Assembly Language Report (Week 3)

Group 7

104502517戴逸任

104502518劉冠聲

Program CODE:

.data

Val1 SBYTE 03h ;Val1 = 03h

Val2 SBYTE 02h ;Val2 = 02h

Val3 SBYTE 8fh ;Val3 = 8fh

Rval SDWORD ? ;Rval = 0

; The code section is a phrase used to refer to a portion of memory or of an object file

; that contains executable instructions.

.code

; main procedure

start@0 PROC

movsx eax,Val1 ;eax = Val1

movsx ebx,Val2 ;ebx = Val2

movsx ecx,Val3 ;ecx = Val3

mov edx,0 ;edx = 0

add eax,ebx ;eax = Val1 + Val2

shl eax,1 ;eax =(Val1+Val2)\*2

add edx,eax ;edx =(Val1+Val2)\*2

shl eax,1 ;eax =(Val1+Val2)\*4

add edx,eax ;edx =(Val1+Val2)\*6

shl eax,1 ;eax =(Val1+Val2)\*8

add edx,eax ;edx =(Val1+Val2)\*14

sub ecx,edx ;ecx = Val3-(Val1+Val2)\*14

neg ecx ;ecx = -ecx

mov Rval,ecx ;Rval = -(Val3 - 14\*(Val1+Val2))

exit

start@0 ENDP

END start@0

Program Step & Register state:

movsx eax,Val1 ; eax=0x03 ebx=0x00 ecx=0x00 edx=0x00

movsx ebx,Val2 ; eax=0x03 ebx=0x02 ecx=0x00 edx=0x00

movsx ecx,Val3 ; eax=0x03 ebx=0x02 ecx=0xffffff8f edx=0x00

mov edx,0 ; eax=0x03 ebx=0x02 ecx=0xffffff8f

edx=0x00

add eax,ebx ; eax=0x05 ebx=0x02 ecx=0xffffff8f

edx=0x00

shl eax,1 ; eax=0x0a ebx=0x02 ecx=0xffffff8f

edx=0x00

add edx,eax ; eax=0x0a ebx=0x02 ecx=0xffffff8f

edx=0x0a

shl eax,1 ; eax=0x14 ebx=0x02 ecx=0xffffff8f

edx=0x0a

add edx,eax ; eax=0x14 ebx=0x02 ecx=0xffffff8f

edx=0x1e

shl eax,1 ; eax=0x28 ebx=0x02 ecx=0xffffff8f

edx=0x1e

add edx,eax ; eax=0x28 ebx=0x02 ecx=0xffffff8f

edx=0x46

sub ecx,edx ; eax=0x28 ebx=0x02 ecx=0xffffff49

edx=0x46

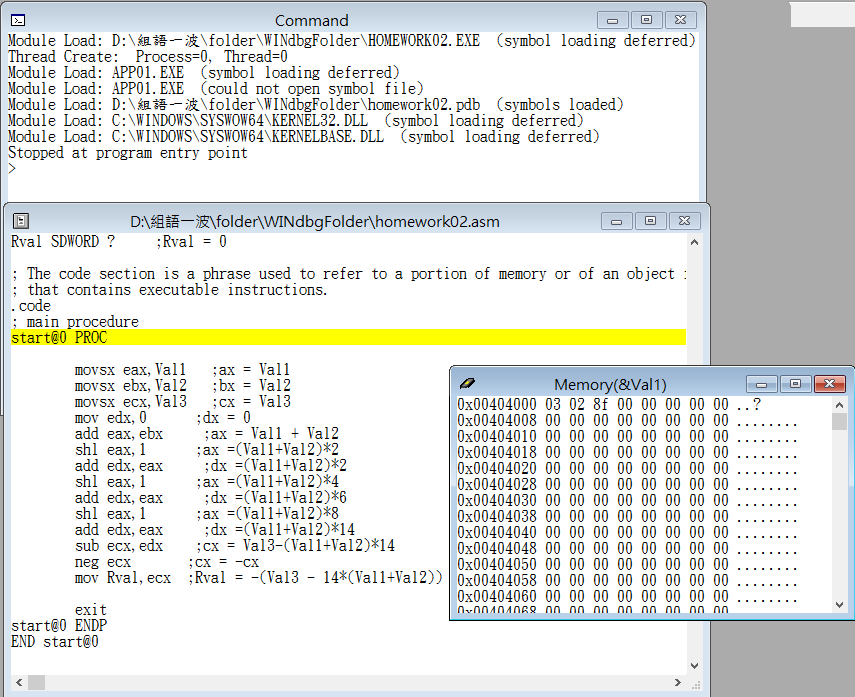
neg ecx ; eax=0x28 ebx=0x02 ecx=0x000000b7

edx=0x46

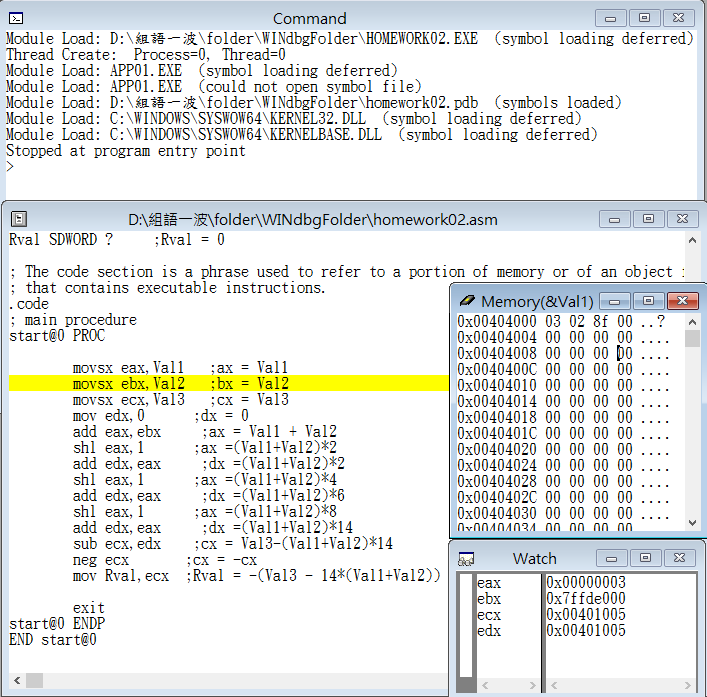
mov Rval,ecx ; eax=0x28 ebx=0x02 ecx=0x000000b7

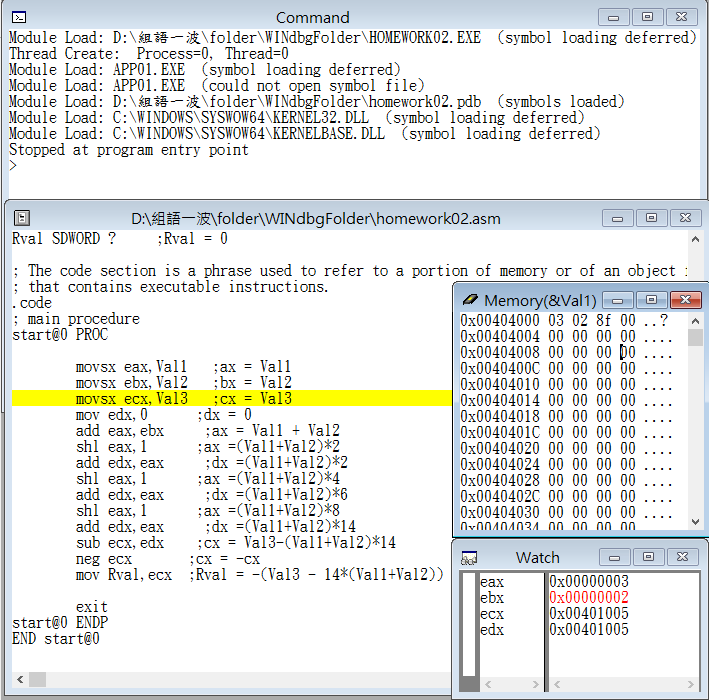
edx=0x46

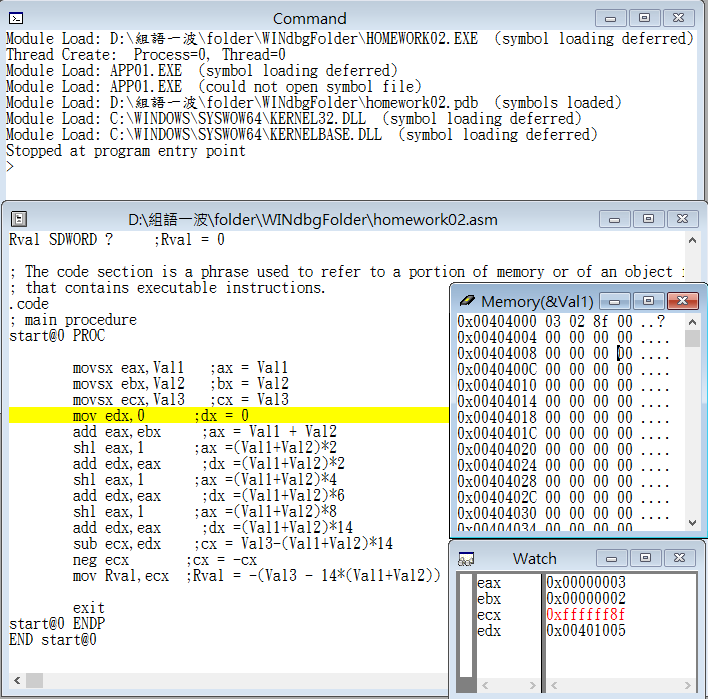
Picture & Discription:



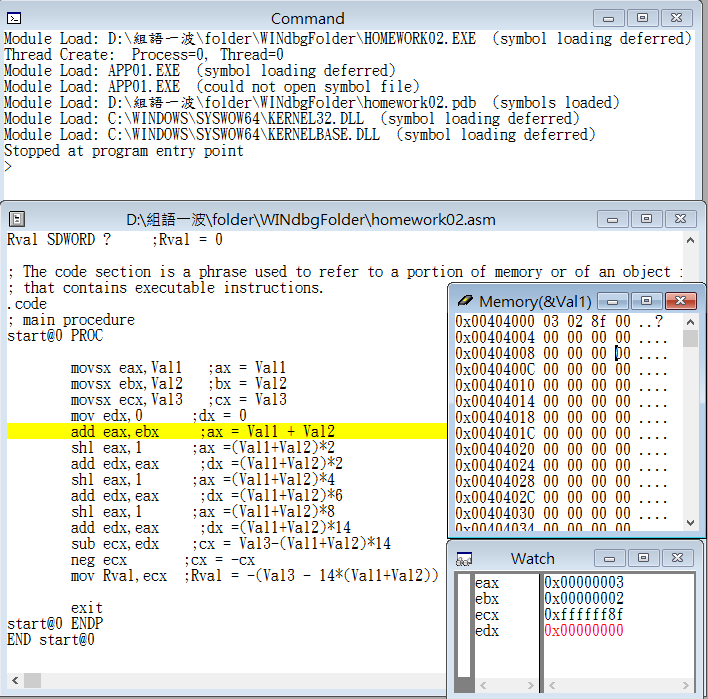
Step1: Start the program



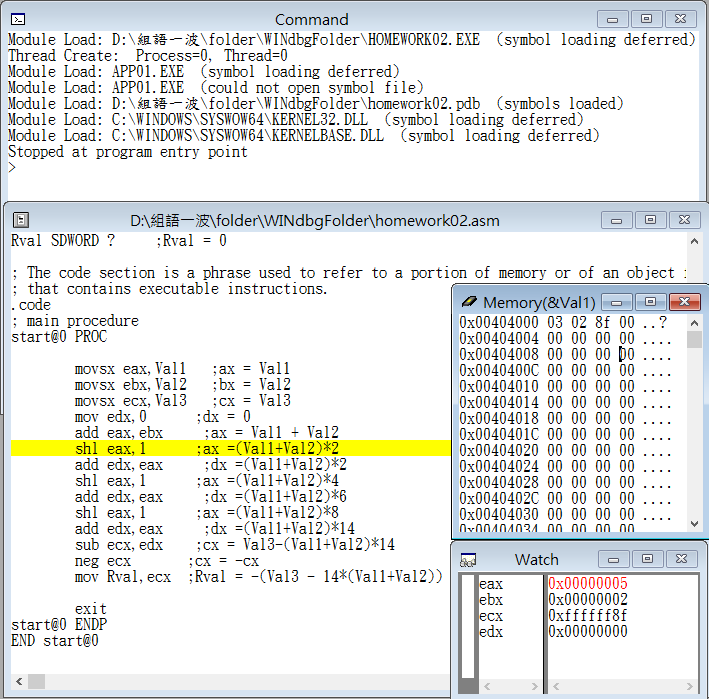




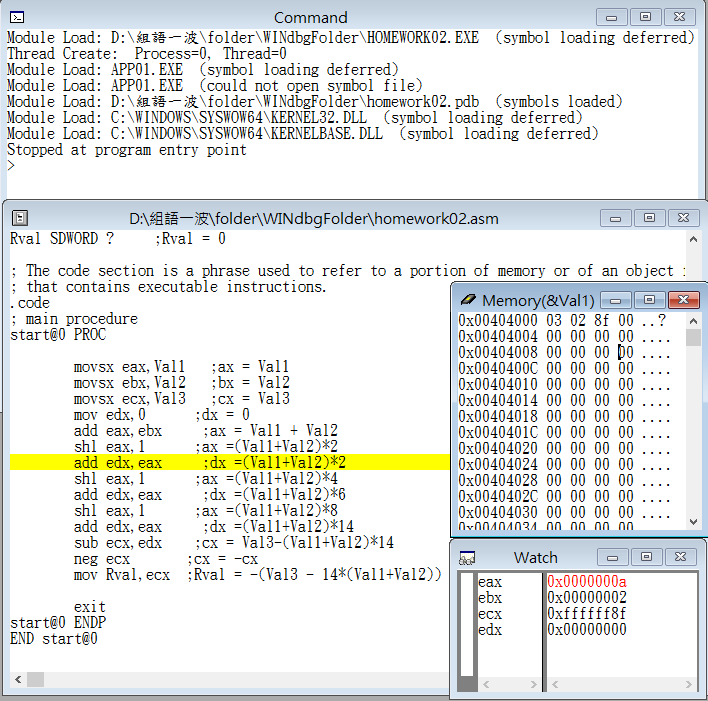
Step2: move Val1, Val2, Val3 into bigger register eax, ebx, and ecx



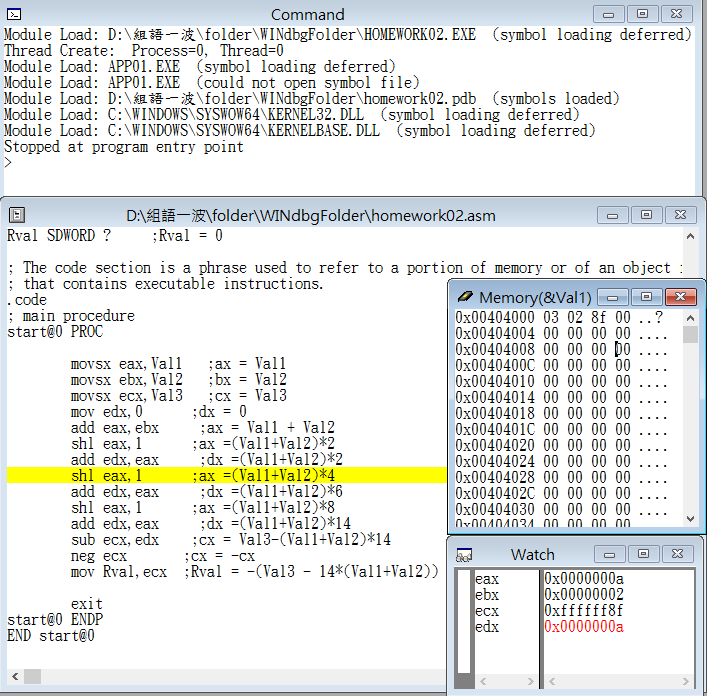
Step3: store the value in edx is 0



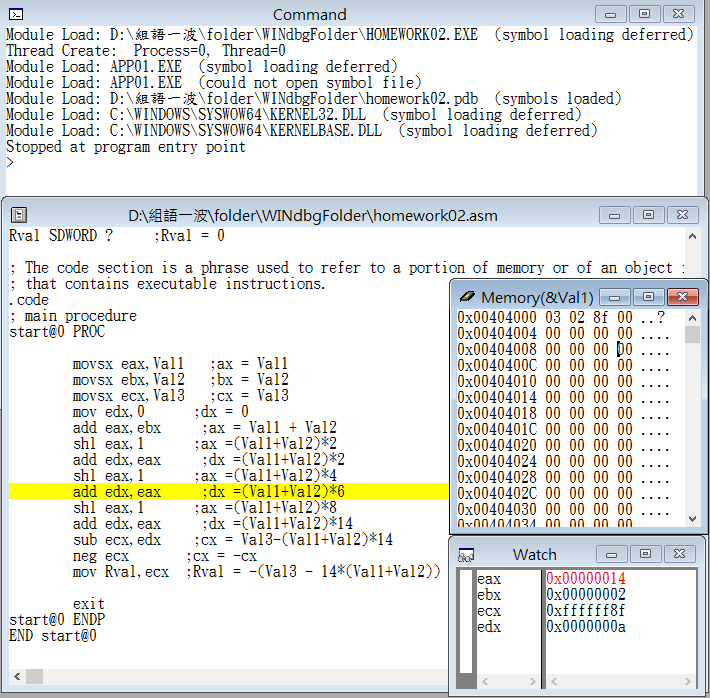
Step4: store the sum of eax and ebx into eax (Val1+Val2)



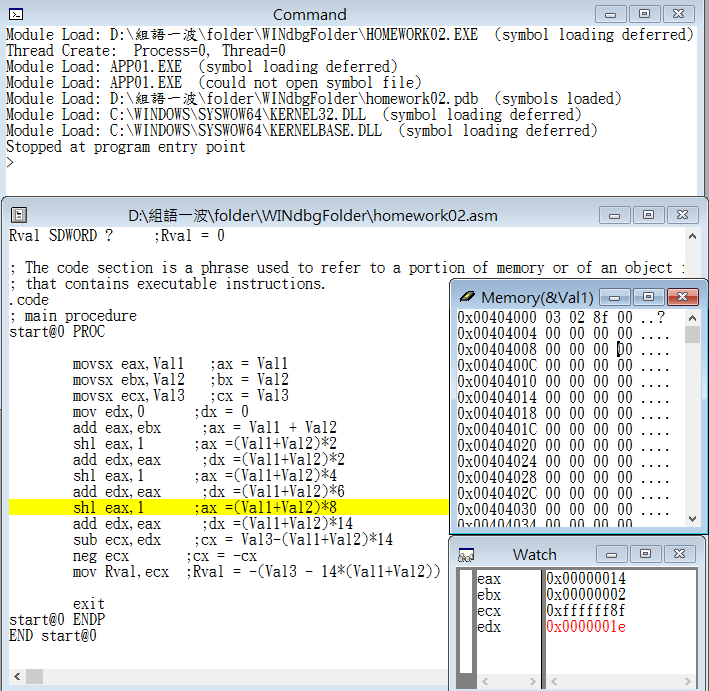
Step5: shift eax 1 time left, it means that eax value become double ((Val1+Val2)\*2)



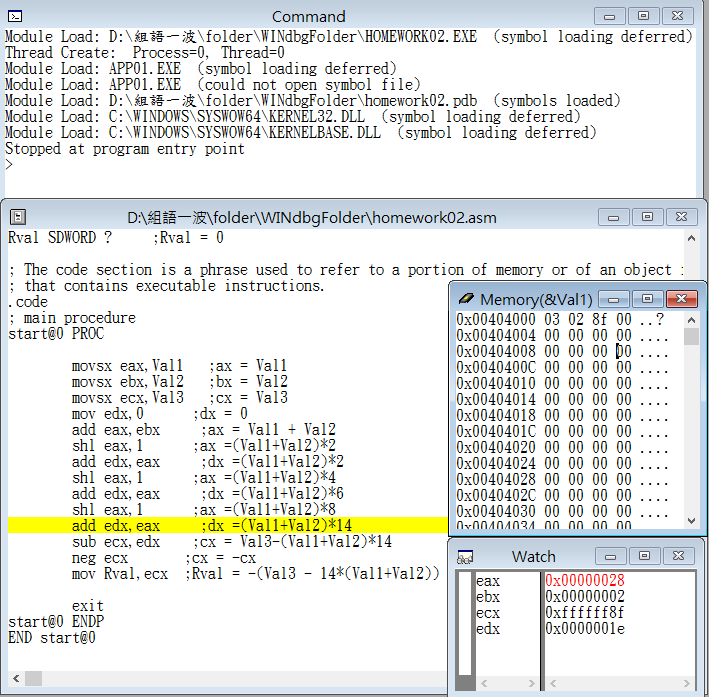
Step6: store the sum of eax and edx into edx ((Val1+Val2)\*2)



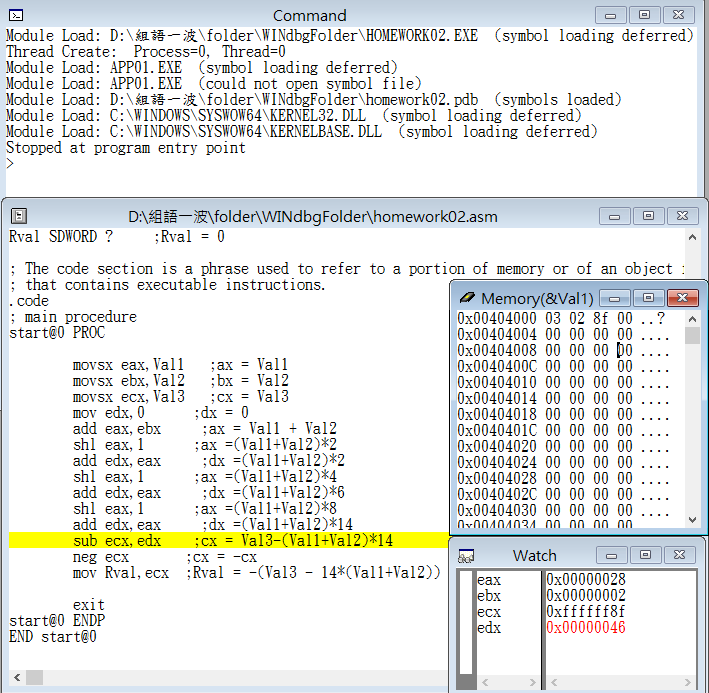
Step7: shift eax 1 time left again, it means that eax become four times of its initial value ((Val1+Val2)\*4)



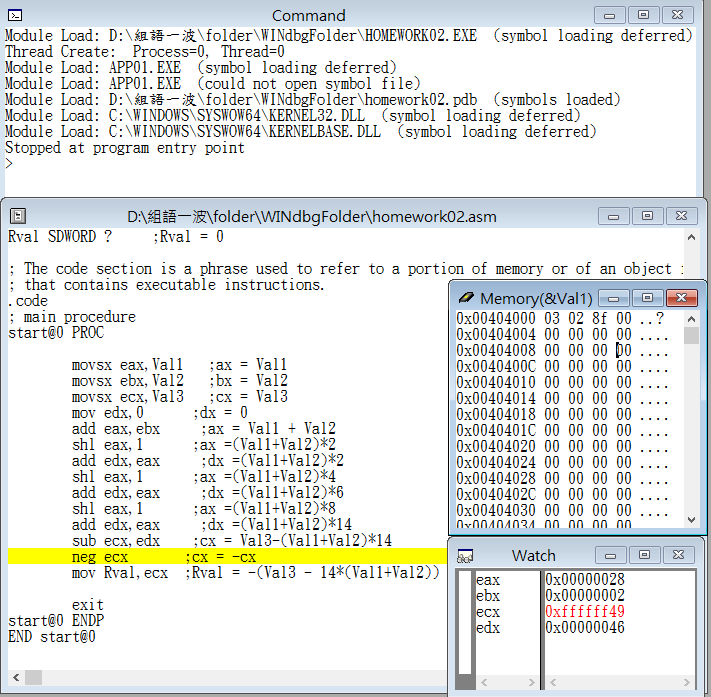
Step8: store the sum of eax and edx into edx ((Val1+Val2)\*6)



Step9: shift eax 1 time left again, it means that eax become eight times of its initial value ((Val1+Val2)\*8)

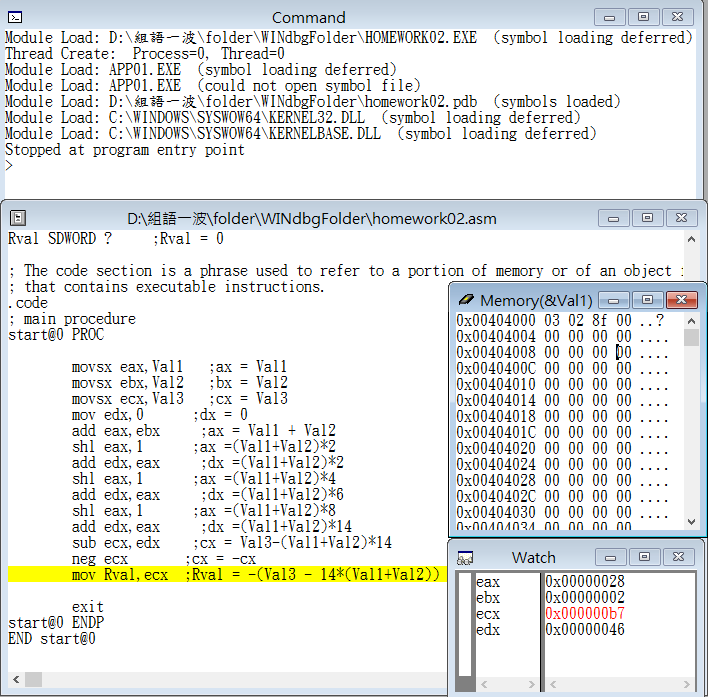


Step10: store the sum of eax and edx into edx ((Val1+Val2)\*14)

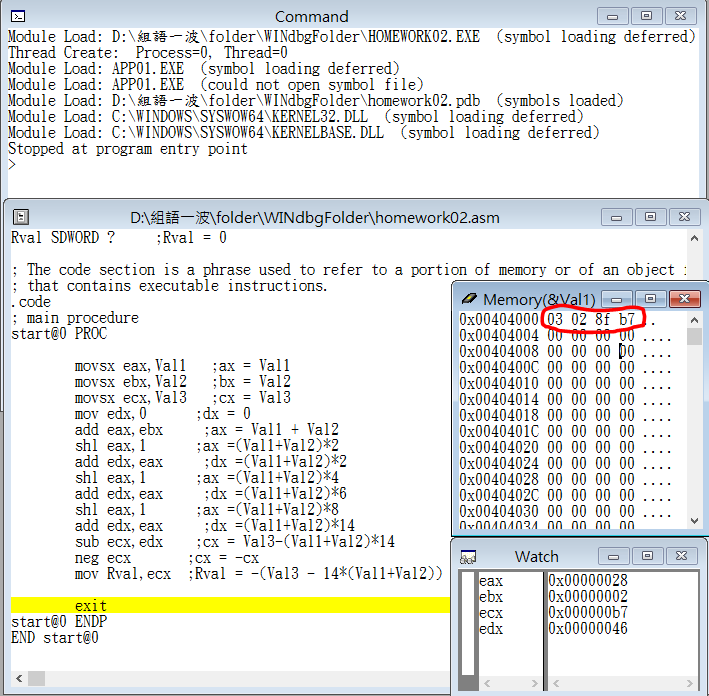


Step11: store the value of ecx sub edx into ecx

(Val3-(Val1+Val2)\*14)



Step12: turn the sign of ecx (-(Val3-(Val1+Val2)\*14))



Step13: store the value of ecx into the variable Rval(the end)

Review:

Through the experience which we learned from the practice last week, we realized how to use the function in the windbg such as watch and memory. The practice this week is about calculating of Assembly language, and our code was not right. We found some mistakes after checking the code, then we knew that our code about storing memory was not correct. We previously use ‘ax’ to store the memory, so we changed the ‘ax’ into ‘eax’ because the value would overflow. The practice is still pretty superficial in the class, and we hope that we can solve the other problems successfully.