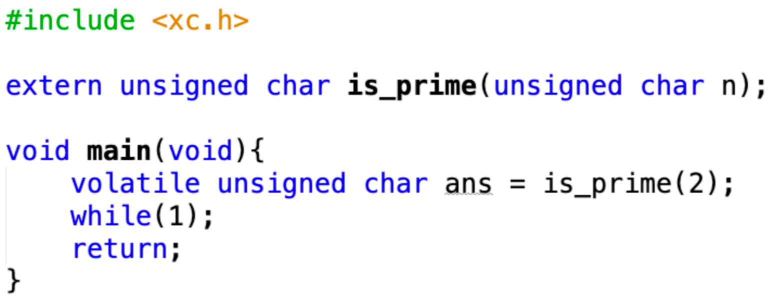
Lab 5: Mixing with C

- **Link**
  - Document
  - Video

- **Basic** (50%)
  - **Description**

    The program **main.c** will call the function is_prime to check whether an input number is a prime number or not. Please complete the is_prime function using PIC18F assembly language. The input of the function will be an 8-bit unsigned char. The return value should be 0x01 if the input number is a prime number, otherwise 0xFF if the input is not a prime. The result should be stored in an unsigned char variable named ***ans***.

    Please be aware that you will need two files, **main.c** and **is_prime.asm**, to fulfill this requirement.



```
#include <xc.h>

extern unsigned char is_prime(unsigned char n);

void main(void){
    volatile unsigned char ans = is_prime(2);
    while(1);
    return;
}
```

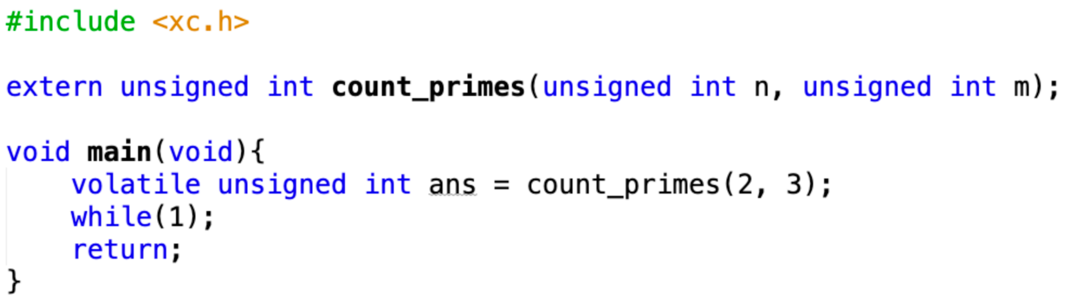
  - **Example**
    1. is_prime(1) = 0xFF
    2. is_prime(2) = 0x01

  - **Standard of grading**
    1. Mixing with C: Implement the function in assembly and call it from the main C program.
    2. The function name and the variable name in **main.c** must be the same as described.
    3. Please show the result in the *Watches* window.
    4. All input test cases will fall between 1 and 255.
    5. You are not allowed to hardcode a list of prime numbers and directly compare the input with the list.

Advance (30%)

Description

The program **main.c** will call the function count_primes to calculate how many prime numbers lie within a given inclusive interval. Please complete the count_primes function using PIC18F assembly language. The inputs of the function will be two 16-bit unsigned integers ***n*** and ***m*** with ***n*** < ***m***. The result should be stored in an unsigned integer variable named ***ans*** that represents the number of prime numbers in the inclusive range [***n***, ***m***].
Please be aware that you will need two files, **main.c** and **count_primes.asm**, to fulfill this requirement.



```c
#include <xc.h>

extern unsigned int count_primes(unsigned int n, unsigned int m);

void main(void){
    volatile unsigned int ans = count_primes(2, 3);
    while(1);
    return;
}
```

Example

1. count_primes(2, 3) = 2
2. count_primes(114, 514) = 67

Standard of grading

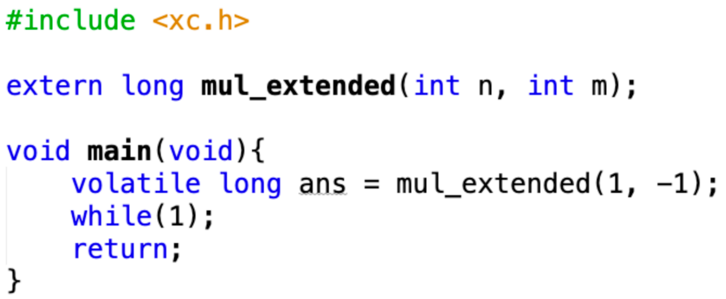
1. Mixing with C: Implement the function in assembly and call it from the main C program.
2. The function name and the variable name in **main.c** must be the same as described.
3. Please show the result in the *Watches* window.
4. All input test cases will fall between 1 and 65535.
5. You are not allowed to hardcode a list of prime numbers and directly compare the input with the list.

- **Hard** (20%)

  - **Description**

    The program **main.c** will call the function mul_extended to perform a signed multiplication. Please complete the mul_extended function using PIC18F assembly language. The function takes two 16-bit signed integer ***n*** and ***m*** as inputs. The result should be stored in a signed long variable named ***ans*** that represents the product of ***n*** and ***m***. All signed inputs use two's complement representation.

    Please be aware that you will need two files, **main.c** and **mul_extended.asm**, to fulfill this requirement.



```c
#include <xc.h>

extern long mul_extended(int n, int m);

void main(void){
    volatile long ans = mul_extended(1, -1);
    while(1);
    return;
}
```

  - **Example**

    1. mul_extended (1, -1) = -1
    2. mul_extended (79, 997) = 78763

  - **Standard of grading**

    1. Mixing with C: Implement the function in assembly and call it from the main C program.
    2. The function name and the variable name in **main.c** must be the same as described.
    3. Please show the result in the *Watches* window.
    4. All input test cases will fall between -32768 and 32767.
    5. Please make use of the hardware multiplier rather than implementing multiplication by repeated addition.