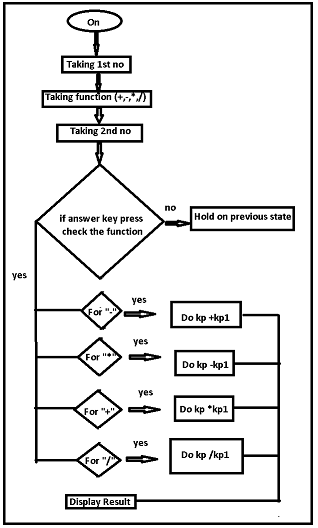
**PROJECT TITLE**: A SIMPLE CALCULATOR

**AIM**: Create a Simple calculator for doing operation on one bit numbers.

**THEORY**:



**Features of this calculator**

* You can give any single digit input from 0 to 9.
* You can press ‘*ON/C*‘ button at any time to reset the calculator
* 4 functions are implemented i-e addition, subtraction, multiplication and division.

**Methodology:**

In the main code, keypad and LCD are initialized at first. Then the code waits for the first number from the keypad. After getting this number LCD screen is cleared. And this number is displayed on the LCD. After that, code waits for the function keyfrom the user. After getting the function key, code waits for the second number[1] and then the equal sign. After getting the equal sign, according to the desired function the result is calculated and displayed on the screen.

**ASM CODE:-**

LIST p=16f877a

#include "P16f877a.inc"

; 0x36,0x35 are used for storing the numbers respectively

; 0x37 is used to store the condition

; 0x38, 0x39 are used to store the loop variables

; 0x40 is used for storing the quotient

; 0x41 ,0x42 are used for converting a hex to decimal (2 digit decimal)

;----------------------------------------------------------------------------------------------------------------

; configuring as input and output ports

bsf STATUS, RP0 ; select bank 1

;FOR KEYPAD

movlw b'11110000'

movwf TRISB

;FOR LCD CONFIG

movlw b'11111000'

movwf TRISD

;FOR LCD OUTPUT

movlw 0x00

movwf TRISC

bcf STATUS, RP0 ; select bank 0

;-----------------------------------------------------------------------------------------------------------------

;Main-------------------------------------------------------------------------------------------------

begin:

call check\_keypad

goto begin

;-----------------------------------------------------------------------------------------------------

;The part which scans keypad-----------------------------------------------------------------------------

check\_keypad ; This routine will scan the keypad for any key presses.

bsf PORTB, 0 ; scan the 1st column of keys

;========================================================

btfsc PORTB, 4 ; has the ON/OFF key been pressed? if yes then

call ON

btfsc PORTB, 5 ; has the 1 key been pressed? if yes then

call ONE

btfsc PORTB, 6 ; has the 4 key been pressed? if yes then

call FOUR

btfsc PORTB, 7 ; has the 7 key been pressed? if yes then

call SEVEN

bcf PORTB, 0 ; finished column 1

bsf PORTB, 1 ; scan the 2nd column of keys

;=========================================================

btfsc PORTB, 4 ; has the 0 key been pressed? if yes then

CALL ZERO

btfsc PORTB, 5 ; has the 2 key been pressed? if yes then

CALL TWO

btfsc PORTB, 6 ; has the 5 key been pressed? if yes then

CALL FIVE

btfsc PORTB, 7 ; has the 8 key been pressed? if yes then

CALL EIGHT

bcf PORTB, 1 ; finished column 2

bsf PORTB, 2 ; scan the 3rd column of keys

;=====================================================

btfsc PORTB, 4 ; has the = key been pressed? if yes then

CALL EQUAL

btfsc PORTB, 5 ; has the 3 key been pressed? if yes then

CALL THREE

btfsc PORTB, 6 ; has the 6 key been pressed? if yes then

CALL SIX

btfsc PORTB, 7 ; has the 9 key been pressed? if yes then

CALL NINE

bcf PORTB, 2 ; finished column 3

bsf PORTB, 3 ; scan the 4th column of keys

;========================================================

btfsc PORTB, 4 ; has the + key been pressed? if yes then

call PLUS

btfsc PORTB, 5 ; has the - key been pressed? if yes then

call MINUS

btfsc PORTB, 6 ; has the x key been pressed? if yes then

call MULT

btfsc PORTB, 7 ; has the / key been pressed? if yes then

call DIV

bcf PORTB, 3 ; finished column 4

return ; and now return to the main routine

;----------------------------------------------------------------------------------------------------------------

;----------------------------------------------------------------------------------------------------------------

;KEY FUNTIONS

ZERO:

movlw 0x00

movwf 0x35

movlw '0'

call display\_digit

return

ONE:

movlw 0x01

movwf 0x35

movlw '1'

call display\_digit

return

TWO:

movlw 0x02

movwf 0x35

movlw '2'

call display\_digit

return

THREE:

movlw 0x03

movwf 0x35

movlw '3'

call display\_digit

return

FOUR:

movlw 0x04

movwf 0x35

movlw '4'

call display\_digit

return

FIVE:

movlw 0x05

movwf 0x35

movlw '5'

call display\_digit

return

SIX:

movlw 0x06

movwf 0x35

movlw '6'

call display\_digit

return

SEVEN:

movlw 0x07

movwf 0x35

movlw '7'

call display\_digit

return

EIGHT:

movlw 0x08

movwf 0x35

movlw '8'

call display\_digit

return

NINE:

movlw 0x09

movwf 0x35

movlw '9'

call display\_digit

return

;-------------------------------------------------------

PLUS:

;storing the first operand to 0x36 so that

;the second operand would come to 0x35

movf 0x35,w

movwf 0x36

;sending the value 00 for addition operation

;0x37 is the varaible that stores the conditions

movlw 0x00

movwf 0x37

movlw '+'

call display\_digit

return

;--------------------------------------------------------

;---------------------------------------------------------

MINUS:

movf 0x35,w

movwf 0x36

movlw 0x01

movwf 0x37

movlw '-'

call display\_digit

return

;----------------------------------------------------------

;----------------------------------------------------------

MULT:

movf 0x35,w

movwf 0x36

movlw 0x02

movwf 0x37

movlw '\*'

call display\_digit

return

;-----------------------------------------------------------

;-----------------------------------------------------------

DIV:

movf 0x35,w

movwf 0x36

movlw 0x03

movwf 0x37

movlw '/'

call display\_digit

return

;------------------------------------------------------------------

EQUAL:

movlw 0x3D

call display\_digit

;CALCULATION PART-------------------------------------------------------

BTFSS 0X37,1

GOTO TCOND0 ;0X

GOTO TCOND1 ;1X

TCOND0:

BTFSS 0X37,0

GOTO COND00

GOTO COND01

TCOND1:

BTFSS 0X37,0

GOTO COND10

GOTO COND11

;Addition

COND00:

MOVF 0X36,W

ADDWF 0X35,W

addlw 0x30

call display\_digit

RETURN

;Substration

COND01:

SUBSTRACTION:

MOVF 0X35,W

SUBWF 0X36,W

addlw 0x30

call display\_digit

RETURN

;Multiplication

COND10:

MOVLW 0X00

LOOP2:

ADDWF 0X36,W

DECF 0X35,F

BTFSS STATUS,Z

GOTO LOOP2

addlw 0x30

call display\_digit

RETURN

;Division 0x36/0x35

;0x40 will serve as quotient

COND11:

clrf 0x40

MOVF 0X35,W

LP1:

incf 0x40,f

SUBWF 0X36,F

BTFSC STATUS,C

GOTO LP1

;initially it is incremented before substration

decf 0x40

movf 0x40,w

addlw 0x30

call display\_digit

RETURN

;END OF CALCULATION-----------------------------------------------

return

ON:

movlw 0x01

call DISPLAY

clrf 0x35

clrf 0x36

clrf 0x37

return

;----------------------------------------------------------------------------------------------------------------

;Display the digit in the LCD------------------------------------------------------------------------------

; LCD INITIALIZATION

;If

;RS=0 Instruction command Code register is selected, allowing user to send command

;RS=1 Data register is selected allowing to send data that has to be displayed.

;R\W=0 Reading

;R\W=1 Writing

;E- Enable

;The enable Pin is used by the LCD to latch information at its data pins. When data is supplied to data pins,

;a high to low pulse must be applied to this pin in order for the LCD to latch the data present in the data pins.

;E should Toggle

;Data Mode: RS=1, R\W=0, E=1\0

display\_digit:

BSF PORTD,0; CONTROL SIGNAL TO RS

BCF PORTD,1; CONTROL SIGNAL TO R/W

BSF PORTD,2; CONTROL SIGNAL TO 'E'

;Here the value already stored in the W reg is send to DISPLAY

call DISPLAY

;initialises the display

MOVLW 0X38 ; setting up LCD for 8 bit interface, 2 lines

CALL DISPLAY

MOVLW 0X0E ; dont blink the cursor

CALL DISPLAY

BSF PORTD,0

RETURN

DISPLAY:

MOVWF PORTC

BCF PORTD,2

CALL DELAY1

BSF PORTD,2

CALL DELAY1

BCF PORTD,0

RETURN

DELAY1:

MOVLW D'13' ;A VERY SMALL DELAY

MOVWF 0X38

MOVLW D'251'

MOVWF 0X39

LOOP: DECFSZ 0X39

GOTO LOOP

DECFSZ 0X38

GOTO LOOP

RETURN

RETURN

return

;----------------------------------------------------------------------------------------------------------------

end

**PROTEUS SIMULATION:-**

