# Lab 1: Reverse engineer assembly code

## Purpose

The learning objective of this lab is for students to become familiar with assembly code and practice basic reverse engineering. This skill will become useful later in the course when you will work with exploits and try to understand their behavior at the level of machine code.

## Task 1:

Write C-pseudocode for assembly code shown below. You should be able to recognize specific C-constructs such as assignmens, loops, conditionals and functions.

```
push      ebp
mov       ebp, esp
mov       eax, [ebp+8]
sub       eax, 41h
jz  short loc_caseA
dec       eax
jz  short loc_caseB
dec       eax
jz  short loc_caseC
mov       al, 5Ah
movzx eax, al
pop       ebp
retn
loc_caseC:
mov       al, 43h
movzx     eax, al
pop       ebp
retn
...
...
loc_caseB:
mov    al, 42h
movzx  eax, al
pop.    ebp
retn
loc_caseA:
mov    al, 41h
movzx eax, al
pop    ebp
retn
```

## Task 2:

Write C-pseudocode for assembly code shown below. You should be able to recognize specific C-constructs such as assignmens, loops, conditionals and functions.

```
    cmp edi, 5
    ja       short loc_10001141
    jmp      ds:off_100011A4[edi*4]
loc_10001125:
    mov      esi, 40h
    jmp      short loc_10001145
loc_1000112C:
    mov      esi, 25h
    jmp      short loc_10001145
loc_10001133:
    mov      esi, 39h
    jmp      short loc_10001145
loc_1000113A:
    mov      esi, 10h
    jmp      short loc_10001145
loc_10001141:
    mov     esi, [esp+0Ch]
...
...
off_100011A4:
    dd offset loc_10001125
    dd offset loc_10001125
    dd offset loc_1000113A
    dd offset loc_1000112C
    dd offset loc_10001133
    dd offset loc_1000113A
```

## Task 3:

The code below calls a function. What type of calling convention is being used? What are the fundamental operations of the call convention you have identified?

```
00401002  mov    edi, ds:printf
00401008  xor    esi, esi
0040100A  lea    ebx, [ebx+0]
00401010 loc_401010:
00401010  push   esi
00401011  push   offset StrFormat        ; "%d\n"
00401016  call   edi                     ; printf
00401018  inc    esi
```

```
09: 00401019  add   esp, 8
10: 0040101C  cmp   esi, 0Ah
11: 0040101F  jl    short loc_401010
12: 00401021  push  offset aEnd          ; "end\n"
13: 00401026  call  edi                  ; printf
14: 00401028  add   esp, 4
```

## Task 4:

The assembly code shown below belongs to a C-function called from main(). The function takes two parameters: a pointer to an array and the length of the array. Assume that when the function is called the first argument points to a byte array containing the following values in sequence: *21, 7, 0, 16, 10, 12, 18*. The second argument is set to the length of the array, that is equal to value *7* in this case.

Write C-pseudocode for assembly code shown below. You should be able to recognize specific C-constructs such as assignmens, loops, conditionals and functions.

The *puts()* function called att offset 0x6D5 prints out a word from the English language. What word is it? The only argument to *puts()* is pushed on the stack on the line before.

```
.text:00000627 ; |||||||||||||| S U B R O U T I N E ||||||||||||||||||||||||||||||||
.text:00000627
.text:00000627 ; Attributes: bp-based frame
.text:00000627
.text:00000627                 public _Z1fPhj
.text:00000627 _Z1fPhj         proc near            ; CODE XREF: main+53↑p
.text:00000627
.text:00000627 var_5C          = dword ptr -5Ch
.text:00000627 var_50          = dword ptr -50h
.text:00000627 var_4C          = dword ptr -4Ch
.text:00000627 var_47          = dword ptr -47h
.text:00000627 var_43          = dword ptr -43h
.text:00000627 var_3F          = dword ptr -3Fh
.text:00000627 var_3B          = dword ptr -3Bh
.text:00000627 var_37          = dword ptr -37h
.text:00000627 var_33          = dword ptr -33h
.text:00000627 var_2F          = word ptr -2Fh
.text:00000627 var_2D          = byte ptr -2Dh
.text:00000627 var_2C          = byte ptr -2Ch
.text:00000627 var_C           = dword ptr -0Ch
```

```
.text:00000627 var_4         = dword ptr -4
.text:00000627 arg_0         = dword ptr  8
.text:00000627 arg_4         = dword ptr  0Ch
.text:00000627
.text:00000627               push  ebp
.text:00000628               mov   ebp, esp
.text:0000062A               push  ebx
.text:0000062B               sub   esp, 64h
.text:0000062E               call  __x86_get_pc_thunk_bx
.text:00000633               add   ebx, 199Dh
.text:00000639               mov   eax, [ebp+arg_0]
.text:0000063C               mov   [ebp+var_5C], eax
.text:0000063F               mov   eax, large gs:14h
.text:00000645               mov   [ebp+var_C], eax
.text:00000648               xor   eax, eax
.text:0000064A               mov   [ebp+var_47], 'DCBA'
.text:00000651               mov   [ebp+var_43], 'HGFE'
.text:00000658               mov   [ebp+var_3F], 'LKJI'
.text:0000065F               mov   [ebp+var_3B], 'PONM'
.text:00000666               mov   [ebp+var_37], 'TSRQ'
.text:0000066D               mov   [ebp+var_33], 'XWVU'
.text:00000674               mov   [ebp+var_2F], 'ZY'
.text:0000067A               mov   [ebp+var_2D], 0
.text:0000067E               sub   esp, 4
.text:00000681               push  20h          ; size_t
.text:00000683               push  0            ; int
.text:00000685               lea   eax, [ebp+var_2C]
.text:00000688               push  eax          ; void *
.text:00000689               call  _memset
.text:0000068E               add   esp, 10h
.text:00000691               mov   [ebp+var_50], 0
.text:00000698
.text:00000698 loc_698:                         ; CODE XREF: _Z1fPhj+A5↓j
.text:00000698               mov   eax, [ebp+var_50]
.text:0000069B               cmp   [ebp+arg_4], eax
.text:0000069E               jbe   short loc_6CE
.text:000006A0               mov   edx, [ebp+var_50]
.text:000006A3               mov   eax, [ebp+var_5C]
.text:000006A6               add   eax, edx
.text:000006A8               movzx eax, byte ptr [eax]
.text:000006AB               movzx eax, al
.text:000006AE               mov   [ebp+var_4C], eax
.text:000006B1               mov   edx, [ebp+var_4C]
.text:000006B4               mov   eax, [ebp+var_50]
.text:000006B7               add   eax, edx
```

```
.text:000006B9             movzx  eax, byte ptr [ebp+eax+var_47]
.text:000006BE             lea     ecx, [ebp+var_2C]
.text:000006C1             mov     edx, [ebp+var_50]
.text:000006C4             add     edx, ecx
.text:000006C6             mov     [edx], al
.text:000006C8             add    [ebp+var_50], 1
.text:000006CC             jmp    short loc_698
.text:000006CE ; ---------------------------------------------------------------------------
.text:000006CE
.text:000006CE loc_6CE:                          ; CODE XREF: _Z1fPhj+77↑j
.text:000006CE             sub     esp, 0Ch
.text:000006D1             lea     eax, [ebp+var_2C]
.text:000006D4             push    eax            ; char *
.text:000006D5             call    _puts
.text:000006DA             add     esp, 10h
.text:000006DD             mov     eax, 0
.text:000006E2             mov     ecx, [ebp+var_C]
.text:000006E5             xor     ecx, large gs:14h
.text:000006EC             jz     short loc_6F3
.text:000006EE             call    __stack_chk_fail_local
.text:000006F3
.text:000006F3 loc_6F3:                          ; CODE XREF: _Z1fPhj+C5↑j
.text:000006F3             mov     ebx, [ebp+var_4]
.text:000006F6             leave
.text:000006F7             retn
.text:000006F7 _Z1fPhj  endp
```