPERATION OF THE CIRCUIT:

- ❖ The microcontroller sends the address on both ports **P0** and **P2**. Then **ALE** signal is activated.
- ❖ The **ALE** signal enables the latch. The address on the input of the latch is propagated to the output of the latch.
- ❖ Then microcontroller will deactivate **ALE** signal. This disables the latch and the output of the latch which is the address remains fixed. When the microcontroller removes the address on the port **P0**, the output of the latch will not change. Thus the output of latch will continue to hold the address.
- ❖ The port **P0** now will not contain address and it is said to be de-multiplexed.
- ❖ The port **P0** can now be used for data communication.
- ❖ The output of the latch and contents of port **P2** forms the address bus. The address bus will select one of the registers of the memory. After the memory is selected by the address on the address bus, the READ/WRITE operation is decided by the control signals.



Circuit operation:

- *For read operation* microcontroller activates one of the two control signals $\overline{PSEN}$ or $\overline{RD}$. For $\overline{\mathbf{PSEN}}$ signal the program memory transfers the contents of the selected memory register onto the data bus. When microcontroller activates $\overline{\mathbf{RD}}$ signal, the data on the data bus is from the selected register of the data memory.
- *For write operation,* in response to the $\overline{\mathbf{WR}}$ signal the data on the data bus is transferred into the selected memory register of the data memory.

2. B.Explain with example the various addressing modes of 8051.

Addressing Modes.

The CPU can acces data in various ways. The data could be in a register, or in memory or be provided as an immediate value. The various ways of accessing a data are called addressing modes.

There are fine different types of Addressing mode

1. Immediate
2. Register
3. Direct
4. Register Indirect
5. Indexed.

1. Immediate Addressing Mode :-

→ In this addressing mode, the source operand is a constant i.e immediate data.

→ The immediate data must be preceded by the pound sign "#".

→ This addressing mode can be used to load information into any of the registers including the DPTR register.

```
Eg:- ① MOV A, #56H        ; load 56H into Accumulator
       MOV R3, #62        ; load the decimal value 62
                            into R3
       MOV B, #40H        ; Load 40H into B
       MOV DPTR, #4521H   ; DPTR = 4512H.

     ② MOV DPTR, #2550H
        Or
       MOV DPL, #50H
       MOV DPH, #25H.
```

2. Register Addressing Mode:-

It involves the use of registers to hold the data to be Manipulated.

Eg :-

```
MOV  A, R0     ; copy the contents of R0 into A
MOV  R2, A     ; copy the contents of A into R2
ADD  A, R5     ; Add the contents of A ~~to~~ the
                 contents of R5
ADD  A, R7     ; add the contents of R7 to
                 contents of A.
MOV  R6, A     ; save the accumulator in R6
```

→ The size of the source & destination should be matched.

Eg:- ① MOV DPTR, #25F5H → valid.

② MOV DPTR, A → invalid

source is A → 8 bit register

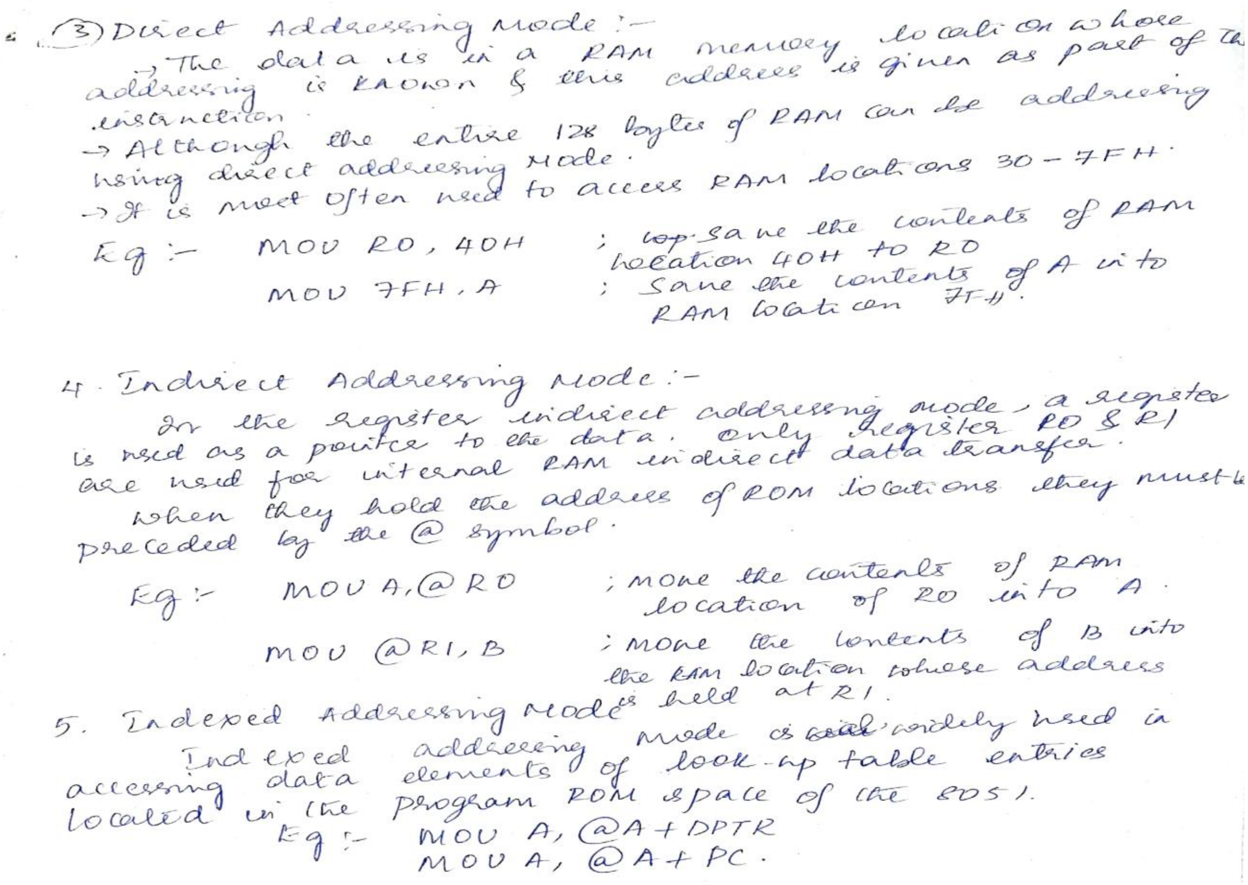
Destinatination is → 16 bit register

→ The data can be moved between the accumulator & Rn (for n = 0 to 7), but the movement of data between Rn is not allowed.

Eg :-

MOV A, R0 valid.

MOV R0, R7 invalid.



(3) Direct Addressing Mode :-

→ The data is in a RAM memory location whose addressing is known & this address is given as part of the instruction.

→ Although the entire 128 bytes of RAM can be addressing using direct addressing mode.

→ It is most often used to access RAM locations 30 – 7FH.

Eg :- MOV R0, 40H ; ~~cop~~ save the contents of RAM location 40H to R0

MOV 7FH, A ; save the contents of A into RAM location 7FH.

4. Indirect Addressing Mode :-

In the register indirect addressing mode, a register is used as a pointer to the data. Only register R0 & R1 are used for internal RAM indirect data transfer. When they hold the address of ROM locations they must be preceded by the @ symbol.

Eg :- MOV A,@R0 ; move the contents of RAM location of R0 into A.

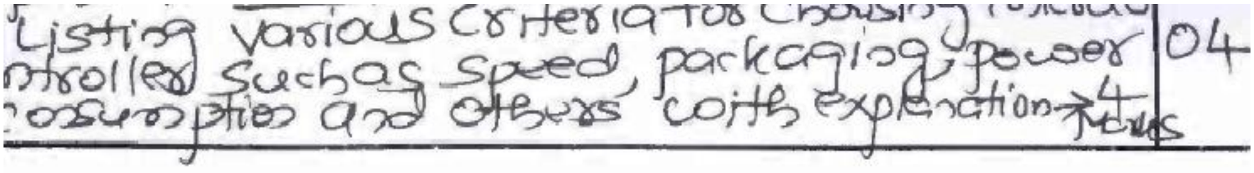
MOV @R1, B ; move the contents of B into the RAM location whose address is held at R1.

5. Indexed Addressing Mode

Indexed addressing mode is ~~[illegible]~~ widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051.

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

2. c.



Listing various Criteria for choosing microcontrollers such as speed, packaging, power consumption and others with explanation→4 Marks	04

3. A.Define assembler directives. With example explain all the assembler directives supported by 8051 microcontroller.

Data Byte Directive:

- **The DB directive is the most widely used data directive in the assembler.**
- **It is used to define the 8-bit data.**
- **When DB is used to define data, the numbers can be in decimal, binary, hex, or ASCII formats. For decimal, the “D” after the decimal number is optional, but using “B” (binary) and “H” (hexadecimal) for the others is required.**

ORG Directive

- **The ORG directive is used to indicate the beginning of the address.**

- The number that comes after ORG can be either in hex or in decimal.0000 or 0000D MOV A,30
- If the number is not followed by H, it is decimal and the assembler will convert it to hex.
- Some assemblers use “. ORG” (notice the dot) instead of “ORG” for the origin directive. Check your assembler.

EQU

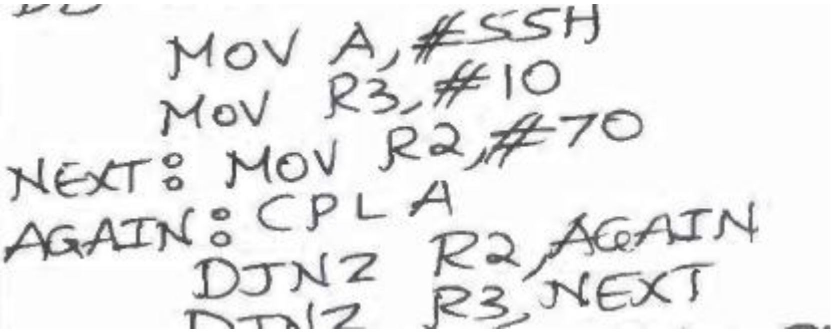
- This is used to define a constant without occupying a memory location.
- The EQU directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program, it constant value will be substituted for the label.
- The following uses EQU for the counter constant and then the constant is used to load the R3 register.
- Eg: COUNT EQU 25

```
        COUNT EQU 25
        . . . . . . . .  ..
        MOV R3, # COUNT
```

END Directive

- Another important pseudocode is the END directive.
- This indicates to the assembler the end of the source (asm) file.
- The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler.
- Some assemblers use “. END” (notice the dot) instead of “END”.

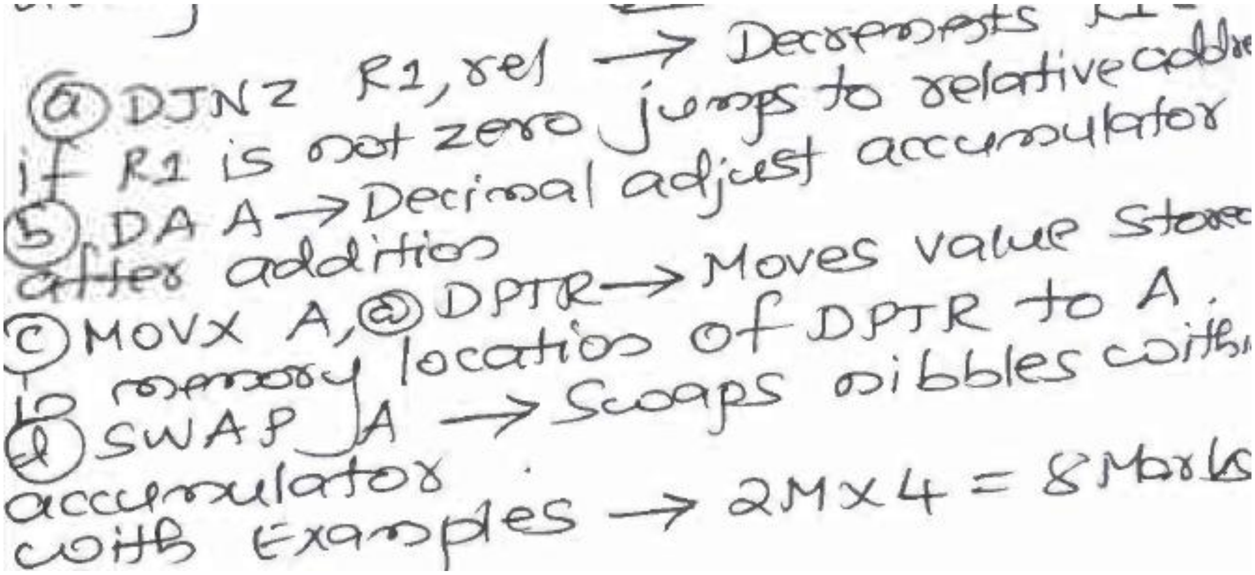
3.b.



```
        MOV A,#55H
        MOV R3,#10
NEXT:   MOV R2,#70
AGAIN:  CPL A
        DJNZ R2,AGAIN
        DJNZ R3,NEXT
```

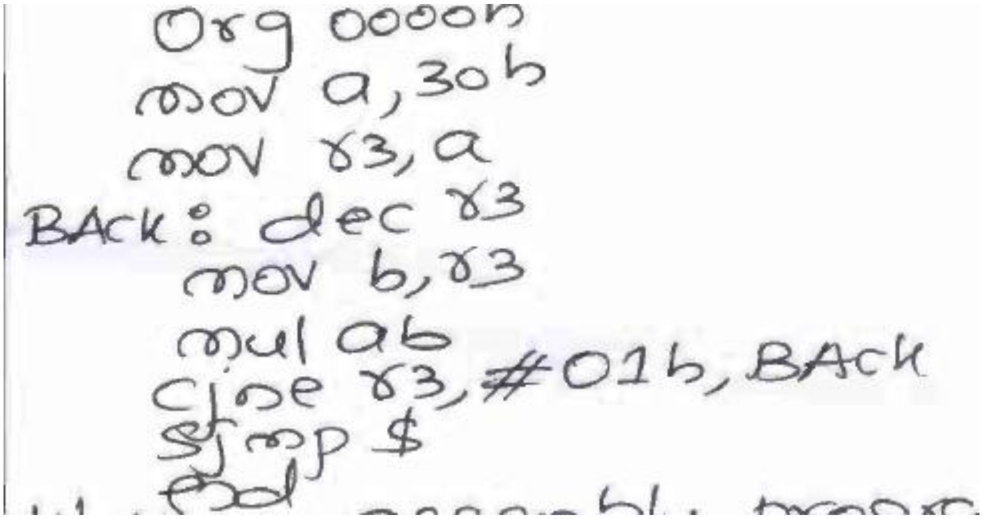
3.c.

4.a.



(a) DJNZ R1, rel → Decrements R1
if R1 is not zero jumps to relative addre
(b) DA A → Decimal adjust accumulator
after addition
(c) MOVX A,@DPTR → Moves value store
to memory location of DPTR to A.
(d) SWAP A → Swaps nibbles with
accumulator
with Examples → 2M×4 = 8Marks

4. **b.**



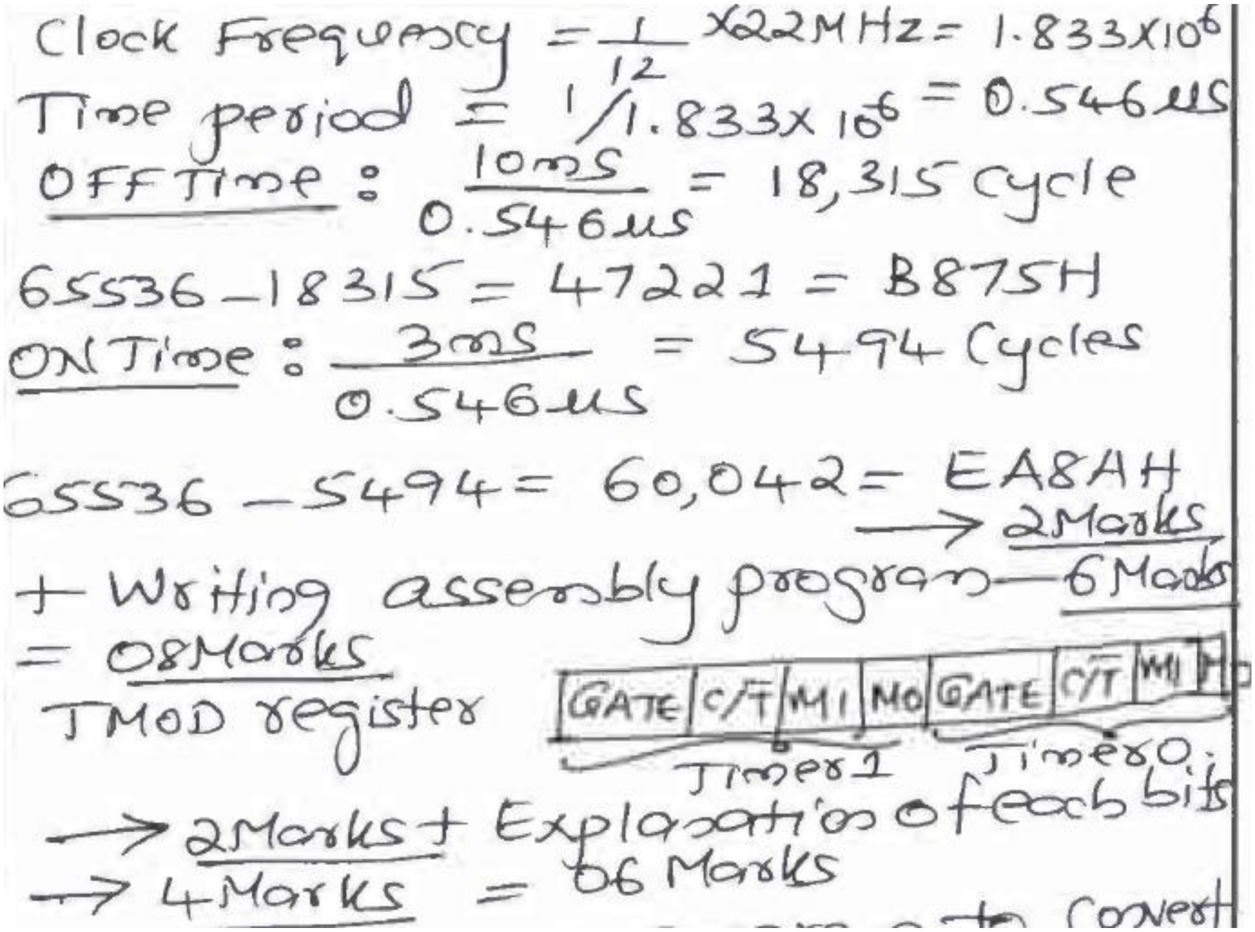
```
    Org 0000h
    mov a,30h
    mov r3,a
BAck: dec r3
    mov b,r3
    mul ab
    cjne r3,#01h,BAck
    sjmp $
    end
```

4.c.

```
        ORG 0000H
MOV R0,#08H
BACK: MOV A,#55H
MOV P1,A
ACALL DELAY
 MOV A,#0AAH
MOV P1,A
ACALL DELAY
```

```
DELAY:DJNZ R0,BACK
SJMP $
END
```

5. a.b.



Clock Frequency $= \frac{1}{12} \times 22 MHz = 1.833 \times 10^6$

Time period $= 1/1.833 \times 10^6 = 0.546 \mu s$

OFF Time : $\frac{10 ms}{0.546 \mu s} = 18,315$ cycle

$65536 - 18315 = 47221 = B875H$

ON Time : $\frac{3 ms}{0.546 \mu s} = 5494$ Cycles

$65536 - 5494 = 60,042 = EA8AH$

→ 2Marks

+ Writing assembly program — 6 Marks

= 08Marks

TMOD register

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M

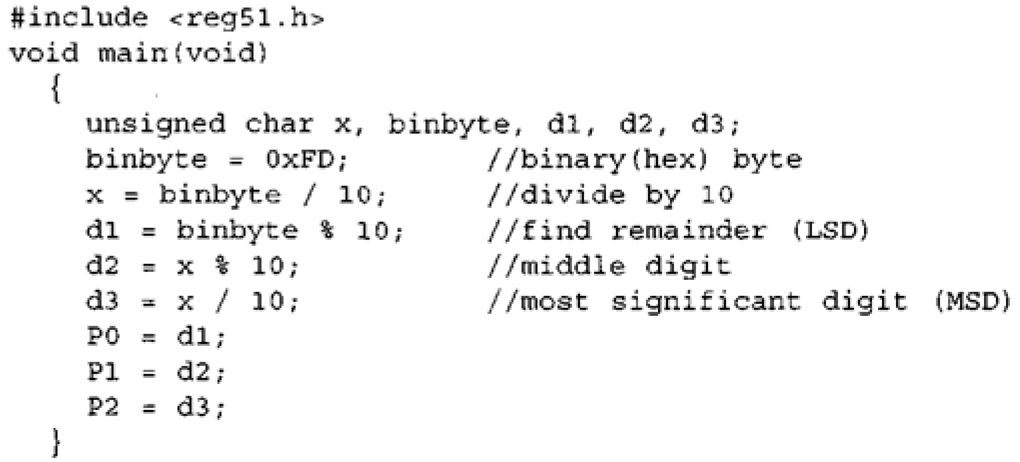
Timer1 Timer0

→ 2Marks + Explanation of each bits

→ 4Marks = 06 Marks

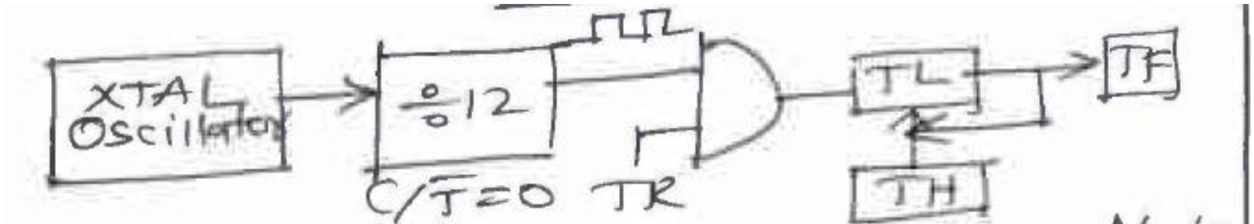
to Convert

5.c.



```
#include <reg51.h>
void main(void)
  {
    unsigned char x, binbyte, d1, d2, d3;
    binbyte = 0xFD;          //binary(hex) byte
    x = binbyte / 10;        //divide by 10
    d1 = binbyte % 10;       //find remainder (LSD)
    d2 = x % 10;             //middle digit
    d3 = x / 10;             //most significant digit (MSD)
    P0 = d1;
    P1 = d2;
    P2 = d3;
  }
```

6.a.



with Explanation → 04 Marks

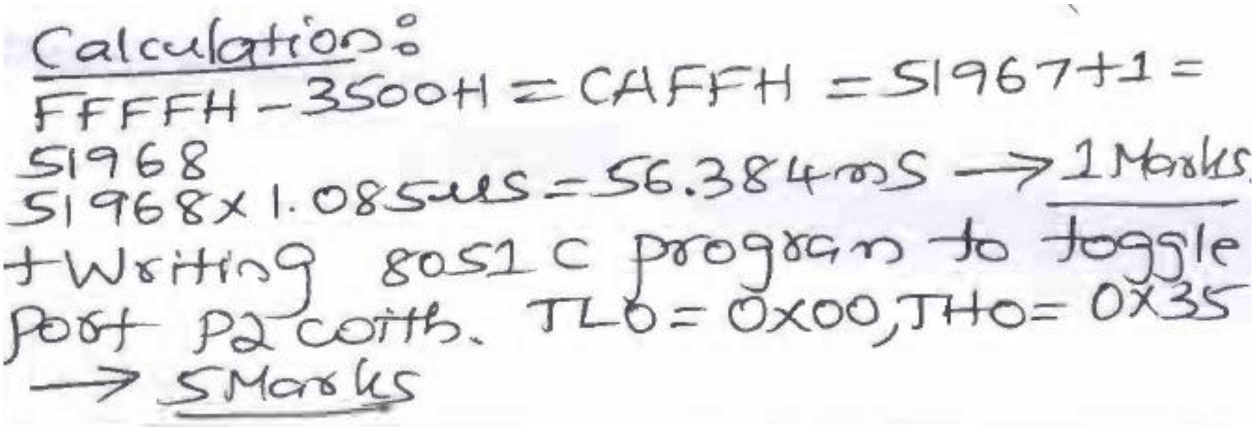
+ Writing steps to program in Mode2 → 04 Marks = 08 Marks.

6.b.

```
# include <reg51.h>

Sbit P1b0 =P1^0;

Sbit regAMSB= Acc^7;

Void main(void)

{

Unsigned char conbyte =0x44;

Unsigned char x;

Acc = conbyte;

For( x=0; x<8; x++)

{
```

```
AMSB  = P1b0;
Acc = Acc>>1;
}
P2=Acc;
}
```

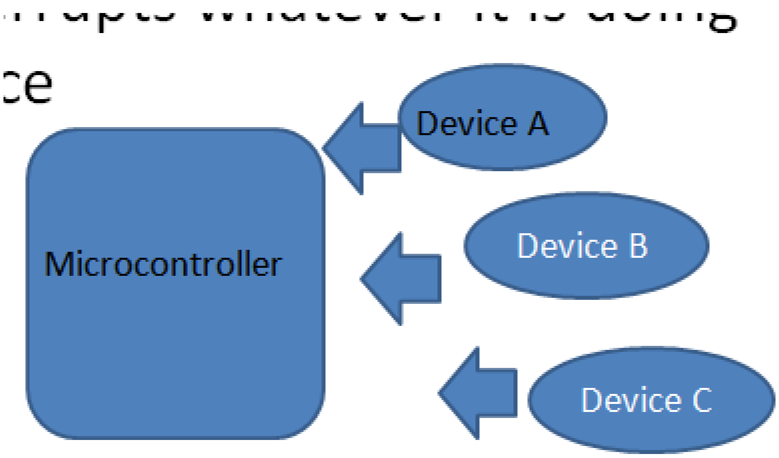
6.c.



Calculation:
FFFFH - 3500H = CAFFH = 51967+1 =
51968
51968x 1.085μs = 56.384mS → 1 Marks
+Writing 8051 C program to toggle
Port P2 with. TL0 = 0x00, TH0 = 0X35
→ 5Marks

7.a. In Interrupts, Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal

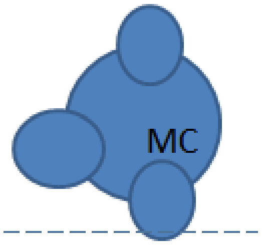
Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device



Polling

- The microcontroller continuously monitors the status of all devices in Round-robin manner.
- When the conditions are for a given device are met, it performs the service.

- After that, it moves on to monitor the next device until every one is serviced

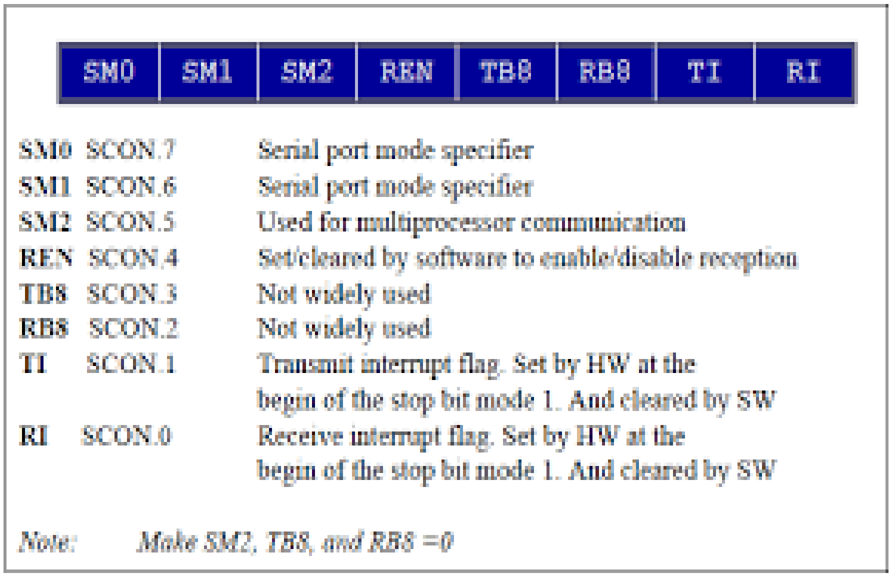


```
b. #include<reg51.h>
Void serTx(unsigned char);
void main(void)
{
TMOD=0x20;
TH1=0xFD;   TH1=‘-3’
SCON=0x50; 0101 0000
TR1=1;
while(1)
  {
serTx(‘Y’);
 serTx(‘E’);
serTx(‘S’);
    }
}
Void serTx(unsigned char x)
{
SBUF=x;
while(TI==0);
TI=0;
}
}
```

7.c. In the 8051 only one interrupt is set aside for serial communication.

- This interrupt is used to both send and receive data.
- If the interrupt bit in the IE register (IE.4) is enabled, when RI or TI is raised the 8051 gets interrupted and jumps to memory address location 0023H to execute the ISR.
- In that ISR we must examine the TI and RI flags to see which one caused the interrupt and respond accordingly.
- The serial interrupt is used mainly for receiving data and is never used for sending data serially.
- This is like receiving a telephone call, where we need a ring to be notified.
- If we need to make a phone call there are other ways to remind ourselves and so no need for ringing.
- In receiving the phone call, however, we must respond immediately no matter what we are doing or we will miss the call. Similarly, we use the serial interrupt to receive incoming data so that it is not lost.

8.a.

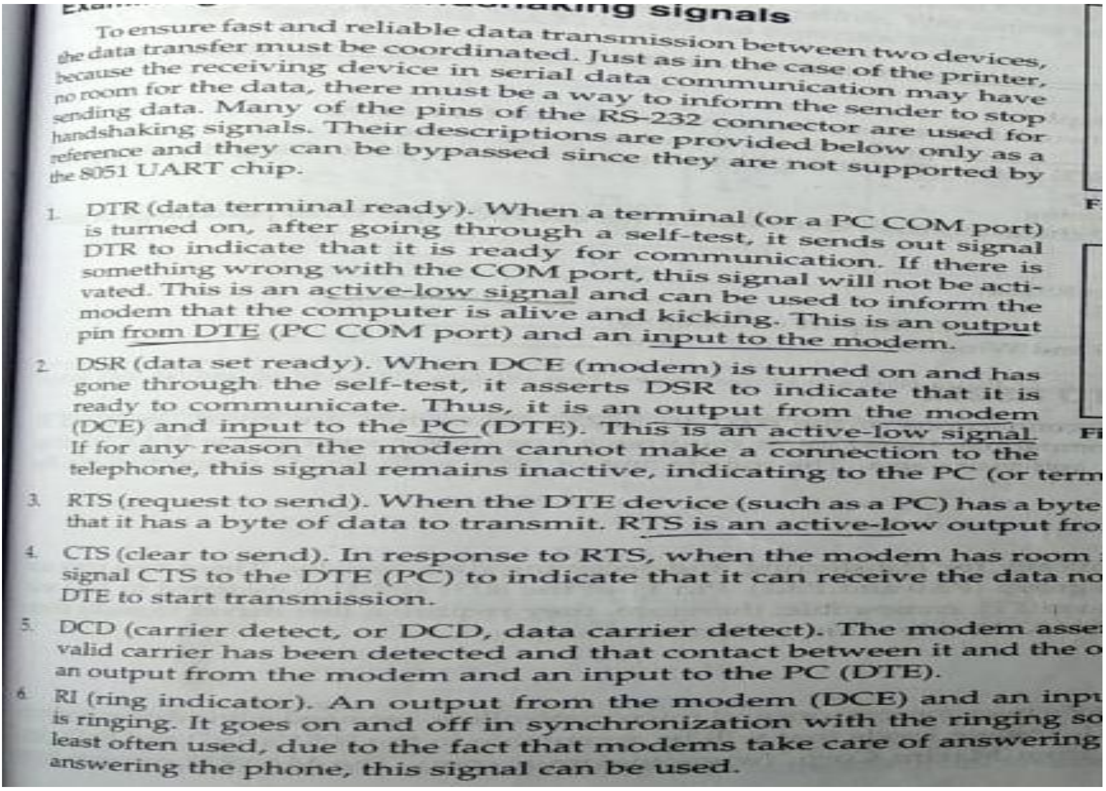


SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	Transmit interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

Note: Make SM2, TB8, and RB8 =0

b.



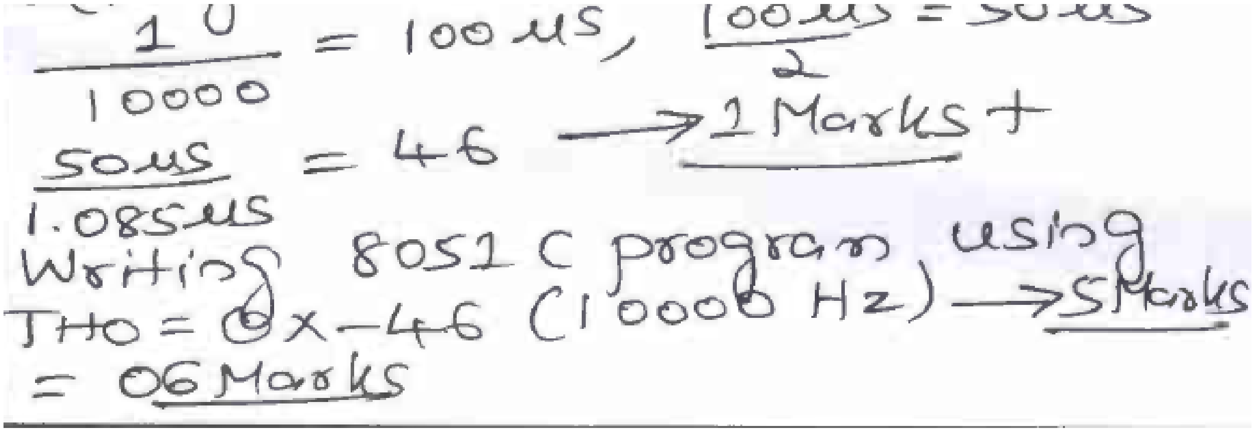
Exa aking signals

To ensure fast and reliable data transmission between two devices, the data transfer must be coordinated. Just as in the case of the printer, because the receiving device in serial data communication may have no room for the data, there must be a way to inform the sender to stop sending data. Many of the pins of the RS-232 connector are used for handshaking signals. Their descriptions are provided below only as a reference and they can be bypassed since they are not supported by the 8051 UART chip.

1. DTR (data terminal ready). When a terminal (or a PC COM port) is turned on, after going through a self-test, it sends out signal DTR to indicate that it is ready for communication. If there is something wrong with the COM port, this signal will not be activated. This is an active-low signal and can be used to inform the modem that the computer is alive and kicking. This is an output pin from DTE (PC COM port) and an input to the modem.
2. DSR (data set ready). When DCE (modem) is turned on and has gone through the self-test, it asserts DSR to indicate that it is ready to communicate. Thus, it is an output from the modem (DCE) and input to the PC (DTE). This is an active-low signal. If for any reason the modem cannot make a connection to the telephone, this signal remains inactive, indicating to the PC (or term
3. RTS (request to send). When the DTE device (such as a PC) has a byte that it has a byte of data to transmit. RTS is an active-low output fro
4. CTS (clear to send). In response to RTS, when the modem has room signal CTS to the DTE (PC) to indicate that it can receive the data no DTE to start transmission.
5. DCD (carrier detect, or DCD, data carrier detect). The modem asse valid carrier has been detected and that contact between it and the o an output from the modem and an input to the PC (DTE).
6. RI (ring indicator). An output from the modem (DCE) and an inpu is ringing. It goes on and off in synchronization with the ringing so least often used, due to the fact that modems take care of answering answering the phone, this signal can be used.

.

c.



$\frac{1}{10000} = 100\,\mu s$, $\frac{100\,\mu s}{2} = 50\,\mu s$

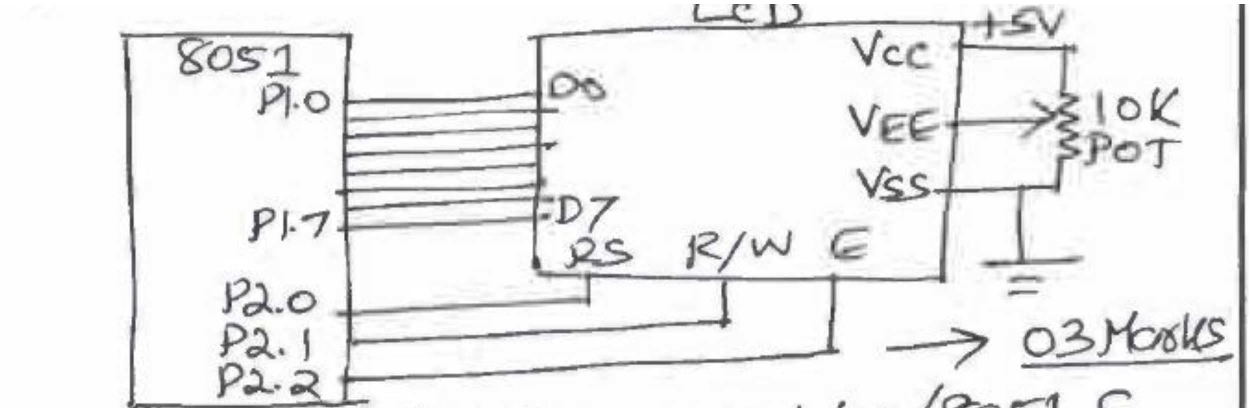
$\frac{50\,\mu s}{1.085\,\mu s} = 46 \rightarrow \underline{1\text{ Marks}} +$

Writing 8051 C program using

$TH0 = 0x{-46}\ (10000\text{ Hz}) \rightarrow \underline{5\text{ Marks}}$

$= \underline{06\text{ Marks}}$

9.a



→ 03 Marks

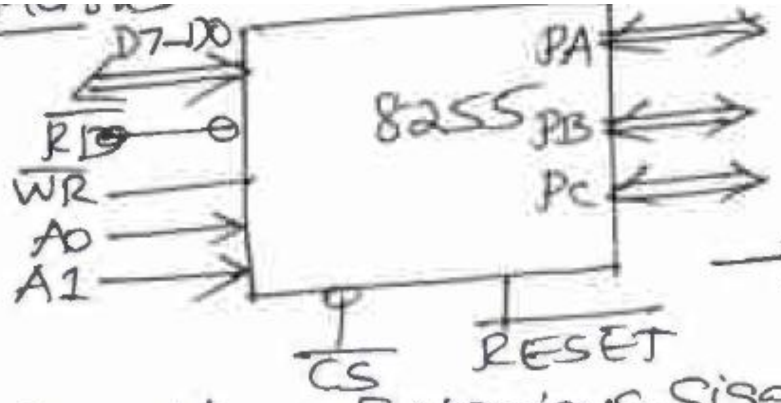
+ Writing 8051 assembly/8051 C program to interface 8051 to LCD using busy flag → 05 Marks = 08 Marks

Stepper motor

b.

```
ORG 00H
Main:Mov A,#OCH //clockwise direction first sequence
Acall Delay
Mov A,#06H     //clockwise direction second sequence
Acall Delay
Mov A,#03H   //clockwise direction third sequence
Acall Delay
Mov A,#09H   //clockwise direction fourth sequence
Acall Delay
SJMP Main
Delay:Mov TMOD,#10H    //Timer1 in Mode1
Mov TL1,#00H            //Load TL1 with Initial value
Mov TH1,#00H        //Load TH1 with Initial value
SETB TR1             //Start timer1
Here:JNB TF1,Here   //Monitor Timer1 Overflow flag
CLR TF1             //Clear Timer1 Overflow flag
CLR TR1              //Stop Timer
RET
END
```

c.



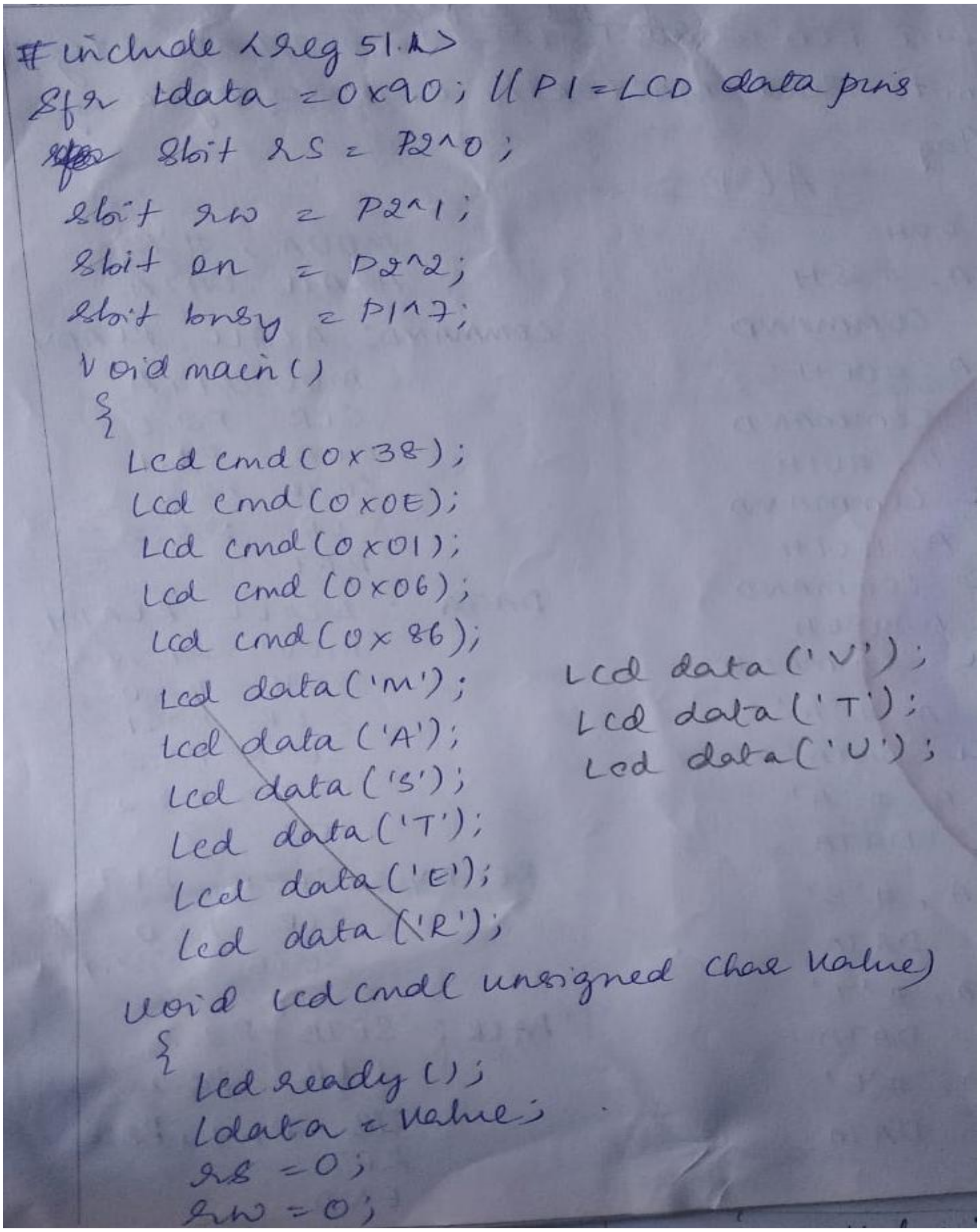
→ 2Marks

+ Explanation of various Signals → 4Ma

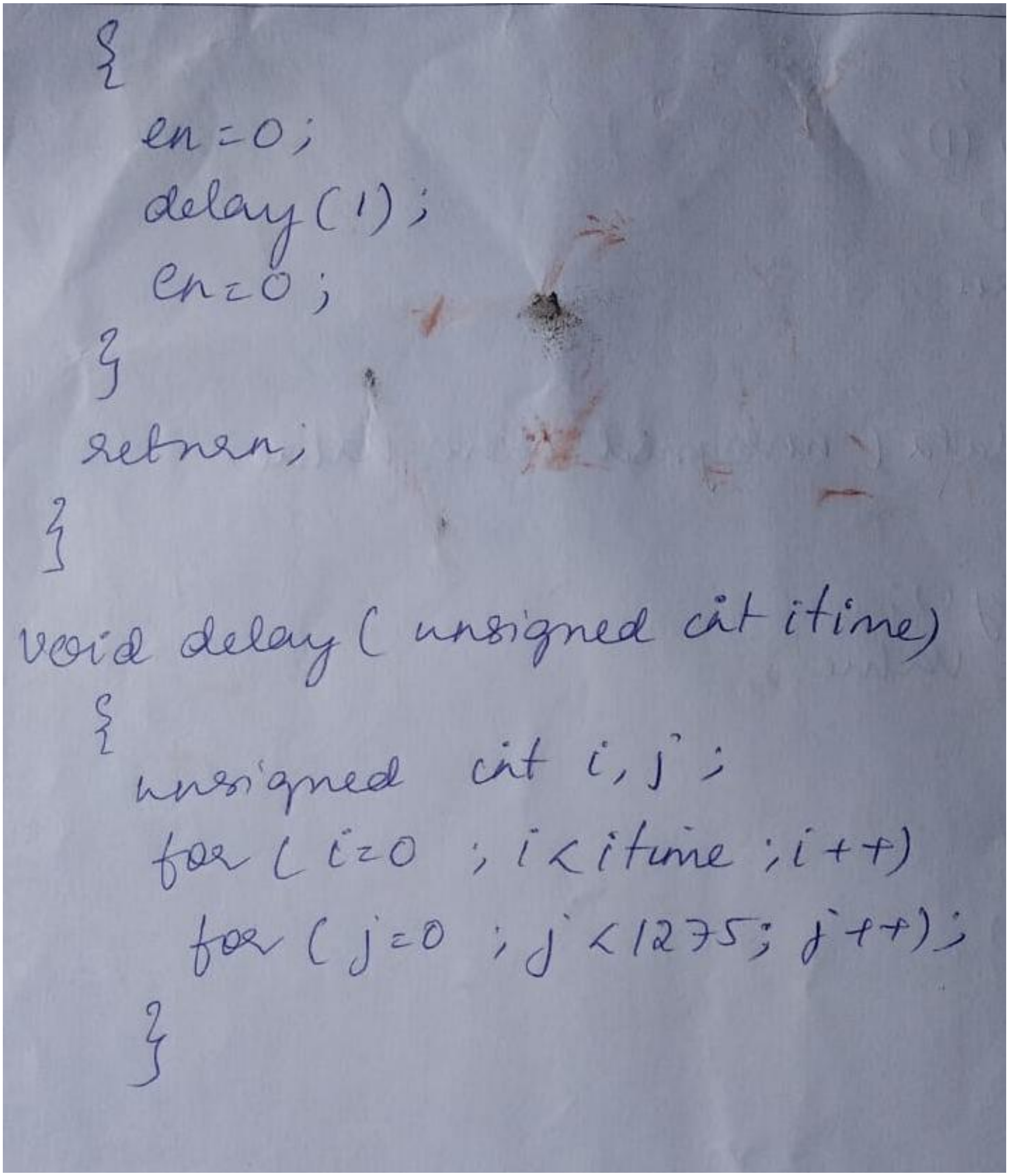
= 06Marks

OR

9.a.

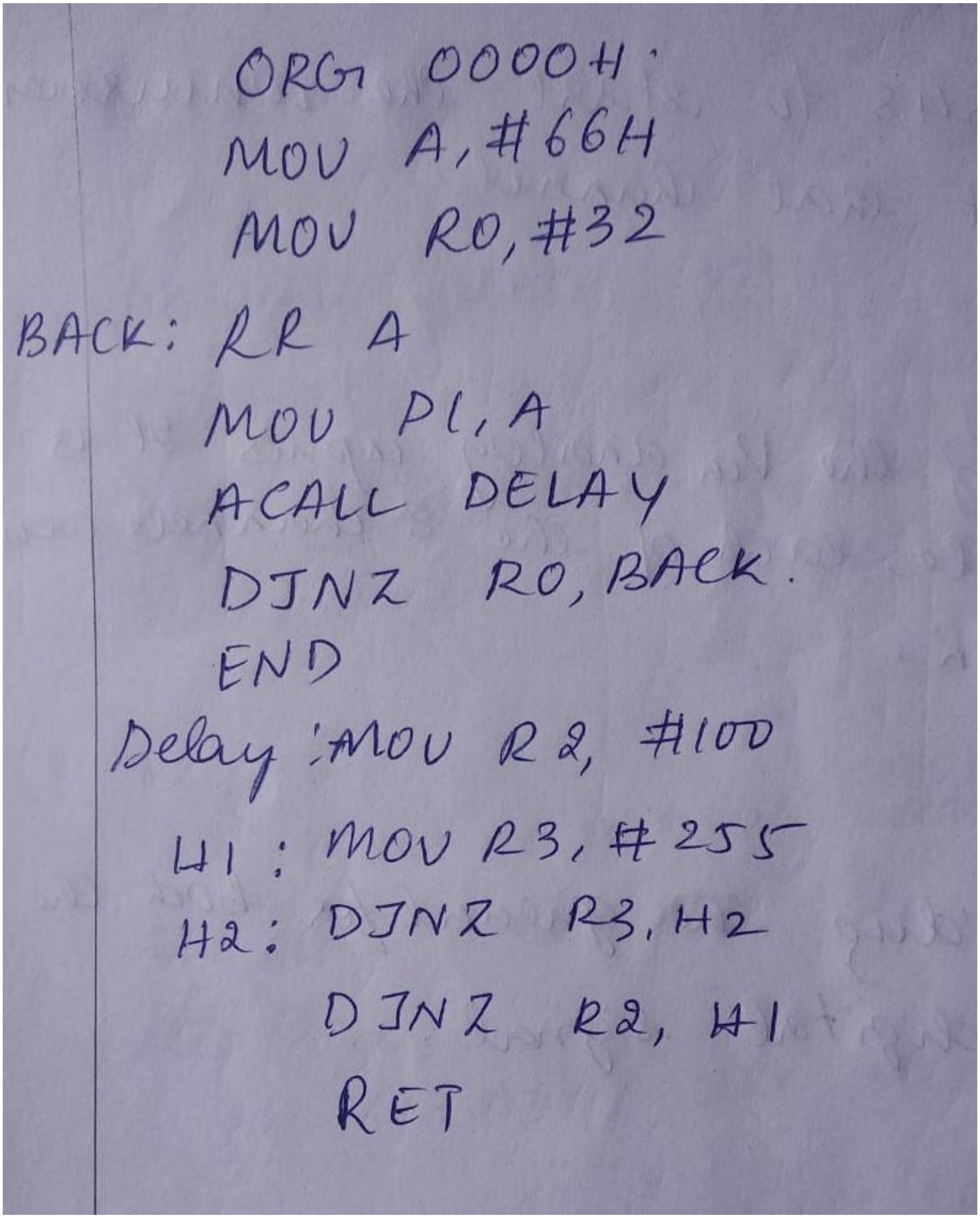


```
#include <reg 51.h>
sfr ldata = 0x90; //P1=LCD data pins
 ~~sfr~~ sbit rs = P2^0;
 sbit rw = P2^1;
 sbit en = P2^2;
 sbit busy = P1^7;
 void main()
 {
   Lcd cmd(0x38);
   Lcd cmd(0x0E);
   Lcd cmd(0x01);
   Lcd cmd(0x06);
   Lcd cmd(0x86);
   Lcd data('M');          Lcd data('V');
   Lcd data('A');          Lcd data('T');
   Lcd data('S');          Lcd data('U');
   Led data('T');
   Lcd data('E');
   Lcd data('R');
 void lcd cmd( unsigned char value)
  {
   lcd ready();
   ldata = value;
   rs = 0;
   rw = 0;
   en =1;
   delay (1);
   en =0;
   return;
 }
void lcddata( unsigned char value)
 {
  lcdready();
  ldata = value;
  rs =0;
  rw =0;
  en =1;
  delay (1);
  en =0;
 return;
}
void lcd ready ()
  {
   busy =1;
   rs=0;
   rw=1;
   while (busy ==1)
   busy =1;
   rs =0;
   rw =1;
   while(busy ==1)
```



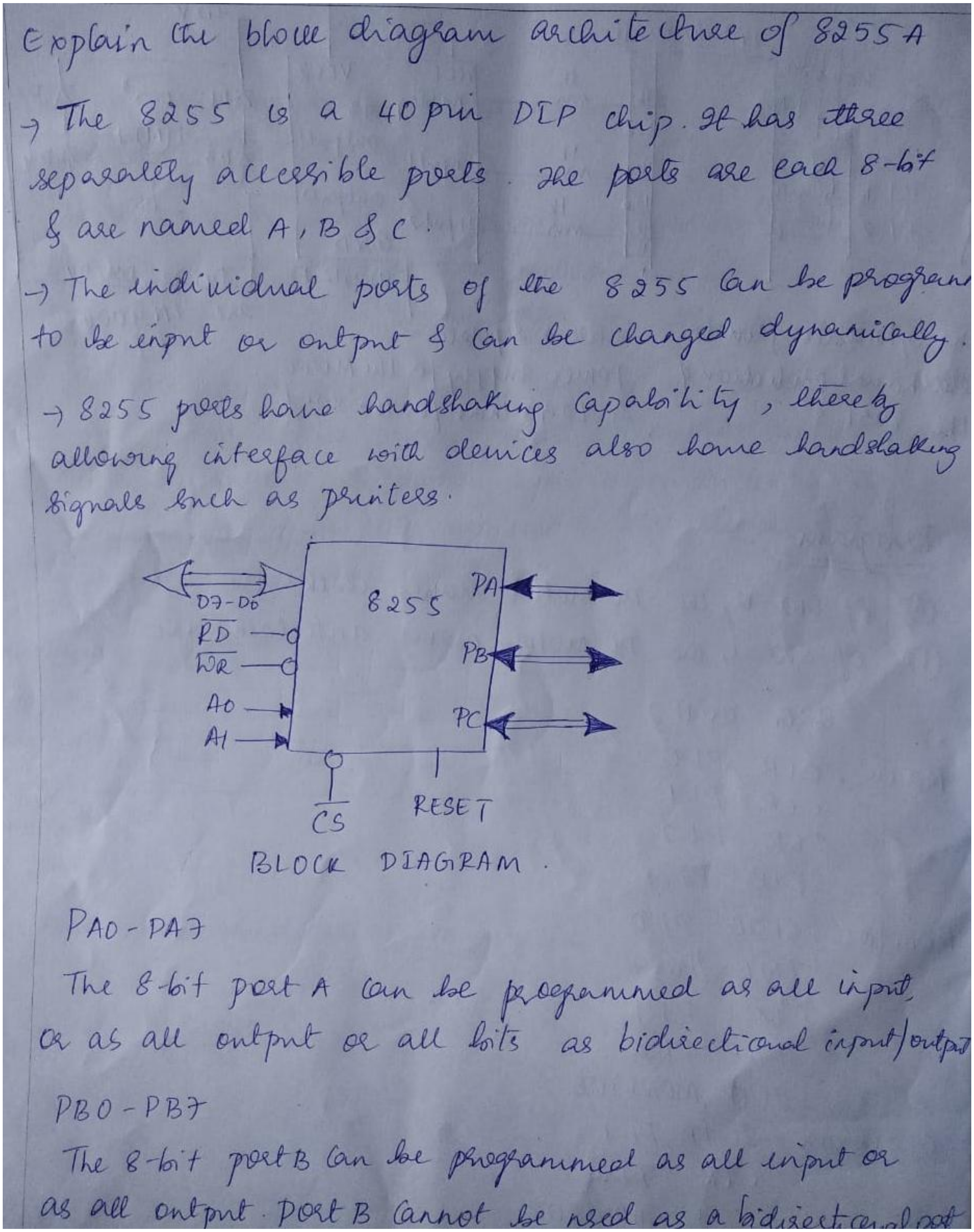
```
  {
    en=0;
    delay(1);
    en=0;
  }
  return;
 }
void delay(unsigned int itime)
 {
   unsigned int i,j;
   for(i=0 ;i<itime ;i++)
    for(j=0 ;j<1275; j++);
  }
```

9.b.



```
        ORG 0000H:
        MOV A,#66H
        MOV R0,#32
BACK:   RR A
        MOV P1,A
        ACALL DELAY
        DJNZ R0,BACK.
        END
Delay :MOV R2, #100
   H1 : MOV R3, #255
   H2: DJNZ R3,H2
        DJNZ R2, H1
        RET
```

9.c.



Explain the block diagram architechure of 8255A

→ The 8255 is a 40 pin DIP chip. It has three separately accessible ports. The ports are each 8-bit & are named A, B & C.

→ The individual ports of the 8255 can be program to be input or output & can be changed dynamically.

→ 8255 ports have handshaking capability, thereby allowing interface with devices also have handshaking signals such as printers.

BLOCK DIAGRAM.

PA0 - PA7

The 8-bit port A can be programmed as all input, or as all output or all bits as bidirectional input/output

PB0 - PB7

The 8-bit port B can be programmed as all input or as all output. Port B cannot be used as a bidirectional port

PC0 - PC7

This 8-bit port C can be all input or all output. It can also be split into two ports, CU (upper bits PC0 - PC3). Each can be used for input or output.

$\overline{RD}$ and $\overline{WR}$

These two active-low control signals are inputs to the 8255. The $\overline{RD}$ & $\overline{WR}$ signals from the 8031/51 are connected to these inputs

D0-D7 data pin

The data pins of the 8255 are connected to the data pins of the microcontroller allowing it to send data back & forth b/w the controller & the 8225 chip

RESET

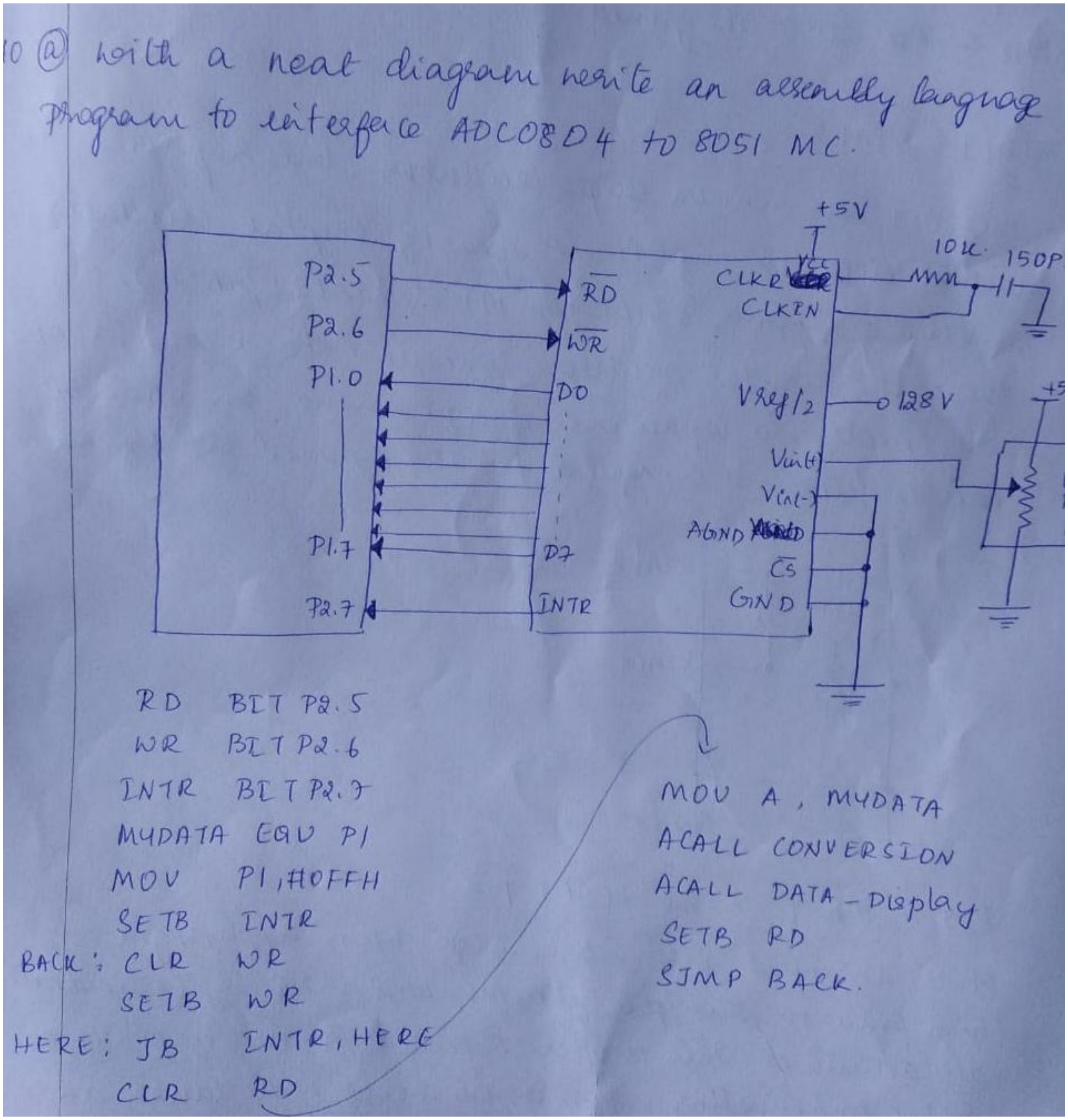
An active high signal i/p into the 8255 used to clear the control register. when RESET is activated, all ports are initialized as i/p ports.

This pin is connected to the RESET o/p of the s/m bus or grounded to make it inactive.

A0, A1 & $\overline{CS}$

$\overline{CS}$ (Chip Select) selects the entire chip, it is A0 & A1 that select specific ports. These three pins are used to access ports A, B, C or the control register as shown in fig.

10.c



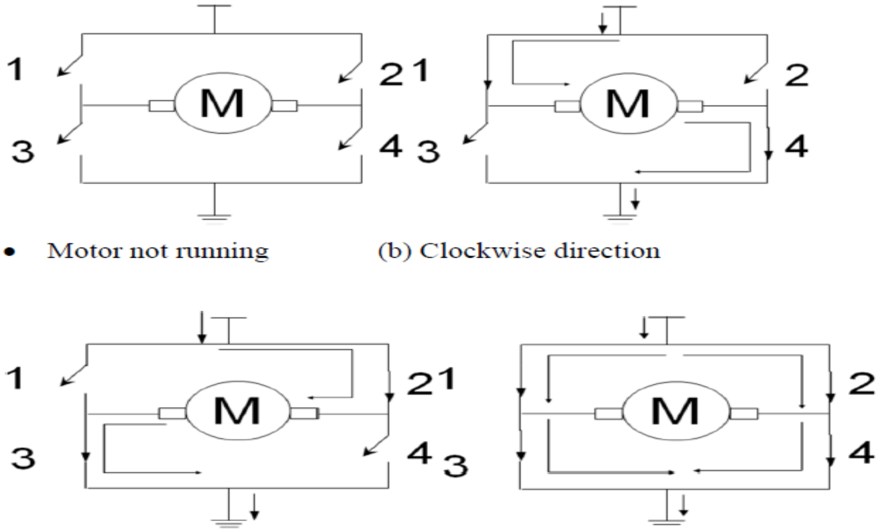
10 (a) with a neat diagram write an assembly language program to interface ADC0804 to 8051 MC.

+5V
Vcc
P2.5
P2.6
P1.0
P1.7
P2.7
RD
WR
D0
D7
INTR
CLKR
CLKIN
10k
150P
Vref/2
o 128 V
+5
Vin(+)
Vin(-)
AGND
CS
GND

```
        RD      BIT P2.5
        WR      BIT P2.6
        INTR    BIT P2.7
        MYDATA  EQU P1
        MOV     P1,#0FFH
        SETB    INTR
BACK:   CLR     WR
        SETB    WR
HERE:   JB      INTR,HERE
        CLR     RD
        MOV     A , MYDATA
        ACALL   CONVERSION
        ACALL   DATA_Display
        SETB    RD
        SJMP    BACK.
```

10.a

Bidirectional Control:



- Motor not running (b) Clockwise direction

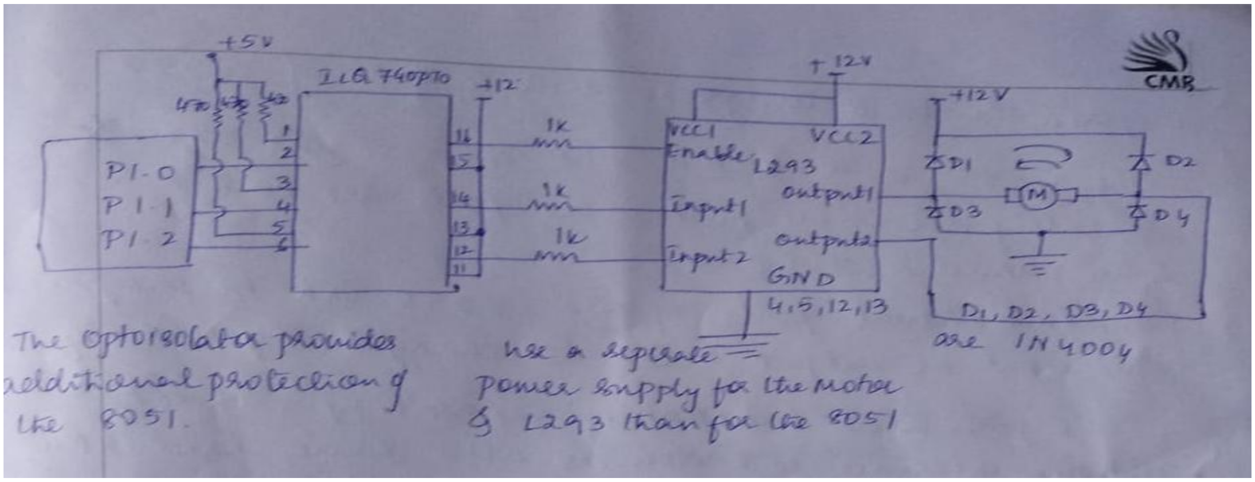
(c) Counter clockwise direction (d) Invalid state (short circuit)

Figure : H-Bridge Motor Configuration

Figure shows the H-Bridge motor configuration. It consists of four switches and based on the closing and opening of these switches the motor either rotates in clockwise or anti-clockwise direction.

As seen in figure 4a, all the switches are open hence the motor is not running. In b, turning of the motor is in one direction when the switches 1 and 4 are closed that is clockwise direction.

Similarly, in c the switches 2 and 3 are closed so the motor rotates in anticlockwise direction, while in figure 4d all the switches are closed which indicates a invalid state or a short circuit.

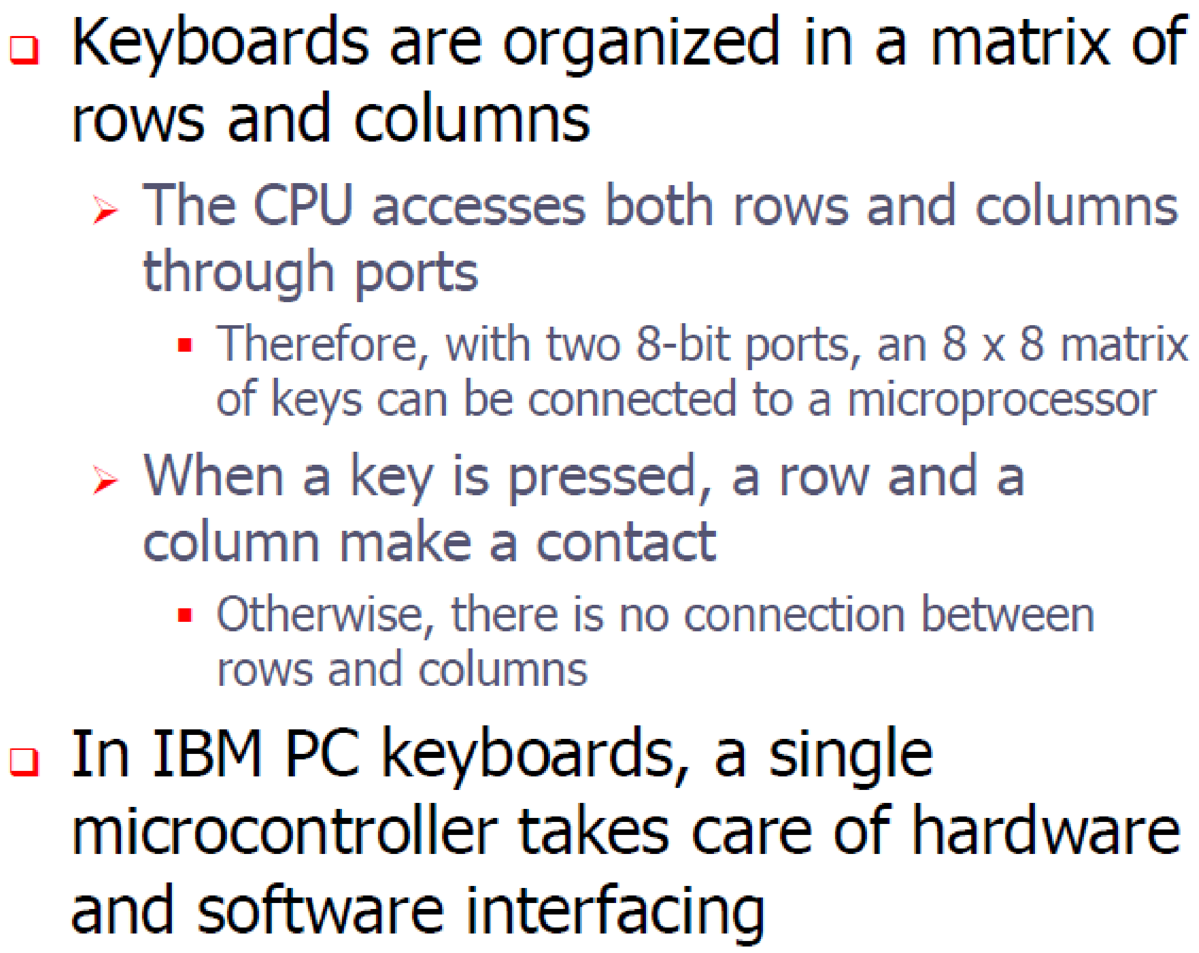


The optoisolator provides additional protection of the 8051.

use a separate power supply for the motor & L293 than for the 8051

D1, D2, D3, D4 are 1N4004

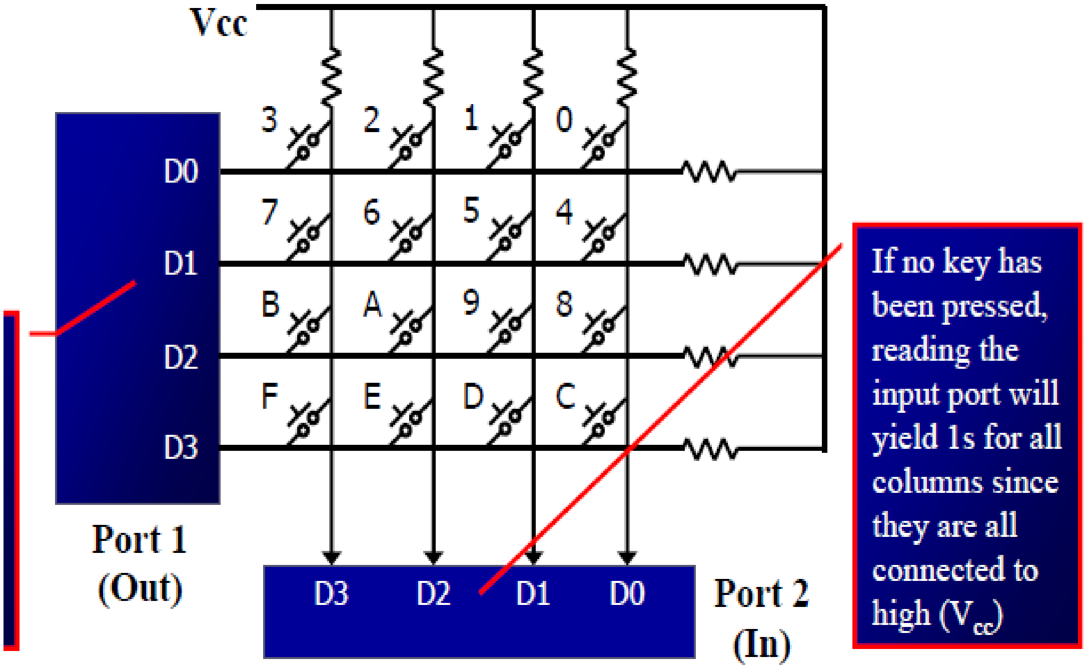
10.
b.



- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
  - When a key is pressed, a row and a column make a contact
    - Otherwise, there is no connection between rows and columns
- In IBM PC keyboards, a single microcontroller takes care of hardware and software interfacing

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port and the columns are connected to an input port

Matrix Keyboard Connection to ports



If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (V_{cc})

- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed
- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns
  - If the data read from columns is D3 – D0 = 1111, no key has been pressed and the process continues till key press is detected
  - If one of the column bits has a zero, this means that a key press has occurred
    - For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed
    - After detecting a key press, microcontroller will go through the process of identifying the key

- Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
  - It reads the columns, if the data read is all 1s, no key in that row is activated and the process is moved to the next row
- It grounds the next row, reads the columns, and checks for any zero
  - This process continues until the row is identified
- After identification of the row in which the key has been pressed
  - Find out which column the pressed key belongs to