Benha University

Benha Faculty of Engineering

Dept. of Electrical Engineering

“Simple Calculator”

Submitted By

Mohamed shaaban Ahmed

Mohamed Mahmoud Ahmed

Mohamed Wael Mohamed

Mahmoud Shehta Ahmed

Mariam Fathi Ali

Yosra Nasser Mansour

Dr. Ahmed El_awamry

Eng. Wael Zaki

Dept. of Electrical Engineering

(Computer and Communications Engineering)

Benha Faculty of Engineering

Benha University

2nd of decameter, 2023

Abstract

This code is an implementation of a calculator .There are two modes in calculator :

a-Normal mode

1-This mode do four operation on two two digits numbers .

Operations:

a-addtion ab+cd=ans
b-subtraction ab+cd=ans
c-multiplication ab+cd=ans
d-division ab+cd=ans

2-If user enter only one number and press equal it will print only the entered number

b-Advanced mode

1-This mode do four operation on one three digits number max(255) .

Operations:

a-convert from decimal format to binary
b convert from decimal format to hexa
c- convert from decimal format to BCD
d-square root

2-If user enter only one number and press equal it will print only the entered number

Features

1-calculator contain two modes(a-normal mode→(+ , - , * , / , =) ,b- advanced mode)
2-User can press ON button to go to select mode in any time while using calculator →don't need to restart calculator

a-features in normal mode

1-do all operation on two digits not one digit
2-any answer printed with no additional zeroes on the left
- a- 22+22=44
- b- 2+3=5 →not 05
- c- 1*3=3 →not 0003
- d- 7/4=1.75 →not 01.75000

3-user can input numbers with any format for all operations
- a- 1+3= → doesn't have to input 01+03
- b- 1/33=
- c- 10*33=
- d- …and so on

4-calculator covers all input formats as three digit number or many operations in format
- a- 1++3= →errror
- b- +12*36= →error
- c- 123+12 →error
- d- ….and so on

5-calculator do multiplication on any two numbers from to digit and print a correct answer

a- 99*99=9801
b- 3*22=66
c- ….and so on

6-calculator do division and print exponent part and fraction part with no error in calculation and cover corner case input

a-divition by zere →error

7-In subtraction it print a correct ans even ans is positive or negative

b-advanced mode

1-it works on a three digit number from 0 to 255
2-it have four operations

a-convert decimal to binary
b-convert decimal to hexa
c-convert decimal to BCD with no additional zeroes
d-square root

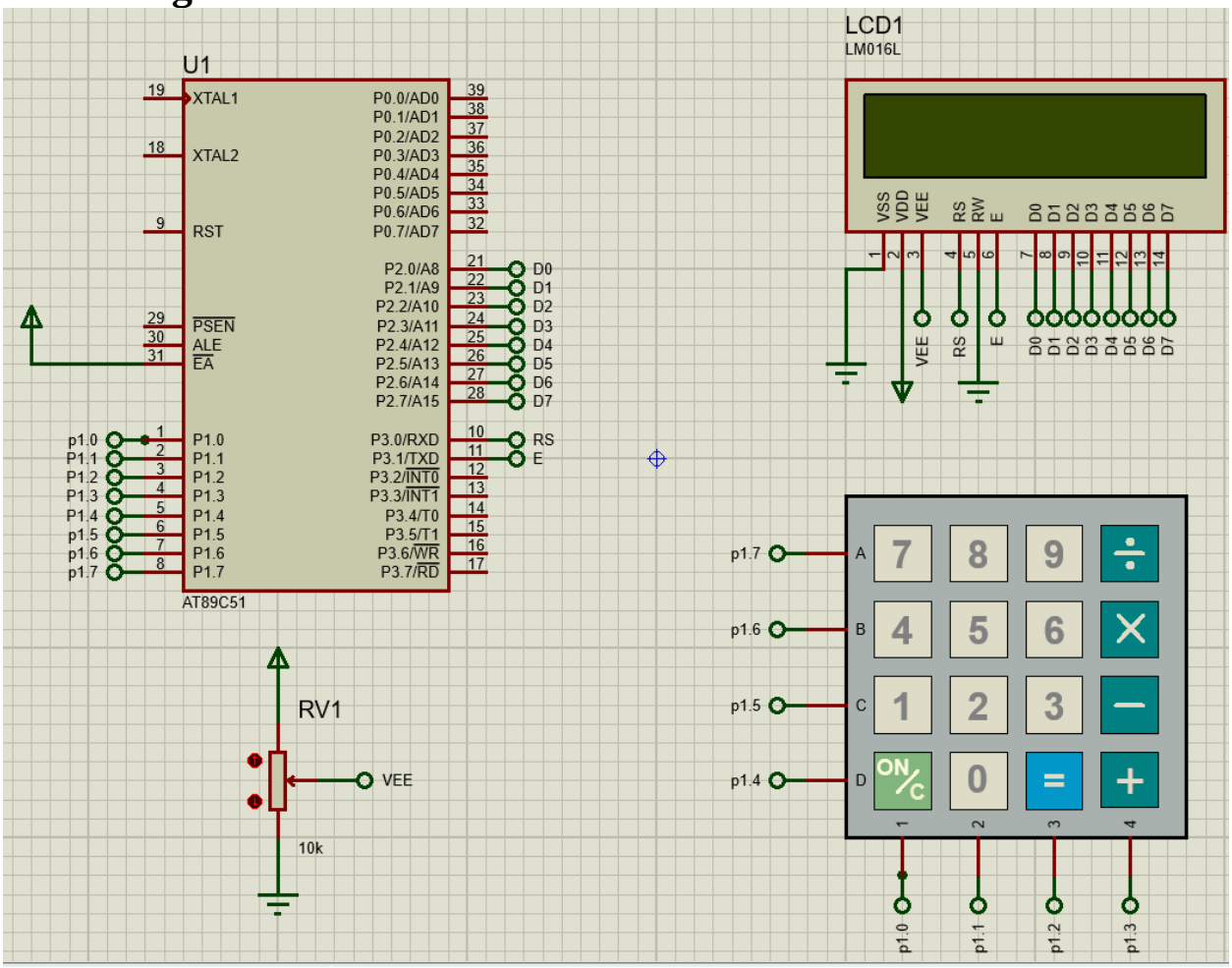
3-if user enter number then equal it will print number not error
4-it covers wrong input formats

a- 1235=→error
b- 1236956=→error
c- …and so on

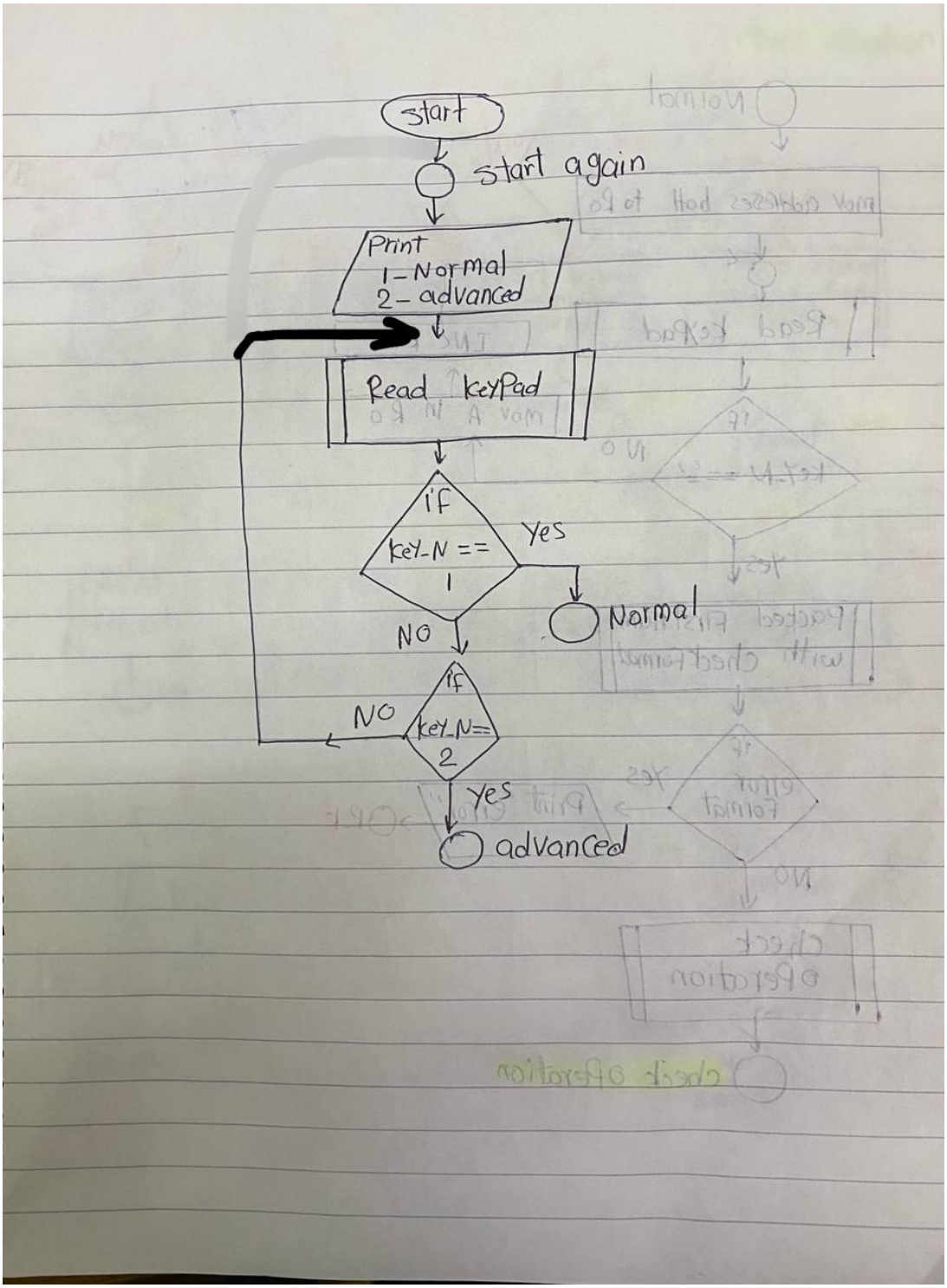
Components

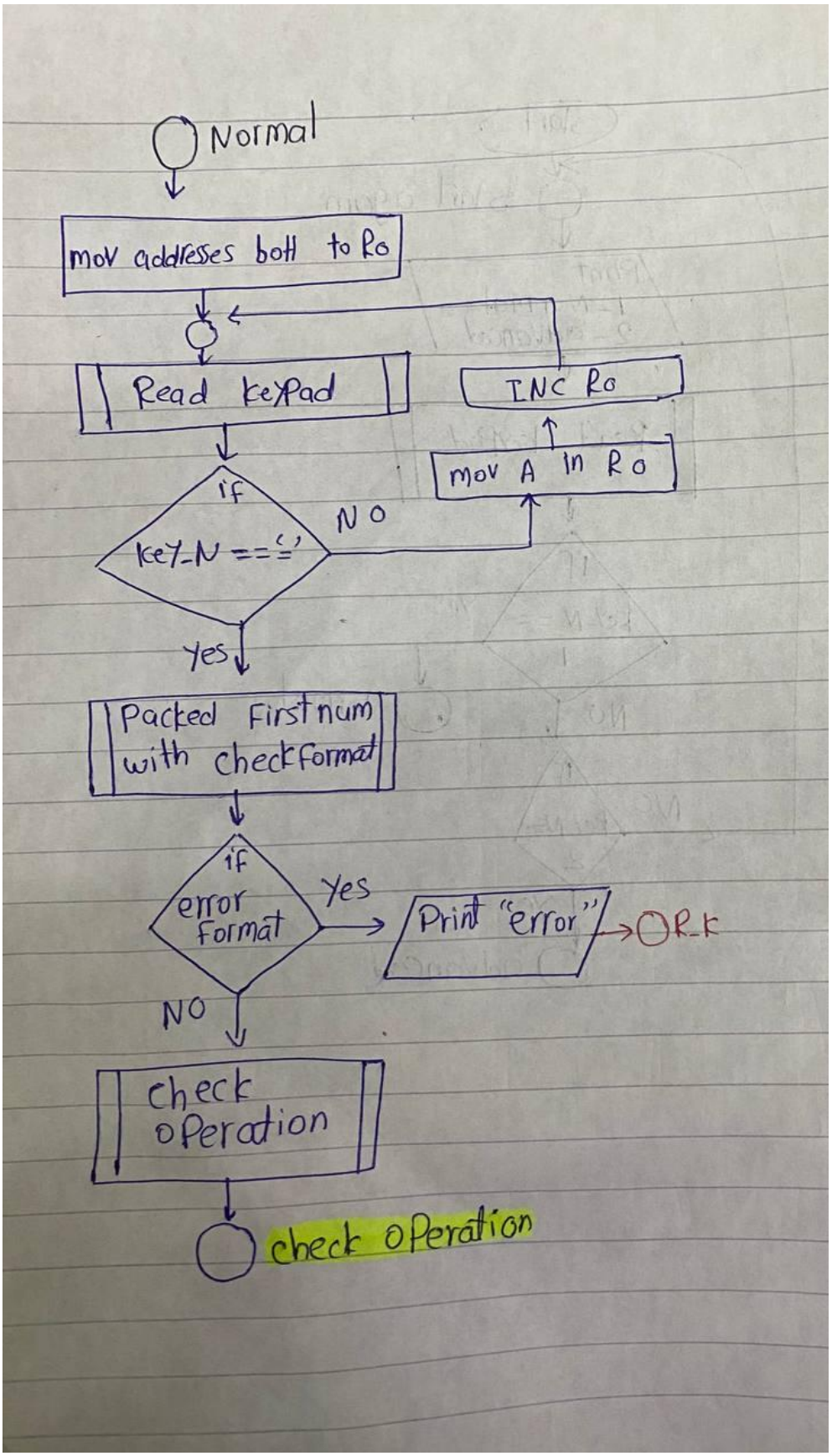
1-80s51 Microcontroller
2-LCD
3-Keypad
4-crystal 11.0952 MHZ
5- capacitors (10uf , 30uf)
6-resistances (10kΩ)
7-variable resistance(10kΩ)

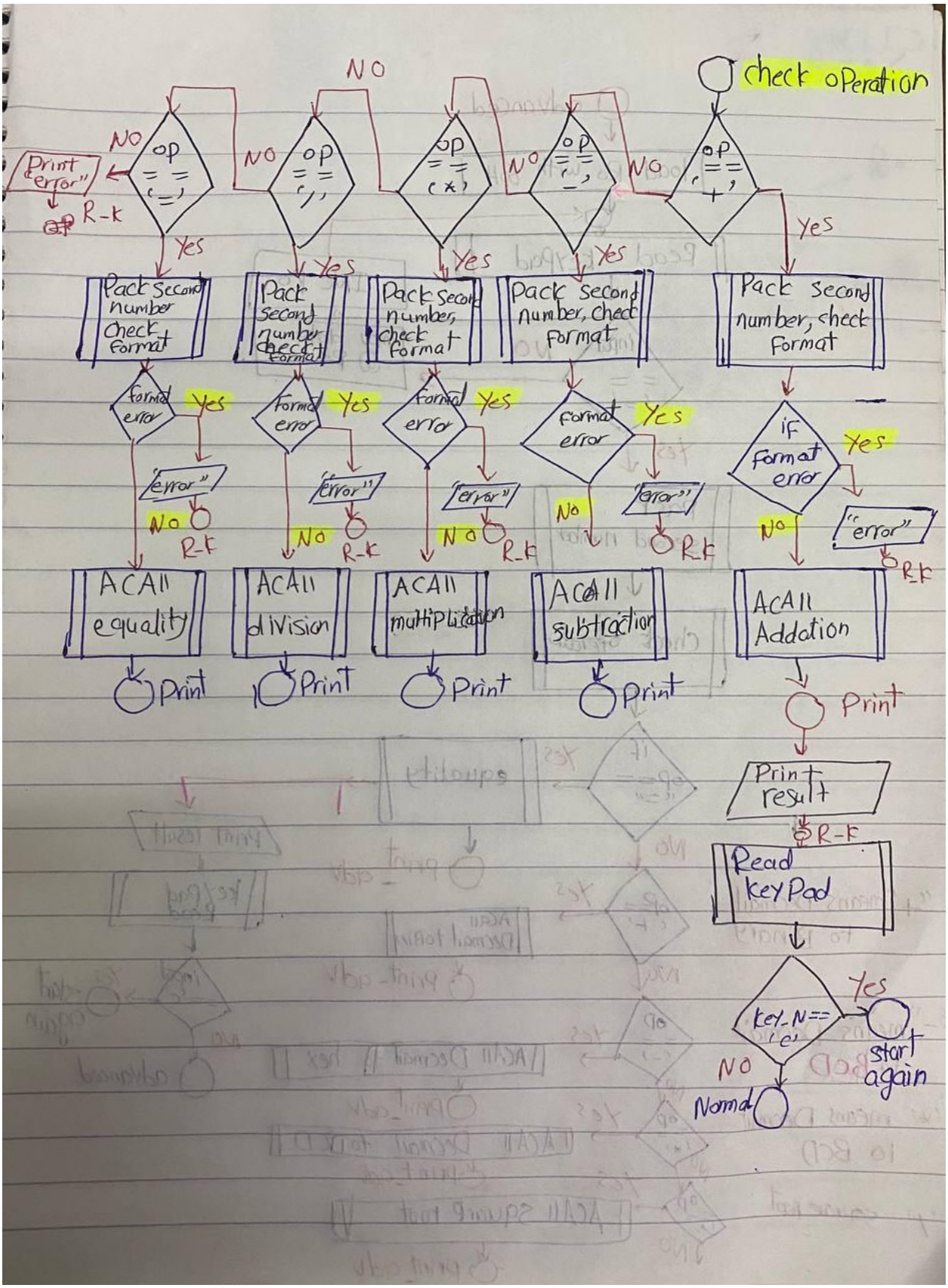
Block diagram

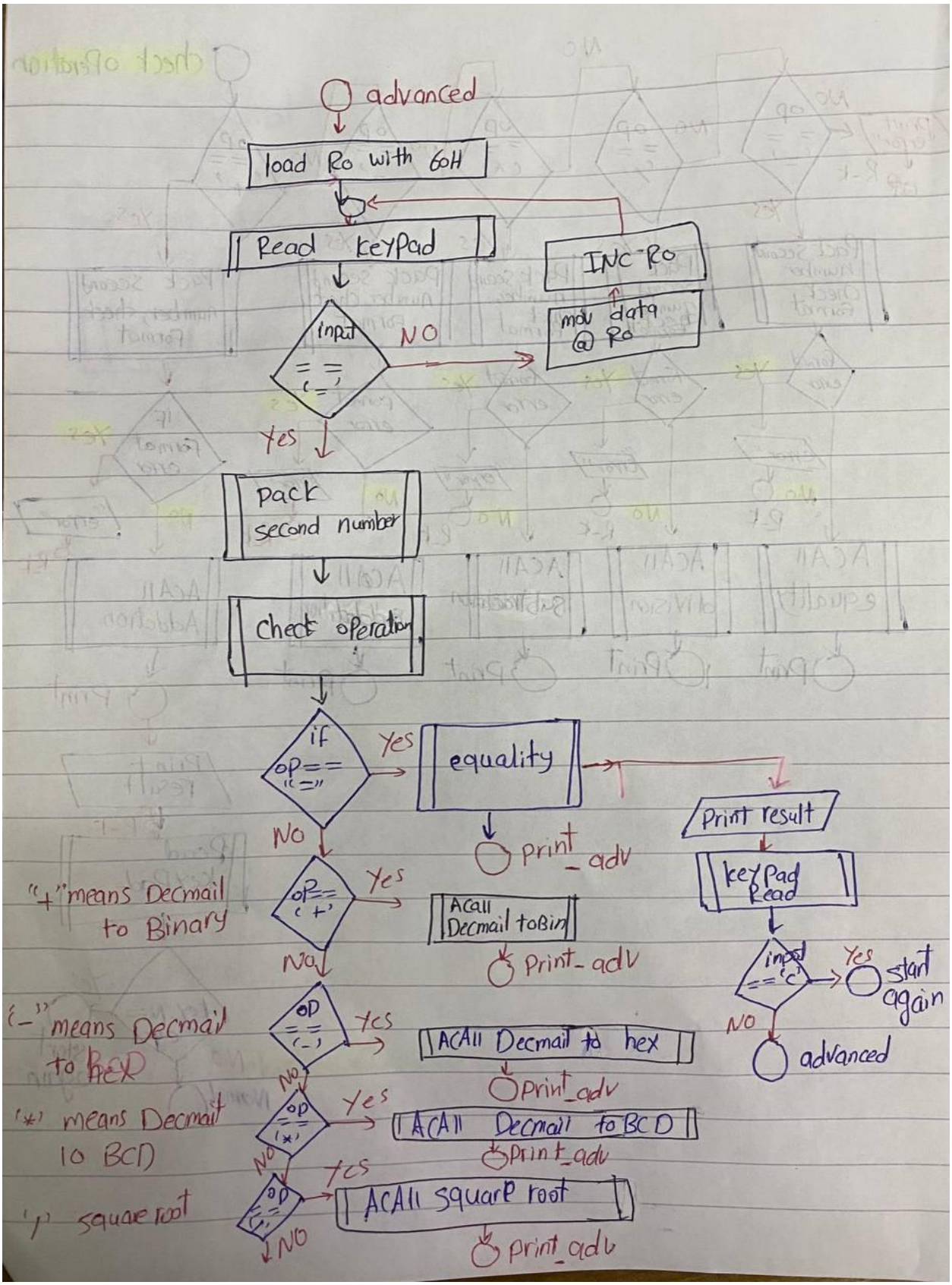


Flowchart


start
start again
Print
1-Normal
2-advanced
Read keyPad
if
key-N ==
1
Yes
Normal
NO
if
key-N==
2
NO
Yes
advanced


Normal
mov addresses both to R0
Read keyPad
INC R0
mov A in R0
if
key_N == '='
NO
yes
Packed Firstnum with checkformat
if
error format
Yes
Print "error"
ORG
NO
check operation
check operation


check operation
op == '='
op == '/'
op == '*'
op == '-'
op == '+'
NO
Yes
Print "error"
R-t
Pack second number check format
Pack second number, check format
format error
"error"
No
ACALL equality
ACALL division
ACALL multiplication
ACALL subtraction
ACALL Addation
Print
Print result
Read keyPad
key_N == 'e'
start again
Nomal


advanced
load R0 with 60H
Read keypad
INC R0
mov data @ R0
input
NO
Yes
pack second number
check operation
if op == "="
Yes
equality
print_adv
No
"+" means Decmail to Binary
op == '+'
Yes
ACall Decmail to Bin
Print_adv
No
"-" means Decmail to hex
op == '-'
Yes
ACall Decmail to hex
Print_adv
No
"*" means Decmail to BCD
op == '*'
Yes
ACall Decmail to BCD
Print_adv
No
"/" square root
op == '/'
Yes
ACall square root
print_adv
No
Print result
keypad Read
input == 'c'
Yes
start again
NO
advanced

Assembly code

```
KEYPAD_PORT EQU P1
C1 BIT P1.0
C2 BIT P1.1
C3 BIT P1.2
C4 BIT P1.3
RO4 BIT P1.4
RO3 BIT P1.5
RO2 BIT P1.6
RO1 BIT P1.7

LCD_PORT EQU P2
Rs BIT P3.0
E BIT P3.1

;INITIAL MESSAGE

ORG 00H
MOV KEYPAD_PORT,#0FFH          ;KEYPAD PORT AS INPUT
LCALL LCD_INIT
MOV DPTR,#MESSAGE              ;PRINT THE INITIAL MESSAGE
AGAIN:
CLR A
MOVC A,@A+DPTR
JZ HERE
LCALL LCD_DATA
INC DPTR
SJMP AGAIN

HERE:
LCALL DELAY
LCALL DELAY
LCALL LCD_INIT
MOV A,#'1'
LCALL LCD_DATA
MOV A,#'-'
LCALL LCD_DATA
MOV A,#'N'
LCALL LCD_DATA
```

```
MOV A,#'O'
LCALL LCD_DATA
MOV A,#'R'
LCALL LCD_DATA
MOV A,#'M'
LCALL LCD_DATA
MOV A,#'A'
LCALL LCD_DATA
MOV A,#'L'
LCALL LCD_DATA
MOV A,#0C0H
LCALL LCD_COMMAND
MOV A,#'2'
LCALL LCD_DATA
MOV A,#'-'
LCALL LCD_DATA
MOV A,#'A'
LCALL LCD_DATA
MOV A,#'D'
LCALL LCD_DATA
MOV A,#'V'
LCALL LCD_DATA
MOV A,#'A'
LCALL LCD_DATA
MOV A,#'N'
LCALL LCD_DATA
MOV A,#'C'
LCALL LCD_DATA
MOV A,#'E'
LCALL LCD_DATA
MOV A,#'D'
LCALL LCD_DATA
CHECK_MOD:
LCALL READ_KEY
CJNE A,#'1',CHECK_ADV
SJMP NORMAL
CHECK_ADV:
CJNE A,#'2',CHECK_MOD
LJMP ADVANCED
NORMAL:
LCALL LCD_INIT
;-----------------RELEASE USED BYTES---------------------------------
;CALCULATOR
MOV 66H,#'Y'          ;TO CHECK THAT USER USED ONLY SIX BYTES OR LESS
MOV R0,#60H           ;START LOCATION TO STORE INPUT DATA
MOV R1,#5
```

```
CLEAR_LOOP:          ; LOOP TO CLEAR 5 BYTES USED IN NUMBER
MOV @R0,#0
INC R0
DJNZ R1,CLEAR_LOOP
MOV R0,#60H          ;SRART INPUT FROM LOCATION WITH ADDRESS H
;---------------------------------RECIEVE INPUT FROM USER----------------------
INPUT_LOOP:
LCALL READ_KEY
LCALL LCD_DATA             ;PRINT ANY INPUT VALUE
MOV @R0,A                  ;INDIRECT SAVE FOR DARA IN RESERVED LOCATION
INC R0                     ;TO SAVE IN NEXT LOCATION
CJNE A,#'=',INPUT_LOOP     ;USER INPUT DATA UNTIL PRESS '='
MOV A,66H                  ;CHECK BYTE 66H IF = 'Y'-->ACESS OPERATION   ELSE--
->WRONG
CJNE A,#'Y',ERORROR        ;ERROR INPUT FORMAT
SJMP START_CALCULATE
ERORROR:LJMP ERROR         ;ERROR MESSAGE
;-----------------------------------------------------------------------
START_CALCULATE:
MOV R0,#60H
MOV A,@R0
LCALL CHECK_BYTE           ;CHECK THAT FIRST BYTE IS A DIGIT
CJNE R7,#0,ILLEAGAL_INPUT  ;NOT DIGIT -->ERROR
SJMP LEAGAL_1              ;DIGIT -->START PACKING OF FIRST NUMBER
ILLEAGAL_INPUT:
LJMP ERROR
LEAGAL_1:
ANL A,#0FH                 ;GET BCD NUMBER FROM ASCII INPUT
MOV R1,A                   ;SAVE FIRST NUMBER IN R0
INC R0                     ;GET NEXT LOCATION OF INPUT DATA
MOV A,@R0
LCALL CHECK_BYTE           ;CHECK IF NEXT BYTE IS DIGIT OR OPERATION
CJNE R7,#0,CHECK_OPERATON  ;IF BYTE CONTAIN ANY SIGN --->GO TO CHECK
OPERATION
ANL A,#0FH                 ;ELSE IF BYTE CONTAIN DIGIT WE CONTINUE IN PACKING
FIRST NUMBER
MOV R3,A                   ;SAVE VALUE TEMPRARY IN R3
MOV A,R1                   ;NUMBER=10*R1+R3
MOV B,#10
MUL AB                     ;10R1
ADD A,R3                   ;10*R1+R3
MOV R1,A
INC R0                     ;GET THIRD INPUR BYTE
MOV A,@R0
                    ;BYTE MUST BE SIGN OF OPERATION
```

```
                                        ;IF BYTE NOT SIGN THIS MEAN THAT USER
INPUT NUMBER FROM 3 DIGITS
;-------------------------------CHECK OPERATION-------------------------

CHECK_OPERATON:
LCALL CHECK_BYTE
CJNE R7,#1,NEXT_OPERATION
SJMP EQUALITY
NEXT_OPERATION:CJNE R7,#2,NEXT_OPERATION1
SJMP ADDITION
NEXT_OPERATION1:CJNE R7,#3,NEXT_OPERATION2
SJMP SUBTRACTION
NEXT_OPERATION2:CJNE R7,#4,NEXT_OPERATION3
LJMP MULTIPLICATION
NEXT_OPERATION3:CJNE R7,#5,ERROR_BYTE  ; ENTERED NUMBER IS 3 DIGIT SO
CALCLATOR SHOULD PRINT ERROR
LJMP DIVISION
ERROR_BYTE:
LJMP ERROR                      ;PRINT ERROR


;*******************************OPERATIONS**********************************
;-----------------------------EQUALITY----------------------------------

EQUALITY:       ;WILL PRINT ENTERED NUMBER(FIRST NUMBER)
MOV A,R1        ;FIRST NUMBER SAVED IN R1 SO WE WANT TO PRINT IT
LCALL PRINT          ;FUNTION TO PRINT ANY DATA IN ONE BYTE IN DECIMAL
FORMAT
LCALL READ_KEY    ;PRESS ANY KEY TO START NEXT OPERATION
LJMP NORMAL

;---------------------------ADDITOIN--------------------------------
ADDITION:
ACALL PACING_SECOND ;FUNCTION TO PACK SECOND NUMVER (IT ALSO CHECK
ERROR IN INPUT  IN SECOND DIGIT)-->1+
ADD A,R1
LCALL PRINT
LCALL READ_KEY
LJMP NORMAL

;-------------------------SUBTRACTION-------------------------------
SUBTRACTION:
ACALL PACING_SECOND
CLR C            ;BECAUSE ALL SUB IS A=A-R-C
MOV R4,A         ;R4=SECOND NUMBER
MOV A,R1
```

```
SUBB A,R4
JNC POSITIVE        ;IF C=0 THEN RESULT IS POSITIVE
MOV B,A             ;SAVE RESULT VALUE
MOV A,#'-'          ;PRINT NEGATIVE SIGN
LCALL LCD_DATA
MOV A,B             ;PREPARE NEGATIVE VALUE TO PRINT BY 2'S COMPLEMENT
CPL A               ;2'S = 1'S+1
INC A
POSITIVE:           ;PRINT RESULT
LCALL PRINT
LCALL READ_KEY
LJMP NORMAL

;----------------------------------------MULTIPLICATION--------------------------------
MULTIPLICATION:                 ;A*R1
ACALL PACING_SECOND                 ;A=SECOND NUMBER
MOV B,R1
MUL AB
;-----------------------------------PRINT RES OF MULTIPLICATION--------------------------
MOV R6,#0
SJMP SUB_1000
INC_COUNT:
INC R6
SUB_1000:
CLR C
SUBB A,#0E8H
XCH A,B
SUBB A,#03H
XCH A,B
JNC INC_COUNT
MOV R7,A            ;SAVE VALUE OF ACC TO GET NEXT REMAINDER
MOV 1,B             ;SAVE VALUE OF B TO GET NEXT REMAINDER
MOV A,R6
MOV 11H,A           ;SAVE FIRST DIGIT TO CHECK WHICH ZEROES WON'T BE PRINTED
-->(0025)
JZ DOONT_PRINT      ;0521-->DON'T PRINT 0
LCALL PRINT
DOONT_PRINT:
CLR C
MOV A,R7            ;ADD WITH 1000 AGAIN TO GET REMAINTER
ADD A,#0E8H
XCH A,1             ;GET B VALUE FROM ADDRESS 1 BY XCH IT WITH A
ADDC A,#03H
XCH A,1             ;RETURN VALUE OF A
MOV R6,#0
MOV B,1             ;VALUE OF B FROM LOC 1
```

```
SJMP SUB_100
;-----------------------GET THIRD DIGIT TO PRINT BY SUBRACT FROM 100 IN LOOP-------
------------------------
SUB_INC_2:
INC R6
SUB_100:
CLR C
SUBB A,#100
XCH A,B
SUBB A,#0
XCH A,B
JNC SUB_INC_2
MOV R7,A
MOV A,R6
MOV 12H,A           ;USED IN PRINT WHICH ZEROES WON'T BE PRINTED -->(0005)
JNZ DONNT_PR2
MOV A,11H
JNZ DONNT_PR2
SJMP ADD_100
DONNT_PR2:
MOV A,12H
LCALL PRINT
ADD_100:
MOV A,R7
ADD A,#100
MOV B,#10
DIV AB
MOV 13H,A           ;0001-->NO ZEROES WILL BE PRINT
JNZ NOOO1           ;0901-->901 WILL BE PRINTED
MOV A,12H           ;9001-->ALL NUMBER WILL BE PRINTED
JNZ NOOO1           ;SO WE USE EACH SAVED VALUE TO PRINT CORRECT FORMAT
MOV A,11H
JNZ NOOO1
SJMP PRINT_LAST_DIGIT
NOOO1:
MOV A,13H
ADD A,#30H
LCALL LCD_DATA
PRINT_LAST_DIGIT:
MOV A,B
ADD A,#30H
LCALL LCD_DATA
LCALL READ_KEY
LJMP NORMAL
;---------------------------------------DIVISION----------------------------------------
DIVISION:                   ;R1/A------>A/B
```

```
ACALL PACING_SECOND
CJNE A,#0,DIVIDE                ;A MUST 1= ZERO TO BE ABLE TO DO OPERATION
LJMP ERROR
DIVIDE:
MOV B,A
MOV 58H,A                       ;SAVE VALUE DIVIDED BY IT TO USE IN REMAINDER
MOV A,R1                        ;A/B
DIV AB
MOV R7,#0                       ;REMINDER FLAG        INTTIOL WITH ZERO  (0-->NO
REM)**(1-->REM)
XCH A,B                         ;TO BE ABLE TO CHECK   B=0-->NO REM   B=VALUE -->REM
CJNE A,#0,REM_FLAG
SJMP NO_REM                     ;NO REM SO SKIP INST OF SET FLAG (MOV R7,#1)
REM_FLAG:
MOV R7,#1                       ;SET REM FLAG BECAUSE B CONTAIN VALUE
NO_REM:
XCH A,B                         ;RETURN A VALUE TO PRINT INTEGER PART   (A CONTAINS
INT ),(B CONTAINS REM)
MOV 57H,B                       ;SAVE REM VALUE TO USE IN REMAINDER
MOV B,#10
DIV AB
JZ DONT_PRINTT                  ;IF NUM=04 -->WE WON'T PRINT 0
ADD A,#30H
LCALL LCD_DATA
DONT_PRINTT:
MOV A,B                         ;VALUE OF B WILL BE PRNTED ANY WAY BECASE WE DIV
2 DIGITS SO MAX RES IS IN 2 DIGITS (00) WILL PRINT ONE ZERO AS RESULT
ADD A,#30H
LCALL LCD_DATA
CJNE R7,#1,NOO_REM              ;USE FLAG TO CHECK IF THERE IS A REM
MOV A,#'.'                  ;FALG =1 SO WE PRINT '.' THEN PRINT VALUE OF REMAINDER
LCALL LCD_DATA
MOV R7,#5                   ;4 NUMBER FOR REMAINDER
MOV B,57H                   ;REMAINDER VALUE ASSINED TO B
REMAINDER:
MOV A,#10                   ;R=(10*REM)/DIVBY
MUL AB                      ;10REM
XCH A,B                     ;TO CHECK IF VALUE IN ONE OR TWO BYTES  (VALUE>255)-
->DIV USING SUBTRACTION     ELSE-->USE DIV INST
CJNE A,#0,SUB_REM
XCH A,B                     ;RETURN VALUE OF A
MOV B,58H                   ;VALUE DIIDED BY
DIV AB
CJNE A,#0,PR_REM            ;WE CONTIUE IN DETERMINING REMAINDER VALUE
UTIL A,B REACH 0
```

```
XCH A,B                    ;IF A=0  BUT B=VALUE WE SHOULD PRINT ZERO THEN MUL BY 10
CJNE A,#0,NEE_REM          ;IF A,B=0 END OF DETERMINING REMAINDER
SJMP NOO_REM
NEE_REM:                   ;WE PRINT VALUE IN A"0" ANY WAY EXPECT IF B ALSO = '0' WE END DETERMINING REMAINDER
XCH A,B
MOV A,#30H
LCALL LCD_DATA
SJMP REMAINDER1
PR_REM:                    ;WE WRITE TWO DIFF INST FOR PRINT BEACAUSE IF A EQUAL VALUE WE DON'T WANT TO CHECK IF B=0 BCUASE A PRINTED ANY WAY
ADD A,#30H
LCALL LCD_DATA
REMAINDER1:
DJNZ R7,REMAINDER          ;DEC COUNTER
SJMP NOO_REM               ;END OF LOOP IF R7=0
SUB_REM:                   ;TO GET DIV ANS BY SUB
XCH A,B                    ;RETURN VALUE OF A BECOUSE WE DO XCH A,B BEFORE LAST CHECK
MOV R6,#0                  ;COUNTER OF NUMBER OF SUB
SJMP SUBBBBB               ;TO SKIP INC IN FIRST LOOP
SUBBBBB1:
INC R6
SUBBBBB:
CLR C
SUBB A,58H                 ;SEB A FROM VALUE DIVIDED BY IT UNTIL REACH VALUE LESS THAN VALUE DIVIDED BY
XCH A,B
SUBB A,#0                  ;AFTER SUBB MAX VALUE IS 2 DIGITS DON'T REACH B SO WE DO B-C
XCH A,B
JNC SUBBBBB1
MOV 70H,A                  ;SAVE VALUE TO USE IN NEXT REM
MOV A,R6                   ;RES TO PRINT
ADD A,#30H
LCALL LCD_DATA
MOV A,70H
ADD A,58H                  ;GET NEXT REMAINDER
XCH A,B                    ;REMAINDER ALWAYS IN B
DJNZ R7,REMAINDER          ;DECREASE COUNTER
NOO_REM:
;CLR C
LCALL READ_KEY
LJMP NORMAL
;-----------------------------------------ADVANCED---------------------------
```

```
ADVANCED:
LCALL LCD_INIT
MOV 65H,'Y'
MOV R0,#60H
MOV R5,#4
CLEAR_LOCATIONS:
MOV @R0,#0
INC R0
DJNZ R5,CLEAR_LOCATIONS
MOV R0,#5FH
READ_AGAIN:
INC R0
LCALL READ_KEY
MOV R6,A
CJNE A,#'+',HEXAAA
MOV A,#'D'
SJMP PRINT_BUTTON
HEXAAA:CJNE A,#'-',DECCIMALL
MOV A,#'D'
SJMP PRINT_BUTTON
DECCIMALL:CJNE A,#'*',BINAAARY
MOV A,#'D'
SJMP PRINT_BUTTON
BINAAARY:CJNE A,#'/',PRINT_BUTTON
MOV A,#'^'
LCALL LCD_DATA
MOV A,#'.'
LCALL LCD_DATA
MOV A,#'5'
PRINT_BUTTON:
LCALL LCD_DATA
MOV A,R6
MOV @R0,A
CJNE A,#'=',READ_AGAIN
PACK_NUMBER:
MOV A,65H
CJNE A,'Y',ER_ROR
SJMP C_PACK
ER_ROR:
LJMP ERROR2
C_PACK:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOV R0,#60H
MOV A,@R0
```

```
LCALL CHECK_BYTE              ;CHECK THAT FIRST BYTE IS A DIGIT
CJNE R7,#0,ILLEAGAL_INPUT2      ;NOT DIGIT -->ERROR
SJMP LEAGAL_12                ;DIGIT -->START PACKING OF FIRST NUMBER
ILLEAGAL_INPUT2:
LJMP ERROR2
LEAGAL_12:
ANL A,#0FH                     ;GET BCD NUMBER FROM ASCII INPUT
MOV R1,A                      ;SAVE FIRST NUMBER IN R0
INC R0                      ;GET NEXT LOCATION OF INPUT DATA
MOV A,@R0
LCALL CHECK_BYTE              ;CHECK IF NEXT BYTE IS DIGIT OR OPERATION
CJNE R7,#0,CHECK_OPERATON2      ;IF BYTE CONTAIN ANY SIGN --->GO TO CHECK
OPERATION
ANL A,#0FH                     ;ELSE IF BYTE CONTAIN DIGIT WE CONTINUE IN PACKING
FIRST NUMBER
MOV R3,A                      ;SAVE VALUE TEMPRARY IN R3
MOV A,R1                      ;NUMBER=10*R1+R3
MOV B,#10
MUL AB                      ;10R1
ADD A,R3                      ;10*R1+R3
MOV R1,A
INC R0                      ;GET THIRD INPUR BYTE
MOV A,@R0
LCALL CHECK_BYTE
CJNE R7,#0,CHECK_OPERATON2
ANL A,#0FH                     ;ELSE IF BYTE CONTAIN DIGIT WE CONTINUE IN PACKING
FIRST NUMBER
MOV R3,A                      ;SAVE VALUE TEMPRARY IN R3
MOV A,R1                      ;NUMBER=10*R1+R3
MOV B,#10
MUL AB                      ;10R1
ADD A,R3                      ;10*R1+R3
MOV R1,A
INC R0
MOV A,@R0
LCALL CHECK_BYTE

                    ;BYTE MUST BE SIGN OF OPERATION
                                          ;IF BYTE NOT SIGN THIS MEAN THAT USER
INPUT NUMBER FROM 3 DIGITS
;-------------------------------CHECK OPERATION-------------------------

CHECK_OPERATON2:
CJNE R7,#1,NEXT_OPERATION22
LJMP EQUALITY2
NEXT_OPERATION22:CJNE R7,#2,NEXT_OPERATION122
```

```
SJMP BINARY
NEXT_OPERATION122:CJNE R7,#3,NEXT_OPERATION222
LJMP HEXA
NEXT_OPERATION222:CJNE R7,#4,NEXT_OPERATION322
LJMP DICIMAL_BCD
NEXT_OPERATION322:CJNE R7,#5,ERROR_BYTE2  ; ENTERED NUMBER IS 3 DIGIT SO CALCLATOR SHOULD PRINT ERROR
LJMP SQUARE_ROOT
ERROR_BYTE2:
LJMP ERROR2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EQUALITY2:        ;WILL PRINT ENTERED NUMBER(FIRST NUMBER)
MOV A,R1          ;FIRST NUMBER SAVED IN R1 SO WE WANT TO PRINT IT
LCALL PRINT           ;FUNTION TO PRINT ANY DATA IN ONE BYTE IN DECIMAL FORMAT
LCALL READ_KEY    ;PRESS ANY KEY TO START NEXT OPERATION
LJMP ADVANCED

BINARY:
MOV R0,#8
MOV A,R1
PRINT_BINARY:
RLC A
XCH A,R1
JC PRINT_B1
MOV A,#30H
LCALL LCD_DATA
SJMP COUNT_BINARY
PRINT_B1:
MOV A,#31H
LCALL LCD_DATA
COUNT_BINARY:
XCH A,R1
DJNZ R0,PRINT_BINARY
MOV A,#'B'
LCALL LCD_DATA
LCALL READ_KEY
LJMP ADVANCED
HEXA:
MOV R5,#2
MOV A,R1
SJMP HEXA2
HEXA_REPEAT:
MOV A,R1
SWAP A
HEXA2:
```

```
SWAP A
ANL A,#0FH
CJNE A,#10,CARRY_CHECK
MOV A,#'A'
LCALL LCD_DATA
SJMP END_HEXA
CARRY_CHECK:
JC LESS_THAN_TEN
CJNE A,#11,TWELVE
MOV A,#'B'
LCALL LCD_DATA
SJMP END_HEXA
TWELVE:CJNE A,#12,THERTEEN
MOV A,#'C'
LCALL LCD_DATA
SJMP END_HEXA
THERTEEN:CJNE A,#13,FOURTEEN
MOV A,#'D'
LCALL LCD_DATA
SJMP END_HEXA
FOURTEEN:CJNE A,#14,FIFTEEN
MOV A,#'E'
LCALL LCD_DATA
SJMP END_HEXA
FIFTEEN:CJNE A,#15,END_HEXA
MOV A,#'F'
LCALL LCD_DATA
SJMP END_HEXA
LESS_THAN_TEN:
ADD A,#30H
LCALL LCD_DATA
END_HEXA:
DJNZ R5,HEXA_REPEAT
MOV A,#'H'
LCALL LCD_DATA
LCALL READ_KEY
LJMP ADVANCED

DICIMAL_BCD:
MOV A,#0C0H
LCALL LCD_COMMAND
MOV A,60H
ANL A,#0FH
MOV R5,#4
SWAP A
PRINT_AGAAIN:
```

```
RLC A
XCH A,R0
JNC PRINT_ZZERO
MOV A,#31H
LCALL LCD_DATA
XCH A,R0
SJMP DEC_COUNT
PRINT_ZZERO:
MOV A,#30H
LCALL LCD_DATA
XCH A,R0
DEC_COUNT:
DJNZ R5,PRINT_AGAAIN
MOV A,61H
LCALL CHECK_BYTE
CJNE R7,#0,END_BCD
MOV A,61H
ANL A,#0FH
MOV R5,#4
SWAP A
PRINT_AGAAIN1:
RLC A
XCH A,R0
JNC PRINT_ZZERO1
MOV A,#31H
LCALL LCD_DATA
XCH A,R0
SJMP DEC_COUNT1
PRINT_ZZERO1:
MOV A,#30H
LCALL LCD_DATA
XCH A,R0
DEC_COUNT1:
DJNZ R5,PRINT_AGAAIN1
MOV A,62H
;;;;;;;;;;;;;;;
LCALL CHECK_BYTE
CJNE R7,#0,END_BCD
MOV A,62H
;;;;;;;;;;;;;;;;;;
ANL A,#0FH
MOV R5,#4
SWAP A
PRINT_AGAAIN2:
RLC A
XCH A,R0
```

```
JNC PRINT_ZZERO2
MOV A,#31H
LCALL LCD_DATA
XCH A,R0
SJMP DEC_COUNT2
PRINT_ZZERO2:
MOV A,#30H
LCALL LCD_DATA
XCH A,R0
DEC_COUNT2:
DJNZ R5,PRINT_AGAAIN2
END_BCD:
MOV A,#'B'
LCALL LCD_DATA
MOV A,#'C'
LCALL LCD_DATA
MOV A,#'D'
LCALL LCD_DATA
LCALL READ_KEY
LJMP ADVANCED
SQUARE_ROOT:
CJNE R1,#220,HIHIHIHHI
SJMP SQUARE_CALCULATE
HIHIHIHHI:
JC SQUARE_CALCULATE
LJMP ERROR2

SQUARE_CALCULATE:
MOV DPTR ,#SQUARE
CLR A
MOV R0,#0 ; R0 IS THE COUNTER
MOV 5AH,R1 ; SAVE THE VALUE IN R2
AGAINNN:
MOVC A,@A+DPTR
XCH A,R1
CLR C
SUBB A,R1
JNC LOOP ; IF CARRY=0 JUMP
MOV A,R0
LCALL PRINT
MOV A,5CH
CJNE A,#0 ,LLL
SJMP FINISH
LLL: MOV A,#'.'
LCALL LCD_DATA ;DISPLAY IS THE FUNCTION OF PRINT
MOV A,R0
```

```
MOV B,#2
MUL AB ;A=2*COUNTER
MOV 5BH,A
MOV B,5CH ; THE VALUE OF SUBTRACTION
MOV A,#10
MUL AB ; A= 10 * SUBTRACTION VALUE
MOV B,5BH
DIV AB
LCALL PRINT
SJMP FINISH
LOOP : INC DPTR
MOV 5CH ,A ; KEEP THE VALUE OF SUBTRACTION TO THE NEXT OP
CLR A
INC R0  ; INCREAMENT COUNTER
MOV R1,5AH ; SAVE THE VALUE OF  SUBTRACTION
SJMP AGAINNN
FINISH:
LCALL READ_KEY
LJMP ADVANCED


;----------------functions-------------------------

READ_KEY:
CHECK_1:                        ;CHECK IF A BUTTON WAS PRESSED
CLR RO1
CLR RO2
CLR RO3
CLR RO4
;DEPOUNCE
MOV A,KEYPAD_PORT
ANL A,#0FH
CJNE A,#0FH,CHECK_1
CHECK_2:                        ;MAYBE IT WAS JUST A SPIKE NOISE SO, WAIT FOR
DEBOUNCE AND CHECK AGAIN
ACALL DELAY                      ;DEPOUNCE DELAY
MOV A,KEYPAD_PORT
ANL A,#0FH
CJNE A,#0FH,CHECK_ROW
SJMP CHECK_1                     ;NOT PRESSED

CHECK_ROW:                      ;CHECK IN WHICH ROW THE PRESSED BUTTON IS
CLR RO1
SETB RO2
SETB RO3
SETB RO4
```

```
MOV A,KEYPAD_PORT
CJNE A,#01111111B,ROW_1

SETB RO1
CLR RO2
SETB RO3
SETB RO4
MOV A,KEYPAD_PORT
CJNE A,#10111111B,ROW_2

SETB RO1
SETB RO2
CLR RO3
SETB RO4
MOV A,KEYPAD_PORT
CJNE A,#11011111B,ROW_3

SETB RO1
SETB RO2
SETB RO3
CLR RO4
MOV A,KEYPAD_PORT
CJNE A,#11101111B,ROW_4

LJMP CHECK_1

ROW_1:                          ;PUT THE ADDRESS OF THE RIGHT ROW IN DPTR
MOV DPTR,#ROW1
SJMP FIND
ROW_2:
MOV DPTR,#ROW2
SJMP FIND
ROW_3:
MOV DPTR,#ROW3
SJMP FIND
ROW_4:
MOV DPTR,#ROW4
SJMP FIND

FIND:
MOV R1,#4                       ;FIND IN WHICH COLUMN THE PRESSED BUTTON IS
RRC A
JNC MATCH
INC DPTR
DJNZ R1, FIND
SJMP READ_KEY
```

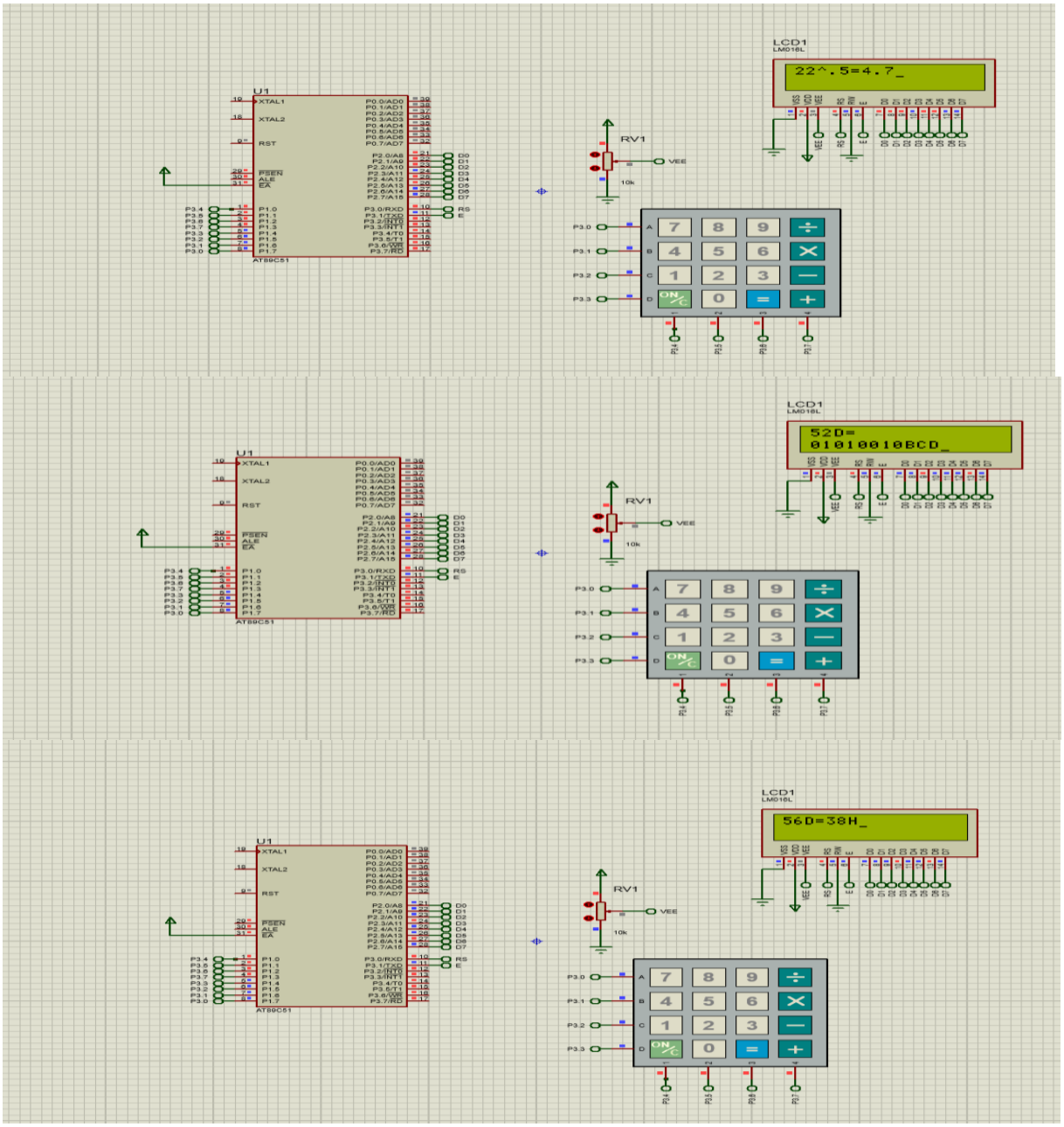
```
MATCH:                          ;TAKE THE NUBER AND WRITE IT ON THE LCD
CLR A
MOVC A,@A+DPTR
CJNE A,#'C',PPO
LCALL LCD_INIT
LJMP HERE                       ;RESET CALCULATOR
PPO:
RET
ERROR:                          ;PRINT THE ERROR MESSAGE
LCALL LCD_INIT
MOV DPTR,#ERR
AGAIN_2:
CLR A
MOVC A,@A+DPTR
JZ HOO
LCALL LCD_DATA
INC DPTR
SJMP AGAIN_2
HOO:
LCALL READ_KEY                  ;TO STOP MESSAGE ON SCREEN
LJMP NORMAL
ERROR2:                         ;PRINT THE ERROR MESSAGE
LCALL LCD_INIT
MOV DPTR,#ERR
AGAIN_22:
CLR A
MOVC A,@A+DPTR
JZ HOO2
LCALL LCD_DATA
INC DPTR
SJMP AGAIN_22
HOO2:
LCALL READ_KEY                  ;TO STOP MESSAGE ON SCREEN
LJMP ADVANCED
PRINT:
MOV B,#10           ;165---->1,5,6
DIV AB              ;GET FIRST DIGIT FROM RIGHT
MOV R4,B            ;SAVE DIGIT IN R4
MOV B,#10
DIV AB              ;GET SECOND DIGIT
MOV R5,A            ;A=LAST DIGIT ****SAVE DIGIT IN R5 TO USE IT IN SECOND
CHECK
JZ NO1              ;IF NUMBER =012  --->DON'T PRINT 0
ADD A,#30H          ;DIGIT !=0 SO PRINT DIGIT
LCALL LCD_DATA
```

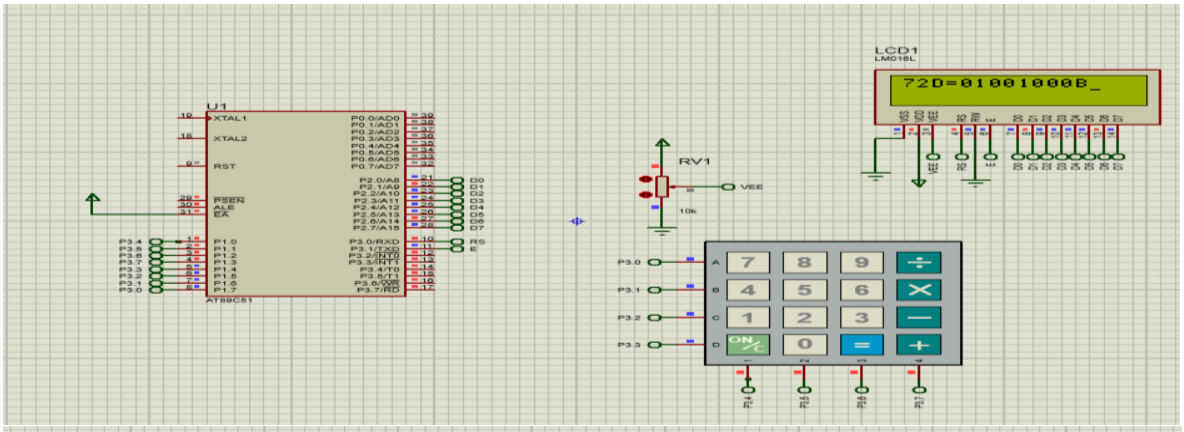
```
NO1:
MOV A,B              ;B CONTAIN SECOND DIGIT
JNZ NO2              ;IF DIGIT =0---->WE HAVE TWO CONDITIONS PRINT IT OR NO (001)-
-->WON'T PRINT   (101)WILL PRINT
MOV A,R5             ;CHECK LAST DIGIT IF =0 OR NOT
JZ NO3
MOV A,B
NO2:                 ;DIGIT !=ZERO SO PRINT IT ANY WAY   (010)--->WILL PRINT 1   (210)-
-->WILL PRINT 1
ADD A,#30H
LCALL LCD_DATA
NO3:
MOV A,R4             ;FIRST DIGIT WILL BE PRINTED IN ALL CINDITIONS
ADD A,#30H
LCALL LCD_DATA
RET

;-----------------------------------------LCD INTERFACING---------------------------------

LCD_INIT:
MOV A,#38H                      ;2 LINES AND 5*7 MATRIX
ACALL LCD_COMMAND
MOV A,#0EH                      ;DISPLAY ON, CURSOR BLINKING
ACALL LCD_COMMAND
MOV A,#01H                      ;CLEAR DISPLAY SCREEN
ACALL LCD_COMMAND

RET

LCD_COMMAND:                        ;APPLY A COMMAND TO THE LCD
MOV LCD_PORT,A
CLR Rs
SETB E
CLR E
ACALL DELAY
RET

LCD_DATA:                       ;WRITE TO THE LCD
MOV LCD_PORT,A
SETB Rs
SETB E
CLR E
ACALL DELAY
RET

DELAY:                      ;USED FOR BOTH LCD AND DEBOUNCING DELAY
```

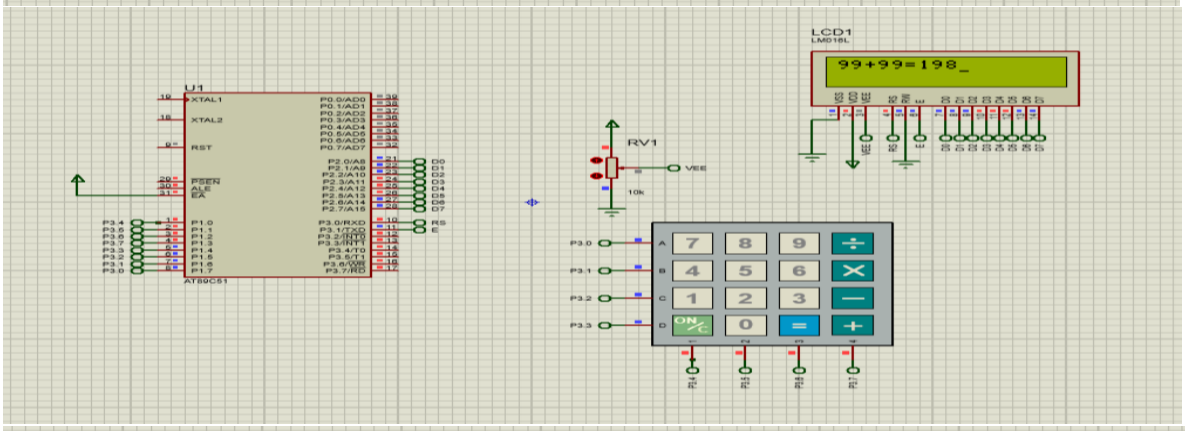
```
MOV R3,#255
LOOP2: MOV R2,#100
LOOP1: DJNZ R2,LOOP1
DJNZ R3,LOOP2
RET
;------------------------FUNCTION CHACK VALUE IN BYTE AND RETURN (0 IF DIGIT , 1 IF'
=' ,.............)IN R7
CHECK_BYTE:
MOV R7,#0            ;ASSUME THAT BYTE CONTAIN DIGIT --->IF TRUE VALUE WON'T
CHANGE
CJNE A,#'=',CHECK_ADD
MOV R7,#1
SJMP GO11
CHECK_ADD:
CJNE A,#'+',CHECK_SUB
MOV R7,#2
SJMP GO11
CHECK_SUB:
CJNE A,#'-',CHECK_MUL
MOV R7,#3
SJMP GO11
CHECK_MUL:
CJNE A,#'*',CHECK_DIV
MOV R7,#4
SJMP GO11
CHECK_DIV:
CJNE A,#'/',GO11
MOV R7,#5
GO11:
RET
;--------------------------FUNCTION TO PACK SECOND NUMBER ALSO CHECK IF INPUT IS
ERROR-------------
PACING_SECOND:
INC R0               ;LOCATION OF FIRST DIGIT IN SECOND NUMBER
MOV A,@R0            ;MOV BYTE TO ACCUMILATOR
ACALL CHECK_BYTE
MOV A,R7
JZ PACK_DIGIT1       ;IF VALUE EQUAL ZERO THEN BYTE CONTAINS DIGIT
LJMP ERROR           ;IF BYTE CONTAINS SIGN THEN IT IS AN ERROR FORMAT
PACK_DIGIT1:
MOV A,@R0
ANL A,#0FH
MOV R2,A             ;SAVE VALUE IN R2 TEMPORARY
INC R0               ;GET SECOND DIGIT IN SECOND NUMBER
MOV A,@R0
ACALL CHECK_BYTE
```

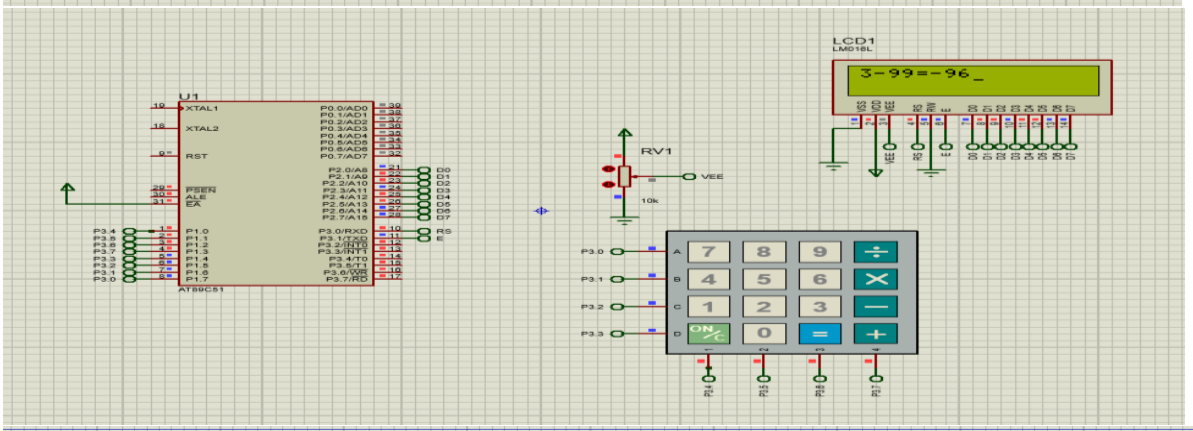
```
CJNE R7,#0,CHECK_NEXT_DIGIT  ;SECOND BYTE SHOULD BE (= OR DIGIT)
ANL A,#0FH                   ;SECOND BYTE IS A DIGIT SO WILL CONTINUE IN PACKING
MOV R3,A                     ;SAVE VALUE OF A IN R3 TEMPORARY
MOV A,R2                     ;N2=10*R2+R3
MOV B,#10
MUL AB                      ;10*R2
ADD A,R3                     ;10*R2+R3
MOV R4,A
INC R0
MOV A,@R0
ACALL CHECK_BYTE
MOV A,R7
CJNE A,#1,ERRROR
MOV A,R4
SJMP END_PACKING2           ;WE GET NUMBER SO END PACKING
ERRROR:
LJMP ERROR
                ;CHECKING IF SECOND BYTE IS NOT A DIGIT
CHECK_NEXT_DIGIT:
CJNE R7,#1,ERROR_FORMAT2    ;IF BYTE CONTAIN ANY SIGN EXCEPT (= OR DIGIT )-
-------->ERROR
MOV A,R2                    ;SECOND BYTE IS =   --->SO NUMBER IS ONLY R2
SJMP END_PACKING2
ERROR_FORMAT2:
LJMP ERROR
END_PACKING2:
RET

;----------------------FUNCTION TO PRINT ANY BYTE WITHOUT ANY ADDITIONAL
ZEROES IN LEFT----------


                    ;INITIAL MESSAGE

MESSAGE: DB 'CASIO',0
                            ;KEYBOARD BUTTONS
ROW1: DB '1','2','3','/'
ROW2: DB '4','5','6','*'
ROW3: DB '7','8','9','-'
ROW4: DB 'C','0','=','+'

                            ;ERROR MESSAGE
ERR: DB 'ERROR',0
SQUARE : DB 1,4,9,16,25,36 ,49,64,81,100,121,144,169,196,225

END
```

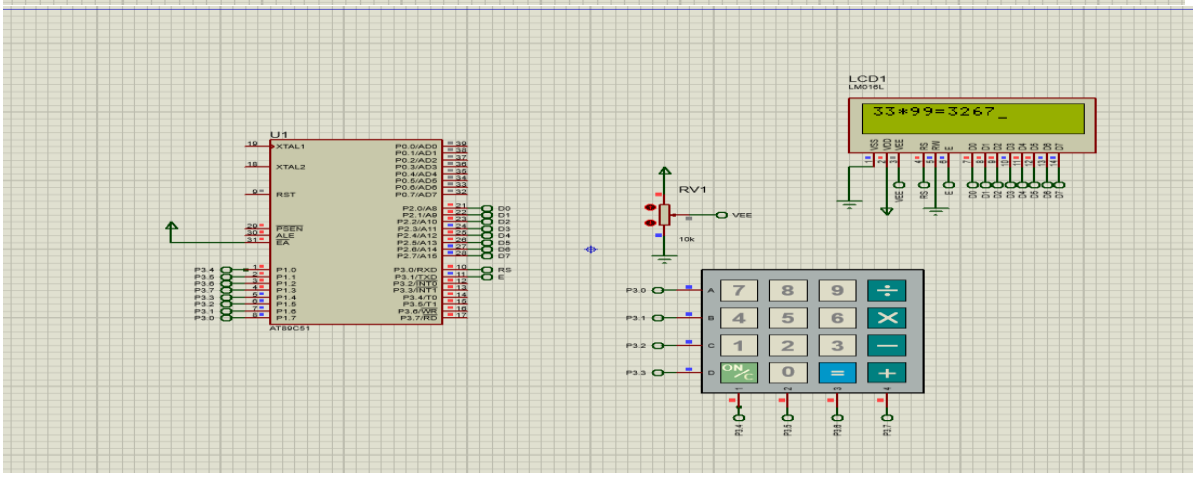
Proteus simulation



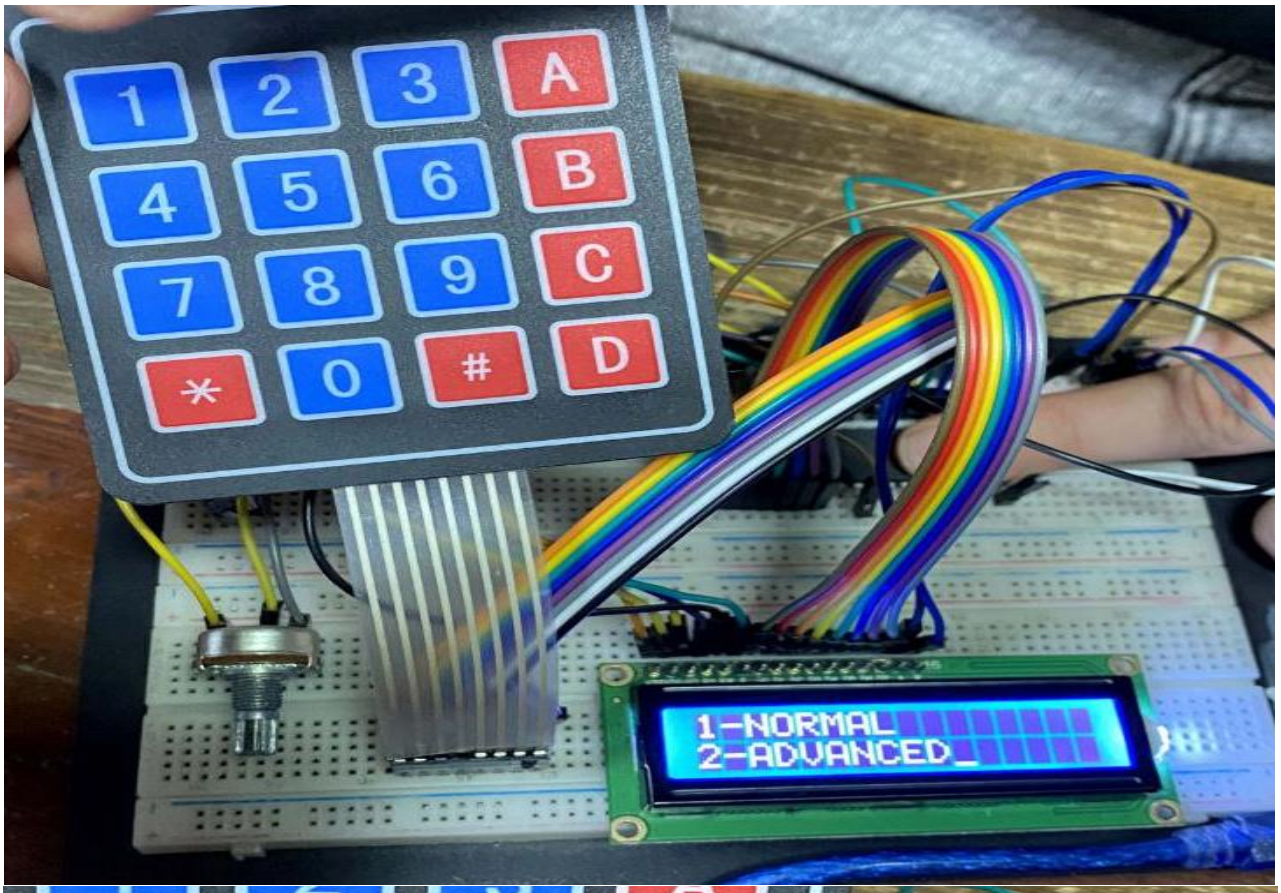

LCD1
LMD16L
72D=01001000B_
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
AT89C51
RV1
VEE
10k
7 8 9 ÷
4 5 6 ×
1 2 3 —
ON/C 0 = +

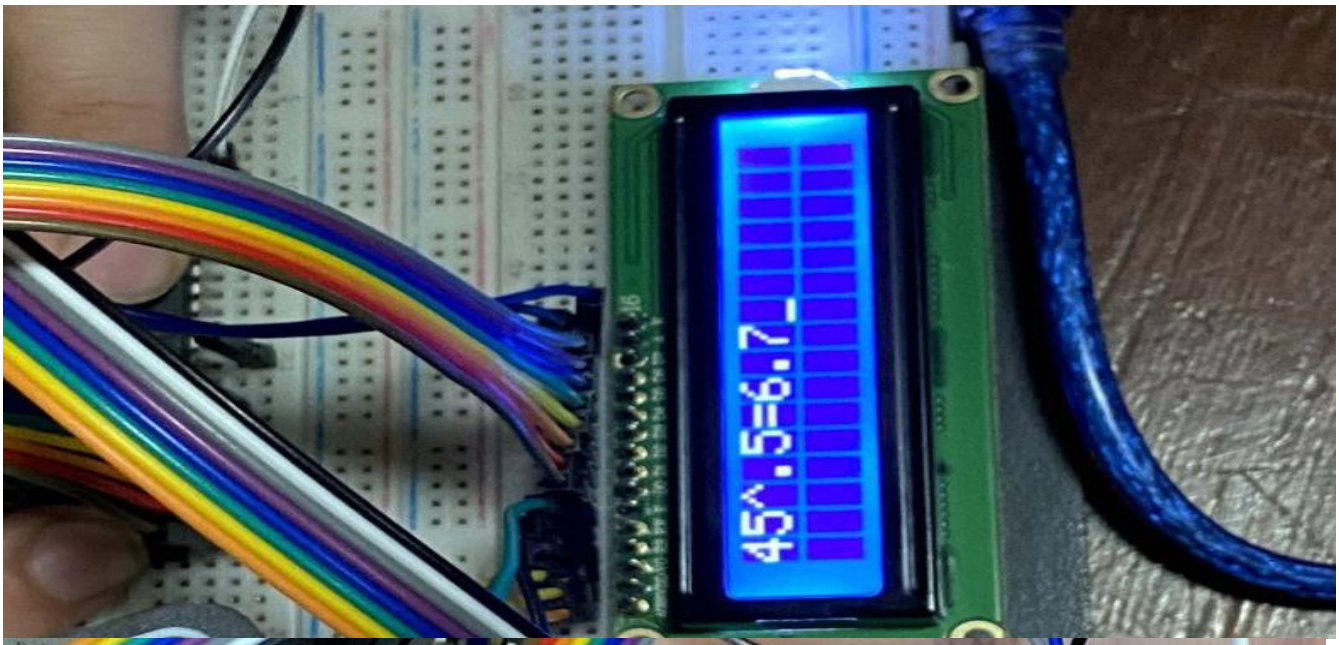

LCD1
LMD16L
99+99=198_
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
AT89C51
RV1
VEE
10k
7 8 9 ÷
4 5 6 ×
1 2 3 —
ON/C 0 = +

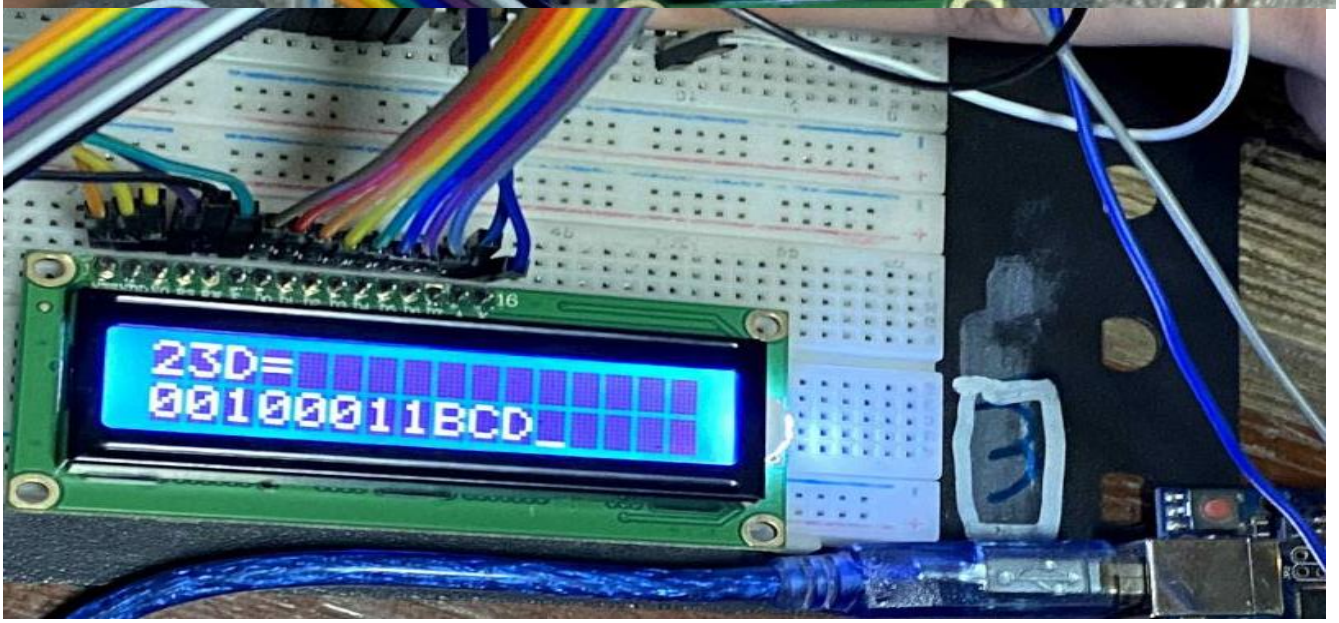

LCD1
LMD16L
3-99=-96_
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
AT89C51
RV1
VEE
10k
7 8 9 ÷
4 5 6 ×
1 2 3 —
ON/C 0 = +


LCD1
LMD16L
33*99=3267_
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
AT89C51
RV1
VEE
10k
7 8 9 ÷
4 5 6 ×
1 2 3 —
ON/C 0 = +

Hardware




45^.5=6.7_


23D=
00100011BCD_