Arithmetic Operations with Call Operations

Abstract:

The main focus of this assignment is on the concepts of arithmetic operation implementation in pic18f452 by writing code in assembly language. It also focuses on how subroutines are written in assembly and how to invoke them using call operation. It also addresses the concept of different type of overflows in multiplication, carry flags in addition which occurs when a calculation exceeds the maximum storage capacity of a system's data type, which can ultimately lead to incorrect results. It also addresses the problem of division of any number by zero.

Objective:

- Demonstrate basic arithmetic operations (addition, subtraction, multiplication, division) for unsigned numbers in Assembly programming.
- Understanding of call instruction for invoking subroutine.
- Efficient Code Writing and improvement in debugging skills.

Question # 1

- Addition and Subtraction Using CALL

Write an Assembly program to perform addition and subtraction of two unsigned numbers by making separate subroutines.

- CODE:

Explanation:

This code implements an assembly program for the PIC18F452 microcontroller that performs addition and subtraction of two 8-bit inputs provided via PORTB and PORTC. The results are displayed on PORTD. If a carry occurs during addition, the first bit of PORTE is set to 1 as an indicator. The program loops infinitely, displaying the results with a delay between operations.

```
;in this code there are two sub routeins one for addition and one for subtraction

;after taking 8 bit input from user, addition and subtraction are performed respectively

;in addition if there is a carry generated, it is represented by setting porte of pic to 1



LIST P=18F452

#include <p18f452.inc>



result EQU 0x32 ;to store the end_result of addition and subtraction
```

```
ORG 0x00 ;sssigning starting memory location

GOTO START ; to jump onto the main part of the code


;---------------------Addition--------------------------------


;subroutein to perform addition on input taken from user

ADD_1:

    MOVF PORTB, W      ;load 8-bit input 1 from port b to working register

    ADDWF PORTC, W     ;add 8-bit input 2 from PORTC to Working register(having input 1)

    MOVWF result       ;store the result,by moving value stored in working register after addition
to file location named as result


    ;to check if there was a carry in the addition

    ;if bit in staus register for carry flag is set than it will skip next instruction of jumping onto
label no_carry

    BTFSS STATUS, C    ;to check if the carry flag is set, ifit is set it will skip next instruction and
will go on instruction of setting porte to 1

    GOTO NO_CARRY      ;if there is no carry, jump to NO_CARRY


    ;setting porte first bit to 1 to indicate the carry flag

    BSF PORTE, 0       ;bit set file here sets the porte first bit to 1, only first bit is set bcz i am only
using first bit for output


;if there is no carry generated than it will simply return to the program insruction from where
specificed subrouteinis called

NO_CARRY:

    RETURN


;---------------------Subtraction--------------------------------


;subroutein to perform subtraction on input taken from user
```

```
SUB_1:

   MOVF PORTC, W      ;load 8-bit input 1 from port b to working register

   SUBWF PORTB, W     ;sbtract 8-bit input 2 (from PORTC) from Working register(having input
1)

   MOVWF result       ;store the result,by moving value stored in working register after
subtraction to file location named as result

   RETURN



;---------------------Delay_Function---------------------------------



;subroutein to give a dealy after displaying one result on output portd

DELAY:

   MOVLW 0xFF            ;loading delay counter to working register

   MOVWF 0x20            ;soring counter at memory location of 0x20

DelayLoop1:

   MOVLW 0xFF            ;loading another delay counter for inner loop into working register

   MOVWF 0x21            ;storing inner loop counter at memory location of 0x21

DelayLoop2:

   DECFSZ 0x21, F        ;decrementing inner counter untill it becomes equal to zero and storing
back at its own location in memory

   GOTO DelayLoop2       ;jumping back to inner loop untill the inner loop counter becomes
zero

   DECFSZ 0x20, F        ;decrementing outer counter untill it becomes equal to zero and storing
back at its own location in memory

   GOTO DelayLoop1       ;jumping back to outer loop untill the outer loop counter becomes
zero

   RETURN



;-------------------Main_Program------------------------------------

START:

   ;configure portb as input for input 1

   MOVLW 0xFF        ;moving 1 to working register
```

```
    MOVWF TRISB       ;moving value from working register to file location basically to set portb
as input port


    ;configure portc as input for input 2

    MOVLW 0xFF        ;moving 1 to working register

    MOVWF TRISC       ;moving value from working register to file location basically to set portc
as input port


    ;configure portd as output for resultant output

    CLRF PORTD        ;clearing portd

    MOVLW 0x00        ;moving 0 to working register

    MOVWF TRISD       ;moving value from working register to file location basically to set portd
as output port


  ;configure porte as output for resultant output

    CLRF PORTE        ;clearing porte

    MOVLW 0x00        ;moving 0 to working register

    MOVWF TRISE       ;moving value from working register to file location basically to set porte
as output port


  ;configure porta as output for resultant output

    CLRF PORTA        ;clearing porta

    MOVLW 0x00        ;moving 0 to working register

    MOVWF TRISA       ;moving value from working register to file location basically to set porta
as output port


MAIN_LOOP:

 ;-------------------calling addition subroutein-----------------------------


    CALL ADD_1        ;calling subrouteine add_1 to perform addition

    MOVF result, W    ;loading result of addition stored in file location result into a working
register
```

```
    MOVWF PORTD        ;loading output of addition from working register to portd for output
display purpose

    CALL DELAY        ;waiting for sometime to differ between different results by creating a delay
of our own choice by calling delay subroutein



;--------------------calling subtraction subroutein------------------------------

    CALL SUB_1        ;calling subrouteine sub_1 to perform subtraction

    MOVF result, W     ;loading result of subtraction stored in file location result into a working
register

    MOVWF PORTD        ;loading output of subtraction from working register to portd for output
display purpose

    CALL DELAY        ;waiting for sometime to differ between different results by creating a delay
of our own choice by calling delay subroutein



    GOTO MAIN_LOOP        ;to repeat process infinetely



    END ; ending the program
```

- <u>Output:</u>

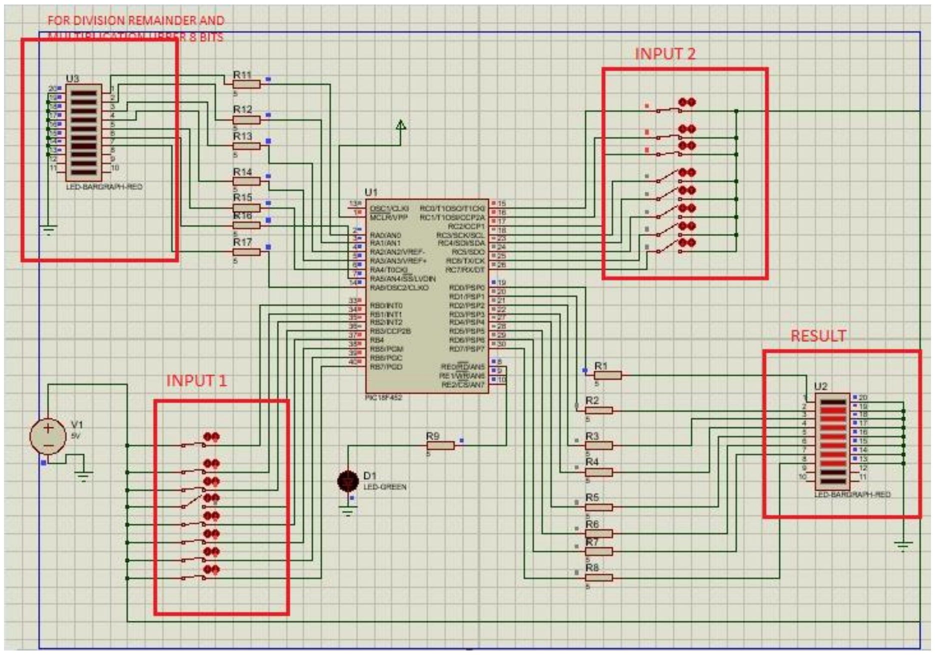

Figure 1: Proteus Circuit to show output of each arithmetic operation

<u>Output after addition between two inputs generating no carry flag:</u>

Input 1= 11110111 in binary and 247 in decimal

Input 2= 00000111 in binary and 7 in decimal

Addition in binary = 11110111 + 00000111 = 11111110 in binary and in decimal 254

Addition in decimal = 247+7 = 254

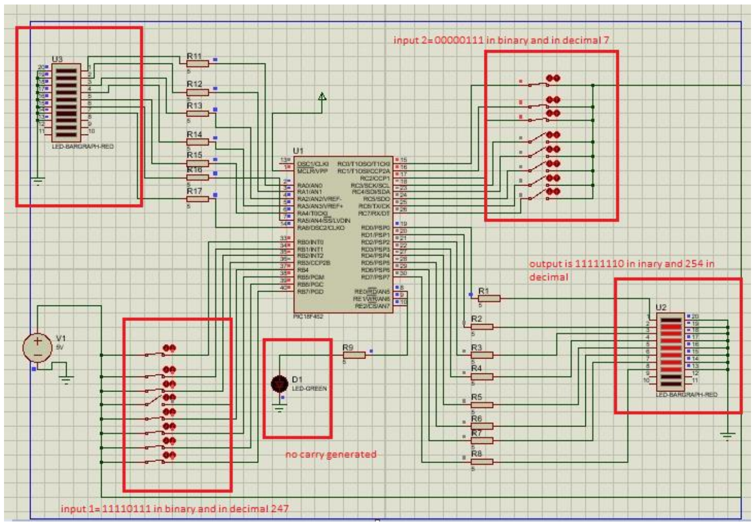
input 2=00000111 in binary and in decimal 7
output is 11111110 in inary and 254 in decimal
no carry generated
input 1=11110111 in binary and in decimal 247

Figure 2: no carry flag is generated when input 1 is added with input 2 of 8-bit number because the resultant was in the range of 8 bit binary (details of inputs and output are highlighted in attached figure)

Output after addition between two inputs generating with a carry flag:

Input 1= 11110101 in binary and 245 in decimal

Input 2= 10000111 in binary and 135 in decimal

Addition in binary = 11110101 + 10000111 = 11111100 in binary with 1 in carry

Addition in decimal = 135+245 = 380

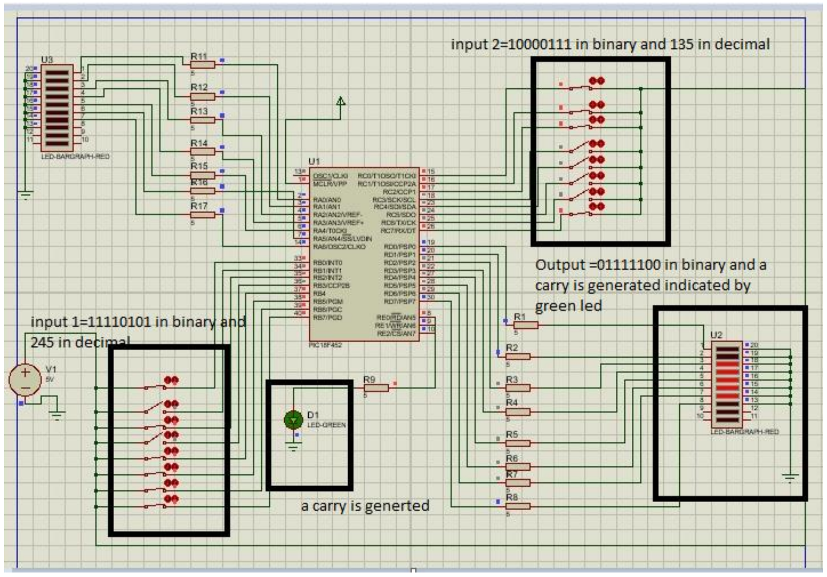
input 2=10000111 in binary and 135 in decimal
Output =01111100 in binary and a carry is generated indicated by green led
input 1=11110101 in binary and 245 in decimal
a carry is generted

Figure 3: carry flag is generated (which is shown by green led) when input 1 is added with input 2 of 8-bit number because the resultant was out of the range of 8 bit binary (details of inputs and output are highlighted in attached figure)

Output after subtraction between two inputs:

Input 1= 11110111 in binary and 247 in decimal

Input 2= 00000111 in binary and 7 in decimal

Subtraction in binary = 11110111 - 00000111 = 11110000 in binary and in decimal 240

Subtraction in decimal = 247+7 = 240

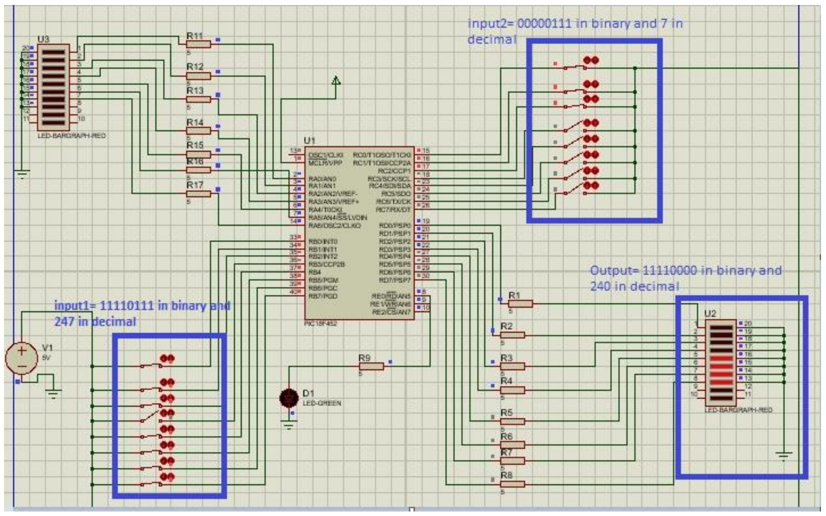
input2= 00000111 in binary and 7 in decimal
Output= 11110000 in binary and 240 in decimal
input1= 11110111 in binary and 247 in decimal
U1
U2
U3
PIC18F452
LED-BARGRAPH-RED
LED-GREEN
D1
V1
R1
R2
R3
R4
R5
R6
R7
R8
R9
R11
R12
R13
R14
R15
R16
R17

Figure 4: when input 1 is subtracted from input 2 of 8-bit number and result is shown with led bar-graph (details of inputs and output are highlighted in attached figure)

Question # 2

- Multiplication and Division Using CALL

Write an Assembly program to perform Multiplication and Division of two unsigned numbers by making separate subroutines.

- CODE:

Explanation:

This code uses the PIC18F452 microcontroller to perform multiplication and division of two 8-bit inputs from PORTB and PORTC. For multiplication, lower 8 bits are displayed on PORTD and upper bits on PORTA, handling overflow. For division, the quotient is shown on PORTD, the remainder on PORTA, and PORTE is used to indicate division by zero. A delay routine separates result displays.

```
;in this code there are two sub routeins one for multiplication and one for division

;after taking 8 bit input from user, multiplication and division are performed respectively

;in division if divisor is zero than led at porte is turned on, it is represented by setting porte of pic to 1

;porte first bit is set to 1 to indicate that division by zero is not possible

;in division quotient and remainder are shown at different ports

;quotient is displayed on portd while remainder is displayed on porta
```

```
;in case of multiplication there is chances of number exceeding 8 bits

;exceeding of bits is possible because when two 8 bits number are multiplied their range increases

;to show correct result of multiplication lower 8 bits are displayed on portd

;the upper bits of multiplication is displayed over porta


LIST P=18F452

#include <p18f452.inc>

temp_num1  EQU 0x30  ;memory location reserved for temporary storage of input 1 in multiplication

temp_num2  EQU 0x31  ;memory location reserved for temporary storage of input 2 in multiplication

result     EQU 0x32  ;to store lower 8 bits of result of multiplication and in case of division to store quotient

result2    EQU 0x33  ;to store upper 8 bits of result of multiplication and in case of division to store remainder but because of only 7 pins available we are omitting one bit

carry      EQU 0x34  ;temporary carry in multiplication to help in code to check if resultant of multiplication has exceeded 8 bits



ORG 0x00 ;sssigning starting memory location

GOTO START ;to jump onto the main part of the code


;--------------------Division---------------------------------

;subroutein for division of two numbers

DIV_1:              ;division by repeated subtractions

    CLRF result       ;clearing result to avoid garbage values

    MOVF PORTB, W      ;loading dividend from portb to working register

    MOVWF 0x31        ;storing dividend temporarily at memory location 0x31

    MOVF PORTC, W      ;loading divisor from portc into working register
    BTFSS PORTC, 0     ;to check if bit in portc are zero, if it is not zero it will skip next instruction
else it will jump to div_zero

    GOTO DIV_ZERO      ;on having divisor equals to zero jumping at label div_zero


DIV_Quotient:          ;for calculation of division quotient

    MOVF 0x31,W        ;loading dividend into working register

    MOVWF 0x32         ;storing dividend into memory location 0x32 for keeping track of
remainder

    MOVF PORTC, W      ;loading portc value basically divisor into working register

    SUBWF 0x31, F      ;subtracting divisor from dividend and storing back into same location

    BTFSS STATUS, C    ;to check if on dividing the remainder is negative if it is negative jumping
on the label div_remainder else skipping the jump on specified label

    GOTO DIV_Remainder ;ending division if on subtraction dividend beocme negative


    INCF result, F     ;after each subtraction quotient is incremented to get correct disvison
quotient

    GOTO DIV_Quotient  ;jumping back to label untill division is completely done


DIV_Remainder:         ;for calculation of division remainder

    MOVF 0x32, W       ;loading remainder into working register

    ANDLW 0x7F         ;omitting 1 most significant bit because the port available at a are only 7

    MOVWF result2      ;storing remainder into result2 to display on porte

    RETURN             ;returning to instruction from where call to subroutein is made


DIV_ZERO:

    BSF PORTE, 0       ;if divisor is zero led at porte is turned on because first bit of porte is set to
1

    RETURN             ;returning to instruction from where call to subroutein is made


;----------------------------MULTIPLICATION-------------------------

MUL_1:
    CLRF    result         ;clearing result to avoid garbage values
    CLRF    result2        ;clearing result2 to avoid garbage values
    CLRF    carry          ;clearing carry to avoid garbage values


    MOVF    PORTB, W       ;loading num1 from portb into working register
    MOVWF   temp_num1      ;storing num1 in temporary memory location temp_num1
    MOVF    PORTC, W       ;loading num2 from portb into working register
    MOVWF   temp_num2      ;storing num2 in temporary memory location temp_num2


REPEAT_ADD:
    MOVF    result, W      ;loading current result basically zero to working register
    ADDWF   temp_num1, W   ;adding number 1 to working register and storing in working
register
    MOVWF   result         ;storing the new result after addition into same location for repeated
addition


    ;to check for overflow (carry flag) and handle upper 8 bits
    MOVF    result, W      ;loading lower 8 bits to working register
    CPFSLT  result2        ;value in working register is compared with result2 to check if it is smaller
or not if smaller next instruction is skipped
    BTFSS   STATUS, C      ;if value in working register is not smaller than it is checked if carry flag
is set or not
    MOVWF   carry          ;if there is carry store the carry in the memory location intializazed with
name of carry
    MOVF    carry, W       ;moving the value at memory locaton carry into working register
    ADDWF   result2, F     ;adding the value stored in memory location at carry with working
register and storing at same location


    DECFSZ  temp_num2, F   ;decrementing number 2 to keep check on how many time we have
to perform addition
    GOTO    REPEAT_ADD     ;repeat addition untill the number 2 is equal to zero
```

```
MUL_DONE:
 RETURN            ;returning to instruction from where call to subroutein is made


;---------------------Delay_Function--------------------------------


;subroutein to give a dealy after displaying one result on output portd
DELAY:
    MOVLW 0xFF            ;loading delay counter to working register
    MOVWF 0x20            ;soring counter at memory location of 0x20
DelayLoop1:
    MOVLW 0xFF            ;loading another delay counter for inner loop into working register
    MOVWF 0x21            ;storing inner loop counter at memory location of 0x21
DelayLoop2:
    DECFSZ 0x21, F        ;decrementing inner counter untill it becomes equal to zero and storing
back at its own location in memory
    GOTO DelayLoop2       ;jumping back to inner loop untill the inner loop counter becomes
zero
    DECFSZ 0x20, F        ;decrementing outer counter untill it becomes equal to zero and storing
back at its own location in memory
    GOTO DelayLoop1       ;jumping back to outer loop untill the outer loop counter becomes
zero
    RETURN


;-------------------Main_Program------------------------------------
START:
    ;configure portb as input for input 1
    MOVLW 0xFF        ;moving 1 to working register
    MOVWF TRISB       ;moving value from working register to file location basically to set portb
as input port


    ;configure portc as input for input 2
    MOVLW 0xFF         ;moving 1 to working register

    MOVWF TRISC        ;moving value from working register to file location basically to set portc
as input port



    ;configure portd as output for resultant output

    CLRF PORTD         ;clearing portd

    MOVLW 0x00         ;moving 0 to working register

    MOVWF TRISD        ;moving value from working register to file location basically to set portd
as output port

  ;configure porte as output for resultant output

    CLRF PORTE         ;clearing porte

    MOVLW 0x00         ;moving 0 to working register

    MOVWF TRISE        ;moving value from working register to file location basically to set porte
as output port

  ;configure porta as output for resultant output

    CLRF PORTA         ;clearing porta

    MOVLW 0x00         ;moving 0 to working register

    MOVWF TRISA        ;moving value from working register to file location basically to set porta
as output port

MAIN_LOOP:

 ;--------------------calling multiplication subroutein------------------------------

    CALL MUL_1         ;calling subrouteine mul_1 to perform multiplication

    MOVF result, W     ;loading lower 8 bits of multiplication stored in file location result into a
working register

    MOVWF PORTD        ;loading lower 8 bits of multiplication from working register to portd for
output display purpose

    MOVF result2, W    ;loading upper 8 bits of multiplication stored in file location result into a
working register

    MOVWF PORTA        ;loading upper 8 bits of multiplication from working register to porta for
output display purpose

    CALL DELAY         ;waiting for sometime to differ between different results by creating a delay
of our own choice by calling delay subroutein

;--------------------calling divison subroutein------------------------------
```

```
    CALL DIV_1         ;calling subrouteine div_1 to perform divison

    MOVF result, W     ;loading quotient of divison stored in file location result into a working
register

    MOVWF PORTD        ;loading quotient of divison from working register to portd for output
display purpose

    MOVF result2, W    ;loading reaminder of divison stored in file location result into a working
register

    MOVWF PORTA        ;loading reaminder of divison from working register to porta for output
display purpose

    CALL DELAY         ;waiting for sometime to differ between different results by creating a delay
of our own choice by calling delay subroutein

    GOTO MAIN_LOOP         ;to repeat process infinetely

    END ; ending the program
```

- <u>Output:</u>

<u>Output after multiplication between two inputs within 8-bit range:</u>

Input 1= 00000011 in binary and 3 in decimal

Input 2= 00000011 in binary and 3 in decimal

Multiplication in binary = 00000011 * 00000011 = 00001001 in binary and in decimal 9

Multiplication in decimal = 3*3 = 9

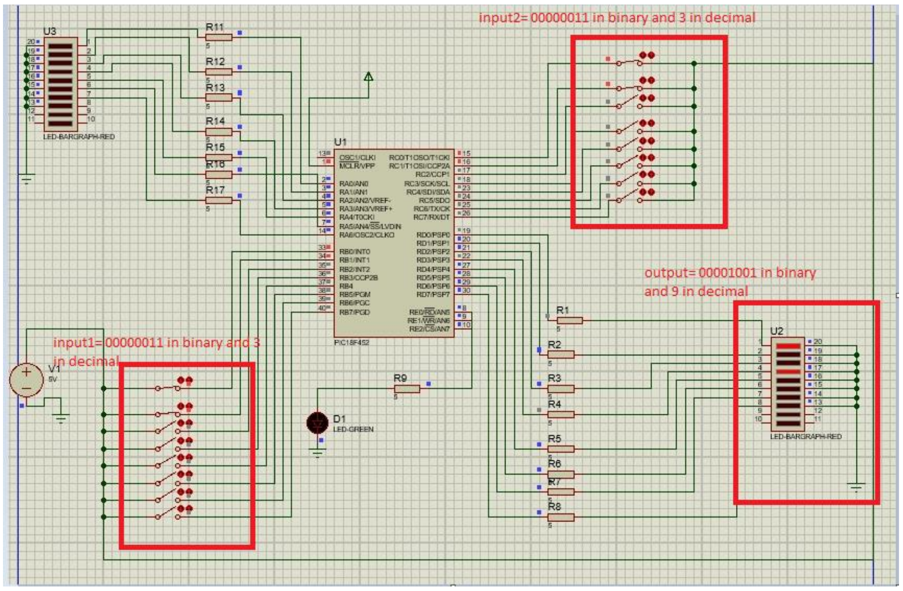
input2= 00000011 in binary and 3 in decimal
output= 00001001 in binary and 9 in decimal
input1= 00000011 in binary and 3 in decimal
U1
PIC18F452
U2
U3
LED-BARGRAPH-RED
D1
LED-GREEN
V1
R1
R2
R3
R4
R5
R6
R7
R8
R9
R11
R12
R13
R14
R15
R16
R17

Figure 5: when input 1 is multiplied with input 2 of 8-bit number and the resultant was in the range of 8 bit binary (details of inputs and output are highlighted in attached figure)

<u>Output after multiplication between two inputs exceeding 8-bit range:</u>

Input 1= 01110111 in binary and 119 in decimal

Input 2= 00000111 in binary and 7 in decimal

Multiplication in binary = 01110111 * 00000111 = 0000011010000001 in binary and in decimal 883

Multiplication in decimal = 119*7 = 883

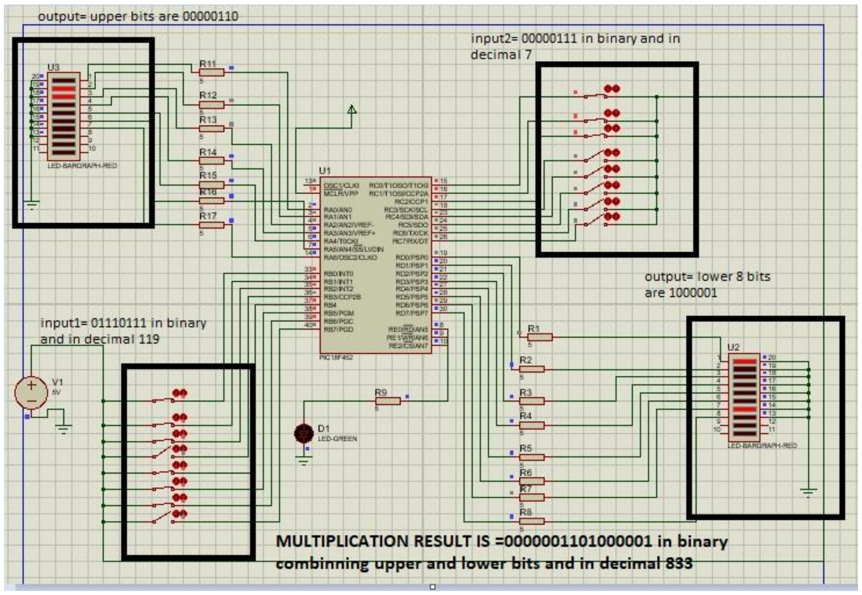
output= upper bits are 00000110
input2= 00000111 in binary and in decimal 7
output= lower 8 bits are 1000001
input1= 01110111 in binary and in decimal 119
MULTIPLICATION RESULT IS =0000001101000001 in binary combinning upper and lower bits and in decimal 883
U1
U2
U3
PIC18F452
LED-BARGRAPH-RED
D1
LED-GREEN
V1

Figure 6: when input 1 is multiplied with input 2 of 8-bit number and the resultant was not in the range of 8 bit binary so lower bits are represented at portd and upper bits at porta (details of inputs and output are highlighted in attached figure)

Output after division between two inputs but divisor is zero:

Input 1= 00000011 in binary and 3 in decimal

Input 2= 00000000 in binary and 0 in decimal

Division in binary = 00000011 / 00000000 = undefined represented by led turned on that no division is possible

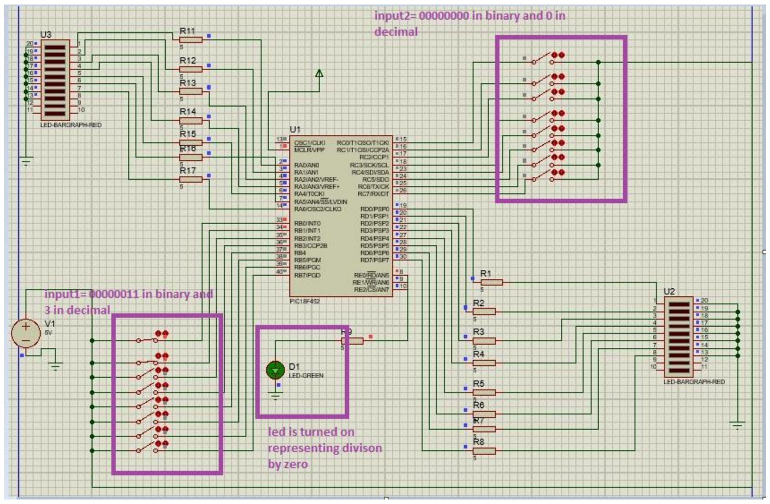
input2= 00000000 in binary and 0 in decimal
input1= 00000011 in binary and 3 in decimal
led is turned on representing divison by zero
U1
U2
U3
PIC18F452
D1
LED-GREEN
V1

Figure 7: when input 1 is divided by input 2 of 8-bit number but input 2 is zero leading to no division possibility led is turned to show division is not possible (details of inputs and output are highlighted in attached figure)

Output after division between two inputs but divisor is not zero and there is a remainder:

Input 1= 11110111 in binary and 247 in decimal

Input 2= 00000111 in binary and 7 in decimal

Division in binary = 11110111 / 00000111 = Quotient 00100011 in binary and in decimal 35 and Remainder is 00000010 in binary and 2 in decimal

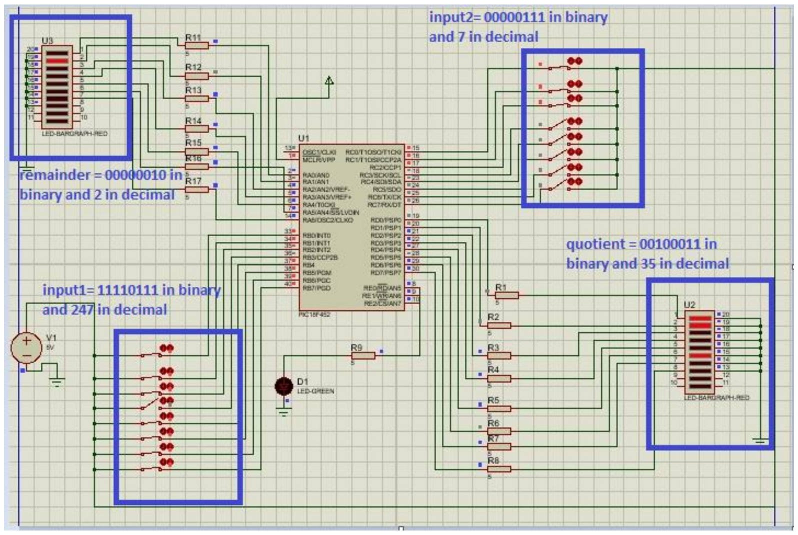
input2= 00000111 in binary and 7 in decimal
remainder = 00000010 in binary and 2 in decimal
quotient = 00100011 in binary and 35 in decimal
input1= 11110111 in binary and 247 in decimal

Figure 8: when input 1 is divided from input 2 of 8-bit number and quotient is shown with led bar-graph at portd and remainder is shown by led bar graph at porta (details of inputs and output are highlighted in attached figure)

Output after division between two inputs but divisor is not zero and there is no remainder:

Input 1= 01110111 in binary and 119 in decimal

Input 2= 00000111 in binary and 7 in decimal

Division in binary = 01110111 / 00000111 = Quotient 00010001 in binary and in decimal 17 and Remainder is 00000000 in binary and 0 in decimal

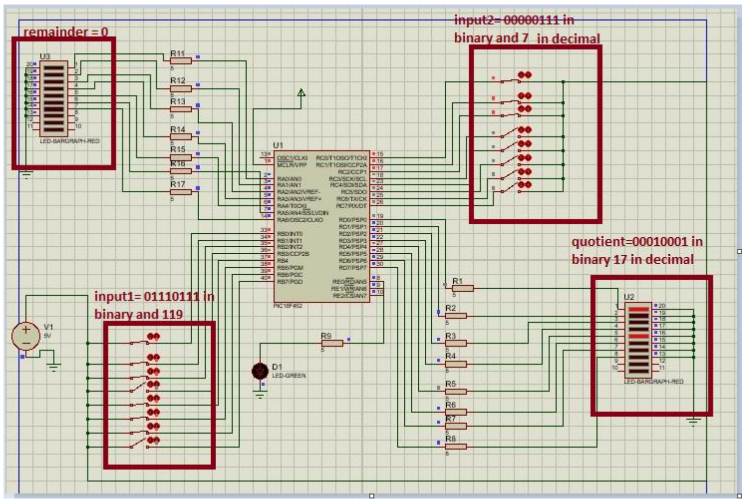
remainder = 0
input2= 00000111 in binary and 7 in decimal
quotient=00010001 in binary 17 in decimal
input1= 01110111 in binary and 119

Figure 9: when input 1 is divided from input 2 of 8-bit number and quotient is shown with led bar-graph at portd and remainder is shown by led bar graph at porta which is zero in this case (details of inputs and output are highlighted in attached figure)