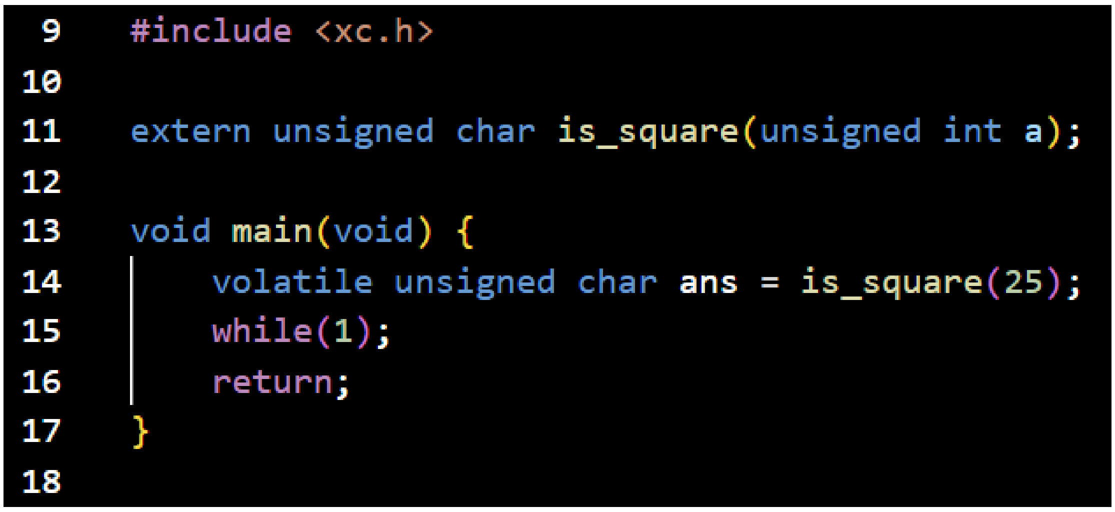
LAB 05 Requirement Description

- **Mixing with C**
  - **Video Link:** [Lab05: Mixing with C – YouTube]
  - **HackMD Link:** [Lab05: Mixing with C – HackMD]
- **Lab requirements:**
  - **Basic (70%):**
    - **Description:**
      The following program "main.c" will call the "is_square" function to check if an input number is a square number or not. Please complete the "is_square" function with PIC18F assembly language. The input of "is_square" function will be an 8-bit unsigned integer. The return value will be either 0x01 if the input integer is a square number or 0xFF if the input integer is not a square number. The result will be stored in an unsigned char named 'ans.' Please be aware that you will need two files, 'main.c' and 'is_square.asm,' to fulfill this requirement.

    - **Example:**
      - is_square(25) = 0x01
      - is_square(99) = 0xFF

    - **Standard of grading:**
      1. Mixing with C. Implement the feature above in asm and call it by main function.
      2. The name of the function and the name of the variable in main.c should be the same as the description.
      3. Please show the output in the WATCHes.
      4. All test cases will fall between 1 and 255.
      5. Listing all square numbers below 256 in your assembly code and checking if the input number is one of the listed numbers is forbidden



```c
#include <xc.h>

extern unsigned char is_square(unsigned int a);

void main(void) {
    volatile unsigned char ans = is_square(25);
    while(1);
    return;
}

```

■ **Advanced (30%):**

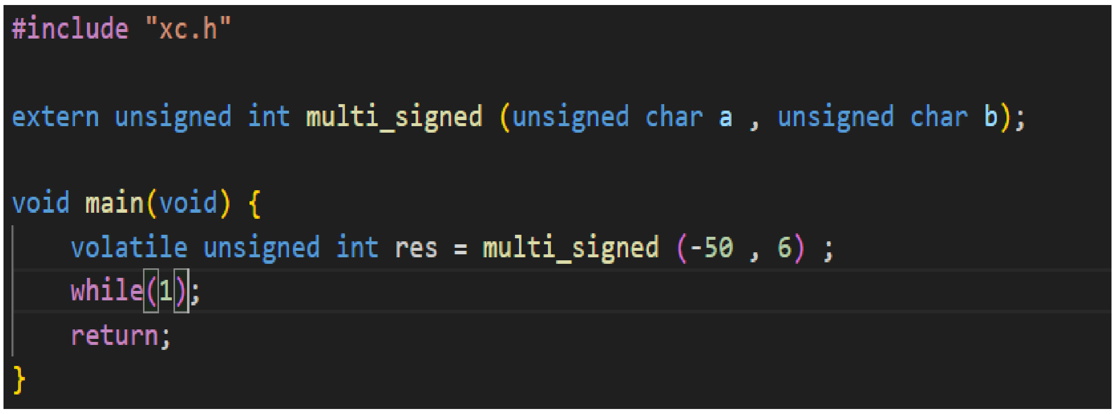
- **Description:**
  The following program "main.c" will call the "multi_signed " function to finish the signed multiplication. Please complete the "multi_signed" function with PIC18F assembly language. The "multi_signed " function takes an 8-bit signed char ***a*** and a 4-bit signed char ***b*** as inputs and returns an unsigned int ***res*** which represents the result of the multiplication of ***a*** and ***b***. The output will be a 16-bit result. The result should be stored in an unsigned int variable, then show it in the WATCHes. Please note that the signed data will be formatted in two's complement.

- **Constrain:** multiplicand (-128~127), multiplier (-8~7).

- **Example:**
  multiplicand = 127, multiplier = -6, multi_signed(127, -6) = 64774
  Notice: The actual test data will not be the same as the example, make sure your code can be executed in any case.

- **Standard of grading:**
  1. You should NOT add more lines of code in C but implement it in asm.
  2. The name of the function and the name of the variable in main.c should be the same as the description.
  3. You should NOT use MULLW or MULWF.

- **Hint: try to predict the sign of outcome before the unsigned result.**



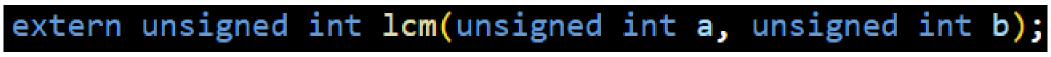
```c
#include "xc.h"

extern unsigned int multi_signed (unsigned char a , unsigned char b);

void main(void) {
    volatile unsigned int res = multi_signed (-50 , 6) ;
    while(1);
    return;
}
```

- **Bonus (20%):**
  - **Description:**

    Given an 8-bit unsigned integer ***a*** and an 8-bit unsigned integer ***b***, please complete the "***lcm***" function with PIC18F assembly language. The function returns a 16-bit unsigned integer that represents the least common multiple of ***a*** and ***b***.

  - **Example:**
    1. a=5, b=15, lcm(5,15) = 15
    2. a=140, b=3, lcm(140,3) = 420

       Notice: The actual test data will not be the same as the example, make sure your code can be executed in any case.
  - **Standard of Grading:**
    1. Mixing with C. Implement the feature above in asm and call it by main function.
    2. Using function signature as follow:

       ```
       extern unsigned int lcm(unsigned int a, unsigned int b);
       ```
    3. You should show the output in the WATCHes and explain your code logic in detail.