Embedded Lab

## Name: Guna sai Kiran Roll: BT20ECE075

Sub: Embedded Sys.

# Experiment:1.1

## Aim: Add 10 numbers

#### Code:

ORG 0000H

SJMP START

ORG 0035H

START: MOV R1, #90D // moving r1 the starting address of the data

CLR A // clearing accumulator

CLR C // clearing carry flag

MOV R0, #0AH // counter R0

REPEAT: ADDC A, @R1 // adding numbers

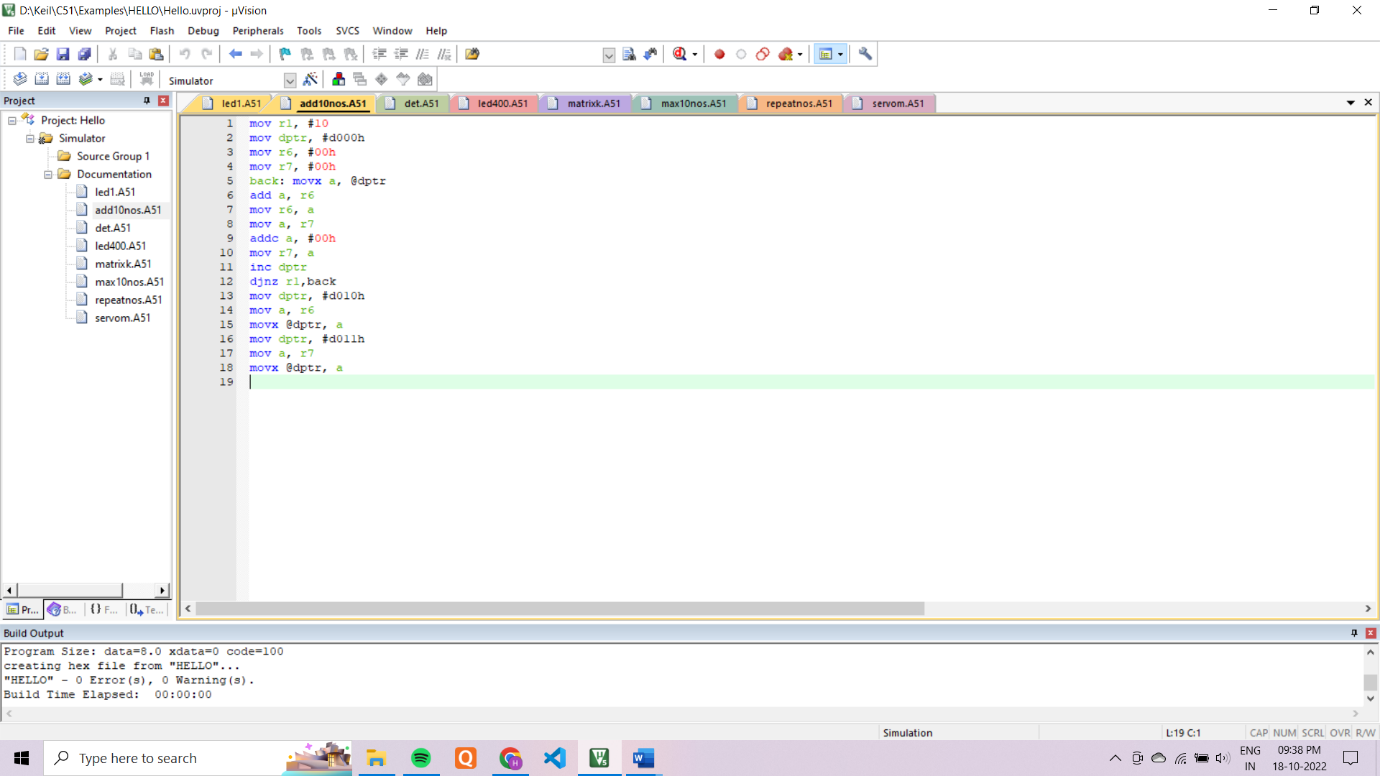
INC R1 // incrementing address to fetch next number

DJNZ R0, REPEAT // decrement R0

// database to store data

ORG 90D LUT: DB 01D DB 02D DB 03D DB 04D DB 05D DB 06D DB 07D DB 08D DB 09D DB 10D

END



# Experiment:1.2

## Aim: Max of 10 numbers

#### Code:

ORG 0000H

SJMP START

// database to store data

ORG 0064H

LUT: DB 01D DB 02DD DB 03D DB 04D DB 05D DB 06D DB 07D DB 08D DB 09D DB 10D

ORG 0035H

START:

MOV A, #00H // clear accumulator

MOV R0, #00H // clearing R0 to store maximum number

MOV DPTR, #64H // putting the starting address of the data in DPTR

MOV R1, #0AH // storing 10 in R1 to find maximum among 10 numbers

LOOP: MOV B, @DPTR // moving data stored at address pointed by DPTR in B

CLR C // clearing Carry flag

INC DPTR // incrementing DPTR to fetch next data

SUBB A, B // comparing maximum number till now with the newly fetched number

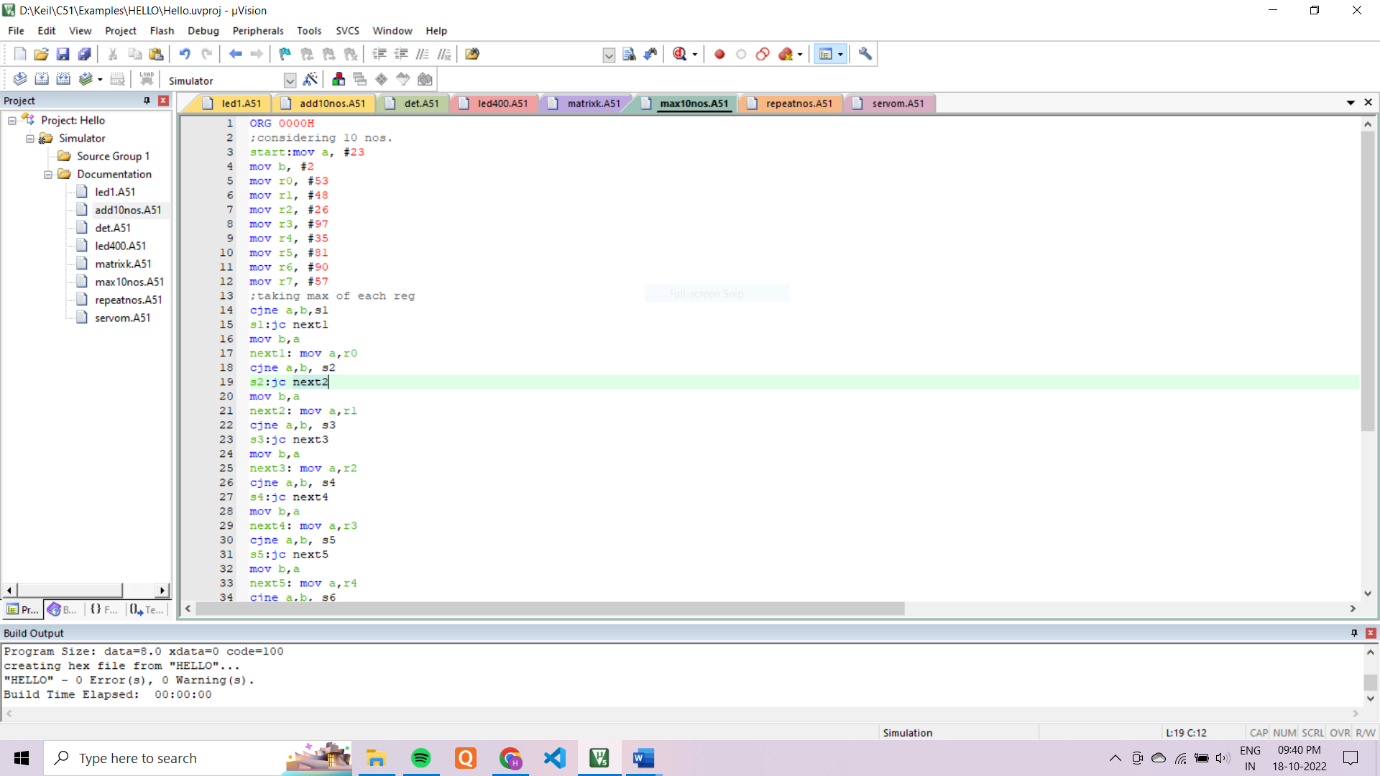
JNC SKIP // If there is carry then data in B is maximum till now

MOV RO, B // store maximum number in R0

SKIP: MOV A, R0 // move maximum number in A to compare with next number

DJNZ R1, LOOP // Repeat for 10 times

END //

****

# Experiment:1.3

## Aim: Frequency of a number

#### Code:

org 0000h

//Taking 10 nos. in 10 registers

mov a, #1

mov b, #2

mov r0, #3

mov r1, #4

mov r2, #5

mov r3, #6

mov r4, #7

mov r5, #8

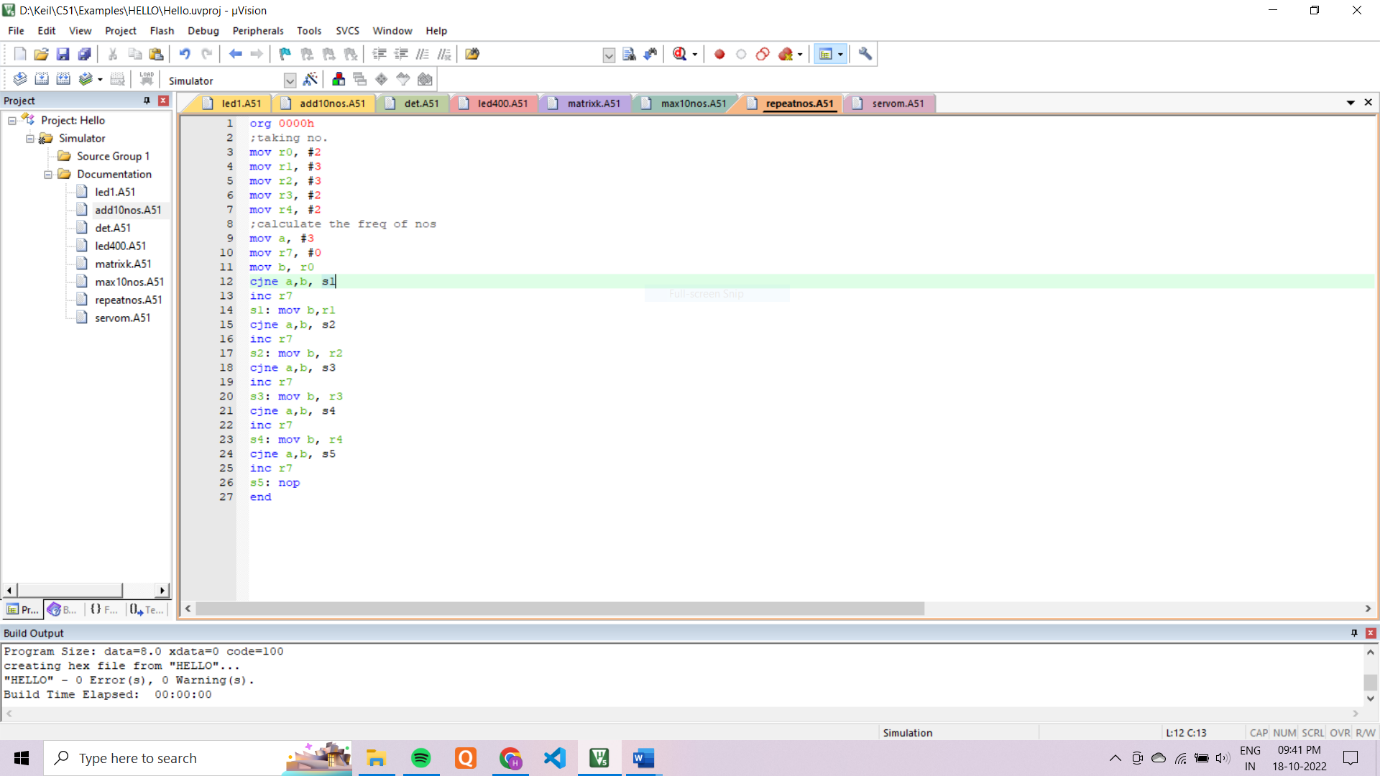
mov r6, #9

mov r7, #10

//Adding 10 nos. and saving output in a register add a,b

add a,r0 add a,r1 add a,r2 add a,r3 add a,r4 add a,r5 add a,r6 add a,r7

end

****

# Experiment: 2

## Aim: Determinant of matrix Code:

**Code:**

ORG 0000H

MOV R6, #00H

MOV R1, #05H // A number at [1 1]

MOV R2, #0AH // D number at [2 2]

MOV R3, #06H // B number at [1 2]

MOV R4, #06H // C number at [2 1]

MOV B, R1

MOV A, R2 LABEL1: MOV R2, A // multiplying A\*D

MOV A, R6 // multiplying using repetitive addition ADD A, B MOV R6, A

MOV A, R2

DEC A

JNZ LABEL1

MOV R5, #00H

MOV B, R3

MOV A, R4

LABEL2: MOV R4, A // multiplying B\*C

MOV A, R5 // multiplying using repetitive addition

ADD A, B

MOV R5, A

MOV A, R4

DEC A

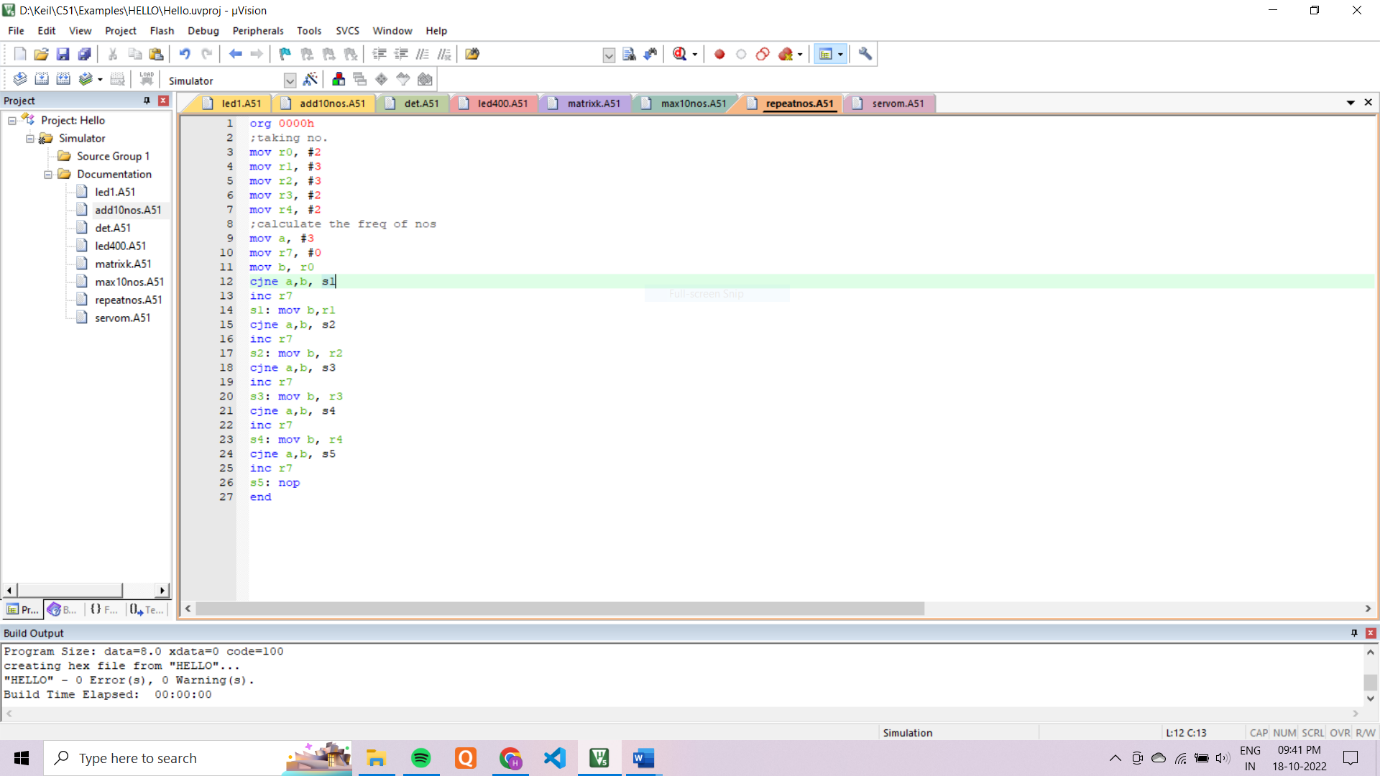
JNZ LABEL2

MOV A, R6

MOV B, R5

SUBB A, B // finding A\*D - B\*C

END

****

# Experiment:3.1

## Aim: Blink LED using timers

## Code:

## ORG 0000H

## SJMP START

## ORG 0035H

## START: MOV TMOD, #01H // using timer 0 in mode 1

## REPEAT: MOV TH0, #0FEH

## MOV TL0, #70H

## CPL P1.0 // complementing a led to blink

## SETB TR0 // starting counter

## CHECK: JNB TF0, CHECK // waiting until TF0 flag is set

## CLR TR0 // clearing TR0 flag to stop timer

## CLR TF0 // clearing TF0 to again use timer

## JMP REPEAT // repeat above process continuously

## END

# Experiment 3.2

## Aim: Blink a Led using Interrupts(1s)

#### Code:

#### ORG 0000H

#### SJMP START

#### ORG 000BH // Interrupt program to generate a delay of 1s

#### MOV TL0, #0B0H // reinitialize TL0 & THO

#### MOV TH0, #3CH

#### DJNZ R0, GO\_BACK

#### CPL P1.0 // BLINK LED

#### MOV R0, #14H

#### GO\_BACK:

#### RETI // return from interrupt

#### ORG 0035H

#### START: CLR P1.0

#### MOV TMOD, #01H // using timer 0 in mode 1

#### MOV R0, #14H // 20 timess 50 ms is equal to 1s

#### MOV TL0, #0B0H // value in TH0 to generate a delay of 50 ms

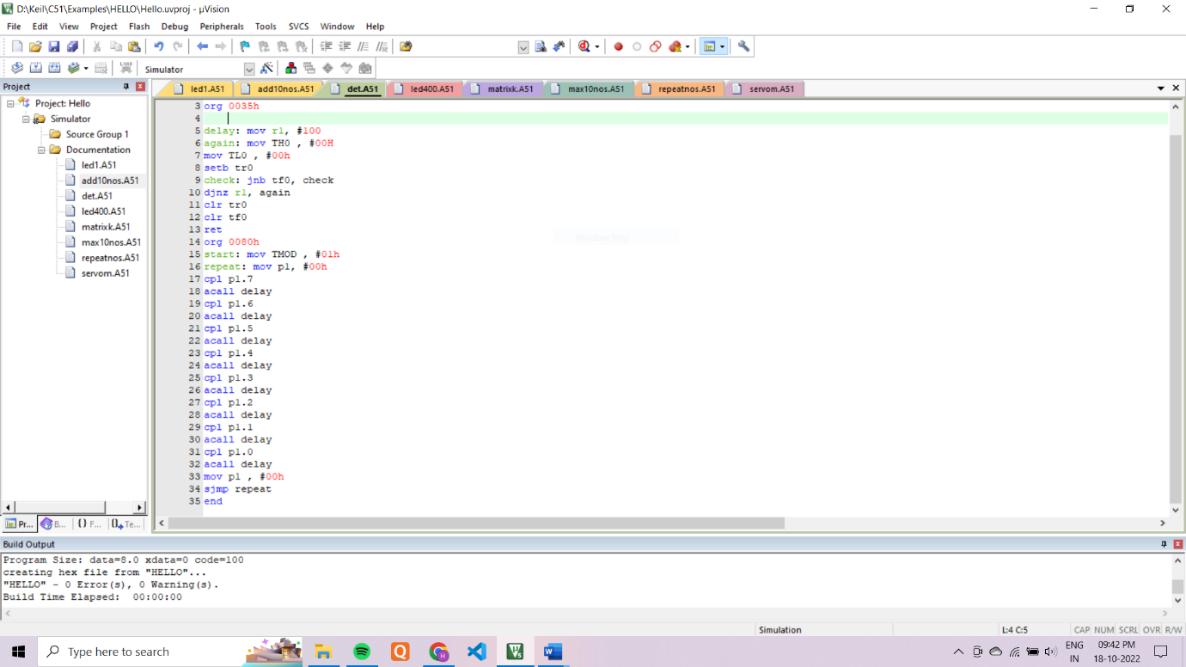
#### MOV TH0, #3CH

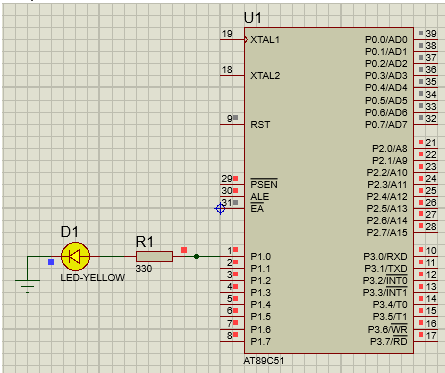
#### MOV IE, #8AH // enable interrupt

#### SETB TR0

#### WAIT: JMP WAIT

#### END

****



# Experiment: 4

## Aim: Gapati Pattern

#### Code:

ORG 0000H

START

DELAY:

MOV TMOD, #01H // Timer 0 in mode 1

MOV TH0,#00H // value in TH0TL0 = 0000h MOV TL0,#00H // produce a delay 0f 65ms

SETB TR0 // start count

CHECK: JNB TF0, CHECK // check if TF0 is set

CLR TR0 // clear TR0 flag

CLR TF0 // clear TF0 flag to produce desired delay

RET // Return from Delay subroutine.

ORG 0035H

START: MOV R0,#08H // moving 8 in R0 to 8 bits in registers.

MOV A,#00H // clearing A

LABEL: SETB C // setting carry flag

RLC A // rotate left with carry to set bitsMOV P1,A // send accumulator data to port 1 to blink leds

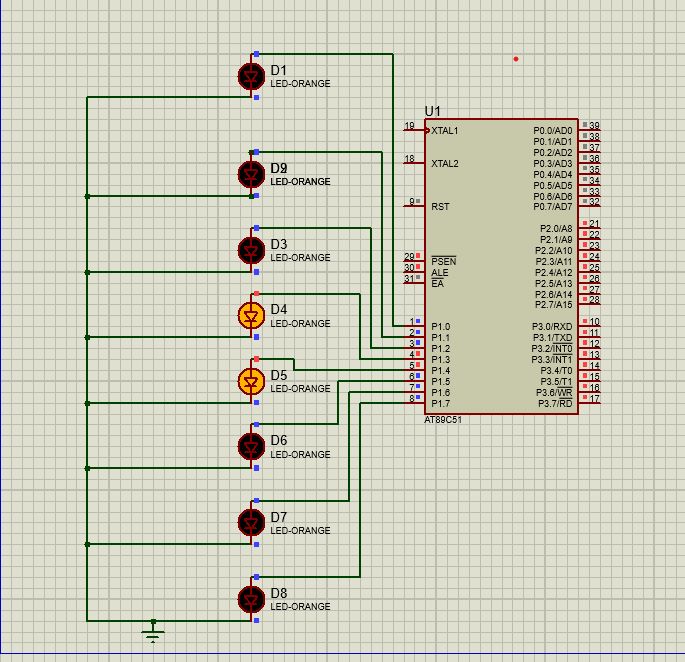
ACALL DELAY // generate a delay of 65ms

ACALL DELAY // generate a delay of 65ms

DJNZ R0, LABEL // repeat above process 8 times.

JMP START // continuously repeat this process.

END



# Experiment: 5

## Aim: Servo Motor

#### Code:

ORG 0000H //Start the program

SJMP MAIN

ORG 0055H

zero\_degrees:

MOV TH0, #0FCH

MOV TL0, #19H

SETB P2.0 ;

SETB TR0 ;

WAIT1:JNB TF0, WAIT1 ;

CLR TF0 ;

CLR TR0 ;

MOV TH0, #0B5H

MOV TL0, #C8H

CLR P2.0 ;

SETB TR0 ;

WAIT1:JNB TF0, WAIT1 ;

CLR TF0 ;

CLR TR0 ;

SJMP zero\_degrees

RET

ORG 0065H

ninety\_degrees:

MOV TH0, #0FAH

MOV TL0, #25H

SETB P2.0 ;

SETB TR0 ;

WAIT2:JNB TF0, WAIT2 ;

CLR TF0 ;

CLR TR0 ;

MOV TH0, #0B7H

MOV TL0, #0BCH

CLR P2.0 ;

SETB TR0 ;

WAIT1:JNB TF0, WAIT1 ;

CLR TF0 ;

CLR TR0 ;

SJMP ninety\_degrees

RET

ORG 0075H

one\_eighty\_degrees:

MOV TH0, #0F8H

MOV TL0, #31H

SETB P2.0 ;

SETB TR0 ;

WAIT3:JNB TF0, WAIT3 ;

CLR TF0 ;

CLR TR0 ;

MOV TH0, #0B9H

MOV TL0, #B0H

CLR P2.0 ;

SETB TR0 ;

WAIT1:JNB TF0, WAIT1 ;

CLR TF0 ;

CLR TR0 ;

SJMP one\_eighty\_degrees

RET

ORG 0085H

delay: //To create a delay of 1sec

MOV R4,#64H ;

LOOP1:MOV R3,#64H

LOOP2:MOV R2,#64H

LOOP3:DJNZ R2,LOOP3

DJNZ R3,LOOP2

DJNZ R4,LOOP1

RET

ORG 0035H

MAIN:CLR P2.0;

MOV TMOD, #01H ;using Timer 0 in Mode 1

LCALL zero\_degrees ;

LCALL delay ;

LCALL ninety\_degrees ;

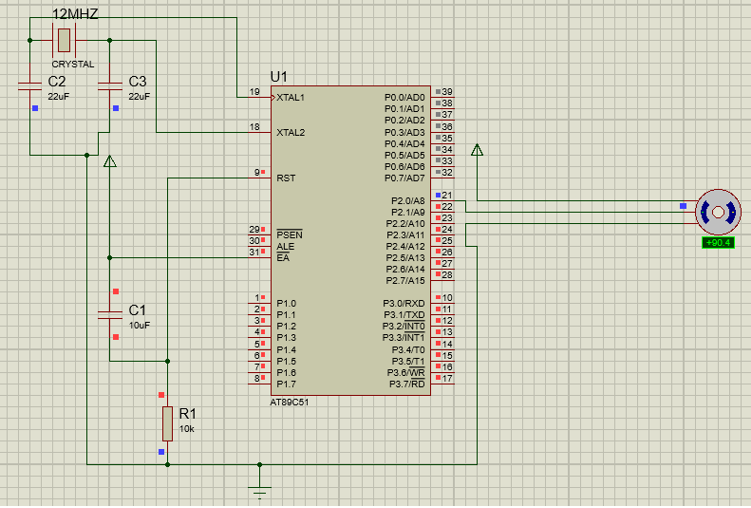
LCALL delay ;

LCALL one\_eighty\_degrees ;

LCALL delay ;

SJMP MAIN ;

END



# Experiment: 6

## Aim: Keypad

#### Code:

ORG 0000H

SJMP START

ORG 0035H

START:

// FOR ROWS

SETB P1.0 //SETTING AS INPUT

SETB P1.1 //SETTING AS INPUT

SETB P1.2 //SETTING AS INPUT

SETB P1.3 //SETTING AS INPUT

//FOR COLOUMNS

SETB P1.4 //SETTING AS OUTPUT

SETB P1.5 //SETTING AS OUTPUT

SETB P1.6 //SETTING AS OUTPUT

SETB P1.7 //SETTING AS OUTPUT

//FOR LEDS

CLR P2.0 //SETTING AS OUTPUT

CLR P2.1 //SETTING AS OUTPUT

CLR P2.2 //SETTING AS OUTPUT

CLR P2.3 //SETTING AS OUTPUT

//FOR R1 IS HIGH

MOV P1,#80H

MOV A,P1

AGAIN:CJNE A,#88H,NEXT

MOV P2,#00H

NEXT:CJNE A,#84H,NEXT1

MOV P2,#01H

NEXT1:CJNE A,#82H,NEXT2

MOV P2,#02H

NEXT2:CJNE A,#80,AGAIN

MOV P2,#03H

//FOR R2 IS HIGH

MOV P1,#40H

MOV A,P1

AGAIN1:CJNE A,#48H,NEXT3

MOV P2,#04H

NEXT3:CJNE A,#44H,NEXT4

MOV P2,#05H

NEXT4:CJNE A,#42H,NEXT5

MOV P2,#06H

NEXT5:CJNE A,#40,AGAIN1

MOV P2,#07H

//FOR R3 IS HIGH

MOV P1,#20H

MOV A,P1

AGAIN2:CJNE A,#28H,NEXT6

MOV P2,#08H

NEXT6:CJNE A,#24H,NEXT7

MOV P2,#09H

NEXT7:CJNE A,#22H,NEXT8

MOV P2,#0AH

NEXT8:CJNE A,#20,AGAIN2

MOV P2,#0BH

//FOR R4 IS HIGH

MOV P1,#00H

MOV A,P1

AGAIN3:CJNE A,#08H,NEXT9

MOV P2,#0CH

NEXT9:CJNE A,#04H,NEXT10

MOV P2,#0DH

NEXT10:CJNE A,#02H,NEXT11

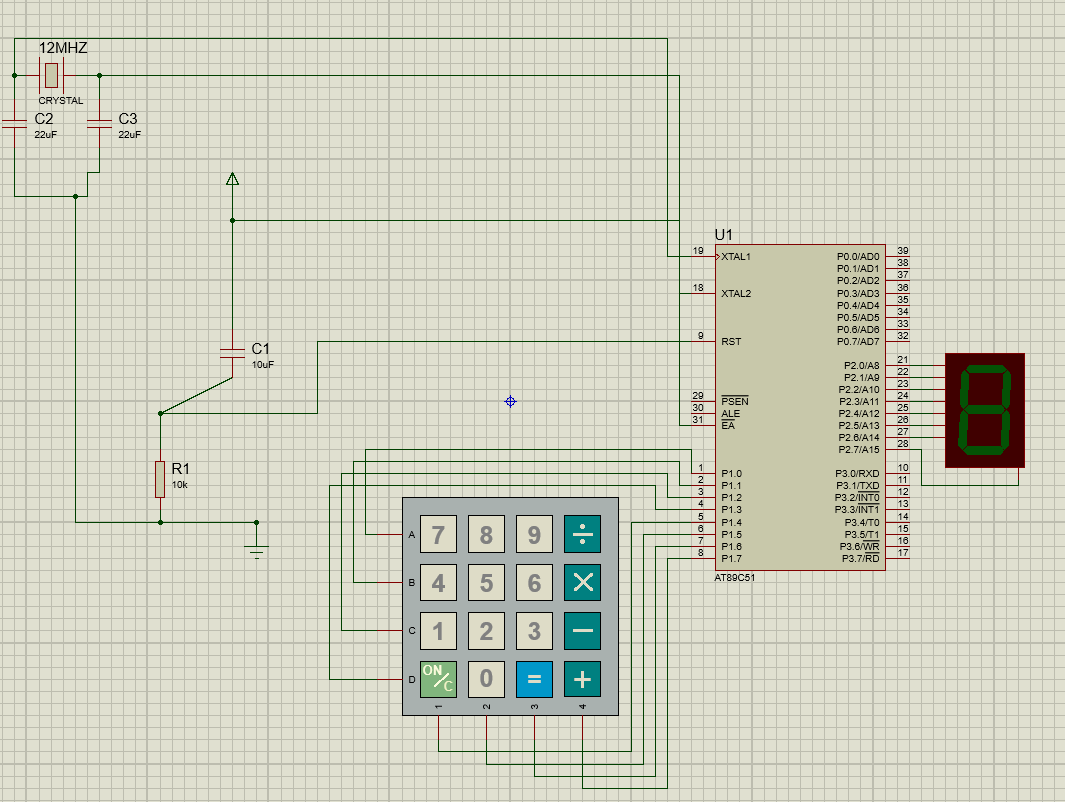
MOV P2,#0EH

NEXT11:CJNE A,#00,AGAIN3

MOV P2,#0FH

END

Circuit diagram:



# Experiment: 7

## Aim: Seven segment display Code:

ORG 4000H

DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH, 0

; Lookup table for digits 0 to 9

ORG 0000H

main: MOV DPTR, #4000H

repeat: CLR A

MOVC A, @A+DPTR ; Copy data from external location to accumulator

MOV P2, A ; Move the pattern of the digit into port P2

ACALL delay ; Call a delay to so that the transition is visible

INC DPTR ; Point to the next pattern

CJNE A, 0, repeat ; Repeat till 0 (Stop bit) is

received

SJMP main ; Run this forever till

externally stopped

; generate a decent enough delay between transitions delay:

MOV R0, #08H LP2: MOV R1, #0FFH LP1: MOV R2, #0FFH LP3: DJNZ R2, LP3

DJNZ R1, LP1

END

DJNZ R0, LP2 RET

### Circuit diagram:

