

Advanced Static Analysis

Aaron Sedlacek

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
mov     eax, [ebp+var_70]
mov     esi, [ebp+var_84]
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Agenda

x86 Assembly

IDA Pro

Code Constructs in Assembly

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

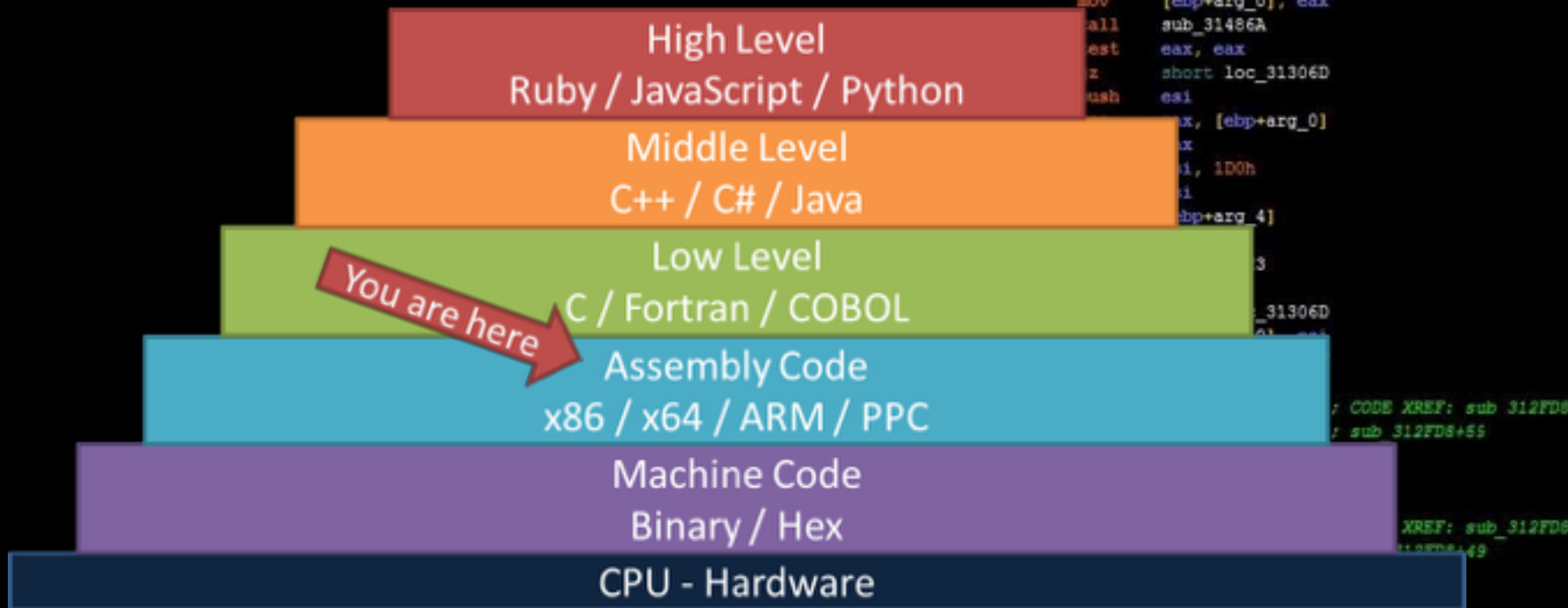
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Abstraction Levels



```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
```

```
mov     eax, [ebp+arg_0]
push    esi
push    esi, 1D0h
push    esi
mov     [ebp+arg_4], esi
```

```
mov     esi, 3
push    esi
call    sub_31306D
mov     esi, esi
```

```
mov     esi, esi
push    esi
CODE XREF: sub_312FD8
sub_312FD8+55
```

```
CODE XREF: sub_312FD8
sub_312FD8+49
```

```
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
CODE XREF: sub_312FD8
```

```
loc_31308C:
mov     [ebp+var_4], eax
CODE XREF: sub_312FD8
```

Reverse Engineering

- Assembly is (usually) the highest level abstraction layer that can be reliably and consistently recovered
 - This is why understanding assembly is so important for a malware analyst
 - x86, x64, MIPS, ARM, PowerPC, etc.

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_31486A
jz      short loc_31306D
push    esi
push    eax
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Why Is x86 So Popular??

- Provides the largest attack surface to malware authors
 - The majority of 32-bit PCs are x86
 - Consumer/business desktops/laptops
 - Commercial servers
 - 64-bit PCs can also run x86 32-bit binaries

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
test    eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Instructions

- Each instruction corresponds to opcodes that tell the CPU what operation to perform
 - ex. `mov ecx, 0x42` → `B9 42 00 00 00`

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


Operands

- Used to identify the data used by an instruction
 - Immediate operands - **mov** eax, **0x42**
 - fixed values, such as **0x42**
 - Register operands - **mov** eax, ecx
 - registers, such as ecx
 - Memory Addresses - **mov** [esp], eax
 - Denoted by brackets, [ecx]

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

```

```

push    esi
push    eax
push    edi
call    [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

Registers

- General Registers
 - Used by the CPU during execution
- Segment Registers
 - Used to track sections of memory
- Status Flags
 - Used to make decisions
- Instruction Pointer (eip)
 - Used to keep track of the next instruction

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

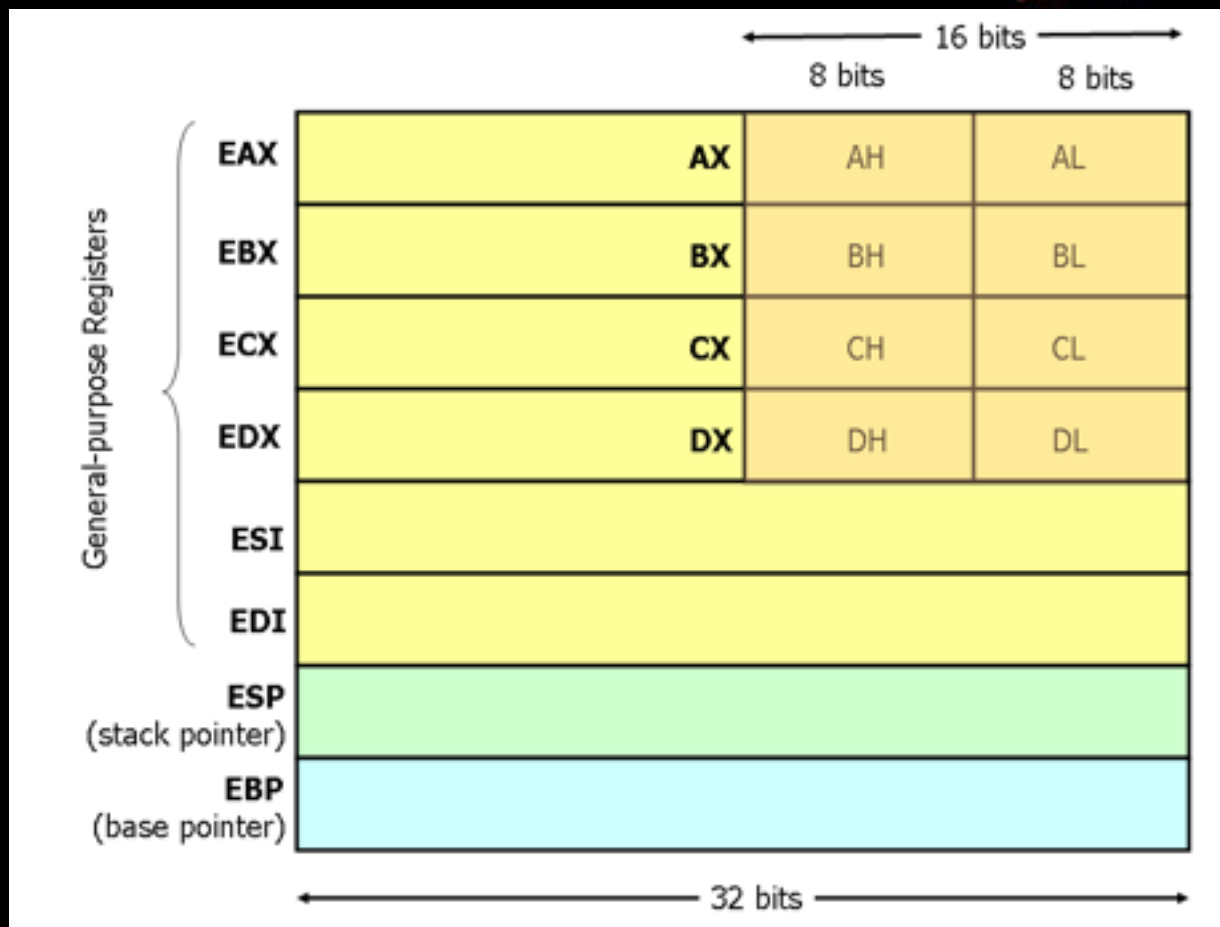
```
loc_313066:                                ; CODE XREF: sub_312FD8
; sub_312FD8+55
call    sub_31411B
```

```
loc_31306D:                                ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
```

```
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


The x86 Registers



Flags

- **EFLAGS** register is 32 bits in size
 - Each bit is a flag, either **set (1)** or **cleared (0)**
 - Flags are used to control CPU operations or indicate results of a CPU operation

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
push    [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Flags That Matter!

- **ZF** - Zero Flag
 - Set when the result of an operation is equal to **zero**
- **CF** - Carry Flag
 - Set when the result of an operation is too large or too small for the destination

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
jz      short loc_313066
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 100h
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Flags That Matter!

- SF - Sign Flag
 - Set when the result of an operation is negative
 - Also set when the most significant bit is set after an arithmetic operation
- TF - Trap Flag
 - Used for debugging, the processor will execute only 1 instruction at a time if this flag is set
 - Single step

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_313066
push    esi
mov     [ebp+arg_4], esi
push    eax
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
cmp     [ebp+arg_0], eax
jz      short loc_31308F
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

mov

- Mov instruction
 - it... moves things...
 - **mov** *destination, source*
 - **mov** `eax, [ebx+8]` puts the value at `ebx+8` into `eax`
 - **mov** `eax, ebx+8` is the same as **lea** `eax, [ebx+8]`, but is invalid

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     eax, 100h
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

mov and lea

- **lea** - Load Effective Address
 - **lea** *destination, source*
 - **lea** `eax, [ebx+8]` puts `ebx+8` into `eax`.

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


Arithmetic

- **add** *destination, value*
- **sub** *destination, value*
- **mul/div** *value*
 - multiplies/divides `eax` by value
 - **mul** stores the result in `edx:eax`
 - **div** stores the result in `eax`, and the remainder in `edx`

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jnz     short loc_31308F
cmp     [ebp+arg_0], eax
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

Arithmetic

- Logical Operators
 - **or**, **and**, **xor**
 - **and** *destination, value*
 - **xor** `eax, eax` ← quick way to set `eax` to zero
 - **shr/shl** *destination, count*
 - used to shift registers right/left
- **ror/rol** - rotation instructions
 - bits are rotated to the other end instead of falling off

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Fh
call sub_3140F3
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

Function Calls

- Function prologue
 - A few lines of code at the start of the function
 - Prepares the stack and registers for use
- Function epilogue
 - A few lines of code at the end of a function
 - Restores the stack and registers to their state before the function was called

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
mov     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    [ebp+arg_4]
test    eax, eax
jnz     short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

The Stack

push, pop, call, leave, enter, and ret instructions

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    esi
mov     [ebp+var_70], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

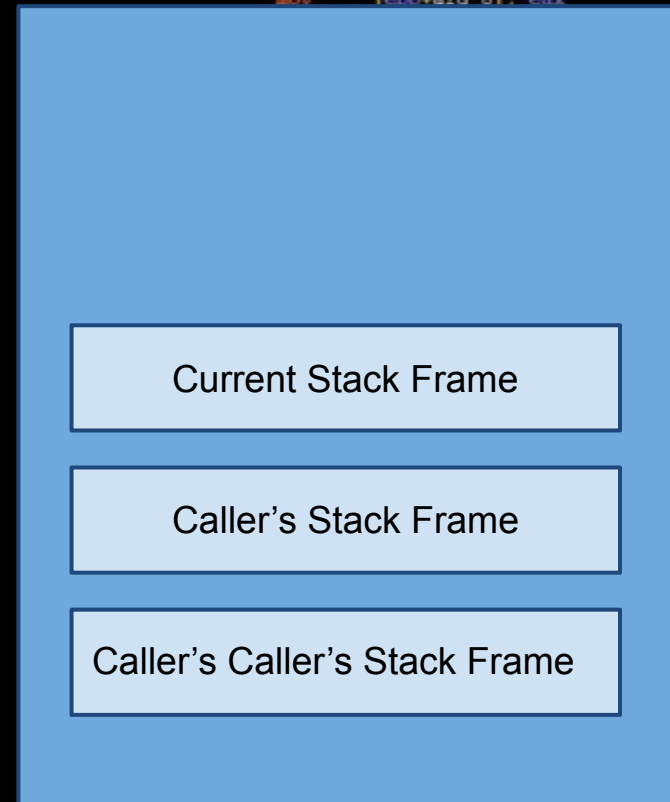
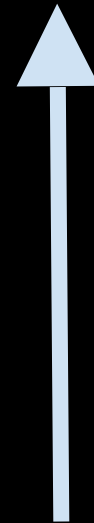
loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Stack Layout

Low Memory Address →

The stack grows upwards
towards zero

High Memory Address →



```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
```

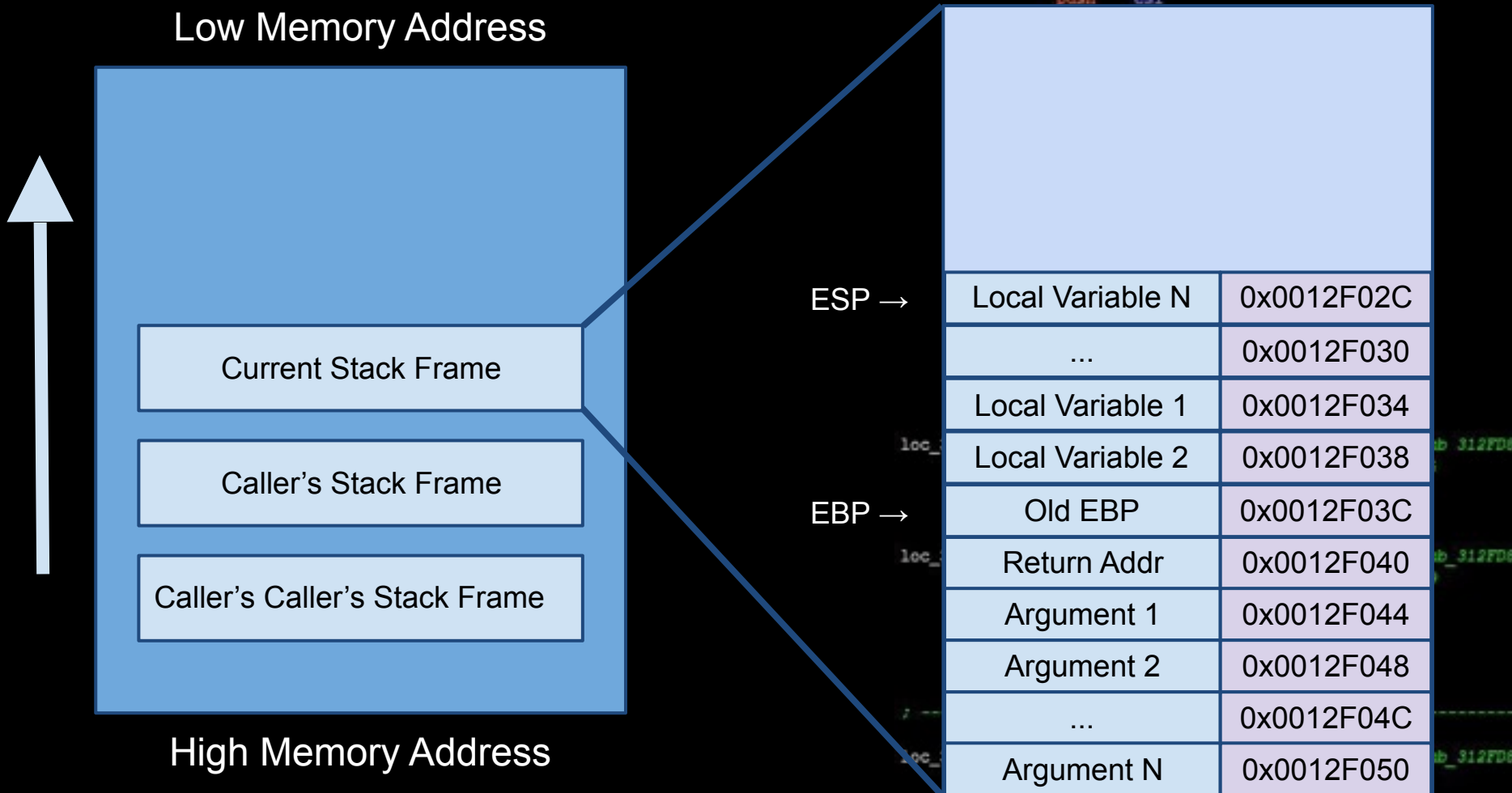
EF: sub_312FD8
FD8+55

EF: sub_312FD8
FD8+49

```
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Stack Layout



```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi

```


Calling Conventions

- **cdecl**
 - Calling function cleans the stack
- **stdcall**
 - Same as **cdecl**, except the called function cleans up the stack before returning
- **fastcall**
 - The first few arguments are passed in registers, usually **edx** and **ecx**
- **thiscall**

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

```

```

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313086:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

How Function Calls Work

- SUPER IMPORTANT:

- Within a given function's stack frame:
 - Local variables will be at a negative offset from ebp
 - Function Arguments will be at a positive offset

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cp      [ebp+arg_0], ebx
jz      short loc_313066
eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
eax, eax
jz      short loc_31306D
push    esi
test    eax, eax
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    44h
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Conditionals - test

- **test** *value, value*
 - Test is identical to the and instruction, but the operands are not modified
 - Only sets the ZF (Zero Flag)
 - **test** eax, eax

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
jz      short loc_313066
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

Conditionals - cmp

- **cmp** *dst, src*
 - Identical to the **sub** instruction, but does not affect operands
 - Zero flag and carry flag may be changed as a result

cmp dst, src	ZF	CF
dst = src	1	0
dst < src	0	1
dst > src	0	0

Branching

- `jmp location`
- A branch is a sequence of code that is conditionally executed
 - Usually uses jump instructions
- See the textbook for an awesome table of the different kinds of jump commands

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

Repeat Instructions

- **rep** instructions
 - Set of instructions for manipulating data buffers
 - Arrays of bytes
- **movsx, cmpsx, stosx, scasx**
 - **x** = **b**, **w**, or **d** for byte, word, and double word
- Ex. **rep movsb**
 - Copies a buffer of bytes
 - Copies from the address in esi to the address in edi
 - ecx is used as a counter

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+55
push    esi
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


Agenda

x86 Assembly

IDA Pro

Code Constructs in Assembly

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

IDA Pro

- Extremely powerful disassembler
 - We will be working with IDA Pro Free (v. 5.0)
- The disassembler of choice for malware analysts, reverse engineers, and vulnerability analysts
 - (feel free to use radare2 if you want)

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
test    [ebp+arg_0], [ebp+arg_0]
push    eax
mov     esi, 1D0h
test    esi, esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

IDA Pro - Interface

- Graph Mode
 - Provides a control flow graph to help you understand how the program works
- Text Mode
 - The more traditional view
 - Used to view data regions of a binary
- The [Spacebar] key switches between graph view and text view

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

IDA Pro - Navigation

- Windows → Reset Desktop
 - restores GUI elements to their defaults
- Cross-References and Links
 - Links are function names, locations, and offsets
 - Clicking a link and pressing [x] allows you to see everywhere that link is referenced and jump around

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1F0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jnz     short loc_31306D
cmp     [ebp+arg_1], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

IDA Pro - Navigation Band

- Presents a color-coded linear view of the loaded binary's address space
 - Light blue is library code (recognized by FLIRT signatures)
 - Red is compiler-generated code
 - Dark blue is user-written code
 - Pink is for imports
 - Gray is for defined data
 - Brown is for undefined data

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
call    sub_314623
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

IDA Pro - Jumping and Searching

- The **[g]** key lets you jump to a specific address or named location
 - ex. sub_401730 or printf
- You can also use the search option in the toolbar
 - Search → Next Code
 - Search → Text
 - Search → Sequence of Bytes

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jz      short loc_313066
mov     eax, [ebp+var_0]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 100h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


IDA Pro - Graphing Options

- IDA Pro can generate other graphs
 - Flow chart of current function
 - Function calls for the entire program
 - xrefs to/from a currently selected xref
 - User specified xref graphs

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

IDA Pro - Ease of Use

- Renaming functions and variables [n]
 - sub_401000 or arg_4 don't tell you much
 - Rename them to something more useful!
- Comments [:]
 - You can add comments to lines too!
- Formatting
 - You can change the format of data
 - ex. 0x61 → 'a' → 97
 - hex: [h/q], decimal: [h], char: [r], binary: [b]

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
push    eax
jz      short loc_31306D
push    esi
mov     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
loc_31306A:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

IDA Pro - Plugins!

- IDA Pro has plenty of plugins
 - Interface by using the IDA API
 - IDC
 - IDAPython (not available in IDA Free)
 - Some Useful Plugins:
 - Fentanyl - Patch Assembly in IDA
 - Hex-Rays Decompiler (Costs fat stacks of cash)
 - KANAL - Krypto ANALyzer

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
loc_313066:                                ; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Agenda

x86 Assembly

IDA Pro

Code Constructs in Assembly

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

LAB 6-01

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----

loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```

LAB 6-03

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```


Questions?

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

; -----

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

References

1. Sikorski, Michael, and Andrew Honig. Practical Malware Analysis the Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch, 2012. Print.

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    esi
mov     [ebp+arg_0], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```