What is ASLR?

A: Address

S: Space

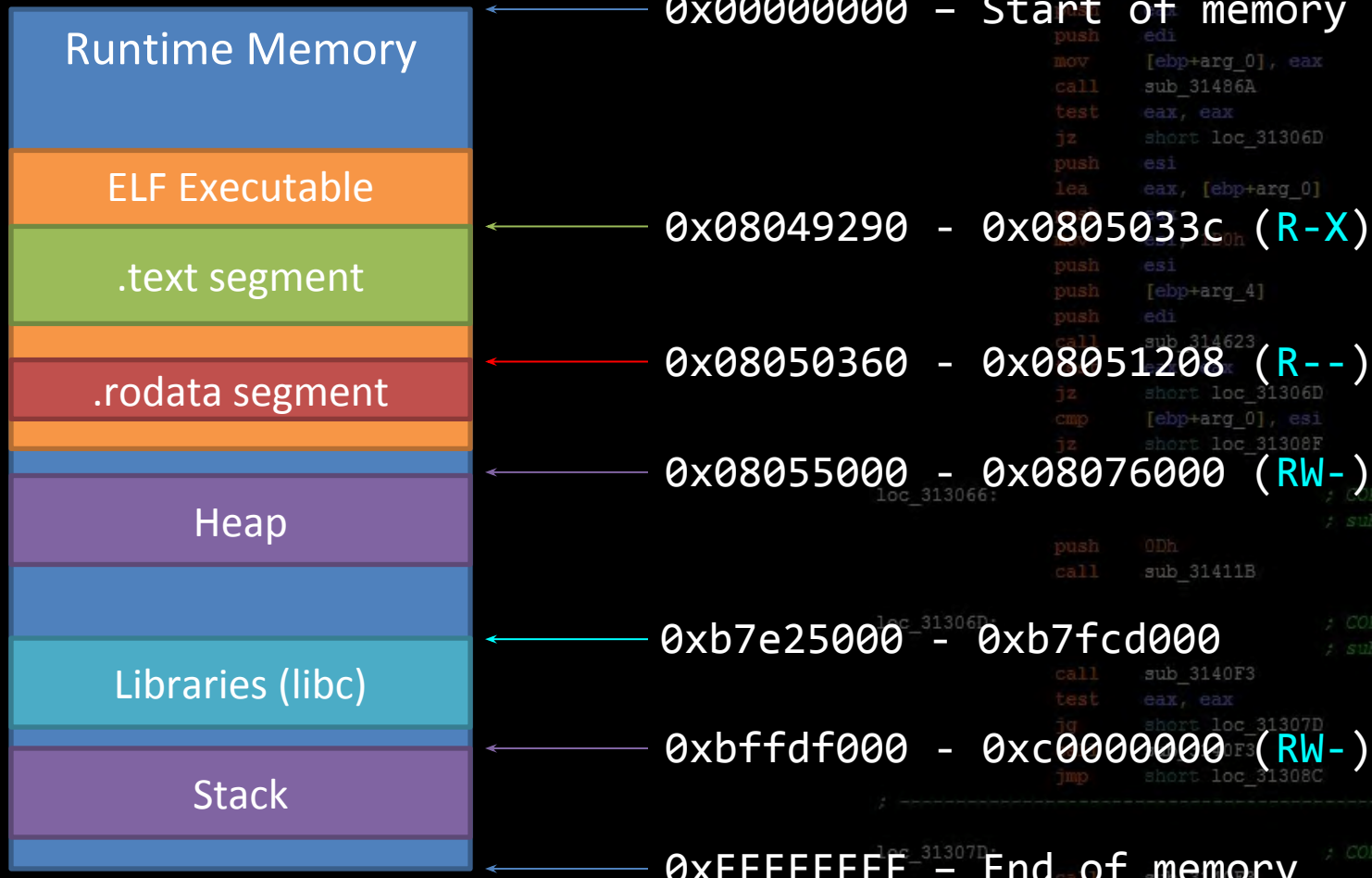
L: Layout

R: Randomization

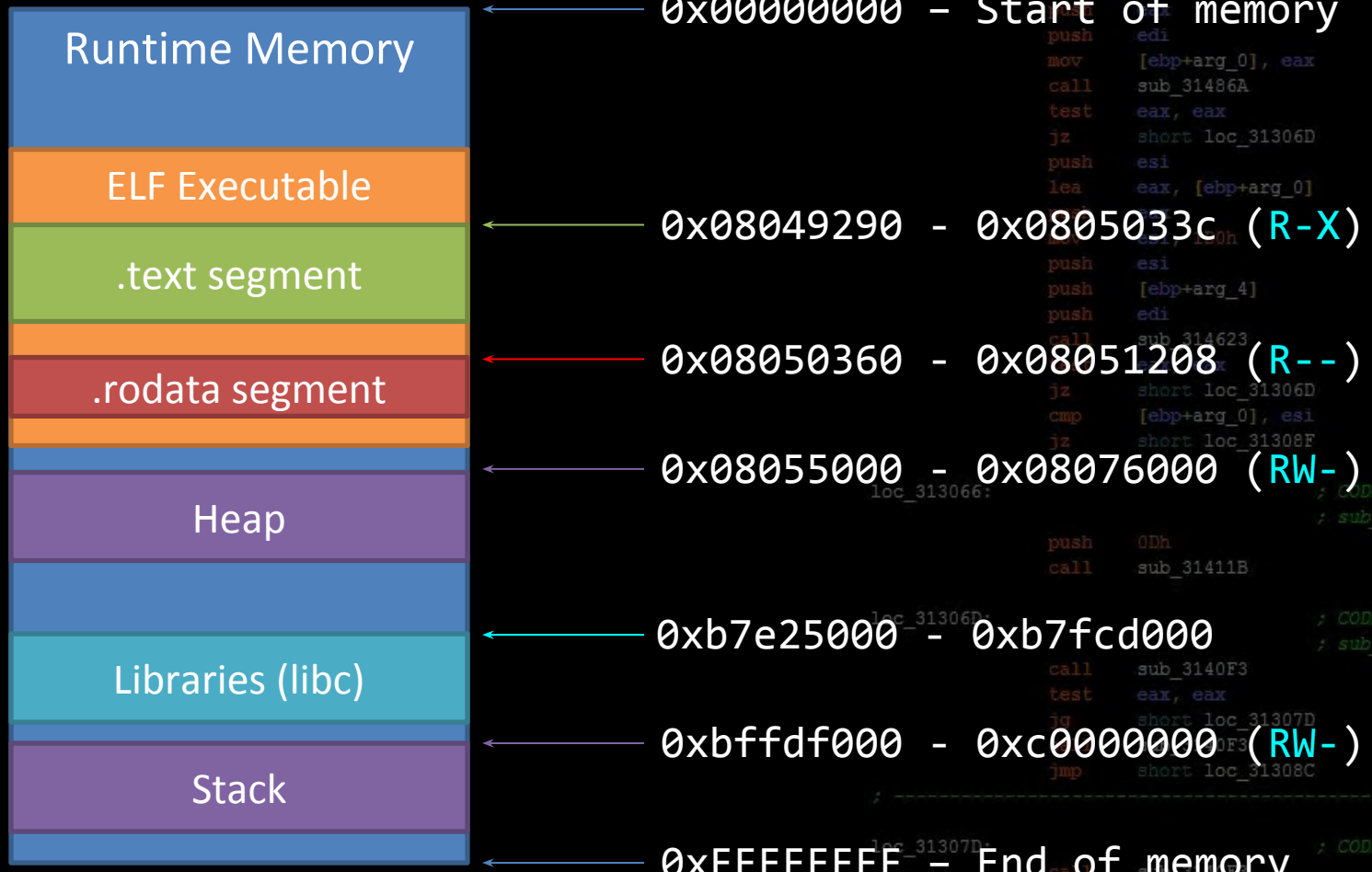
Course Terminology

- Address Space Layout Randomization
  - An exploit mitigation technology used to ensure that address ranges for important memory segments are random for every execution
  - Meant to mitigate exploits leveraging hardcoded stack, heap, code, libc addresses
  - Known as ASLR for short

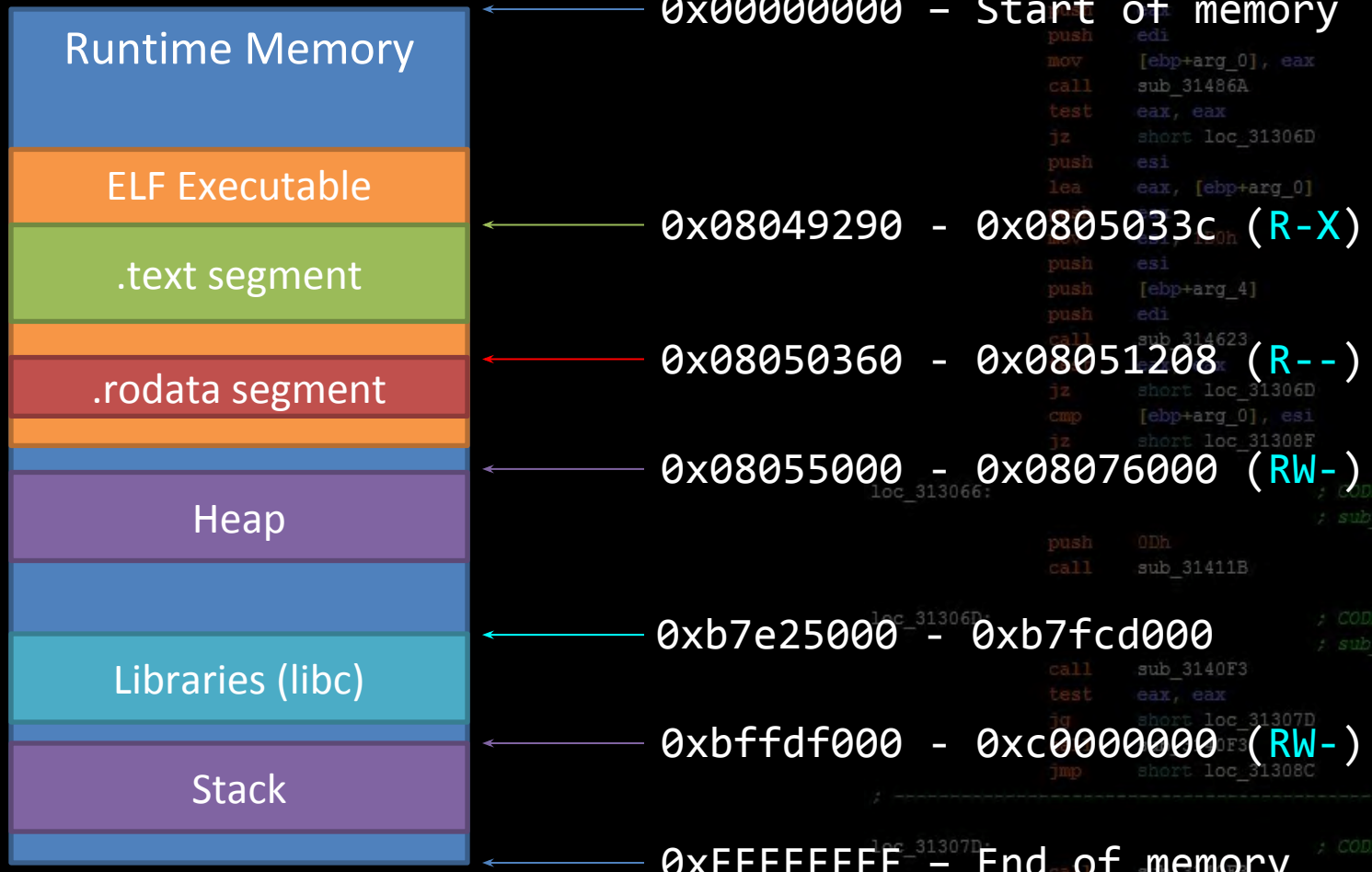
Runtime Process Without ASLR



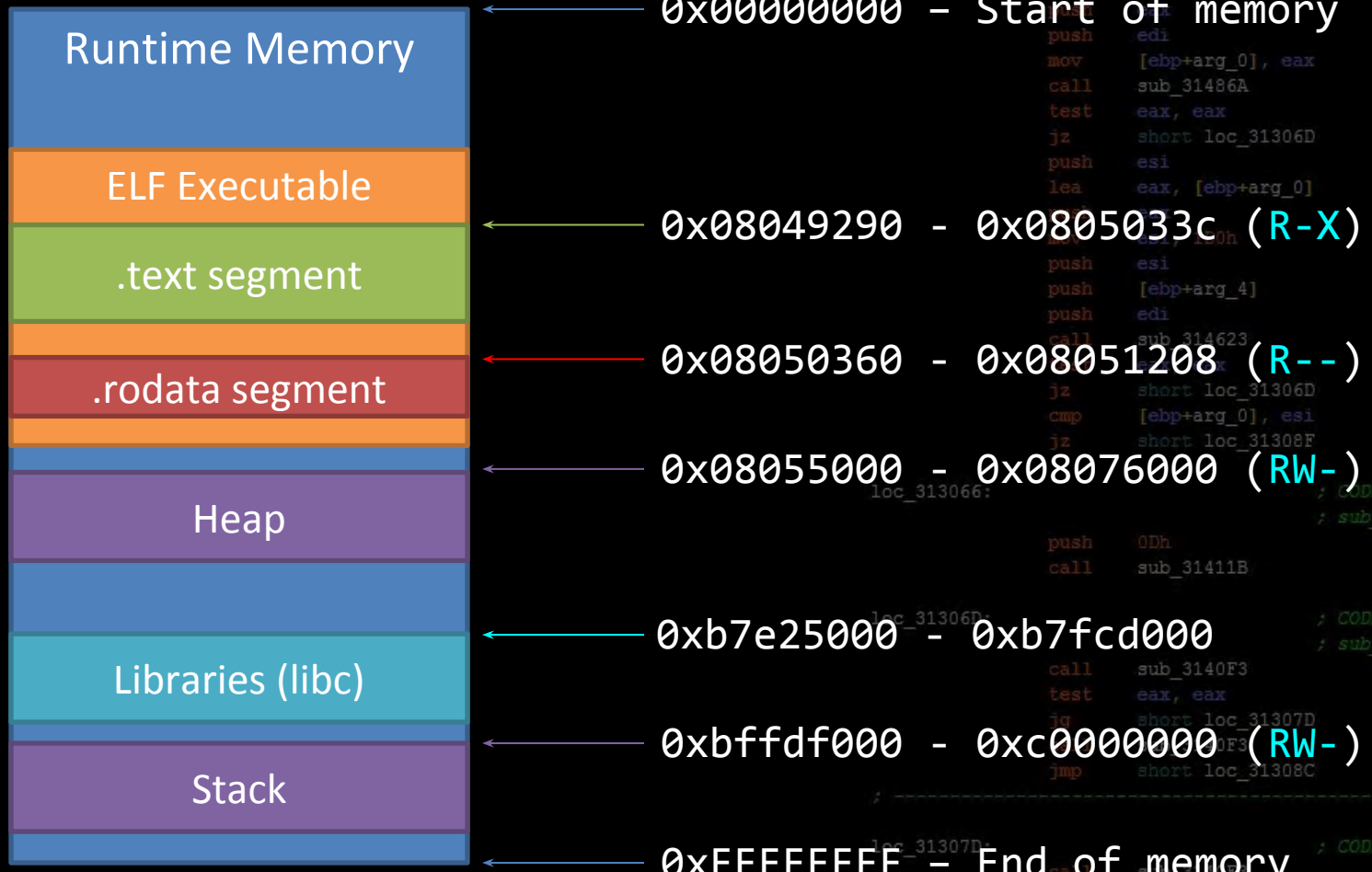
Run #1 Without ASLR



Run #2 Without ASLR



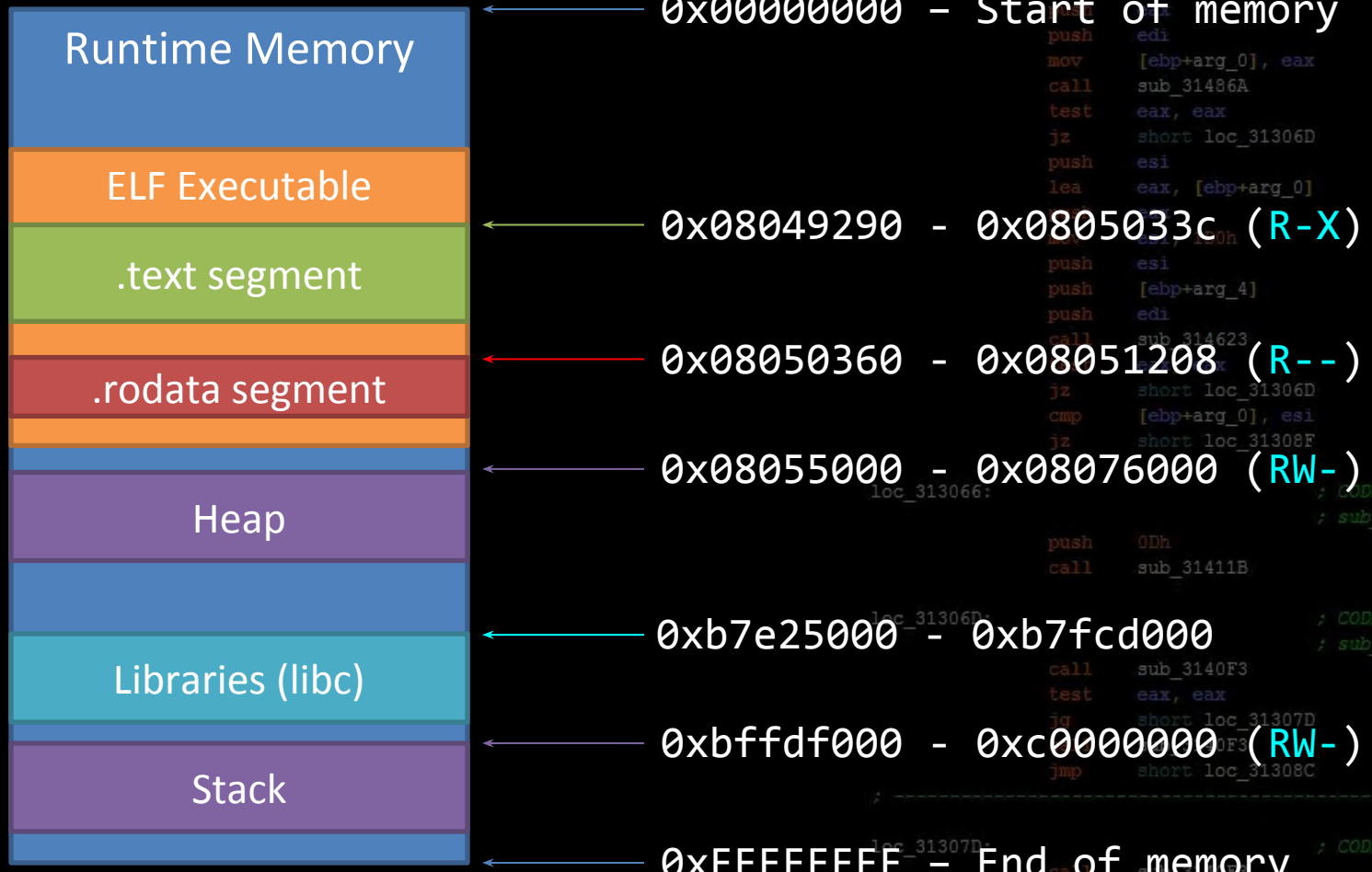
Run #3 Without ASLR



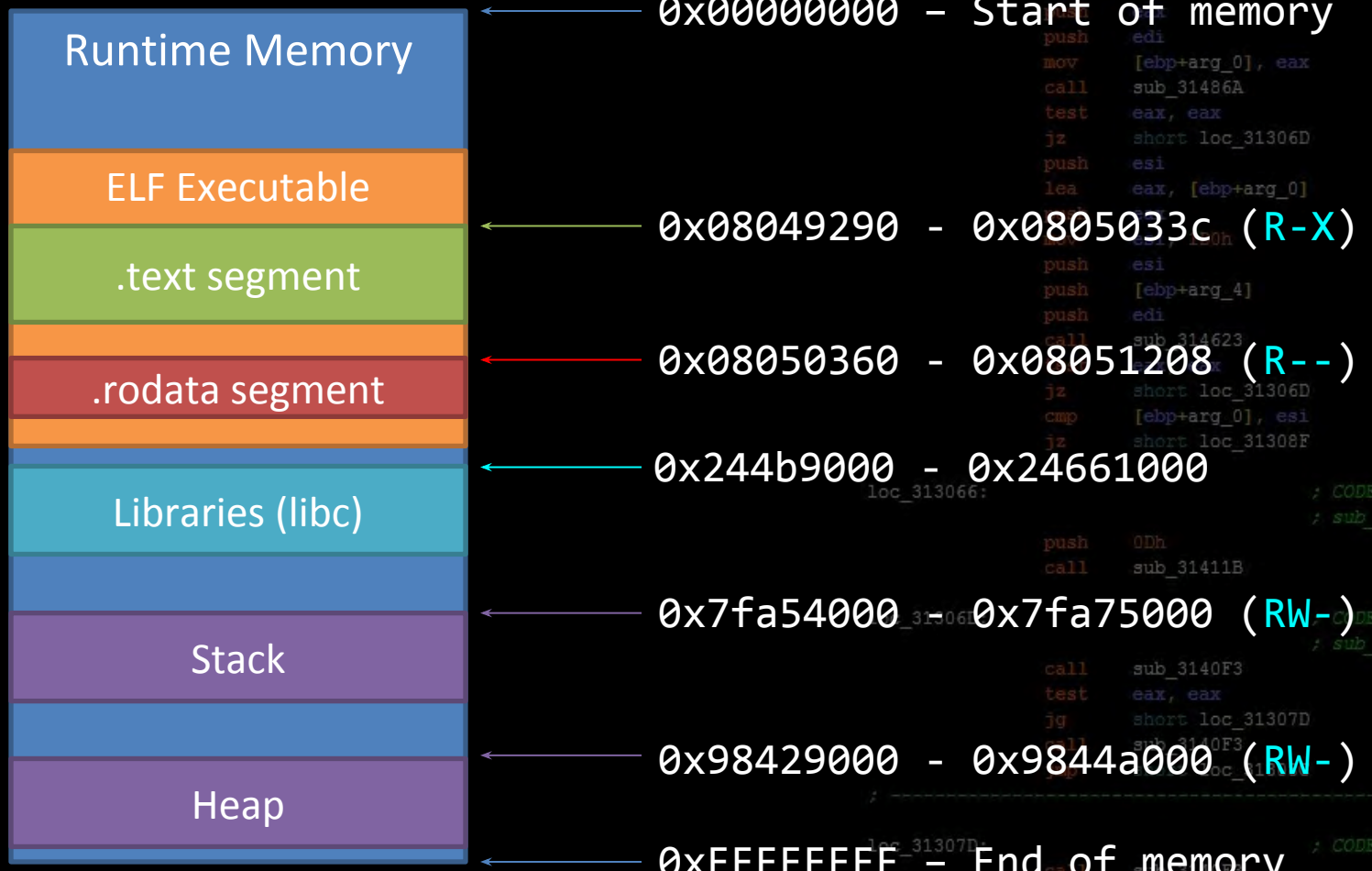
ya so, nothing changes...

```
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+59
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
```

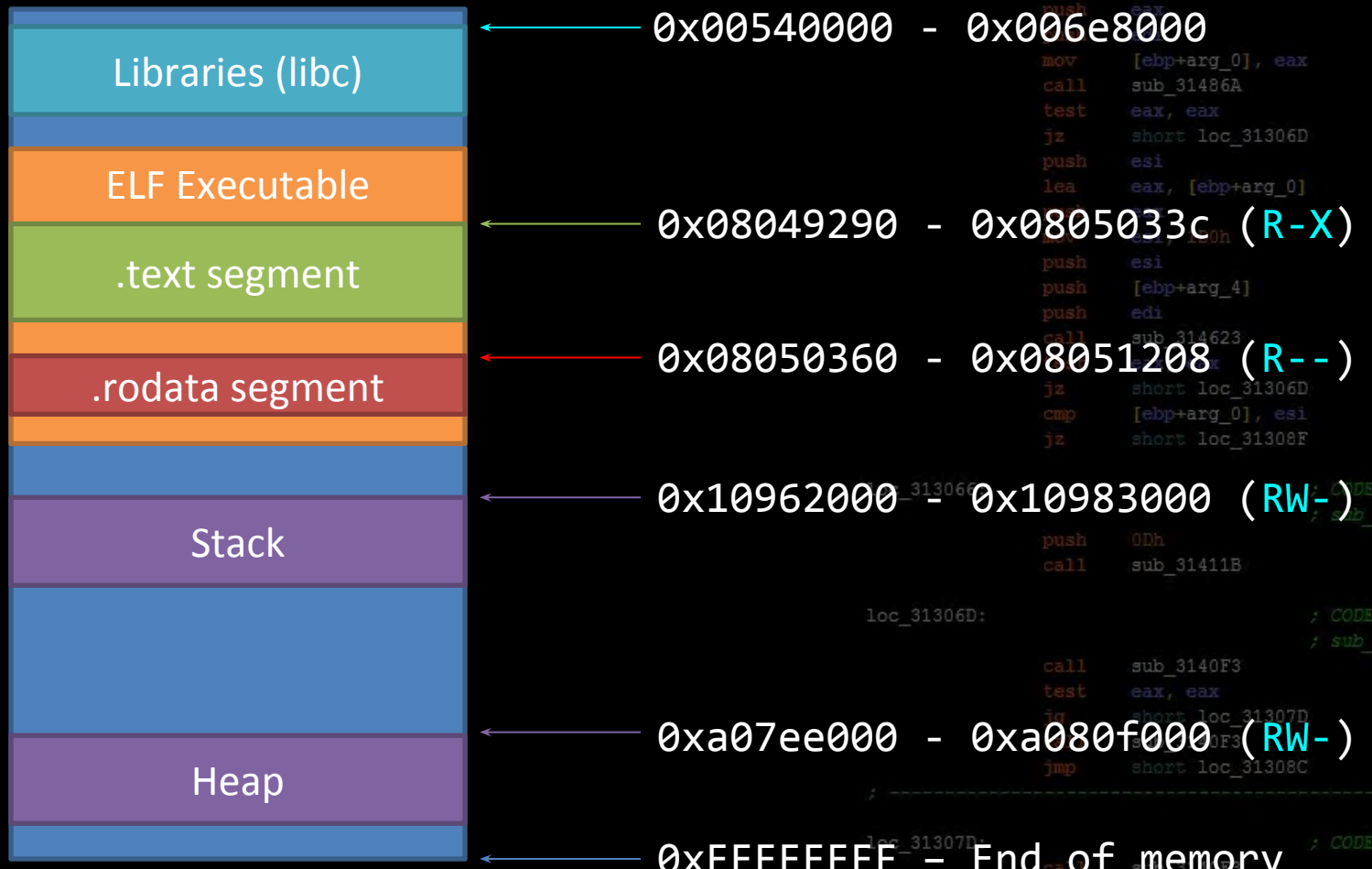
Runtime Process Without ASLR



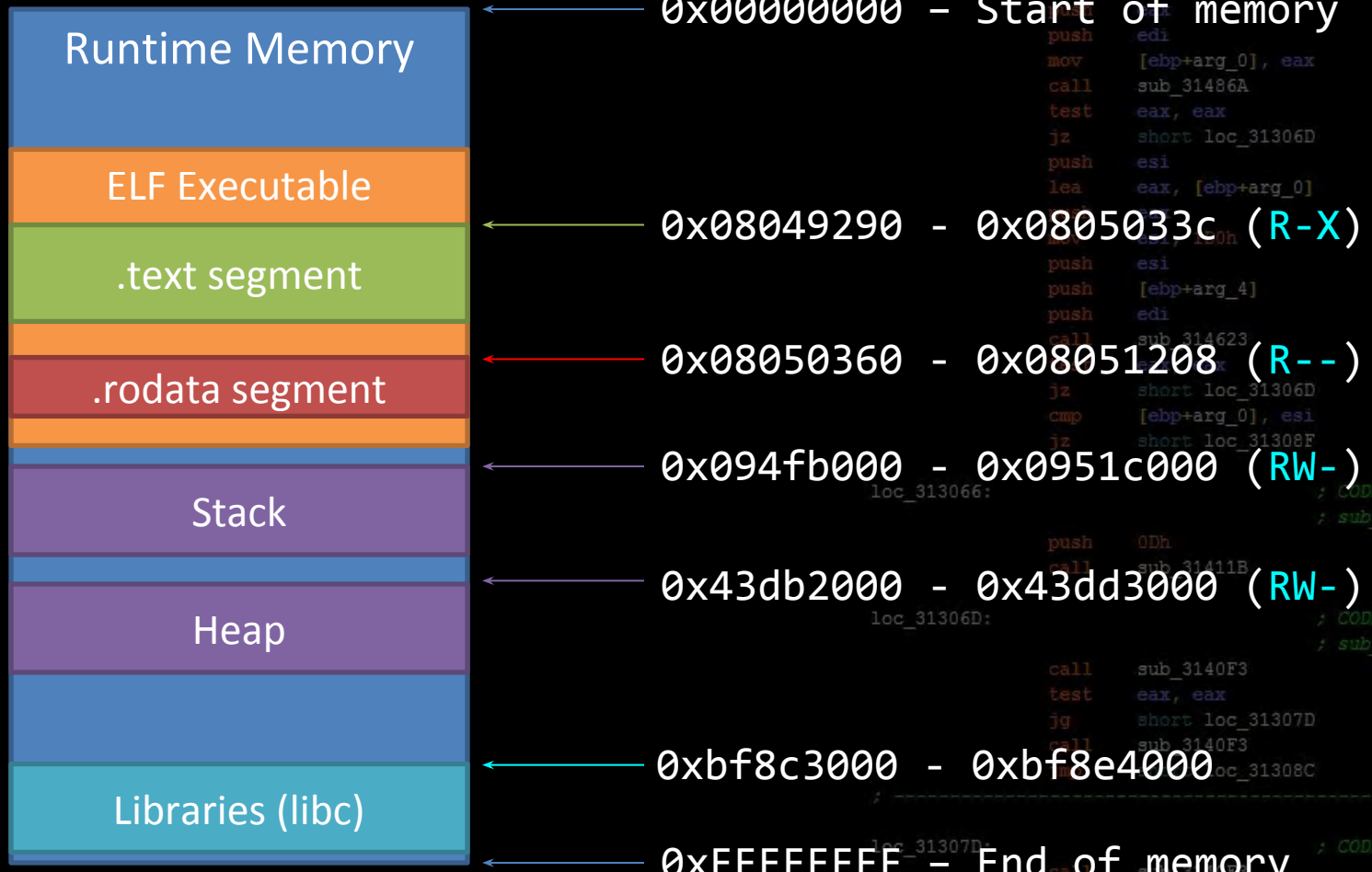
Run #1 With ASLR



Run #2 With ASLR



Run #3 With ASLR



ASLR in Action

> Open up a terminal.

```
                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+59
                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
```

ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

> Repeat a few times :)

ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

> Repeat a few times :)

You’ll see lots of lines like this:

bfe49000-bfe6a000 rw-p 00000000 00:00 0 [stack]

...

bfa23000-bfa44000 rw-p 00000000 00:00 0 [stack]

...

bfdab000-bfdcc000 rw-p 00000000 00:00 0 [stack]

ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

> Repeat a few times :)

- Stack Address Changes

```
                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
```

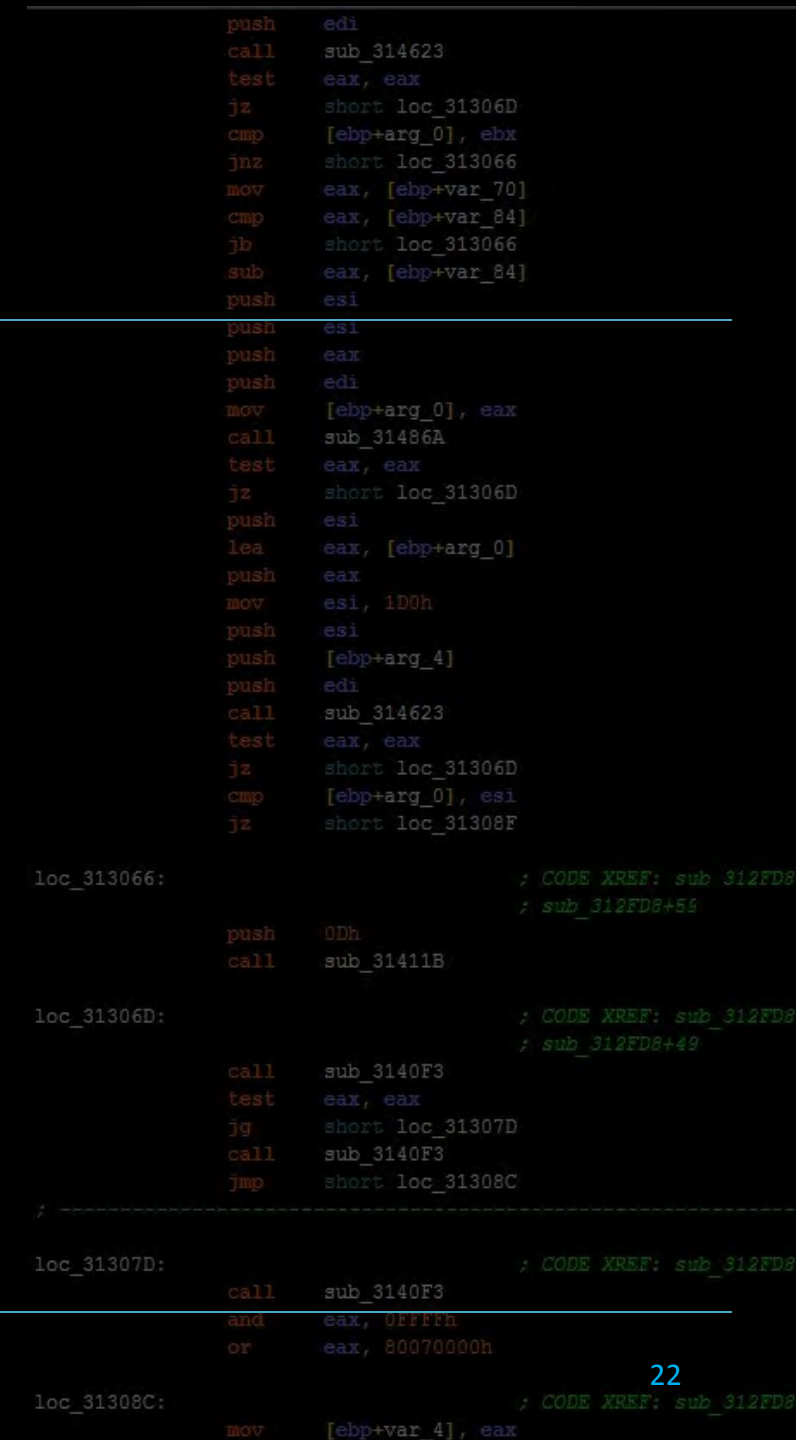
ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

> Repeat a few times :)

- Stack Address Changes
- Heap Address Changes



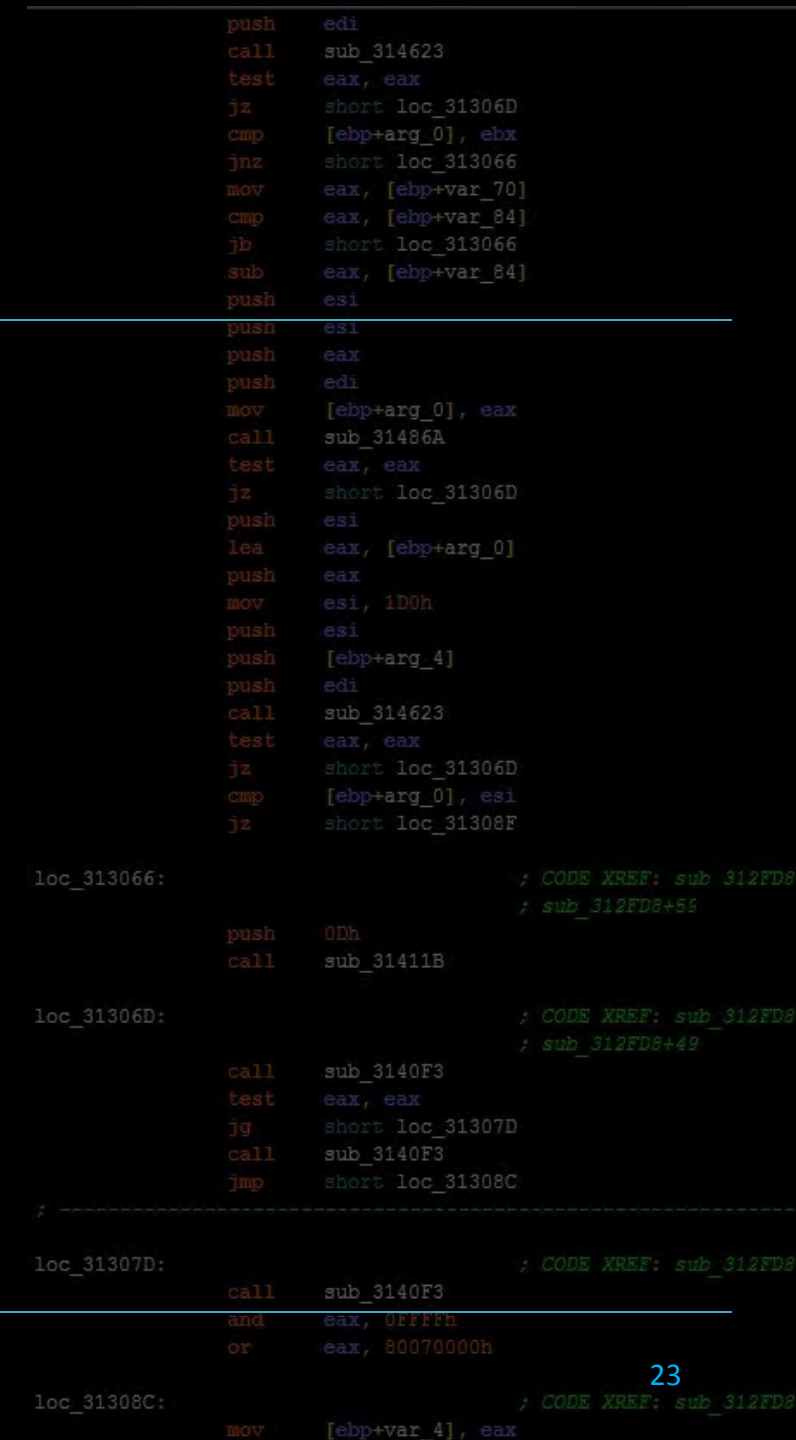
ASLR in Action

> Open up a terminal.

> Type “cat /proc/self/maps”

> Repeat a few times :)

- Stack Address Changes
- Heap Address Changes
- Library Addresses Change

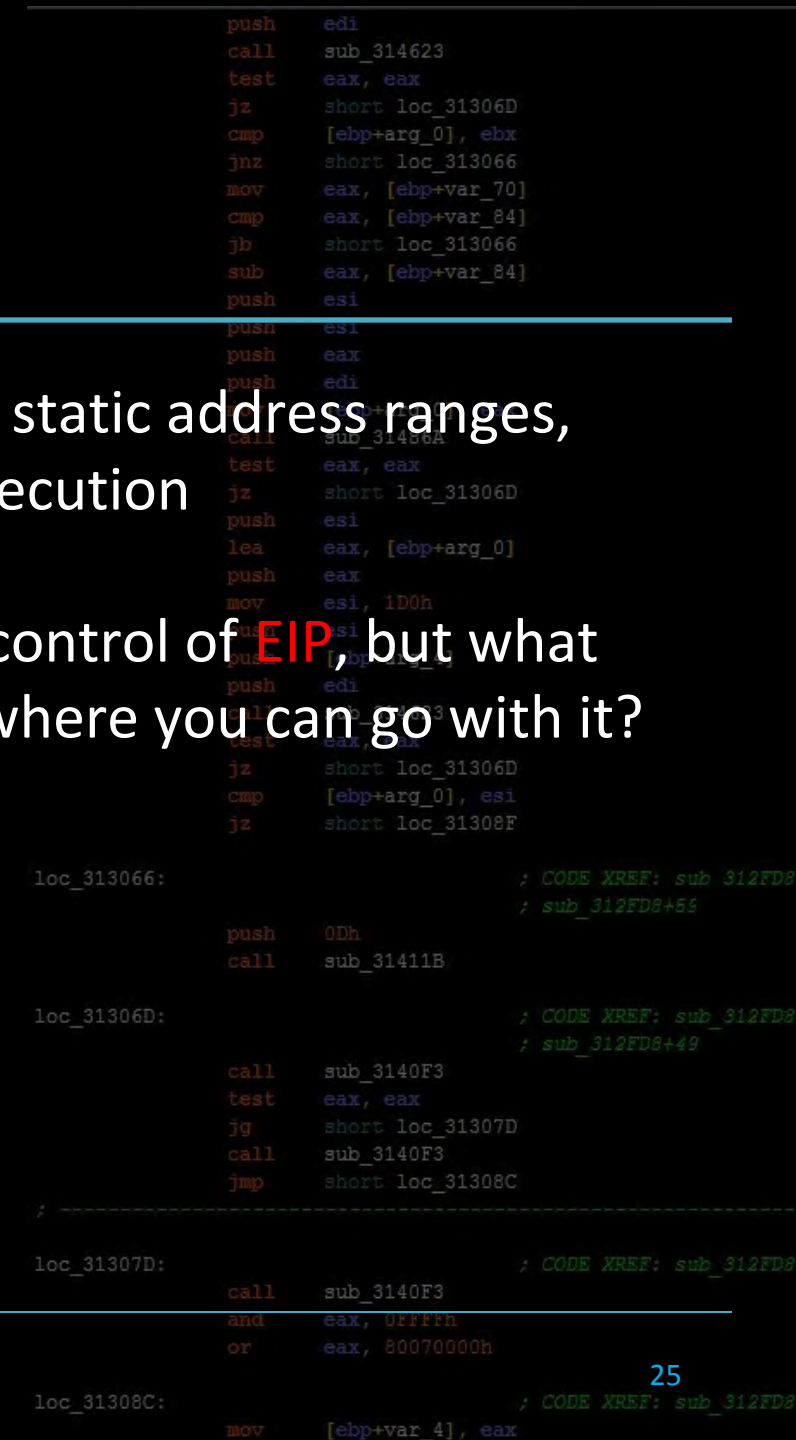


ASLR Basics

- Memory segments are no longer in static address ranges, rather they are unique for every execution

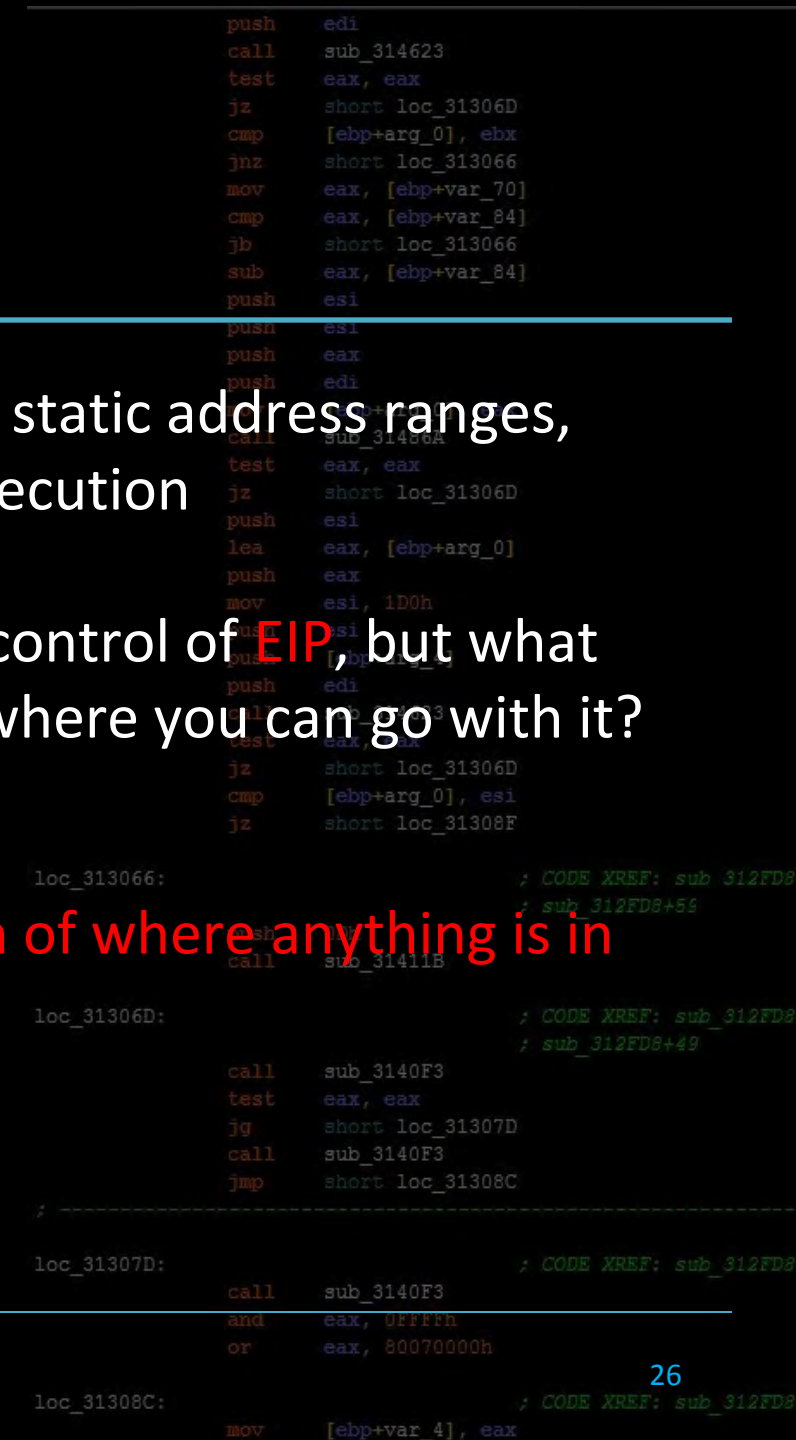
ASLR Basics

- Memory segments are no longer in static address ranges, rather they are unique for every execution
- A simple stack smash may get you control of EIP, but what does it matter if you have no idea where you can go with it?



ASLR Basics

- Memory segments are no longer in static address ranges, rather they are unique for every execution
- A simple stack smash may get you control of EIP, but what does it matter if you have no idea where you can go with it?
  - The essence of ASLR
- You must work with no expectation of where anything is in memory anymore



History of ASLR

- When was ASLR implemented?
  - May 1st, 2004 - OpenBSD 3.5 (mmap)
  - June 17th, 2005 - Linux Kernel 2.6.12 (stack, mmap)
  - January 30th, 2007 - Windows Vista (full)
  - October 26th, 2007 - Mac OSX 10.5 Leopard (sys libraries)
  - October 21st, 2010 - Windows Phone 7 (full)
  - March 11th, 2011 - iPhone iOS 4.3 (full)
  - July 20th, 2011 - Mac OSX 10.7 Lion (full)

History of ASLR

- When was ASLR implemented?
  - May 1st, 2004 - OpenBSD 3.5 (mmap)
  - June 17th, 2005 - Linux Kernel 2.6.12 (stack, mmap)
  - January 30th, 2007 - Windows Vista (full)
  - October 26th, 2007 - Mac OSX 10.5 Leopard (sys libraries)
  - October 21st, 2010 - Windows Phone 7 (full)
  - March 11th, 2011 - iPhone iOS 4.3 (full)
  - July 20th, 2011 - Mac OSX 10.7 Lion (full)

perspective: markus is accepted to RPI

Reminder:

Security is rapidly evolving

Checking for ASLR

```
$ cat /proc/sys/kernel/randomize_va_space
```

Checking for ASLR

```
$ cat /proc/sys/kernel/randomize_va_space
2
```

Checking for ASLR

```
$ cat /proc/sys/kernel/randomize_va_space
2
```

0: No ASLR

1: Conservative Randomization

(Stack, Heap, Shared Libs, PIE, mmap(), VDRO)

2: Full Randomization

(Conservative Randomization + memory managed via brk())