

< 과제 5 >

2017253041_홍성우

1.

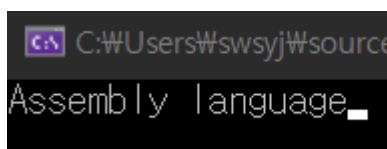
strcpy.asm: source의 문자열을 rep movsb 명령어로 ecx만큼 반복 수행하여 target에 옮겨 출력한 것이다.

```
9: cld ; DF=0 (forward)
00401060 cld
10: mov esi, OFFSET source
00401061 mov esi,offset source (0404000h)
11: mov edi, OFFSET target
00401066 mov edi,offset target (0404012h)
12: mov ecx, LENGTHOF source ; set REP count
0040106B mov ecx,12h
13: rep movsb ; repeat move byte
00401070 rep movsb byte ptr es:[edi],byte ptr [esi]
14:
15: mov edx, OFFSET target
00401072 mov edx,offset target (0404012h) 경과 시간 1ms 이하
16: call WriteString
00401077 call _WriteString@0 (0401014h)
17: mov eax, 0
0040107C mov eax,0
18: call ExitProcess
00401081 call _ExitProcess@4 (04015ABh)
--- 소스 파일이 없습니다. ---
00401086 int 3
00401087 int 3
```

78 %

레지스터

EAX = 0019FFCC EBX = 003FB000 ECX = 00000000 EDX = 00401019 ESI = 00404012
EDI = 00404024 EIP = 00401072 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000246



blcmp.asm: source와 target의 원소를 repe cmpsd 명령어를 이용하여 다룰때까지 비교하는 것이다. 비교한 뒤 다르다면 msg_ne의 주소를 edx에 옮겨 출력하고 같다면 msg_eq의 주소를 edx에 옮겨 출력하게 된다. 이 경우엔 "Not equal"이 출력된다.

```

13:  cld          ; DF=0 (forward)
00401060  cld
14:  mov esi, OFFSET source
00401061  mov     esi,offset source (0404000h)
15:  mov edi, OFFSET target
00401066  mov     edi,offset target (0404010h)
16:  mov ecx, LENGTHOF source
0040106B  mov     ecx,4
17:  repe cmpsd    ; repeat compare dword
00401070  repe cmps    dword ptr [esi],dword ptr es:[edi]
18:
19:  mov edx, OFFSET msg_ne
00401072  mov     edx,offset msg_ne (0404026h)  경과 시간 1ms 이하
20:  jne L1
00401077  jne     L1 (040107Eh)
21:  mov edx, OFFSET msg_eq
00401079  mov     edx,offset msg_eq (0404020h)
22: L1:
23:  call WriteString
0040107E  call    _WriteString@0 (0401014h)
24:  mov eax, 0
00401083  mov     eax,0

```

78 %

레지스터

EAX = 0019FFCC EBX = 003E9000 ECX = 00000002 EDX = 00401019 ESI = 00404008
EDI = 00404018 EIP = 00401072 ESP = 0019FF74 EBP = 0019FF80 EFL = 00000202

C:\Users\W...
Not equal

fill.asm: buffer를 0FFh로 채우는 것이다.

```

8:  cld          ; DF=0 (forward)
00401010  cld
9:  mov edi, OFFSET buffer
00401011  mov     edi,offset buffer (0404000h)
10:  mov ecx, LENGTHOF buffer
00401016  mov     ecx,32h
11:  mov al, 0FFh
0040101B  mov     al,0FFh
12:  rep stosb    ; repeat move byte
0040101D  rep stos    byte ptr es:[edi]
13:
14:  mov eax, 0
0040101F  mov     eax,0  경과 시간 1ms 이하
15:  call ExitProcess
00401024  call    _ExitProcess@4 (040102Fh)
--- 소스 파일이 없습니다. -----
00401029  int     3
0040102A  int     3
0040102B  int     3
0040102C  int     3
0040102D  int     3
0040102E  int     3

```

78 %

레지스터

EAX = 0019FFFF EBX = 0021B000 ECX = 00000000 EDX = 00401005 ESI = 00401005
EDI = 00404032 EIP = 0040101F ESP = 0019FF74 EBP = 0019FF80 EFL = 00000246

2.

call.asm: call dumpregs로 출력을 한번하고 주소를 옮긴 뒤 call esi로 한번 더 출력해준 것이다.

```
5:  mov eax, 1
00401060  mov     eax,1
6:  mov ebx, 2
00401065  mov     ebx,2
7:  mov ecx, 3
0040106A  mov     ecx,3
8:  call DumpRegs
0040106F  call    _DumpRegs@0 (0401023h)
9:
10:  mov esi, offset DumpRegs
00401074  mov     esi,offset _DumpRegs@0 (0401023h)
11:  call esi
00401079  call    esi
12:
13:  mov eax, 0
0040107B  mov     eax,0    경과 시간 3ms 이하
14:  call ExitProcess
00401080  call    _ExitProcess@4 (04015ABh)
--- 소스 파일이 없습니다. -----
00401085  int     3
78 %
```

레지스터

EAX = 00000001	EBX = 00000002	ECX = 00000003	EDX = 00401019	ESI = 00401023
EDI = 00401019	EIP = 0040107B	ESP = 0019FF74	EBP = 0019FF80	EFL = 00000246

```
C:\Users\swsyj\source\repos\System Programming\lab3\Debug...
EAX=00000001  EBX=00000002  ECX=00000003  EDX=00401019
ESI=00401019  EDI=00401019  EBP=0019FF80  ESP=0019FF74
EIP=00401074  EFL=00000246  CF=0   SF=0   ZF=1   OF=0   AF=0   PF=1

EAX=00000001  EBX=00000002  ECX=00000003  EDX=00401019
ESI=00401023  EDI=00401019  EBP=0019FF80  ESP=0019FF74
EIP=0040107B  EFL=00000246  CF=0   SF=0   ZF=1   OF=0   AF=0   PF=1
```

reverse.asm: aName의 문자들을 stack에 push한 다음에 pop하여 원상복구하는 것이다.

```
8:  mov ecx, LENGTHOF aName - 1
00401010  mov     ecx,11h
9:  mov esi,0
00401015  mov     esi,0
10: L1: movzx eax, aName[esi] ; zero extension
0040101A  movzx   eax,byte ptr aName (0404000h)[esi]
11:  push eax
00401021  push    eax
12:  inc esi
00401022  inc     esi
13:  loop L1
00401023  loop    L1    경과 시간 1ms 이하
14:
15:  mov ecx, LENGTHOF aName - 1
00401025  mov     ecx,11h
16:  mov esi, 0
0040102A  mov     esi,0
17: L2: pop eax
78 %
```

레지스터

EAX = 00000059	EBX = 00359000	ECX = 00000011	EDX = 00401005	ESI = 00000001
EDI = 00401005	EIP = 00401023	ESP = 0019FF70	EBP = 0019FF80	EFL = 00000202

sumof.asm: call sumof를 통해 eax에 모두의 합을 저장하는 프로시저를 부르고 원래 위치로 ret하여 eax를 theSum에 옮겨준 것이다.

```

0040102F call SumOf (0401043h)
12: mov theSum, eax
00401034 mov dword ptr [theSum (0404000h)],eax
13:
14: mov eax, 0
00401039 mov eax, 0
15: call ExitProcess
0040103E call _ExitProcess@4 (0401052h)
24: add eax,ebx
00401043 add eax,ebx
25: add eax,ecx
00401045 add eax,ecx
26: ret
00401047 ret
--- 소스 파일이 없습니다. -----
00401048 int 3
00401049 int 3
0040104A int 3
0040104B int 3
0040104C int 3
0040104D int 3
0040104E int 3
0040104F int 3
00401050 int 3
00401051 int 3
00401052 jmp dword ptr [__imp__ExitProcess@4 (0405000h)]
78 %
레지스터
EAX = 00006000 EBX = 00002000 ECX = 00003000 EDX = 0040100A ESI = 0040100A
EDI = 0040100A EIP = 00401047 ESP = 0019FF70 EBP = 0019FF80 EFL = 00000206

```

arraysum.asm: 배열의 합을 구하는 arraysum을 호출하여 계산 후 call dumpregs로 출력한 것이다.

```

Microsoft Visual Studio 디버그 콘솔

EAX=0000F000 EBX=002A8000 ECX=00000000 EDX=0040101E
ESI=00404014 EDI=0040101E EBP=0019FF80 ESP=0019FF74
EIP=00401086 EFL=00000206 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=1

```

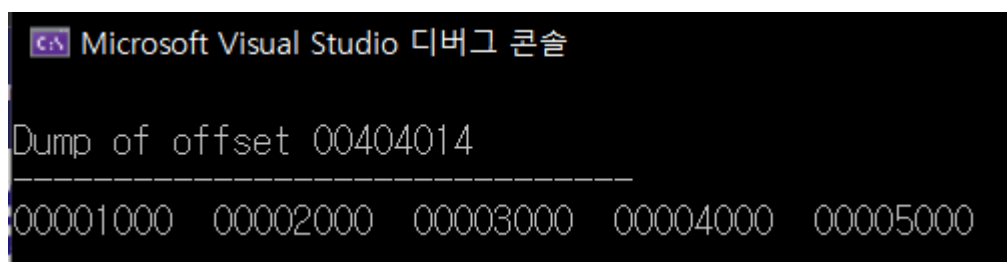
```

10 ArraySum proc
11     mov eax,0           ; set the sum to zero
12 L1: add eax,[esi]       ; add each integer to sum
13     add esi,4           ; point to next integer
14     loop L1             ; repeat for array size
15     ret                경과 시간 1ms 이하
16 ArraySum endp
17
18 .data
19 array DWORD 1000h, 2000h, 3000h, 4000h, 5000h
20 theSum DWORD ?
21 .code
22 main PROC
23     mov esi,OFFSET array
24     mov ecx,LENGTHOF array
25     call ArraySum
26     mov theSum,eax
27     call DumpRegs
28     mov eax, 0
29     call ExitProcess
30 main ENDP
78 %
문제가 검색되지 않음
레지스터
EAX = 0000F000 EBX = 002A8000 ECX = 00000000 EDX = 0040101E ESI = 00404014
EDI = 0040101E EIP = 0040106C ESP = 0019FF70 EBP = 0019FF80 EFL = 00000206

```

3.

```
1  include print.inc
2
3  .data
4  source dword 1000h, 2000h, 3000h, 4000h, 5000h
5  target dword lengthof source dup(?)
6
7  .code
8  main proc
9      mov esi, offset source
10     mov edi, offset target
11     mov ecx, lengthof source
12     call copyArray
13
14     mov esi, offset target
15     mov ecx, lengthof target
16     mov ebx, type target
17     call dumpmem
18
19     mov eax, 0
20     call ExitProcess
21 main endp
22
23 copyArray proc uses esi edi ecx
24     cld
25     rep movsd
26     ret
27 copyArray endp
28
29 end main
30
```



Microsoft Visual Studio 디버그 콘솔

Dump of offset 00404014

00001000 00002000 00003000 00004000 00005000

copyArray 프로시저는 esi, edi, ecx를 레지스터를 통하여 전달 받으므로 uses operator를 사용해주고, source에서 target으로 정수 배열이 잘 복사됐는지 확인하기 위해 call dumpmem을 사용해 출력한 것이다.