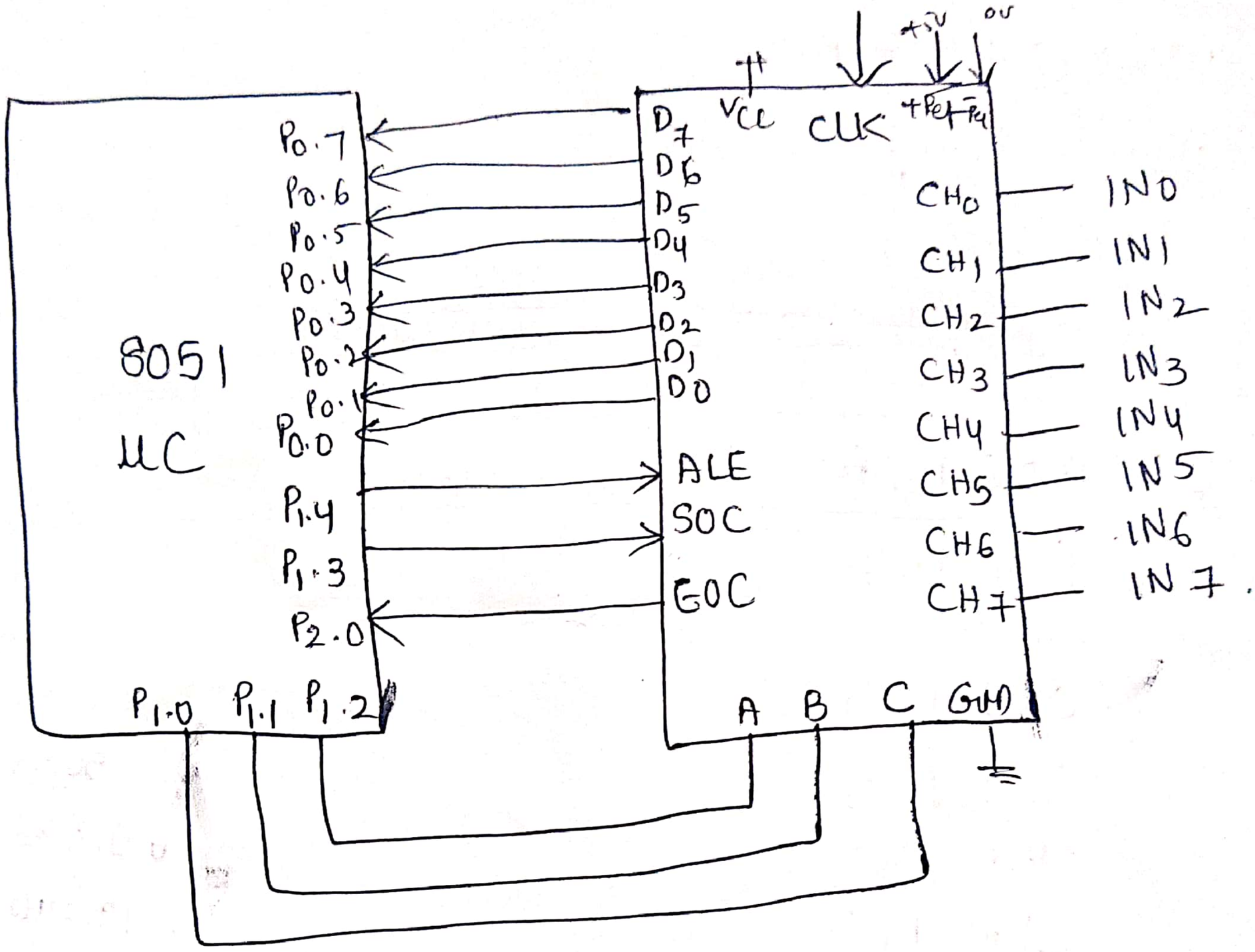
1) Interface 8-bit, 8-bit channel ADC to 8051. write an Assembly language program to convert CH0, CH_3 and CH_7 and store result in external memory location starting from C000H. Repeat procedure for every 1 Sec.

→ 8-Bit, 8-bit channel → ADC 0808



program :-

```
    CLR  P1.3
    CLR  P1.4
    mov  P0, #0FFH
    mov  P2, #0FFH
Back: mov  DPTR, #0C000H
    mov  A, #00H
    Acall  R_ADC
    movx @DPTR, A.
    mov  A, #03H.
```

```
Acall R-ADC
MOVX @DPTR, A
INC DPTR
MOV A, #07H
ACall R-ADC
MOVX @DPTR, A
A call Delay.
SJMP Back.
```

Analog to Digital conversion Routine :-

```
R_ADC : MOV P1, A
        SET P1.4
        NOP.
        CLR P1.4
        SET P1.3
        NOP
        CLR P1.3
wait :  JB P2.0, wait
        mov A, P0
        RET.
```

50ms
65,536 -
1C' 19,240
1 4B27

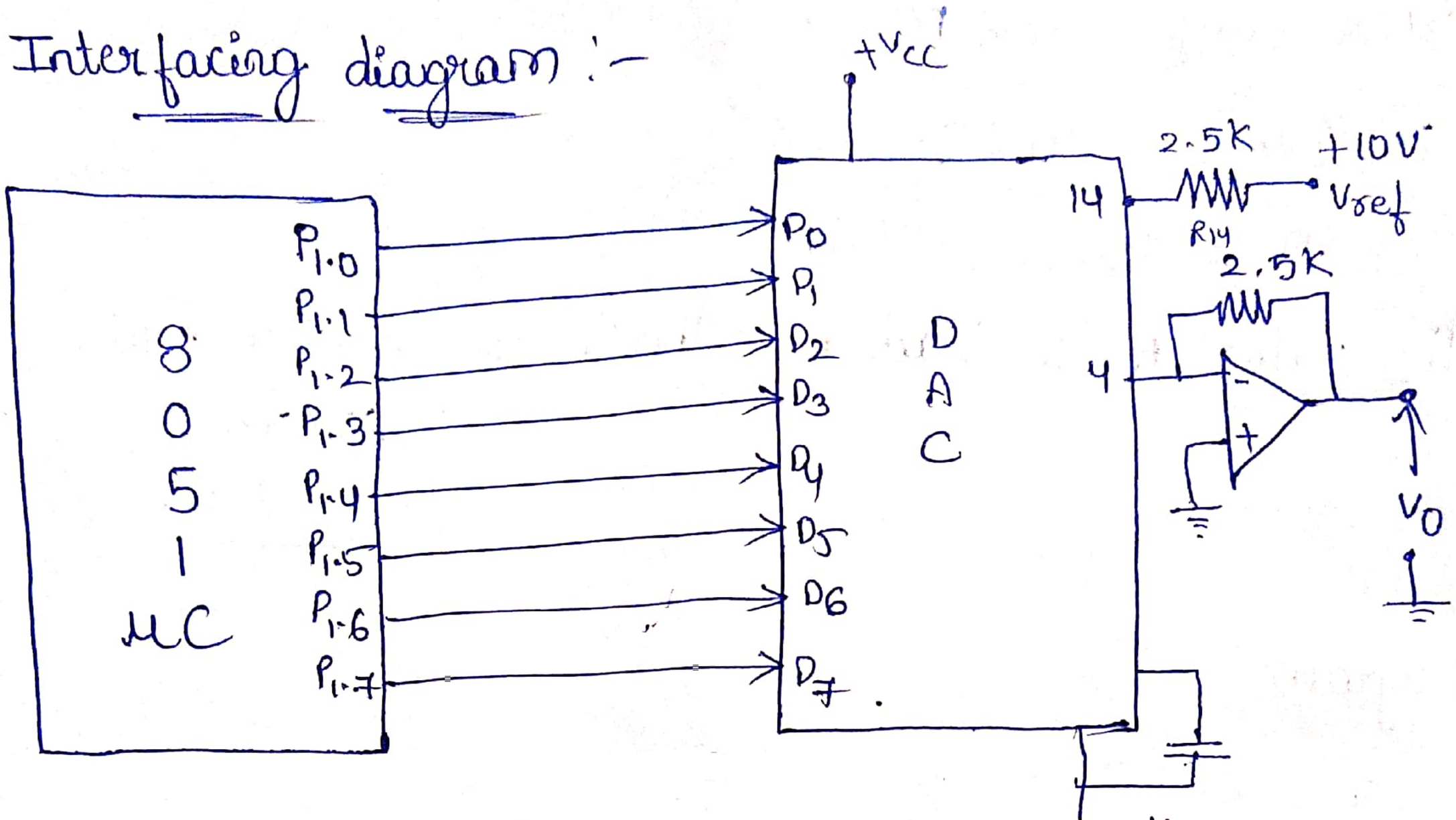
Delay Routine :-

```
Delay : MOV TMOD, #01
        mov R0, #14H.
        mov TL0, #B0H
        mov TH0, #3CH
        SETB TR0
Again : JNB TF0, Again
        CLR TR0
        CLR TF0
        DJNZ R0, Back
        RET.
```

Interfacing DAC to 8051 :-

1) Interface DAC 0808 to 8051. write an Assembly language program to generate a square wave with frequency of 1KHz and 10V.

Interfacing diagram :-



$$I_{out} = \frac{V_{ref}}{R_{14}}\left(\frac{D_0}{2} + \frac{D_1}{4} + \frac{D_2}{8} + \frac{D_3}{16} + \frac{D_4}{32} + \frac{D_5}{64} + \frac{D_6}{128} + \frac{D_7}{256}\right)$$

D = 11111111

$$I_{out} = \frac{10}{2.5k}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$

$$\frac{V_{out}}{2.5k} = \frac{10}{2.5k}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$

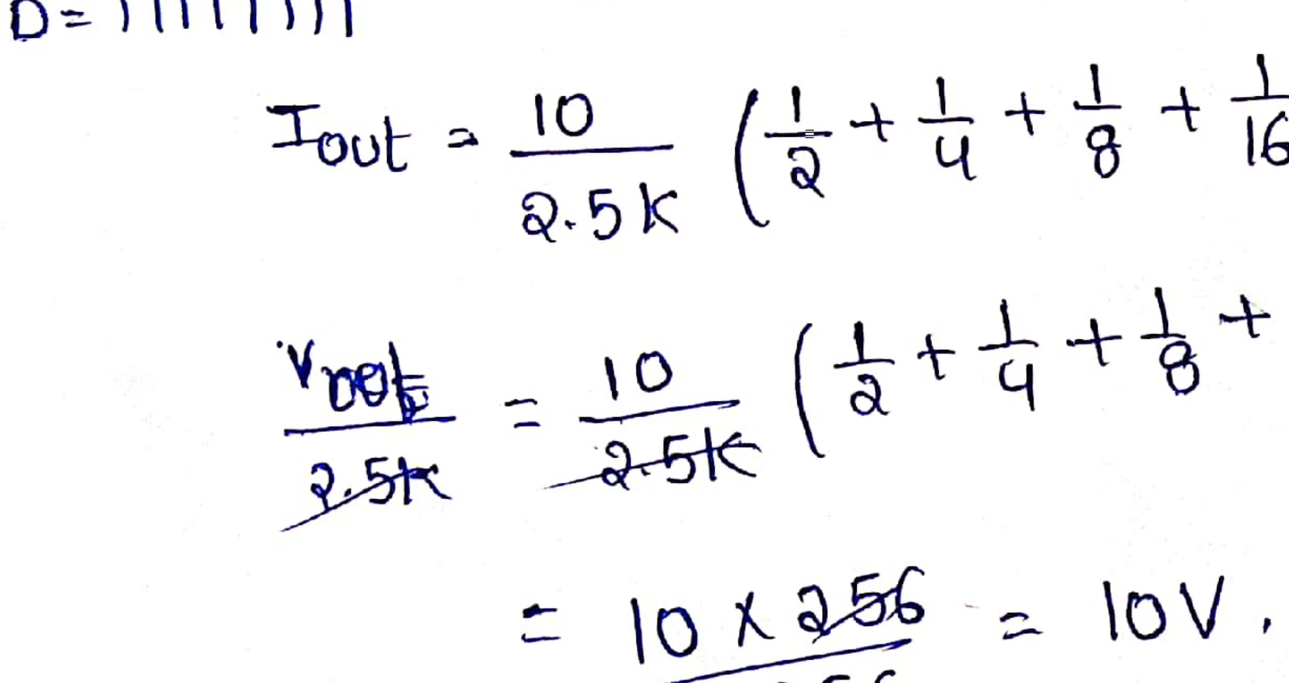
$$= \frac{10 \times 256}{256} = 10V.$$

$f = 1KHz$,

$T = \frac{1}{1K} = 1 \times 10^{-3} = 1msec$

Time periode of square wave is 1msec.

Duty cycle : 50%.



Crystal frequency = 11.0598 MHZ.

Frequency of machine cycle = 11.0592 MHZ/12 = 0.9216 MHZ.

period of machine cycle = 1/0.9216 MHZ = 1.085 μsec

Time delay = n x period of machine cycle.

$0.5 \times 10^{-3} = n \times 1.08 \times 10^{-6}$

$n = 4608 \Rightarrow 461$

the value to be loaded in the timer = 65536 - 461

= 65075

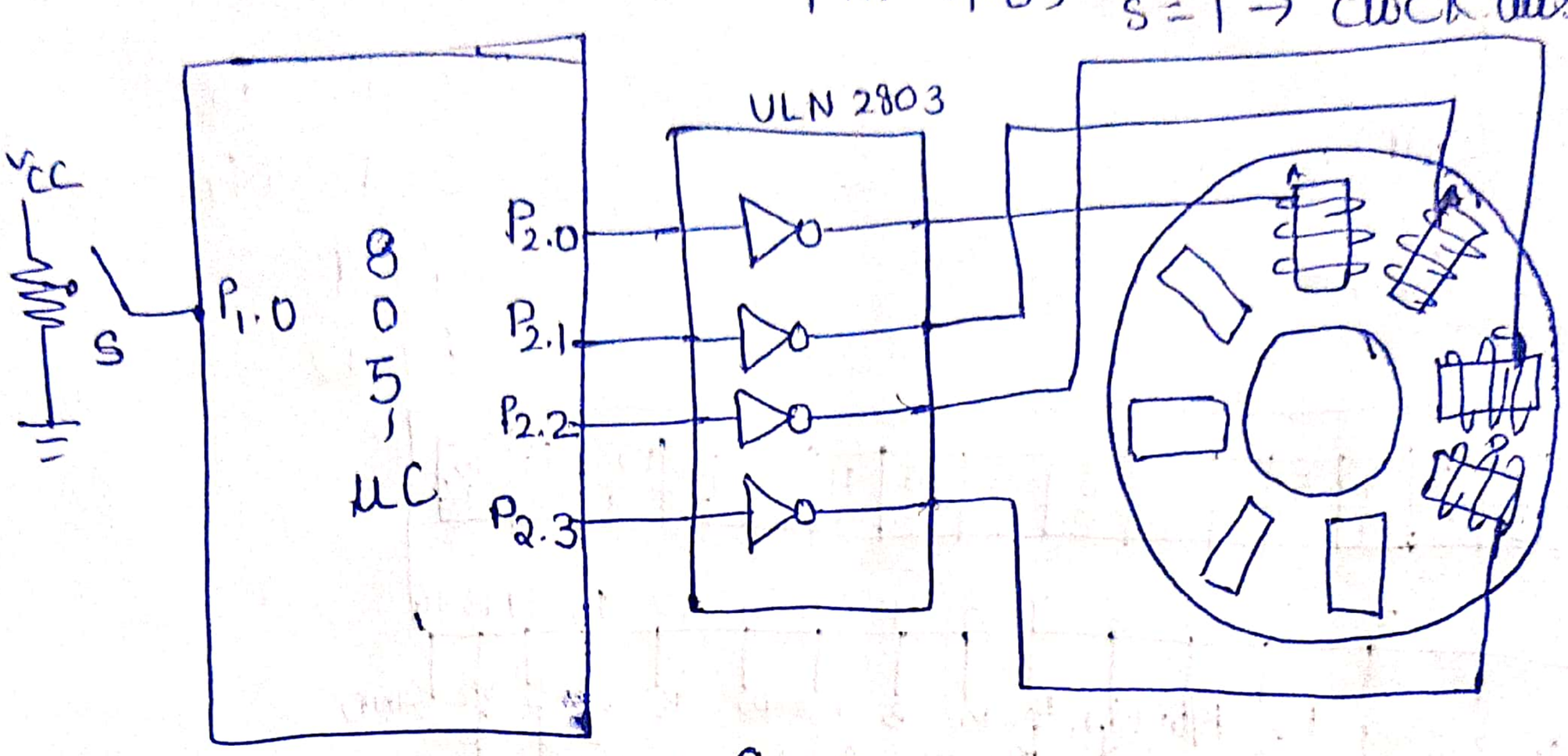
= FE33 H.

program :-

```
        ORG 0000 H.
Back:   mov A, #00 H → ov:
        mov P2, A
        Acall Delay.
        mov A, # FF H
        mov P2, A
        Acall Delay.
        SJMP BACK
Delay:  mov TMOD, #01 H.
~~Delay~~:  mov TL0, 33 H
        mov TH0, FEH
        SETB TR0
Again:  JNB TF0, Again
        CLR TR0
        CLR TF0
        RET
        END.
```

Stepper motor interface :-

1) Interface stepper motor to 8051. write an assembly Language program to control stepper motor in both clock wise and anti clock wise.
(switch is connected to pin $P_{1.0}$) S=0 Anti clock wise
S=1 → clock wise



S = 0

program :-

```
        org   0000 H.
        MOV   P1, #0FFH
        MOV   A, #77 H
        MOV   P2, A.
Return. JNB   P1.0, Antic
        RR A
        Acall Delay.
        MOV P2, A.
        SJmp Return
Antic : RL A
        Acall Delay.
        MOV P2, A.
        SJmp Return.
```

clock wise

	A	B	C	D
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

Anti clock wise

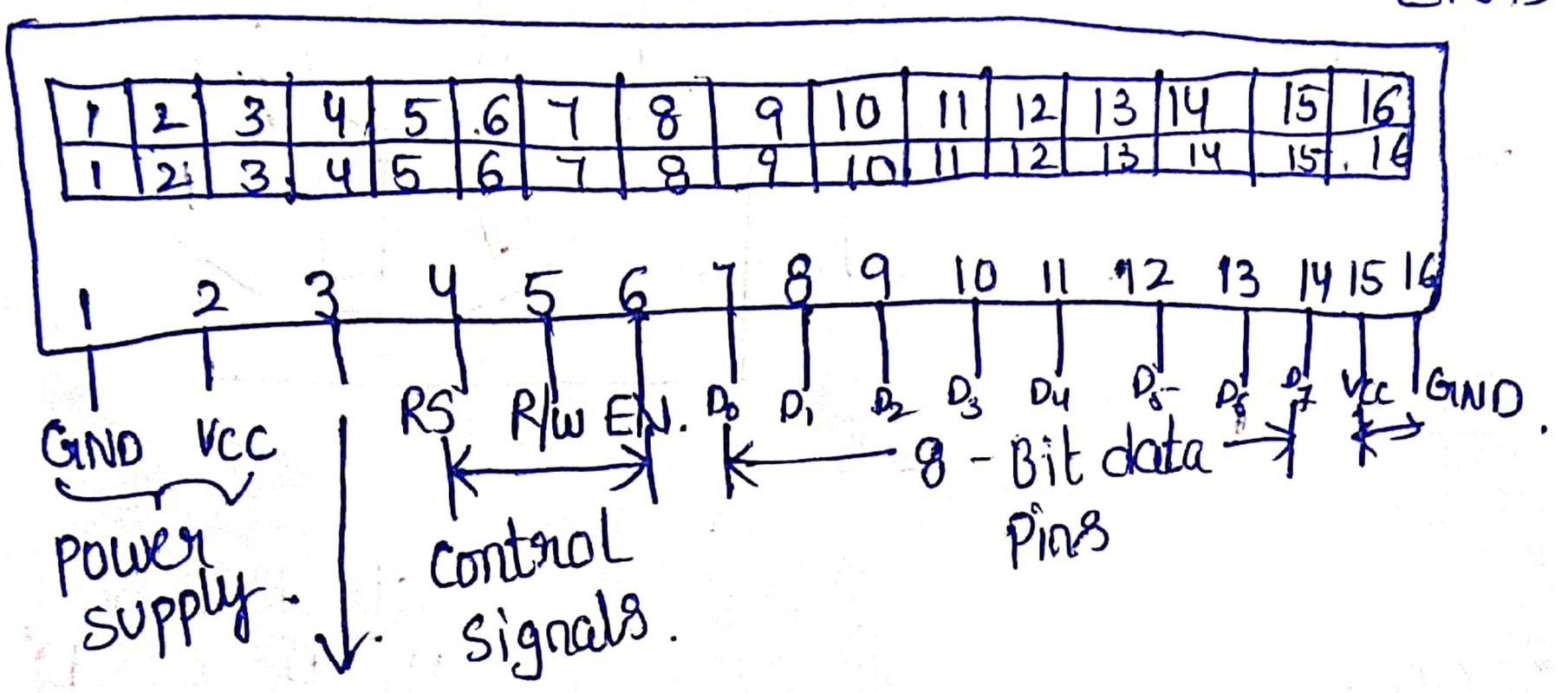
	A	B	C	D
1	0	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	0	1	1

```
Delay : MOV R2, #255
Back  : MOV R3, #255.
Stay  : DJNZ R3, Stay.
        DJNZ R2, Back
        RET
        END.
```

(8051)

```
Delay : MOV TMOD, #01H
        MOV TL0,
        MOV TH0,
        SETB TR0
wait  : JNB TF0 wait
        CLR TR0
        CLR TF0
        RET
        END.
```

16X2 LCD :-



→ Pin 1 & 2 → power supply.

Pin 3 → V_{EE} : contrast adjust.

Pin 4 → RS (Register select).

if RS = 0 → instruction select command register.

RS = 1 → select Data register.

Pin 5 → R/W = 0 → write operation, 1 → Read operation.

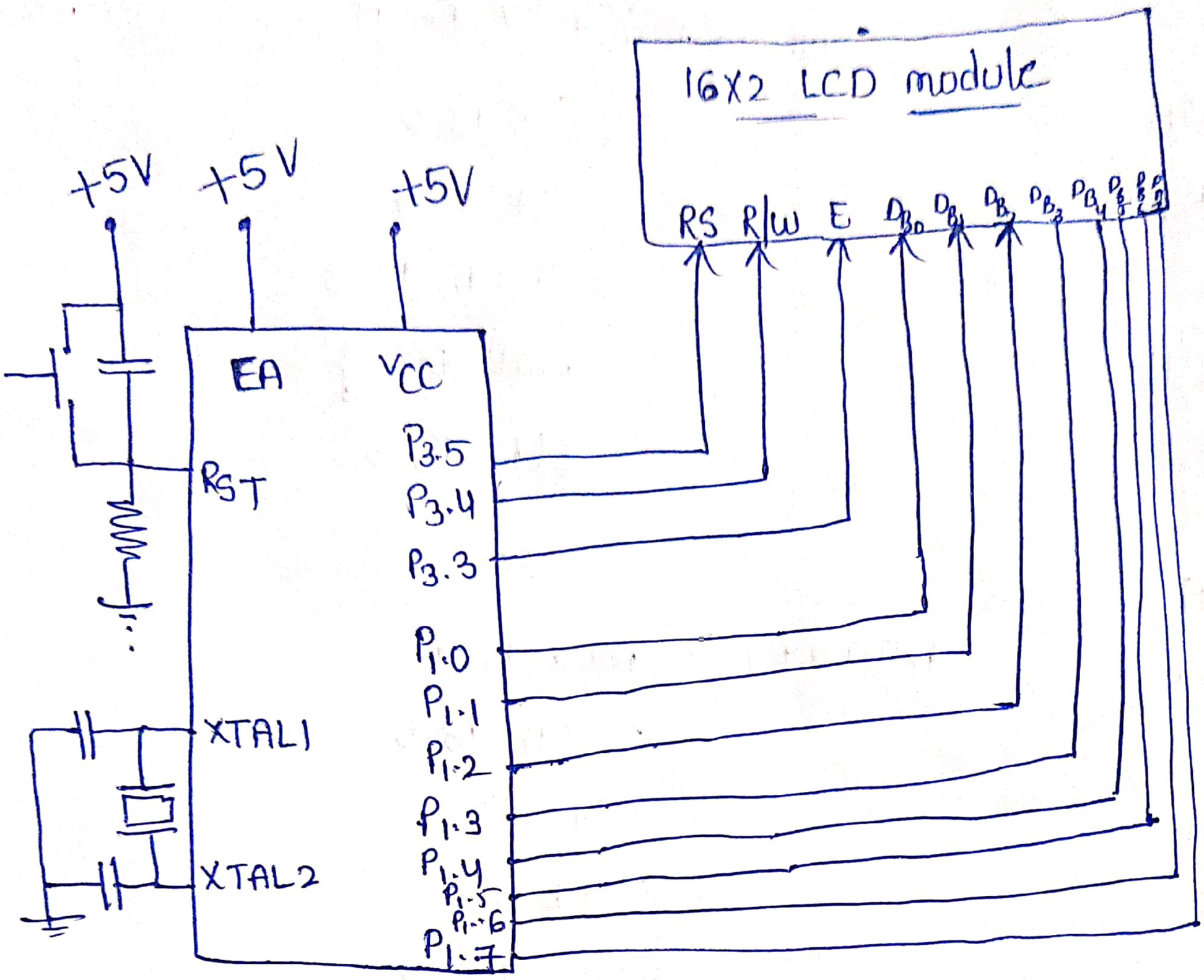
Pin 6 → EN = 1 → instruction execute, 0 → NO execution.

Pin 7 to pin 14 → Data bus lines.

Pin 15 → VCC } For Backlight

Pin 16 → GND } Enable the signal for 2nd Row.

Interfacing LCD to 8051 to Display "TEC" :-



→ LCD Command codes :-

01 → Clear display screen
02 → Return home
04 → Decrement cursor (shift cursor to left)
06 → Increment cursor (shift cursor to right)
05 → Shift display right
07 → Shift display left
08 → Display off, cursor off.
0A → Display off, cursor on.
0C → Display on, cursor OFF.
0E → Display on, cursor blinking.
0F → LCD ON, cursor on.
10 → Shift cursor position to left.
14 → Shift cursor position to right
18 → Shift the entire display to the left
1C → Shift the entire display to the Right

80 → Force cursor to Beginning of 1st line.
C0 → Force cursor to Beginning of 2nd line

38 → 2 lines and 5x7 matrix.

program :-

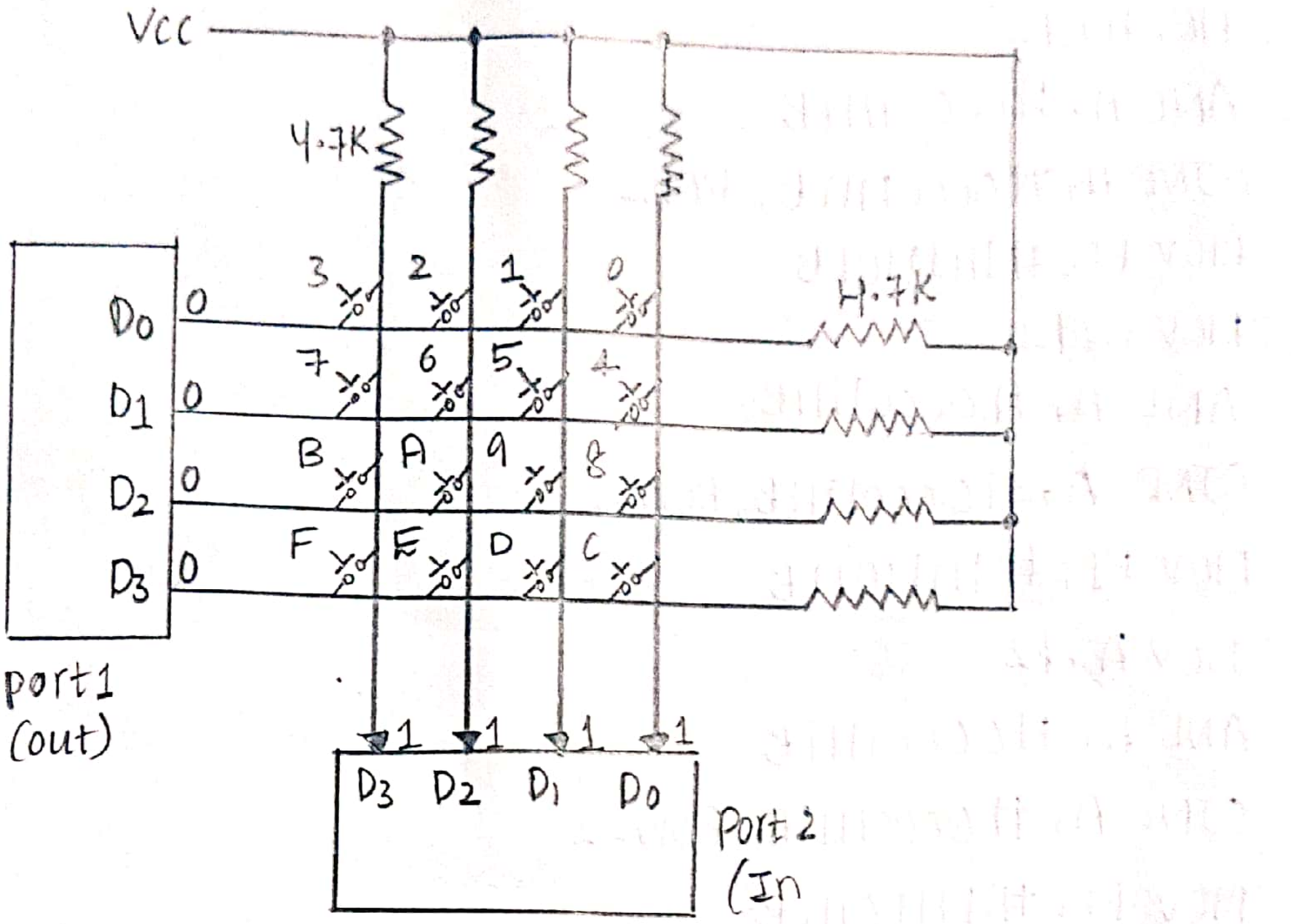
```
      ORG 0000H
Again: MOV A, #38H
       ACall COMNWRT
       ACall Delay
       MOV A, #0EH
       ACall COMNWRT
       ACall Delay
       MOV A, #01H
       ACall COMNWRT
       ACall Delay
       MOV A, #06H
       ACall COMNWRT
       ACall Delay
       MOV A, #80H
       ACall COMNWRT
       ACall Delay
       MOV A, #'T'
       ACall DataWRT
       ACall Delay
       MOV A, #'E'
       ACall DataWRT
       ACall Delay
       MOV A, #'C'
       ACall Delay
       SJmp Again.

COMNWRT : MOV P1, A
          CLR P3.5
          CLR P3.4
          SETB P3.3
          ACall Delay
          CLR P3.3
          RET.

DataWRT :- MOV P1, A
           SETB P3.5
           CLR P3.4
           SETB P3.3
           ACall Delay.
           CLR P3.3
           RET.

Delay : MOV R3, #50
here2 : MOV R4, #255
here  : DJNZ R4, here
        DJNZ R3, here2
        RET
        END.
```

Interfacing concepts of a 4x4 matrix keyboard with 8051 with Diagram:-



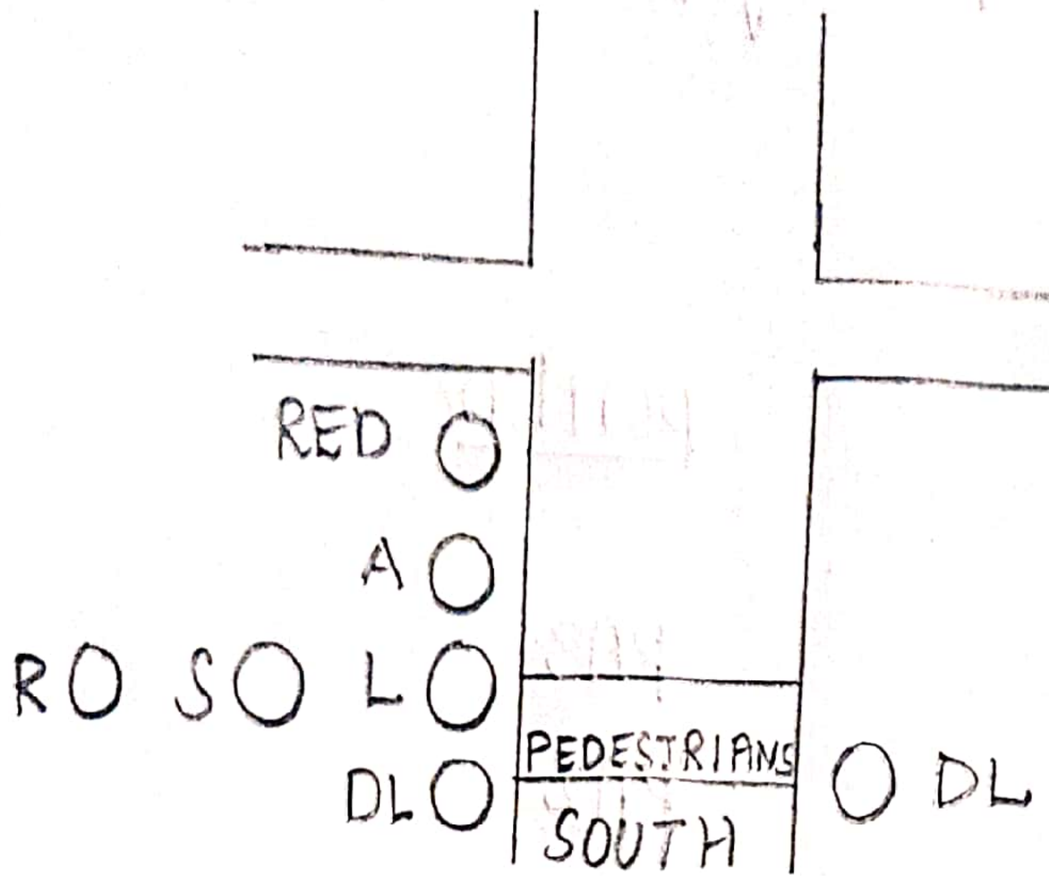
- Program to Access 4x4 Matrix Keyboard:-

$P_{1.0} - P_{1.3}$ connected to rows, $P_{2.0} - P_{2.3}$ connected to columns

```
        MOV P2, #0FFH          ; make P2 an input port
K1 :    MOV P1, #00H           ; ground all rows at once
        MOV A, P2              ; read all col. Ensure all keys open
        ANL A, #00001111B      ; masked unused bits.
        CJNE A, #00001111B, K1 ; check til all keys released
K2 :    ACALL DELAY            ; call 20 ms delay
        MOV A, P2              ; See if any key is pressed
        ANL A, #00001111B      ; mask unused bits
        CJNE A, #00001111B, OVER ; keypressed, await closure
        SJMP K2                ; check if key pressed.
OVER :  ACALL DELAY            ; wait 20 ms debounce
        MOV A, P2              ; check key closure
```

```
        ANL A, #00001111B
        CJNE A, #00001111B, OVER1
        SJMP K2
OVER1:  MOV P1, #11111110B
        MOV A, P2
        ANL A, #00001111B
        CJNE A, #00001111B, ROW_0
        MOV P1, #11111101B
        MOV A, P2
        ANL A, #00001111B
        CJNE A, #00001111B, ROW_1
        MOV P1, #11111011B
        MOV A, P2
        ANL A, #00001111B
        CJNE A, #00001111B, ROW_2
        MOV P1, #11110111B
        MOV A, P2
        ANL A, #00001111B
        CJNE A, #00001111B, ROW_3
        LJMP K2
ROW_0:  MOV DPTR, #KCODE0
        SJMP FIND
ROW_1:  MOV DPTR, #KCODE1
        SJMP FIND
ROW_2:  MOV DPTR, #KCODE2
        SJMP FIND
ROW_3:  MOV DPTR, #KCODE3
        SJMP FIND
FIND:   RRC A
        JNC MATCH
        INC DPTR
        SJMP FIND
Match:  CLR A
        MOVC A, @A+DPTR
        MOV P0, A
        LJMP K1
ASCII look up table for each ROW.
        ORG 3000H
KCode0: DB '0', '1', '2', '3' ; ROW 0
KCode0: DB '4', '5', '6', '7' ; ROW 1
KCode0: DB '8', '9', 'A', 'B' ; ROW 2
KCode0: DB 'C', 'D', 'E', 'F' ; ROW 3.
        END.
```

Traffic Lights Interface:-



Determine port values for the following traffic situation:

Vehicles from WEST are allowed to go NORTH or EAST.

Vehicles from EAST are allowed to go WEST.

Pedestrain crossing is allowed on SOUTH.

No other vehicle movement/pedestrain crossings are allowed.

Now, as per the above interpretation, the status of the LEDs should be as shown below:

	RED	AMBER	LEFT	STRAIGHT	RIGHT	PEDESTRAIN
SOUTH	ON	OFF	OFF	OFF	OFF	GREEN
EAST	OFF	OFF	ON	ON	OFF	RED
NORTH	ON	OFF	OFF	OFF	OFF	RED
WEST	OFF	OFF	OFF	ON	OFF	RED

PA=18H PB=08H PC=B5H

The 24 LEDs and their corresponding portlines are summarized below:

	LED	portline
SOUTH	RED	PA3
	AMBER	PA2
	LEFT	PA0
	STRAIGHT	PC3
	RIGHT	PA1
	PEDESTRIAN	PC6
EAST	RED	PA7
	AMBER	PA6
	LEFT	PA4
	STRAIGHT	PC2
	RIGHT	PA5
	PEDESTRIAN	PC7
NORTH	RED	PB3
	AMBER	PB2
	LEFT	PB0
	STRAIGHT	PC1
	RIGHT	PB1
	PEDESTRIAN	PC4
WEST	RED	PB7
	AMBER	PB6
	LEFT	PB4
	STAIGHT	PC0
	RIGHT	PB5
	PEDESTRIAN	PC5

Interfacing Traffic Light Controller to microcontroller 8051:

Program:

```
#define DELAY 0x0E2EE

        ORG 00H
        SJMP 30H
        ORG 30H
        MOV DPTR,#100H
        MOV P0,#00H
        MOV P1,#00H
        MOV P2,#00H

START : CLR A
        MOVC A,@A+DPTR
        MOV P0,#00H
        MOV P0,A
        INC DPTR
        LCALL DELAY
        LCALL DELAY
        LCALL DELAY
        LCALL DELAY
        CLR A
        MOVC A,@A+ DPTR
        MOV P1,#00H
        MOV P1,A
        INC DPTR
        LCALL DELAY
        L CALL DELAY
        LCALL DELAY
        LCALL DELAY
        CLR A
```

```
MOVC A,@A+DPTR
MOV  P2,#00H
MOV  P2,A
INC  DPTR
LCALL DELAY
LCALL DELAY
LCALL DELAY
LCALL DELAY
SJMP START
ORG  100H
DB 10H,81H,7AH
DB 44H,44H,0F0H
DB 08H,11H,0E5H
DB 44H,44H,0F0H
DB 81H,10H,0DAH
DB 44H,44H,0F0H
DB 11H,08H,0B5H
DB 44H,44H,0F0H
DB 88H,88H,00H
DB 44H,44H,0F0H
DB 00H
END
```