NITTE
EDUCATION TRUST
N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAA

Department of Computer Science and Engineering

Laboratory Manual

Microprocessors and Peripherals

20CS408

Microprocessors and Peripherals Lab

20CS408
IV Sem

Software tools used

1. Windows 10
2. DOSBox 0.74
3. Arduino IDE 1.8.19

Hardware to be used

1.ARM CORTEX-M3 Based Microcontroller Trainer Module
2.Stepper Motor
3.USB cable
2. Wires

Marks distribution:

CIE Marks Distribution

Assessment	Weight-age in Marks
Continuous evaluation	30
MSE	20
Total	**50**

MSE Marks Distribution:

Evaluation	Distribution of Marks
Write-up and Execution of PART- A Program	10
Write-up and Execution of PART- B Program	10
Total	**20**

SEE Marks Distribution:

Evaluation	Distribution of Marks
Write-up and Execution of PART A program	05+15
Write-up and Execution of Part B program	05+10
Modification	05
VIVA	10
Total	**50**

Prepared by:

Dr. Anisha P Rodrigues

Assistant Professor-III

Computer Science and Engineering

NMAMIT, Nitte

List of Experiments

PART A (8086 Assembly Language Programs)

PART B (Microcontroller and Peripherals based programming)

PART A:

1. Write an ALP to search a key element in a list of ‘n’ 8-bit numbers using the Binary search algorithm

```
DATA SEGMENT
        arr db 06H, 12H, 15H, 56H, 65H, 70H, 78H
        len db $-ARR
        mid db ?
        key db 79H
        msg1 db 10, 13, 'KEY NOT FOUND', 10, 13, '$'
        msg2 db 10, 13, 'KEY FOUND AT POSITION $'
DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA

START:
        MOV AX, DATA
        MOV DS, AX
        MOV DL,00H
        MOV DH, len
        MOV BX,0000
        MOV CL, key
UP:
        CMP DL, DH
        JG NOTFOUND
        MOV BL, DL
        ADD BL, DH
        SHR BL, 01H
        MOV mid, BL
        CMP CL, arr[BX]
        JZ FOUND
        JB FIRSTHALF
        INC mid
        MOV DL, mid
        JMP UP

FIRSTHALF:
        DEC mid
        MOV DH, mid
        JMP UP
NOTFOUND:
        LEA DX, msg1
        MOV AH, 09H
        INT 21H
        JMP EXIT
```

```
FOUND:
        LEA DX, msg2
        MOV AH, 09H
        INT 21H
        MOV BL, mid
        CALL DISPHEXA
EXIT:
        MOV AH, 4CH
        INT 21H

DISPHEXA PROC NEAR
        MOV DL, BL
        MOV CL, 04H
        SHR DL, CL
        CMP DL, 09H
        JBE L1
        ADD DL, 07H
L1:
        ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV DL, BL
        AND DL, 0FH
        CMP DL, 09H
        JBE L2
        ADD DL, 07H
L2:
        ADD DL, 30H

        MOV AH, 02H
        INT 21H
RET
DISPHEXA ENDP
CODE ENDS
END START
```

2. Write ALP macros:

i. **To read a character from the keyboard in the module (1) (in a different file).**

ii. **To display a character in module(2) (from different file).**

iii. **Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.**

```
READCHAR MACRO
        MOV AH, 01H
        INT 21H
ENDM

DISPCHAR MACRO
        MOV AH, 02H
        INT 21H
ENDM

INCLUDE F1.MAC
INCLUDE F2.MAC

DATA SEGMENT
        str db 50 DUP(?)
        n db ?
        msg1 db 10, 13, 'ENTER STRING : $'
        msg2 db 10, 13, 'ENTERED STRING IS : $'
DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
START:
        MOV AX, DATA
        MOV DS, AX
        LEA DX, msg1
        MOV AH, 09H
        INT 21H
        LEA SI, str
        CALL READSTRING
        MOV n, CL
        LEA DX, msg2
        MOV AH, 09H
        INT 21H
        LEA SI, str
        MOV CL, n
        CALL DISPSTRING
        MOV AH, 4CH
        INT 21H

READSTRING PROC NEAR
        MOV CL, 00H
UP:
        CMP CL, 50
        JZ L1
        READCHAR
        CMP AL, 0DH
        JZ L1
        MOV [SI], AL
```

```
        INC SI
        INC CL
        JMP UP
L1:
        RET
READSTRING ENDP

DISPSTRING PROC NEAR
UP2:
        CMP CL, 00
        JZ L2
        MOV DL, [SI]
        DISPCHAR
        INC SI
        DEC CL
        JMP UP2
L2:
        RET
DISPSTRING ENDP
CODE ENDS
END START
```

3.Write an ALP to read an alphanumeric character and display its equivalent ASCII code at the center of the screen.

```
CLRSCR MACRO
        MOV AH, 00H
        MOV AL, 02H
        INT 10H
ENDM

SETCURSOR MACRO row, col
        MOV DL, col
        MOV DH, row
        MOV BH, 00H
        MOV AH, 02H
        INT 10H
ENDM


DATA SEGMENT
        n DB ?
        msg1 DB 10, 13, 'ENTER THE CHARACTER : $'
DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
START:
```

```
        MOV AX, DATA
        MOV DS, AX
        LEA DX, msg1
        MOV AH, 09H
        INT 21H
        MOV AH, 01H
        INT 21H
        MOV n, AL
        CLRSCR
        SETCURSOR 12, 40
        MOV BL, n
        CALL DISPHEXA
        MOV AH, 01H
        INT 21H
        MOV AH, 4CH
        INT 21H
        DISPHEXA PROC NEAR
        MOV DL, BL
        MOV CL, 04H
        SHR DL, CL
        CMP DL, 09H
        JBE L1
        ADD DL, 07H
L1:
        ADD DL, 30H
        MOV AH, 02H
        INT 21H
        MOV DL, BL
        AND DL, 0FH
        CMP DL, 09H
        JBE L2
        ADD DL, 07H
L2:
        ADD DL, 30H
        MOV AH, 02H
        INT 21H
        RET
DISPHEXA ENDP
CODE ENDS
END START
```

4. Write an ALP to reverse a given string and check whether it is a palindrome or not.

```
DATA SEGMENT
        str1 DB 20 DUP(?)
        str2 DB 20 DUP(?)
        n DB ?
        M1 DB 10, 13, 'ENTER THE STRING : $'
        M2 DB 10, 13, 'STRING IS PALINDROME $'
        M3 DB 10, 13, 'STRING IS NOT PALINDROME$'
```

```
DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
START:
        MOV AX, DATA
        MOV DS, AX
        LEA SI, str1
        LEA DI, str2
        LEA DX, M1
        MOV AH, 09H
        INT 21H
        CALL READSTRING
L2:
        MOV n, CL
        MOV CL, 00H
        DEC SI
UP1:
        CMP CL, n
        JZ CHECK
        MOV AL, [SI]
        MOV [DI], AL
        DEC SI
        INC CL
        INC DI
        JMP UP1
CHECK:
        LEA SI, str1
        LEA DI, str2
        MOV CL, 00H
UP2:
        CMP CL, n
        JZ PAL
        MOV AL, [SI]
        CMP AL, [DI]
        JNZ NOTPAL
        INC SI
        INC DI
        INC CL
        JMP UP2
NOTPAL:
        LEA DX, M3
        MOV AH, 09H
        INT 21H
        JMP EXIT
PAL:
        LEA DX, M2
        MOV AH, 09H
        INT 21H
EXIT:
```

```
        MOV AH, 4CH
        INT 21H

READSTRING PROC NEAR
        MOV CL, 00H
UP:
        MOV AH, 01H
        INT 21H
        CMP AL, 0DH
        JZ L1
        MOV [SI], AL
        INC SI
        INC CL
        JMP UP
L1:
        RET
READSTRING ENDP
CODE ENDS
END START
```

5. Write an ALP to read your name from the keyboard and display it at a specified location on the screen in front of the message "What is your name?" You must clear the entire screen before display.

```
CLRSCR MACRO
        MOV AH, 00H
        MOV AL, 02H
        INT 10H
ENDM

SETCURSOR MACRO row, col
        MOV DL, col
        MOV DH, row
        MOV BH, 00H
        MOV AH, 02H
        INT 10H
ENDM

DATA SEGMENT
        str DB 30 dup(?)
        n DB ?
        msg1 DB 10,13,'ENTER YOUR NAME:$'
        msg2 DB 'What is your name?$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
```

```
        LEA DX,msg1
        MOV AH,09H
        INT 21H
        LEA SI, str
        CALL READSTRING
        MOV n,CL
        CLRSCR
        SETCURSOR 10,30
        LEA DX,msg2
        MOV AH,09H
        INT 21H
        LEA SI, str
        MOV CL, n
        CALL DISPSTRING
        MOV AH,01H
        INT 21H
        MOV AH,4CH
        INT 21H
READSTRING PROC NEAR
        MOV CL, 00H
UP:
        CMP CL,30
        JZ L1
        MOV AH,01H
        INT 21H
        CMP AL,0DH
        JZ L1
        MOV [SI],AL
        INC SI
        INC CL
        JMP UP
L1:
        RET
READSTRING ENDP

DISPSTRING PROC NEAR
UP1:
        CMP CL,00H
        JZ L2
        MOV DL,[SI]
        MOV AH,02H
        INT 21H
        INC SI
        DEC CL
        JMP UP1
L2:
        RET
DISPSTRING ENDP
CODE ENDS
END START
```

6. Write an ALP to compute the factorial of a positive integer 'n' using recursive procedure.

```
DATA SEGMENT
        n DB 06H
        fact DW ?
        msg DB 'FACTORIAL(6)=$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AX,1
        MOV BL,n
        MOV BH,00
        CALL FACTORIAL
        MOV fact,AX
        MOV BL, AH
        LEA DX,msg
        MOV AH,09H
        INT 21H
        CALL DISPHEXA
        MOV BX,fact
        CALL DISPHEXA
        MOV AH,4CH
        INT 21H
        FACTORIAL PROC
        CMP BX,1
        JE L1
        PUSH BX
        DEC BX
        CALL FACTORIAL
        POP BX
        MUL BX
L1:
        RET
FACTORIAL ENDP

DISPHEXA PROC
        MOV DL,BL
        MOV CL,04
        SHR DL,CL
        CMP DL,09H
        JBE L2
        ADD DL,07H
L2:
        ADD DL,30H
```

```
        MOV AH,02H
        INT 21H
        MOV DL,BL
        AND DL,0FH
        CMP DL,09H
        JBE L3
        ADD DL,07H
L3:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        RET
DISPHEXA ENDP
CODE ENDS
END START
```

7. Write an ALP to generate the first ‘n’ Fibonacci numbers.

```
DATA SEGMENT
        f1 DB 00H
        f2 DB 01H
        f3 DB ?
        N DB 10
        msg DB 'THE FIBONACCI SERIES IS:',10,13,'$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        LEA DX,msg
        MOV AH,09H
        INT 21H
        MOV BL,f1
        CALL DISPHEXA
        MOV DL,' '
        MOV AH,02H
        INT 21H
        MOV BL,f2
        CALL DISPHEXA
        MOV DL,' '
        MOV AH,02H
        INT 21H
        MOV CL,00
        SUB N, 02H
UP:
        CMP CL,N
        JZ EXIT
```

```
        MOV AL,f1
        ADD AL,f2
        MOV f3,AL
        MOV BL,f3
        CALL DISPHEXA
        MOV DL,' '
        MOV AH,02H
        INT 21H
        MOV AL,f2
        MOV f1,AL
        MOV AL,f3
        MOV f2,AL
        INC CL
        JMP UP
EXIT:
        MOV AH,4CH
        INT 21H
        DISPHEXA PROC
        PUSH CX
        MOV DL,BL
        MOV CL,04
        SHR DL,CL
        CMP DL,09H
        JBE L1
        ADD DL,07H
L1:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL,BL
        AND DL,0FH
        CMP DL,09H
        JBE L2
        ADD DL,07H
L2:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        POP CX
        RET
DISPHEXA ENDP
CODE ENDS
END START
```

8. Write an ALP to read the current time from the system and display it in the standard format on the screen.

```
DATA SEGMENT
```

```
        HR DB ?
        MIN DB ?
        SEC DB ?
        msg DB 'THE CURRENT TIME IS:',10,13,'$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        LEA DX,msg
        MOV AH,09H
        INT 21H
        MOV AH,2CH
        INT 21H
        MOV HR,CH
        MOV MIN,CL
        MOV SEC,DH
        MOV AL,HR
        AAM
        MOV BX,AX
        CALL DISPUNPACKDBCD
        MOV DL,':'
        MOV AH,02H
        INT 21H
        MOV AL,MIN
        AAM
        MOV BX,AX
        CALL DISPUNPACKDBCD
        MOV DL,':'
        MOV AH,02H
        INT 21H
        MOV AL,SEC
        AAM
        MOV BX,AX
        CALL DISPUNPACKDBCD
        MOV AH,4CH
        INT 21H

DISPUNPACKDBCD PROC NEAR
        MOV DL,BH
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL,BL
        ADD DL,30H
        MOV AH,02H
        INT 21H
        RET
```

```
DISPUNPACKDBCD ENDP
CODE ENDS
END START
```

9. Write an ALP to create and open a file named myfile.txt, and write a text 'Welcome to MICROPROCESSOR LAB' into this file and close. Later, read the content from the file myfile.txt and display on to the screen.

```
DATA SEGMENT
        BUF DB 30 DUP (?)
        FILE DB 'MYFILE.TXT',0
        MSG DB "WELCOME TO MICROPROCESSOR LAB$"
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE , DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX

        LEA DX,FILE
        MOV CX, 0
        MOV AH, 3CH
        INT 21H

        LEA DX, MSG
        MOV CX, 30
        MOV BX, AX
        MOV AH, 40H
        INT 21H

        MOV AH, 3EH
        INT 21H

        LEA DX, FILE
        MOV AL, 00H
        MOV AH, 3DH
        INT 21H

        LEA DX, BUF
        MOV BX, AX
        MOV CX, 30
        MOV AH, 3FH
        INT 21H

        MOV AH,3EH
        INT 21H

        LEA DX, BUF
        MOV AH,09H
        INT 21H
```

```
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

PART B

1. Write a Program to read 8-bit Boolean input from the serial monitor and display whether it has even or odd number of 1s and display the number of 1s in the input.

```
char rx_byte = 0;              // Store the byte received from the   user
int count;                  // Declare a integer variable to  store the value

void setup() {
  Serial.begin(9600);          // Set baudrate to 9600bps
}

void loop() {
  Serial.println("Enter the 8-bit boolean input:");
  while(1)
  {
  if(Serial.available() > 0){ // To check whether the user wrote  anything on the serial monitor
      rx_byte = Serial.read();  // Each time it receives one  character it is stored in variable
      if (rx_byte == '1'){
        count += 1;     // Increment count by 1 if the input bit  is 1
      }
       else if (rx_byte == '\n')
       {
        break;
        }
      }
    }

  if((count%2) == 0 && count !=0){  //condition to check the  value of count is even or not
         Serial.println("Even number of 1's");
        }
        else if ((count%2)!=0 && count !=0){  //Condition to check  the value of count is odd or
not
         Serial.println("Odd number of 1's");
        }
        else{
          Serial.println("Entered input does not have 1's");    //  if the count value is 0
        }
        Serial.print("Number of 1's in the input :");
```

```
  Serial.println(count);    // Display the number of 1's in the   given string
 Serial.println("_____________________________________________________________");
}
```

2. Write an Embedded C program
A. To control the brightness of LED using the potentiometer.
B. To turn buzzer ON/OFF using switches.

A. To control the brightness of LED using the potentiometer.

```
#define LED_PIN = 8;  // the PWM pin the LED is attached to

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // declare LED pin to be an output:
  pinMode(LED_PIN, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // reads the input on analog pin A0 (value between 0 and 1023)
  int analogValue = analogRead(A0);

  // scales it to brightness (value between 0 and 255)
  int brightness = map(analogValue, 0, 1023, 0, 255);

  // sets the brightness LED that connects to  pin 3
  analogWrite(LED_PIN, brightness);

  // print out the value
  Serial.print("Analog: ");
  Serial.print(analogValue);
  Serial.print(", Brightness: ");
  Serial.println(brightness);
  delay(100);
}
```

B. To turn buzzer ON/OFF using switches.

```
#define switch_1 14   // digital pin 14 is connected with switch_1
#define switch_2 15   // digital pin 15 is connected with switch_2
#define Buzzer 16       // digital pin 13 is connected with Buzzer

void setup() {
  pinMode(Buzzer,OUTPUT);    //Configure Buzzer as output
```

```
  pinMode(switch_1,INPUT); //Configure switch_1 as input
  pinMode(switch_2,INPUT); //Configure switch_2 as input
}

void loop() {
if (digitalRead(switch_1) == LOW && digitalRead(switch_2) == HIGH){ // if the switch_1
is pressed
    digitalWrite(Buzzer,HIGH);   //Turn ON Buzzer
  }
  else if (digitalRead(switch_1) == HIGH && digitalRead(switch_2) == LOW){ // if the
switch_2 is pressed
    digitalWrite(Buzzer,LOW); //Turn OFF Buzzer
  }
  else if (digitalRead(switch_1) == LOW && digitalRead(switch_2) == LOW){
     digitalWrite(Buzzer,HIGH); //Turn ON Buzzer
  }
  else {
    digitalWrite(Buzzer,LOW); //Turn OFF Buzzer
  }
  delay(500);
}
```

3. Write an Embedded C program

A. To display two messages alternatively with flickering effect on LCD for a suitable period of time.

B. To display a message using scrolling effect on LCD for a suitable period of time in both directions simultaneously.

A. To display two messages alternatively with flickering effect on LCD for a suitable period of time .

```
#include <LiquidCrystal_I2C.h>  // Include the library
LiquidCrystal_I2C lcd(0x27,16,2); // Initialize LCD pins

void setup() {
  lcd.backlight();
  lcd.init(); // Initializing LCD
}

void loop() {
  lcd.setCursor(0,0);// Set the cursor to the first position
  lcd.print("Student Name");// Display the student name
  delay(500);//delay of 500ms
  lcd.clear();// Clear the string on LCD
  lcd.setCursor(0,1);
  lcd.print("Institute Name");
  delay(500);
  lcd.clear();
}
```

B. To display a message using scrolling effect on LCD for a suitable period of time in both directions simultaneously.

```
#include <LiquidCrystal_I2C.h> // Include the library
LiquidCrystal_I2C lcd(0x27,16,2); // Initialize LCD pins

int i;
void setup() {
lcd.backlight();
lcd.init(); // Initializing LCD
}
void loop()
{
  int i=0;
  while(i<16)
  {
    lcd.setCursor(i,0);
    lcd.print("Microcontroller");

    lcd.setCursor(15-i,1);
    lcd.print("Programming");
    delay(500);
    lcd.clear();
    i++;
  }

}
```

4. Write an Embedded C program to perform up-down counter on a 7-segment display using switches.

```
#include <TM1637Display.h>

#define switch_1 14
#define switch_2 15

const int CLK = 8; //Set the CLK pin connection to the display
const int DIO = 9; //Set the DIO pin connection to the display

const uint8_t blank[] = {0x00, 0x00, 0x00,0x00};

TM1637Display display(CLK, DIO); //set up the 4-Digit Display.

int num = 0;
bool UP = 0;
bool DOWN = 0;

void setup(){
```

```
    pinMode(switch_1, INPUT);
    pinMode(switch_2,INPUT);
    display.setBrightness(7); //set the display to maximum brightness
    display.setSegments(blank);//clear display
}

void loop(){
   call();
   if(UP){
    if(num == 9999)
      num = 0;
    call();
    num++;// increment 'num'
   }
   else if(DOWN){
    if(num == 0 )
      num = 9999;
    call();
    num--;        // decrement 'num'
   }
   else{
    num = 0;
   }
}

void call(){
  display.showNumberDec(num, true, 4, 0);
  delay(500); //wait 200 milliseconds
  if(!digitalRead(switch_1)){ // When switch_1 is pressed
    UP = 1;
    DOWN = 0;
  }
  if(!digitalRead(switch_2)){ // When switch_2 is pressed
   UP = 0;
   DOWN = 1;
  }
}
```

5. Write an Embedded C program to perform ring counter on a 7-segment display.

```
 #include <TM1637Display.h>

const int CLK = 8; //Set the CLK pin connection to the display
const int DIO = 9; //Set the DIO pin connection to the display

uint8_t blank[] = {0x00, 0x00, 0x00,0x00}; // store the initial input
uint8_t ring[] = {0x06, 0x3f, 0x3f,0x3f}; // the numbers represents the values 1, 0, 0, 0
uint8_t dumy[] = {0x3f, 0x3f, 0x3f, 0x3f};
```

```
TM1637Display display(CLK, DIO); //set up the 4-Digit Display.

int i = 0;

void setup(){
   display.setBrightness(7); //set the diplay to maximum brightness
   display.setSegments(blank);//display the initial value
}

void loop(){
  display.setSegments(ring);
  delay(1000);
  for(i=0;i<3;i++){
   ring_counter();
  }
}

void ring_counter(){
   for (i=0;i<4;i++){
   dumy[i]=ring[(i+3)%4];
  }
  display.setSegments(dumy);
  delay(1000);
  for(i=0;i<4;i++){
   ring[i]=dumy[i];
  }
}
```

6. Write an Embedded C program to control the speed of rotation of a stepper motor using potentiometer.

```
const int stepPin = 3;
const int dirPin = 2;
int customDelay,customDelayMapped; // Defines variables

void setup() {
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  digitalWrite(dirPin,HIGH); //Enables the motor to move in a clockwise direction
}
void loop() {
  customDelayMapped = speedUp();
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(customDelayMapped);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(customDelayMapped);
}

// Function for reading the Potentiometer
int speedUp() {
  int customDelay = analogRead(A0); // Reads the potentiometer
```

```
  int newCustom = map(customDelay, 0, 1023, 1000,4000);
  return newCustom;
}
```

7. Write an Embedded C program to drive a stepper motor clockwise and anticlockwise with the help of switches

```
const int switch_1 = 14;
const int switch_2 = 15;
const int dirPin = 2;
const int stepPin = 3;
bool clockwise =0;
bool counterclockwise =0;

void setup()
{
  // Declare switches as inputs
  pinMode(switch_1,INPUT);
  pinMode(switch_2,INPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop()
{
  call();
  if(clockwise){
    digitalWrite(dirPin,HIGH);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(1000);
  }
  if(counterclockwise){
    digitalWrite(dirPin,LOW);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(1000);
  }
}

void call (){
  if (digitalRead(switch_1) == LOW && digitalRead(switch_2) == HIGH){
    clockwise = 1;
    counterclockwise = 0;
  }
```

```
  if(digitalRead(switch_1) == HIGH && digitalRead(switch_2) == LOW){
    clockwise = 0;
    counterclockwise = 1;
  }
}
```

8. Write an Embedded C program to read the input from 4x4 keypad and simulate operations as a calculator

```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);
const byte ROWS = 4;
const byte COLS = 4;

char keys [ROWS] [COLS] = {
  {'1', '2', '3', '+'},
  {'4', '5', '6', '-'},
  {'7', '8', '9', '*'},
  {'C', '0', '=', '/'}
};
byte rowPins[ROWS] = {22,23,24,25};
byte colPins[COLS] = {26,27,28,29};

Keypad CustomKeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS,
COLS );

boolean PresentValue = false;
boolean Next = false;
boolean Final = false;
String num1, num2;
int answer;
char op;

void setup(){
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Welcome");
  lcd.setCursor(0,1);
  lcd.print("A=+,B=-,C=*,D=/");
  delay(5000);
  lcd.clear();
}

void loop(){
```

```
        char key = CustomKeypad.getKey();

        if (key != NO_KEY &&
(key=='1'||key=='2'||key=='3'||key=='4'||key=='5'||key=='6'||key=='7'||key=='8'||key=='9'||key=='
0'))
        {
          if (PresentValue != true)
          {
            num1 = num1 + key;
            int numLength = num1.length();
            lcd.setCursor(15 - numLength, 0); //to adjust one whitespace for operator
            lcd.print(num1);
          }
          else
          {
            num2 = num2 + key;
            int numLength = num2.length();
            lcd.setCursor(15 - numLength, 1);
            lcd.print(num2);
            Final = true;
          }
        }

        else if (PresentValue == false && key != NO_KEY && (key == '/' || key == '*' || key
== '-' || key == '+'))
        {
          if (PresentValue == false)
          {
            PresentValue = true;
            op = key;
            lcd.setCursor(15,0);
            lcd.print(op);
          }
        }

        else if (Final == true && key != NO_KEY && key == '='){
          if (op == '+'){
            answer = num1.toInt() + num2.toInt();
          }
          else if (op == '-'){
            answer = num1.toInt() - num2.toInt();
          }
          else if (op == '*'){
            answer = num1.toInt() * num2.toInt();
          }
          else if (op == '/'){
            answer = num1.toInt() / num2.toInt();
          }
            lcd.clear();
            lcd.setCursor(15,0);
```

```
      lcd.autoscroll();
      lcd.print(answer);
      lcd.noAutoscroll();
    }
    else if (key != NO_KEY && key == 'C'){
      lcd.clear();
      PresentValue = false;
      Final = false;
      num1 = "";
      num2 = "";
      answer = 0;
      op = ' ';
    }
}
```