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

```
.data
MyID DWORD ?
Digit0 BYTE 2
Digit1 BYTE 5
Digit2 BYTE 6
Digit3 BYTE 7
```

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, $10.(003 \times 10 = 030)$ 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.

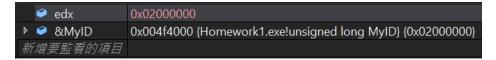
MAIN PROC:

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

LINE 12-14:

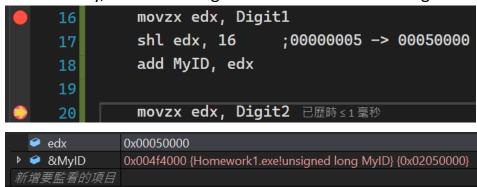
We move the first digit to *edx* using *movzx* due to size limitation, and carry it 6 digits left, which is shifting it 24(6x4) times, and then finally store it to *MyID*.

```
12 movzx edx, Digit0
13 shl edx, 24 ; 000000002 -> 02000000
14 mov MyID, edx
15
16 movzx edx, Digit1
```



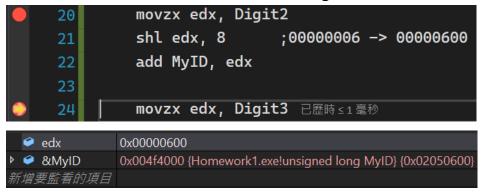
LINE 16-18:

Similarly, the second digit needs 16 times of shifting.



LINE 20-22:

The third one needs 8 times of shifting.



LINE 24-25:

24

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



movzx edx, Digit3

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:

```
INCLUDE Irvine32.inc
     3
        .data
     4 MyID DWORD ?
     5 Digit0 BYTE 2
     6 Digit1 BYTE 5
        Digit2 BYTE 6
        Digit3 BYTE 7
     8
    10
        . code
        Main PROC
           movzx edx, Digit0
           shl edx, 24 ; 00000002 -> 02000000
    13
            mov MyID, edx
    14
            movzx edx, Digit1
            shl edx, 16 ;00000005 -> 00050000
            add MyID, edx
            movzx edx, Digit2
            shl edx, 8 ;00000006 -> 00000600
            add MyID, edx
            movzx edx, Digit3
          add MyID, edx
            exit 已歷時≤1毫秒
    26
        Main ENDP
    27
    28
        END main
    29
91% ▼ ② 找不到任何問題
監看式 1
                                                                                                     ▼ Ҭ X
                         ▶ → 搜尋深度: 3 →
                                                                       類型
  edx
                 0x00000007
                                                                       unsigned int
 ▶ 🔗 &MyID
                 0x004f4000 {Homework1.exe!unsigned long MyID} {0x02050607}
                                                                       unsigned long *
 新增要監看的項目
```