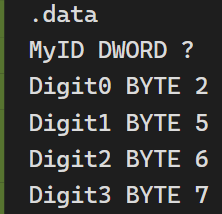
**Assembly Homework 1 #Arithmetic**

**Department:** CSIE 2-B

**Student Number:** 110502567

**Name:** 蔡淵丞Vincent

**DATA SECTION:**



declaring the last 4 digits of my student

number as BYTEs and a variable MyID as

DWORD to store the result

**THOUGHT:**

The main purpose of the program is to learn to use the shifting instruction, and the function of this instruction is to multiply the value by 2 to the power of *n*. Therefore, our main challenge of presenting the result in hexadecimal is how many times we are going to shift to carry the value to the ideal digit.

Observe the rule of carrying digits in decimal, we know that to make a number one digit higher, we must multiply it by the base number, which is in this case, 10.(003 x 10 = 030) Similarly, in hexadecimal display, we can multiply the number by 16 to have the same effect on it.

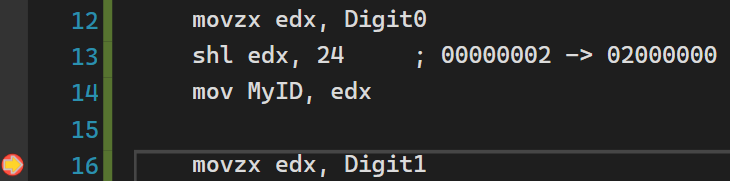
Now we know that by multiplying the number by 16(2^4) we can make it one digit higher in hexadecimal, which happens to be the effect of shifting it left 4 times in assembly language.

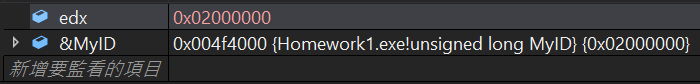
**MAIN PROC:**

We use one register ***edx***to hold and shift one number at a time, and then store it to variable MyID.

**LINE 12-14:**

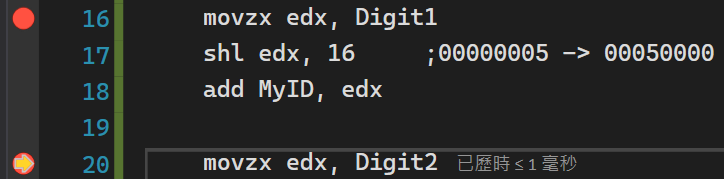
We move the first digit to ***edx*** using *movzx* due to size limitation, and carry it 6 digits left, which is shifting it 24(6x4) times, and then finally store it to ***MyID***.

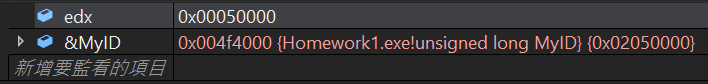




**LINE 16-18:**

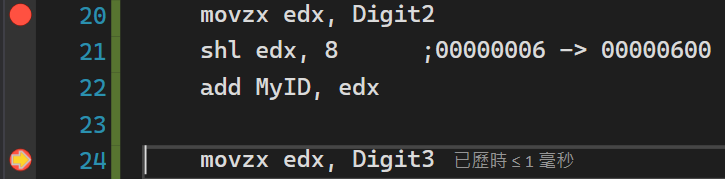
Similarly, the second digit needs 16 times of shifting.

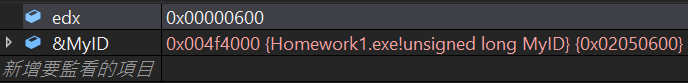




**LINE 20-22:**

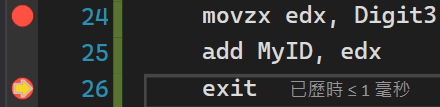
The third one needs 8 times of shifting.

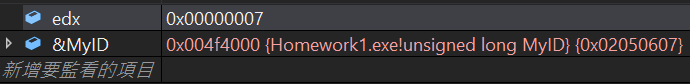
****



**LINE 24-25:**

The fourth digit doesn’t need to be shifted; we simply add it to ***MyID.***

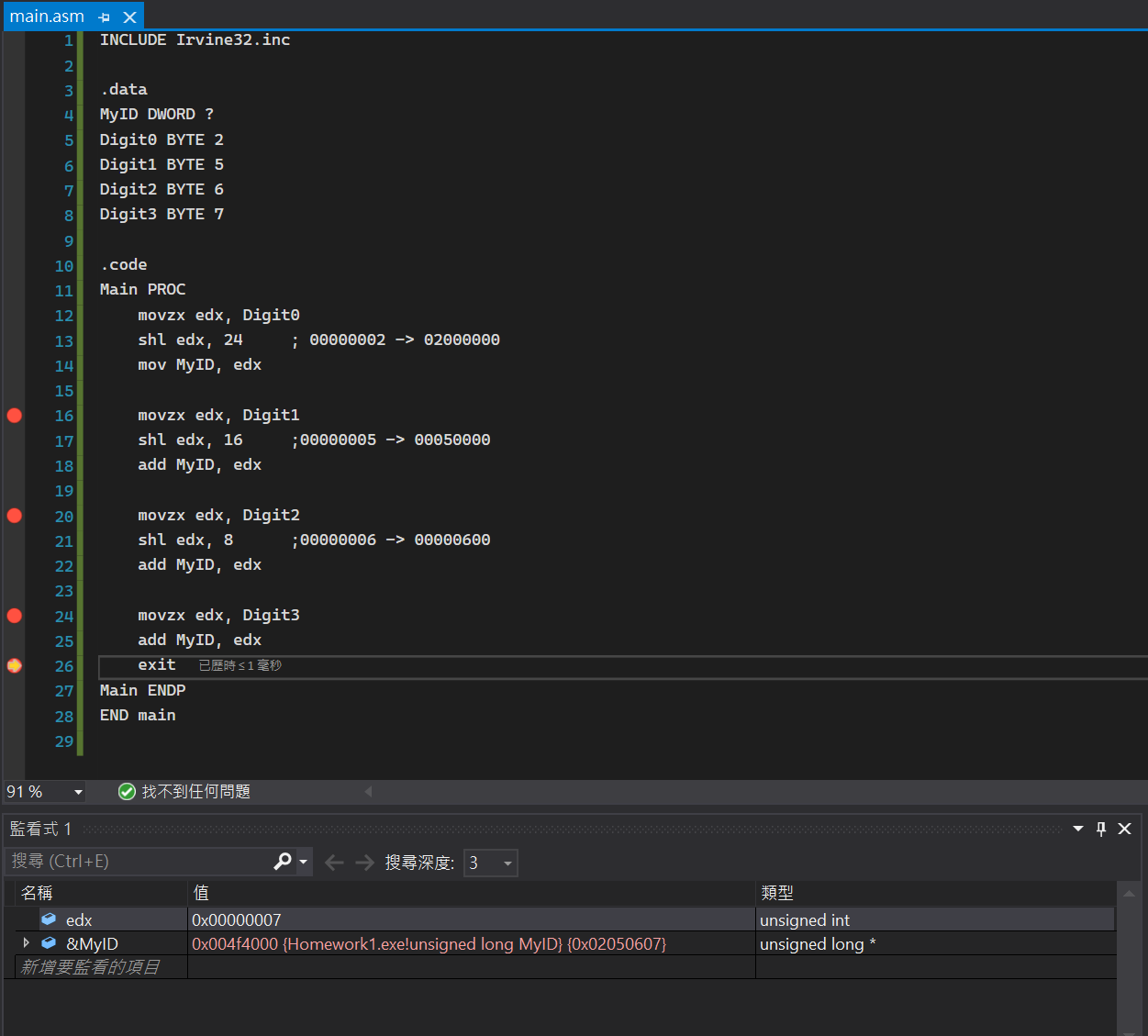




***MyID*** has been successfully altered to 02050607(hex) as wanted.

**REVIEW:**

In my opinion, the step-by-step process of the code in this report is quite unnecessary, because descriptive comments in the code can do it all. But it is still important that we have the ability to monitor the value in registers. I just consider it a little time consuming.

**FULL CODE:** ****