

< 과제 4 >

2017253041_홍성우

1.

mul.asm: mul을 할 때 범위를 넘어서면 edx에 저장되고 unsign인 mul 연산을 할땐 edx로 넘어가니 cf와 of가 1이 되고 sign인 imul을 할땐 sign extension이므로 1이 아닌 0이 된다.

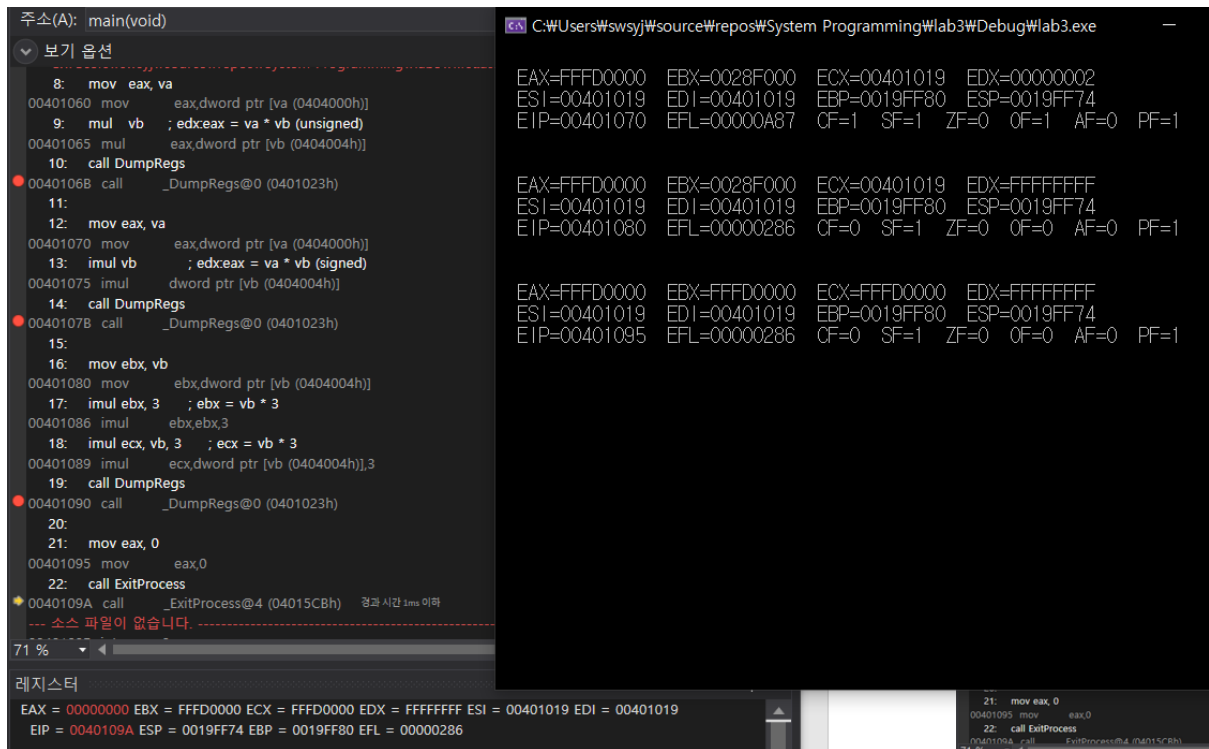
```
디스어셈블리 first.asm
주소(A): main(void)
보기 옵션
0040105F int 3
--- C:\Users\swsyj\source\repos\System Programming\lab3\Debug\lab3.exe
8: mov eax, va
00401060 mov eax, dword ptr [va (0404000h)]
9: mul vb ; edxeax = va * vb (unsigned)
00401065 mul eax, dword ptr [vb (0404004h)]
10: call DumpRegs
0040106B call _DumpRegs@0 (0401023h)
11:
12: mov eax, va
00401070 mov eax, dword ptr [va (0404000h)] 경과 시간 4ms
13: imul vb ; edxeax = va * vb (signed)
00401075 imul dword ptr [vb (0404004h)]
14: call DumpRegs
0040107B call _DumpRegs@0 (0401023h)
15:
16: mov ebx, vb
00401080 mov ebx, dword ptr [vb (0404004h)]
17: imul ebx, 3 ; ebx = vb * 3
00401086 imul ebx, ebx, 3
18: imul ecx, vb, 3 ; ecx = vb * 3
00401089 imul ecx, dword ptr [vb (0404004h)], 3
19: call DumpRegs
00401090 call _DumpRegs@0 (0401023h)
20:
21: mov eax, 0
00401095 mov eax, 0
22: call ExitProcess
0040109A call ExitProcess@4 (04015C8h)
71 %

레지스터
EAX = FFFD0000 EBX = 0028F000 ECX = 00401019 EDX = FFFFFFFF ESI = 00401019 EDI = 00401019
EIP = 00401080 ESP = 0019FF74 EBP = 0019FF80 EFL = 0000286
0x00404004 = FFFD0000

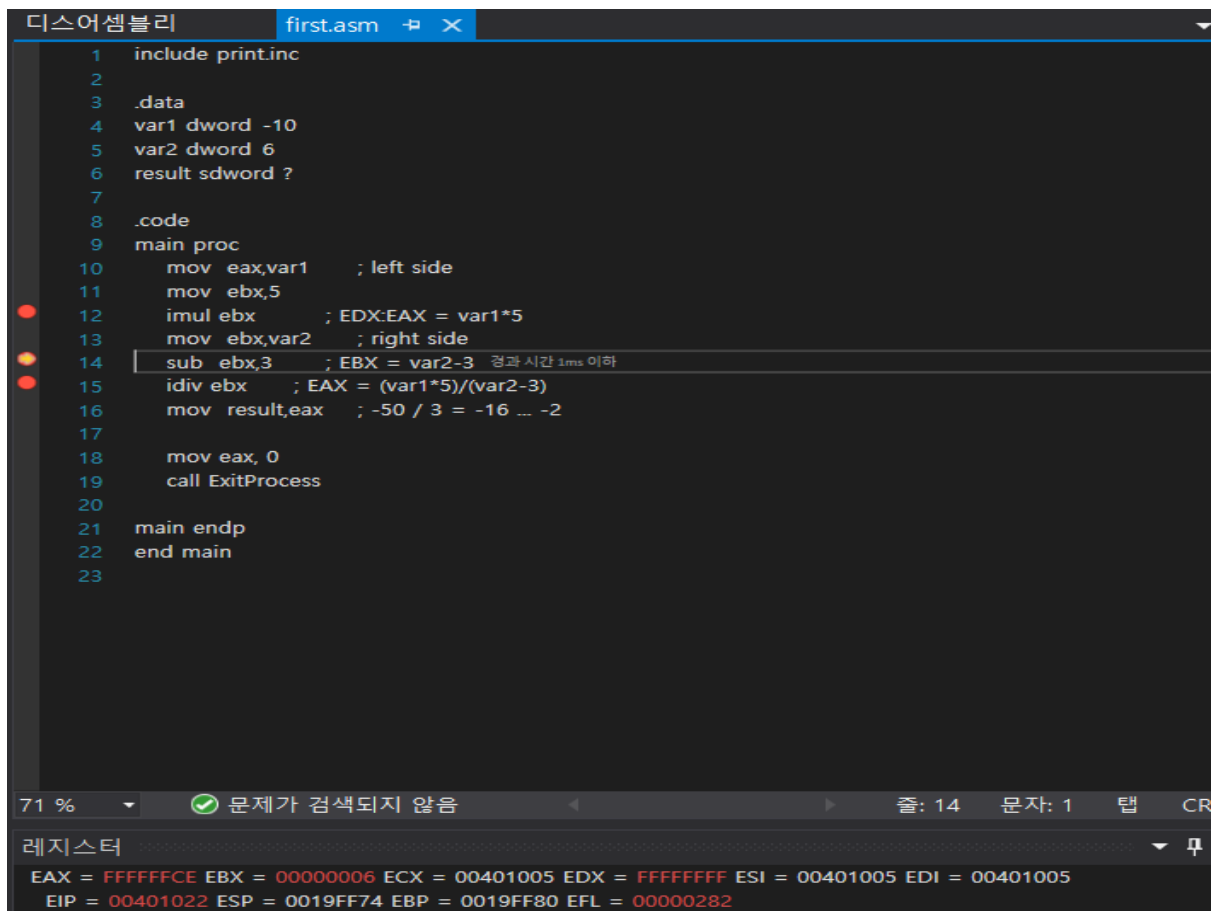
C:\Users\swsyj\source\repos\System Programming\lab3\Debug\lab3.exe
EAX=FFFD0000 EBX=0028F000 ECX=00401019 EDX=00000002
ESI=00401019 EDI=00401019 EBP=0019FF80 ESP=0019FF74
EIP=00401070 EFL=00000A87 CF=1 SF=1 ZF=0 OF=1 AF=0 PF=1

EAX=FFFD0000 EBX=0028F000 ECX=00401019 EDX=FFFFFFFF
ESI=00401019 EDI=00401019 EBP=0019FF80 ESP=0019FF74
EIP=00401080 EFL=00000286 CF=0 SF=1 ZF=0 OF=0 AF=0 PF=1

17: imul ebx, 3 ; ebx = vb * 3
00401086 imul ebx, ebx, 3
18: imul ecx, vb, 3 ; ecx = vb * 3
00401089 imul ecx, dword ptr [vb (0404004h)], 3
19: call DumpRegs
00401090 call _DumpRegs@0 (0401023h)
```



muldiv.asm: 계산하려는 식을 왼쪽, 오른쪽으로 나누어 계산을 하고 sign 계산을 위해 imul, idiv을 사용한다. 마지막엔 결과값을 eax에서 result로 옮겨준다.



```
12    imul ebx     ; EDX:EAX = var1*5
13    mov  ebx,var2 ; right side
14    sub  ebx,3    ; EBX = var2-3
15    idiv ebx     ; EAX = (var1*5)/(var2-3) 경과 시간 1ms 이하
16    mov  result,eax ; -50 / 3 = -16 ... -2
17
18    mov  eax, 0
19    call ExitProcess
20
21  main endp
22  end main
23
```

71 % 문제가 검색되지 않음 줄: 15 문자: 1

레지스터

EAX = FFFFFFFCE EBX = 00000003 ECX = 00401005 EDX = FFFFFFFF ESI = 00401005 EDI = 00401005
EIP = 00401025 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000206

```
14    sub  ebx,3    ; EBX = var2-3
15    idiv ebx     ; EAX = (var1*5)/(var2-3)
16    mov  result,eax ; -50 / 3 = -16 ... -2 경과 시간 1ms 이하
17
18    mov  eax, 0
19    call ExitProcess
20
21  main endp
22  end main
23
```

71 % 문제가 검색되지 않음 줄: 16 문자: 1

레지스터

EAX = FFFFFFFF0 EBX = 00000003 ECX = 00401005 EDX = FFFFFFFE ESI = 00401005 EDI = 00401005
EIP = 00401027 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000206

0x00404008 = 00000000

extadd.asm: value1, value2의 각 원소끼리의 덧셈이니 loop문 전에 clc로 cf를 초기화해주고 carry를 포함 덧셈이니 pushfd, popfd로 carry값을 저장해주고 loop문은 지날때마다 ecx값이 감소하는걸 볼 수 있다. 그리고 result에는 각 원소끼리의 덧셈 값이 총 4개 저장될 것이다.

```
1  include print.inc
2
3  .data
4  value1 DWORD 0def0h, 9abch, 5678h, 1234h
5  value2 DWORD 5678h, 1234h, 0def0h, 9abch
6  result DWORD 4 dup (?)
7
8  .code
9  main proc
10     mov esi, 0
11     mov ecx, LENGTHOF value1
12     clc             ; CF = 0
13
14 L1: mov eax,value1[esi] ; value1's digit(32-bit)
15     adc eax,value2[esi] ; value2's digit(32-bit)
16     pushfd           ; save CF
17     mov result[esi],eax ; store partial sum
18     add esi,4         ; next index
19     popfd            ; restore CF
20     loop L1          ; repeat the loop
21
22     mov eax, 0
23     call ExitProcess
24
25 main endp
26 end main
27
```

71 % 문제가 검색되지 않음 줄: 16 문자: 1

레지스터

EAX = 00013568 EBX = 0032D000 ECX = 00000004 EDX = 00401005 ESI = 00000000 EDI = 00401005
EIP = 00401027 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000202

16
17
18
19
20
21
22
23
24
25
26
27

pushfd ; save CF 경과 시간 1ms 이하

mov result[esi],eax ; store partial sum

add esi,4 ; next index

popfd ; restore CF

loop L1 ; repeat the loop

mov eax, 0

call ExitProcess

main endp

end main

71 %

문제가 검색되지 않음

줄: 16 문자: 1

레지스터

EAX = 000ACF0 EBX = 0032D000 ECX = 00000003 EDX = 00401005 ESI = 00000004 EDI = 00401005
EIP = 00401027 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000216

16
17
18
19
20
21
22
23
24
25
26
27

pushfd ; save CF 경과 시간 1ms 이하

mov result[esi],eax ; store partial sum

add esi,4 ; next index

popfd ; restore CF

loop L1 ; repeat the loop

mov eax, 0

call ExitProcess

main endp

end main

71 %

문제가 검색되지 않음

줄: 16 문자: 1

레지스터

EAX = 00013568 EBX = 0032D000 ECX = 00000002 EDX = 00401005 ESI = 00000008 EDI = 00401005
EIP = 00401027 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000202

16
17
18
19
20
21
22
23
24
25
26
27

pushfd ; save CF 경과 시간 1ms 이하

mov result[esi],eax ; store partial sum

add esi,4 ; next index

popfd ; restore CF

loop L1 ; repeat the loop

mov eax, 0

call ExitProcess

main endp

end main

71 %

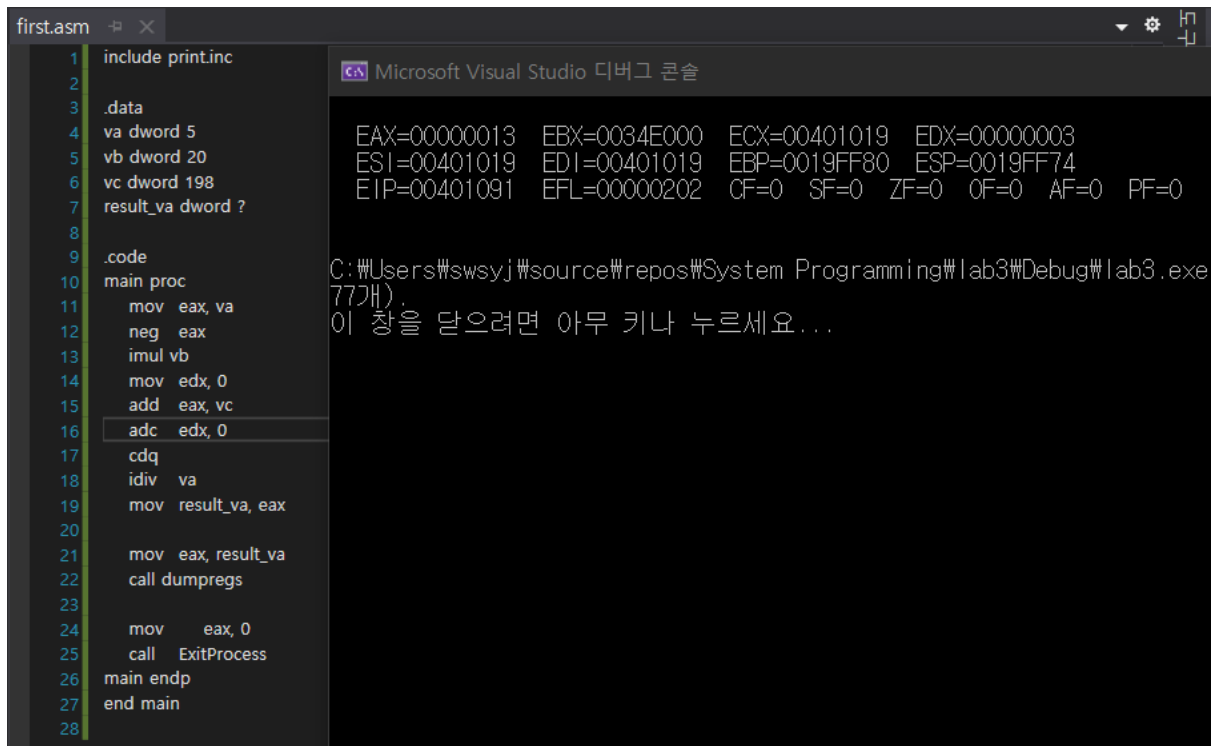
문제가 검색되지 않음

줄: 16 문자: 1

레지스터

EAX = 000ACF0 EBX = 0032D000 ECX = 00000001 EDX = 00401005 ESI = 0000000C EDI = 00401005
EIP = 00401027 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000216

2.



The screenshot shows the Microsoft Visual Studio interface. On the left, the assembly file 'first.asm' is open, displaying the following code:

```
1 include print.inc
2
3 .data
4 va dword 5
5 vb dword 20
6 vc dword 198
7 result_va dword ?
8
9 .code
10 main proc
11     mov     eax, va
12     neg     eax
13     imul    vb
14     mov     edx, 0
15     add     eax, vc
16     adc     edx, 0
17     cdq
18     idiv    va
19     mov     result_va, eax
20
21     mov     eax, result_va
22     call    dumpregs
23
24     mov     eax, 0
25     call    ExitProcess
26 main endp
27 end main
28
```

On the right, the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) is open, showing the following output:

```
EAX=00000013  EBX=0034E000  ECX=00401019  EDX=00000003
ESI=00401019  EDI=00401019  EBP=0019FF80  ESP=0019FF74
EIP=00401091  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

C:\Users\swsyj\source\repos\System Programming\lab3\Debug\lab3.exe
77개)
이 창을 닫으려면 아무 키나 누르세요...
```

초기값들을 설정해주고 결과값 va는 따로 설정해줬다. neg연산으로 -va를 만들고 음수가 됐으므로 vb와 imul을 해준다. 그리고는 vc랑 확장된 덧셈을 해야하니 edx를 0으로 만들어놓고 eax랑 더해 주고 edx를 0과 adc해주어 carry값을 유지해준다. 그 뒤 부호있는 나눗셈을 하기 전 cdq를 통해 부호확장을 해주고 va=5와 idiv를 해주고 result_va에 저장해준다. 최종결과는 eax에 13, edx에 3으로 나타나고 10진수로 나타내면 19와 3이므로 각각 몫과 나머지인 것을 알 수 있고 계산결과가 맞는 것을 볼 수 있다.