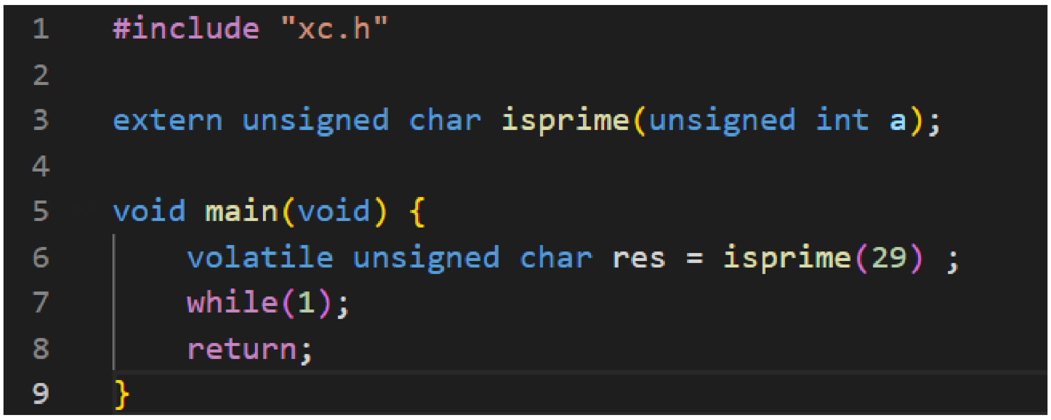
Lab 5: Requirement Description

10/16 (23:37)更新：更新了影片中應使用的測資及基礎題的範例圖片。

- Video：https://www.youtube.com/watch?v=0yHHQaLUj6A
- hackmd: https://hackmd.io/EiCnnaKHRkKDq5Zq_erdHg?view

- Basic (70%):
  - Description: The following program "main.c" will call the "***isprime***" function to check if an input number is prime or not. Please complete the "***isprime***" function with PIC18F assembly language. The input of "***isprime***" function will be an 8-bit unsigned integer. The return value will be either **0x01** if the **input integer is prime** or **0xFF** if the **input integer is not prime**. Then store the value in an unsigned char named "**res**". **Note that, you will need two files "*main.c*" and "*isprime.asm*" to finish the requirement.**
  - Example:
    - isprime(29) = 0x01
    - isprime(15) = 0xFF

    Notice: The actual test data will not be same as example, make sure your code can be executed on any case
  - Standard of grading:
    1. Mixing with C. Implement the feature above in asm and call 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. No need to consider “0” or “1” as input.
    5. If you need to demo online, please show **the execution result and process of isprime(29)** in the video.



```c
#include "xc.h"

extern unsigned char isprime(unsigned int a);

void main(void) {
    volatile unsigned char res = isprime(29) ;
    while(1);
    return;
}
```

- Advanced (30%):
  - Description: The following program "main.c" will call the "***divide_signed*** " function to finish the signed division. Please complete the "***divide_signed*** " function with PIC18F assembly language. The "***divide_signed*** " function inputs an 8-bit signed char divided by a 4-bit signed char and outputs an unsigned int. The outputs will be an **8-bit quotient** and a **4-bit remainder**. The resulted quotient and remainder should be stored in two signed chars, then shown in the WATCHes. **Note that, the signed data will be represented by two's complement.**
    (e.g., If quotient = -21, WATCHes will show **235 (-21 + 256)** in decimal.)
  - Constrain: Dividend (-128~127), divisor (-8~7, without 0)
  - Example:
    - dividend = 127, divisor = -6, devide_signed(127, -6) = (235, 1)

    Notice: The actual test data will not be same as example, make sure your code can be executed on any case
  - Standard of grading:
    1. You should NOT add more line 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 can implement this function in your own way.

4. The relation between quotient and remainder is not fixed as long as you follow the principle that

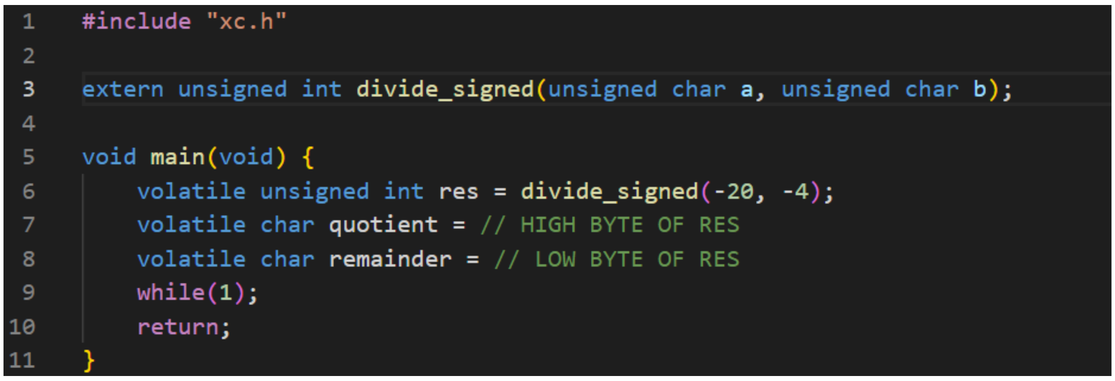
$$|divisor| > |remainder|$$

e.g., devide_signed(127, -6) can be (235, 1) or (234, -5).

$$127 \div (-6) = (-21) \times (-6) + 1 = (-22) \times (-6) + (-5)$$

5. Don’t need to consider “dividing by 0” situation.
6. If you need to demo online, please show **the execution results and process of the example test data** in the video.

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



```c
#include "xc.h"

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

void main(void) {
    volatile unsigned int res = divide_signed(-20, -4);
    volatile char quotient = // HIGH BYTE OF RES
    volatile char remainder = // LOW BYTE OF RES
    while(1);
    return;
}
```

Bonus (20%):

- Description:
  Given **8-bit unsigned integer a** and **8-bit unsigned integer b**, please implement:

$$Pow(a, b)$$

The function returns a 16-bit unsigned integer.

- Hint:
  1. $Pow(a, 0) = 1$
  2. Function definition above can be referred to C standard library:

     https://cplusplus.com/reference/cmath/pow/

- Example:
  1. a=2, b=5, mypow(2,5) = 32,
  2. a=5, b=3, mypow(5,3) = 125

Notice: The actual test data will not be same as example, make sure your code can be executed on any case.

- Standard of Grading:

1. Mixing with C. Implement the feature above in asm and call by main function.
2. Using function signature as follow:

```
extern unsigned int mypow(unsigned int a, unsigned int b);
```

3. You should show the output in the WATCHes and explain your code logic in detail.
4. If you need to demo online, please show **the execution result and process of mypow(5,3)** in the video.