Quang Nguyen

CPE 409 Lab

# Goals

* Learn the power of having a hardware multiplier
* Learn how resources are used in basic math functions
* Learn how to use fixed point data type

# Equipment used

## Hardware

* Microchip Explorer 16 board
* PIC kit 3

## Software

* MPLAB X IDE 2.00

# Design Specifications

* Timer 1 must be used to keep track of time of execution
* Design 1:
  + Part 1
    - Must multiply two unsigned 8 bits numbers together using repetitive addition
    - The result must be represented as a 16 bits number
    - Must measure the time it takes to perform 255 \* 255
  + Part 2
    - Must multiply 255 by 255 using the provided multiply operator
    - Must measure the time it takes to do perform this operation
* Design 2:
  + Part A
    - Must measure the time it takes and the resulting value of each of the following operations
      * Char = char \* char
      * Int = int \* int
      * Long = long \* long
      * Long long = long long \* long long
      * Float = float \* float
      * Long double = long double \* long double
  + Part B
    - Must perform division using the same data types as Part A
* Design 3:
  + Must write a function that will accepts 2 \_Q16 type input
    - It function will return the result of the multiplication of these 2 inputs in \_Q16 format
  + Must compare the execution time of this new function with the execution time of float \*float.
  + Must record the execution time of a \_Q16 addition.

# Design

* Design 1

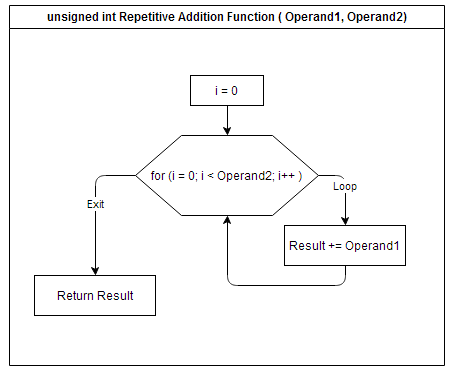


Figure 1: Flow diagram of Repetitive Addition Function

* + Refer to the end of the report for the code of the program
* Design 2

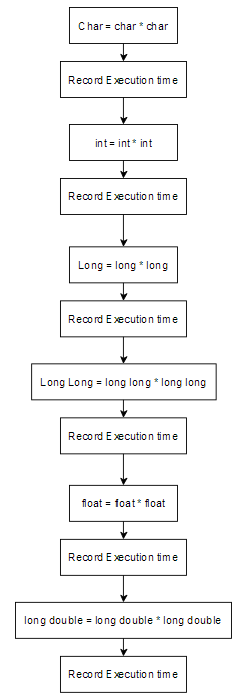


Figure 2: Flow diagram of Part 2

* + Refer to the end of the report for the code of the program
* Design 3
  + Refer to the end of the report for the code of the program

# Verification

### Part 1

* Result of 255 \* 255 = 65025

- Result of Timer 1 with a prescaler of 1 was 2833

- 0.0625 us \* 2833 = 177.06 us or 0.17706 ms



Figure 3: Snippet of the outputs for Design 1

### Part 2

* Part 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Design 2 Multiplication | | | |  |
|  | **Operand1** | **Operand2** | **Result** | **Timer 1** | **Execution time (s)** |
| **char** | 7 | 11 | 77 | 8 | 0.0000005 |
| **float** | 7 | 11 | 77.0 | 6 | 0.000000375 |
| **int** | 7 | 11 | 77 | 19 | 1.1875E-06 |
| **long** | 7 | 11 | 77 | 94 | 0.000005875 |
| **long double** | 7 | 11 | 77.0 | 120 | 0.0000075 |
| **long long** | 7 | 11 | 77 | 317 | 1.98125E-05 |

Table 1: Table for Design 2 Part 1

Figure 3: Graph for Design 2 Part 1

* Part 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Design 2 Division | | | |  |
|  | **Operand1** | **Operand2** | **Result** | **Timer 1** | **Execution time (s)** |
| **char** | 85 | 14 | 6 | 26 | 0.000001625 |
| **float** | 85 | 14 | 6.071429 | 24 | 0.0000015 |
| **int** | 85 | 14 | 6 | 474 | 0.000029625 |
| **long** | 85 | 14 | 6 | 7527 | 0.000470438 |
| **long double** | 85 | 14 | 6.071428 | 379 | 2.36875E-05 |
| **long long** | 85 | 14 | 6 | 1216 | 0.000076 |

Table 2: Table for Design 2 Part 2

Figure 4: Graph for Design 2 Part 2

## Part 3

* Q16.16 multiplication – Operand 1 = 3.1, Operand 2 = 4.1
  + **Time of execution = 28 us**
* Q15.16 Addition – Operand 1 = 85, Operand 2 = 14
  + **Time of execution = 2.18 us**
* Float addition – Operand 1 = 85, Operand 2 = 14
  + **Time of execution = 8.25 us**

# Conclusions and Limitations

* Design 1
  + It was concluded that the hardware multiplier does indeed speed up the multiplication operator by a significant amount
* Design 2
  + It was concluded that it takes the hardware significantly longer to perform multiplication as the number of bits increases.
  + The spike in execution time for long division in part 2 suggests that there might have been an error in the code
* Design 3
  + It was concluded that performing addition in \_Q15.16 format is significantly faster than performing addition in float format.
    - To be specific, it is faster by a factor of 4.

# Design 1 Code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Project: Lab 5 Part 1

\*

\* Name: Quang Nguyen

\*

\* Date: 2/14/2014

\*

\* Function: Multiply 2 unsigned 9 bit numbers together using

\* repetitive addition and place the answer in an

\* unsigned 16 bit number

\* Timer 1 will be used to keep track of how long

\* the additon takes

\*

\* Pins used:

\* Lower byte of PORTA

\* All 8 bits of PORTA will be turned ON if

\* the Timer overflow to indicate that the

\* prescaller needs to be a bigger number

\*

\* Peripherals used:

\* TIMER 1

\* Used to get an accurate measure of time

\* Internal Clock = 16 MHz

\* 1/16 MHz = 0.0625 us

\* with prescaler at 1

\* 0.0625 us \* 1 = 0.0625 us

\*

\* Comments:

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Configuration Bits

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <xc.h>

// FOSCSEL

#pragma config FNOSC = PRIPLL // Oscillator Mode (Primary Oscillator (XT, HS, EC) w/ PLL)

#pragma config IESO = ON // Two-speed Oscillator Start-Up Enable (Start up with FRC, then switch)

// FOSC

#pragma config POSCMD = XT // Primary Oscillator Source (XT Oscillator Mode)

#pragma config OSCIOFNC = OFF // OSC2 Pin Function (OSC2 pin has clock out function)

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Both Clock Switching and Fail-Safe Clock Monitor are disabled)

// FWDT

#pragma config FWDTEN = OFF // Watchdog Timer Enable (Watchdog timer enabled/disabled by user software)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Library includes

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <p33FJ256GP710A.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Constant Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define OP1 7.234

#define OP2 11.567

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Global Variable Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function Prototype

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function name : repetitiveAddition

\*

\* Returns : Unsigned int

\*

\* Parameters : 2 unsigned char

\*

\* Purpose : Perform addition using 8 bit repetitive addition

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned int repetitiveAddition (unsigned char FirstOperand, unsigned char SecondOperand);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Main Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

// setting up everything

initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Infinite Loop

\*

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while (1) {

T1CONbits.TON = 1; // Turn on Timer 1

repetitiveAddition (255, 255);

T1CONbits.TON = 0; // Turn OFF Timer 1

while(1); // Endless loop

} // End of the infinite While loop

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Initialize Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize() {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Setting up for Clock (PLL, M, N1, N2)

\* for 32 MHz and Fcy = 16 MHz

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Fosc = Fin(M/(N1\*N2)) = 8 MHz (32/(2\*4)) = 32 MHz

PLLFBD = 30; // M = 32

// N1 default is 2

// N2 default is 4

// Fcy = Fosc/2 by default

// Setting up Timer 1 module

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Internal Clock = 16 MHz

\* 1/16 MHz = 0.0625 us

\* with prescaler at 1

\* 0.0625 us \* 1 = 0.0625 us

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TCS = 0; // Use internal clock

T1CONbits.TCKPS = 0; // Set prescaler to 1

// Set initial value for PR1 to throw an interrupt if the

// prescaller is too smaller and the timer is overflowing

PR1 = 65535;

TMR1 = 0; // Clear TMR1

\_T1IE = 1; // Turn on TMR1 interrupt

// The default priority is 4 so lets not worry about it for now

// Setting up PORT A

AD1PCFGH = 0xFF; // Turn off ADC for Module 1

TRISA = 0x00; // Port A to output

PORTA = 0x00; // Keep PORTA off to begin with

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Interrupt Service Routine

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void \_\_attribute\_\_ ((interrupt, no\_auto\_psv)) \_T1Interrupt (void)

{ //

// Turn off flag //

IFS0bits.T1IF = 0; //

PORTA = 0xff; // Turn ON PORTA to indicate that TMR1 has overflowed

// and that the prescaler should be increased

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Other Functions

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned int repetitiveAddition (unsigned char FirstOperand, unsigned char SecondOperand)

{

unsigned int ResultOfAddition = 0;

unsigned char i = 0;

for (i = 0; i < SecondOperand; i++ )

{

ResultOfAddition = ResultOfAddition + FirstOperand;

}

return ResultOfAddition;

}

# Design 2

### The only part that changed in design 2 was the main loop of the program

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Main Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

// setting up everything

initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Loop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char ResultOfChar = 0, CharTemp1 = OP1, CharTemp2 = OP2;

int ResultOfInt = 0, IntTemp1 = OP1, IntTemp2 = OP2;

long ResultOfLong = 0, LongTemp1 = OP1, LongTemp2 = OP2;

long long ResultOfLongLong = 0, LongLongTemp1 = OP1, LongLongTemp2 = OP2;

float ResultOfFloat = 0, FloatTemp1 = OP1, FloatTemp2 = OP2;

long double ResultOfLongDouble = 0, LongDoubleTemp1 = OP1, LongDoubleTemp2 = OP2;

int Time[6];

//time[0] : Char

//time[1] : Int

//time[2] : Long

//time[3] : Long Long

//time[4] : Float

//time[5] : Long Double

// Initialize Time

int i;

for (i = 0; i < 6; i++)

Time[i] = 0;

i =0;

while (1) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Char = Char \* Char \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Note: TMR1 should be 0 at this point

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfChar = CharTemp1 / CharTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Int = Int \* Int \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfInt = IntTemp1 / IntTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Long = Long \* Long \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfLong = LongTemp1 / LongTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Long Long = Long Long \* Long Long \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfLongLong = LongLongTemp1 / LongLongTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Float = Float \* Float \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfFloat = FloatTemp1 / FloatTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Long Double = Long Double \* Long Double \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TON = 1; // Turn on Timer 1

ResultOfLongDouble = LongDoubleTemp1 / LongDoubleTemp2;

T1CONbits.TON = 0; // Turn OFF Timer 1

Time[i] = TMR1; // Store value of TMR1

i++; // Increment pointer

TMR1 = 0; // Clear TMR1

while(1); // Endless loop

} // End of the infinite While loop

return 1;

}

# Design 3

### The parts that were changed in design 3 was the main loop of the program and the 2 added functions

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Main Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

// setting up everything

initialize();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Loop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Initialize Time

float Operand1 = 3.1, Operand2 = 4.1;

\_Q16 ResultQ16 = 0;

\_Q16 QOperand1 = 0, QOperand2 = 0;

float ResultFloat = 0;

while (1) {

QOperand1 = \_Q16ftoi(Operand1);

QOperand2 = \_Q16ftoi(Operand2);

T1CONbits.TON = 1; // Turn on Timer 1

// ResultQ16 = Q16Add (QOperand1,QOperand2);

ResultFloat = Operand1 + Operand2;

T1CONbits.TON = 0; // Turn OFF Timer 1

while(1); // Endless loop

} // End of the infinite While loop

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Other Functions

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\_Q16 Q16Mult (\_Q16 Op1, \_Q16 Op2)

{

\_Q16 ResultOfQ16 = 0;

ResultOfQ16 = ((Op1 >> 8) \* (Op2 >> 8));

return ResultOfQ16;

}

\_Q16 Q16Add (\_Q16 Op1, \_Q16 Op2)

{

\_Q16 ResultOfQ16 = 0;

ResultOfQ16 = (Op1 + Op2);

return ResultOfQ16;

}