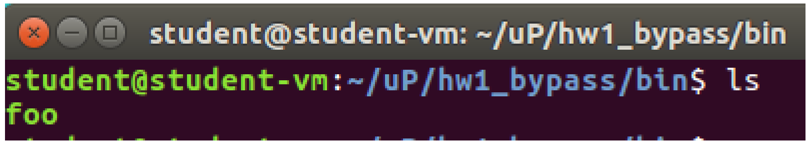
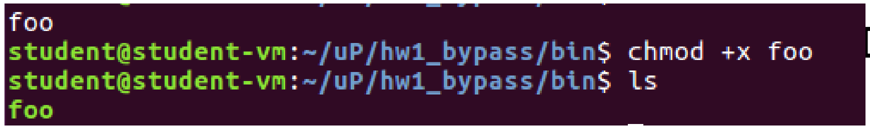
Run the compiled executable code

- Given executable (binary format) code,
  - 1) try to execute



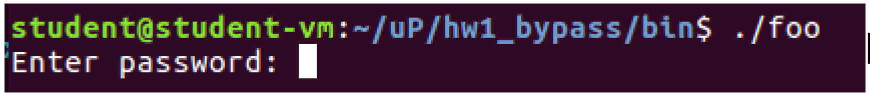
```
student@student-vm: ~/uP/hw1_bypass/bin
student@student-vm:~/uP/hw1_bypass/bin$ ls
foo
```

⇨ List files current directory



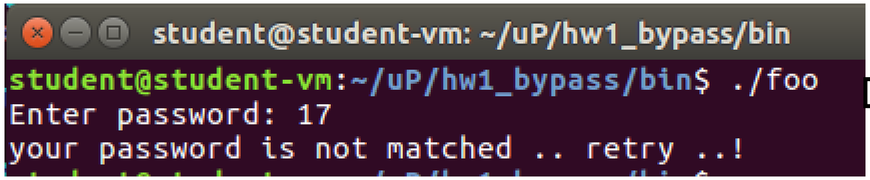
```
foo
student@student-vm:~/uP/hw1_bypass/bin$ chmod +x foo
student@student-vm:~/uP/hw1_bypass/bin$ ls
foo
```

⇨ Allow executable property



```
student@student-vm:~/uP/hw1_bypass/bin$ ./foo
Enter password: █
```

⇨ Execute foo (./foo)

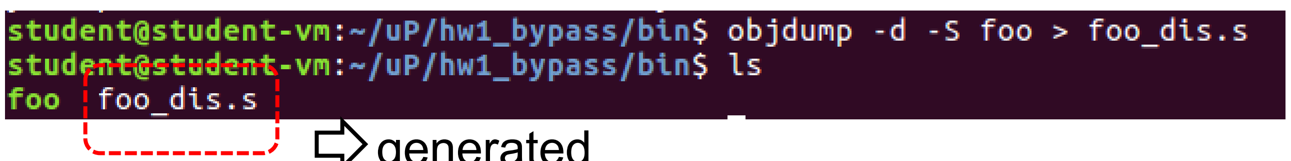


```
student@student-vm: ~/uP/hw1_bypass/bin
student@student-vm:~/uP/hw1_bypass/bin$ ./foo
Enter password: 17
your password is not matched .. retry ..!
```

⇨ You don't know password, but try to enter anything..
Opps!! , failed..

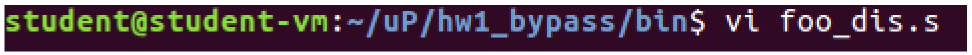
Analyze the control flow of executable code

- Given executable (binary format) code,
  - 2) let's generate disassembled code from executable code



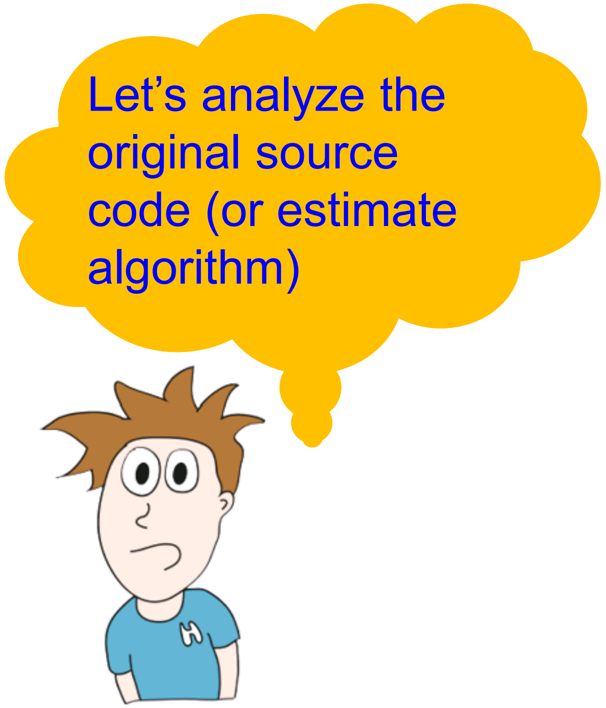
```
student@student-vm:~/uP/hw1_bypass/bin$ objdump -d -S foo > foo_dis.s
student@student-vm:~/uP/hw1_bypass/bin$ ls
foo  foo_dis.s
```

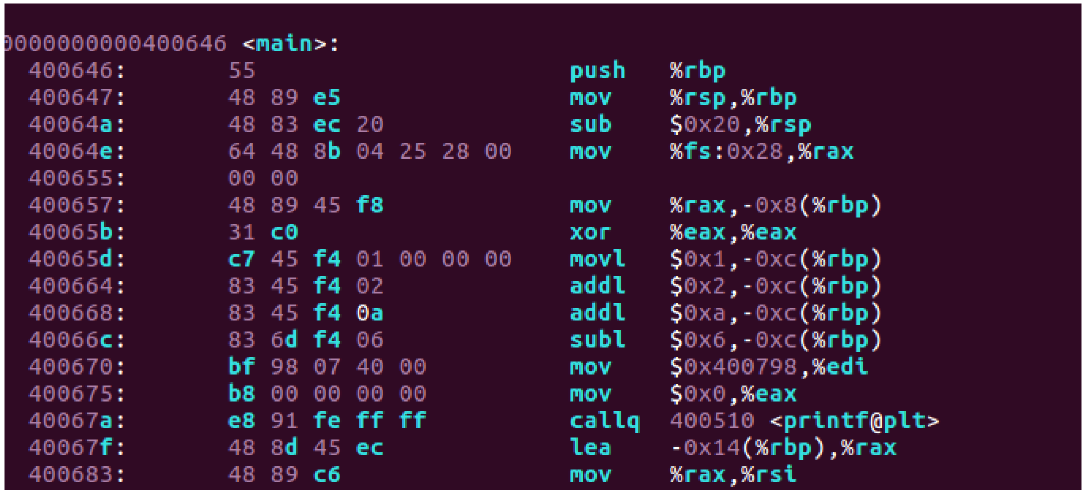
generated



```
student@student-vm:~/uP/hw1_bypass/bin$ vi foo_dis.s
```

Open using vi editor

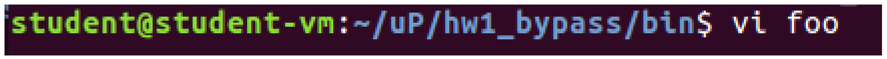

Let's analyze the original source code (or estimate algorithm)



```
0000000000400646 <main>:
  400646:       55                      push   %rbp
  400647:       48 89 e5                mov    %rsp,%rbp
  40064a:       48 83 ec 20             sub    $0x20,%rsp
  40064e:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
  400655:       00 00
  400657:       48 89 45 f8             mov    %rax,-0x8(%rbp)
  40065b:       31 c0                   xor    %eax,%eax
  40065d:       c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
  400664:       83 45 f4 02             addl   $0x2,-0xc(%rbp)
  400668:       83 45 f4 0a             addl   $0xa,-0xc(%rbp)
  40066c:       83 6d f4 06             subl   $0x6,-0xc(%rbp)
  400670:       bf 98 07 40 00          mov    $0x400798,%edi
  400675:       b8 00 00 00 00          mov    $0x0,%eax
  40067a:       e8 91 fe ff ff          callq  400510 <printf@plt>
  40067f:       48 8d 45 ec             lea    -0x14(%rbp),%rax
  400683:       48 89 c6                mov    %rax,%rsi
```

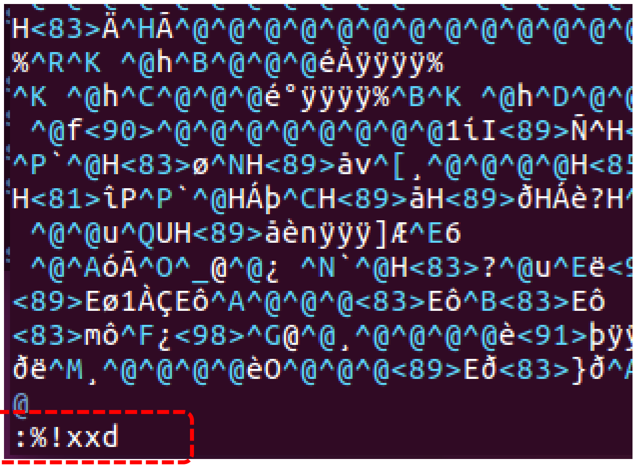
Modify micro-instructions in binary level

- Given executable (binary format) code,
  - **3) Modify binary code itself**



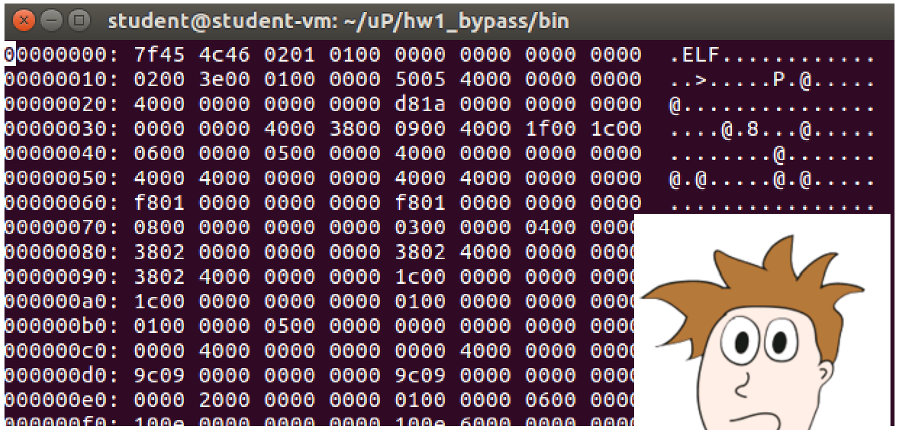
```
student@student-vm:~/uP/hw1_bypass/bin$ vi foo
```

Open binary code using vi editor

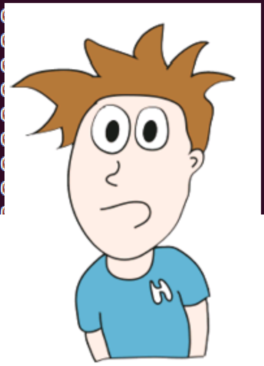


```
H<83>Ä^HĀ^@^@^@^@^@^@^@^@^@^@^@^@^@^
%^R^K ^@h^B^@^@^@éÀÿÿÿÿ%
^K ^@h^C^@^@^@é°ÿÿÿÿ%^B^K ^@h^D^@^
 ^@f<90>^@^@^@^@^@^@^@^@1íI<89>Ñ^H
^P`^@H<83>ø^NH<89>åv^[¸^@^@^@^@H<8
H<81>îP^P`^@HÁþ^CH<89>åH<89>ðHÁè?H
 ^@^@u^QUH<89>åènÿÿÿ]Æ^E6
 ^@^AóÃ^O^_@^@¿ ^N`^@H<83>?^@u^Eë<
<89>Eø1ÀÇEô^A^@^@^@<83>Eô^B<83>Eô
<83>mô^F¿<98>^G@^@¸^@^@^@^@è<91>þÿ
ðë^M¸^@^@^@^@èO^@^@^@<89>Eð<83>}ð^
@
:%!xxd
```

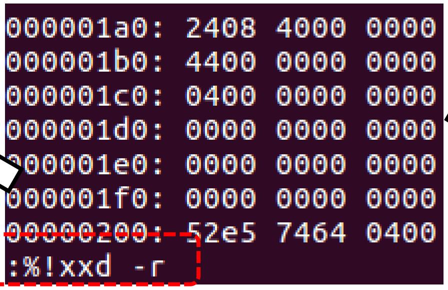
Enter :%!xxd to convert into binary editing mode



```
student@student-vm: ~/uP/hw1_bypass/bin
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0200 3e00 0100 0000 5005 4000 0000 0000  ..>.....P.@.....
00000020: 4000 0000 0000 0000 d81a 0000 0000 0000  @...............
00000030: 0000 0000 4000 3800 0900 4000 1f00 1c00  ....@.8...@.....
00000040: 0600 0000 0500 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 4000 0000 0000 4000 4000 0000 0000  @.@.....@.@.....
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 000
00000080: 3802 0000 0000 0000 3802 4000 0000 000
00000090: 3802 4000 0000 0000 1c00 0000 0000 000
000000a0: 1c00 0000 0000 0000 0100 0000 0000 000
000000b0: 0100 0000 0500 0000 0000 0000 0000 000
000000c0: 0000 4000 0000 0000 0000 4000 0000 000
000000d0: 9c09 0000 0000 0000 9c09 0000 0000 000
000000e0: 0000 2000 0000 0000 0100 0000 0600 000
```

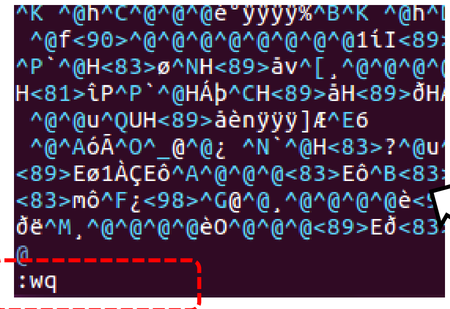


Modify jump address (relative distance)



```
000001a0: 2408 4000 0000
000001b0: 4400 0000 0000
000001c0: 0400 0000 0000
000001d0: 0000 0000 0000
000001e0: 0000 0000 0000
000001f0: 0000 0000 0000
00000200: 52e5 7464 0400
:%!xxd -r
```

Enter :%!xxd –r to restore into original mode

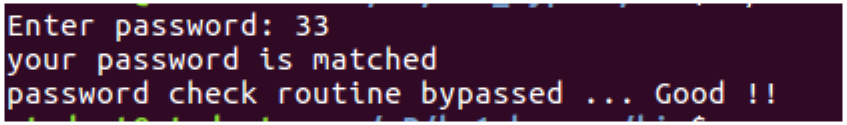


```
^K ^@h^C^@^@^@é°ÿÿÿÿ%^B^K ^@h^
 ^@f<90>^@^@^@^@^@^@^@^@1íI<89
^P`^@H<83>ø^NH<89>åv^[¸^@^@^@^
H<81>îP^P`^@HÁþ^CH<89>åH<89>ðH
 ^@^@u^QUH<89>åènÿÿÿ]Æ^E6
 ^@^AóÃ^O^_@^@¿ ^N`^@H<83>?^@u
<89>Eø1ÀÇEô^A^@^@^@<83>Eô^B<83
<83>mô^F¿<98>^G@^@¸^@^@^@^@è<9
ðë^M¸^@^@^@^@èO^@^@^@<89>Eð<83
@
:wq
```

save

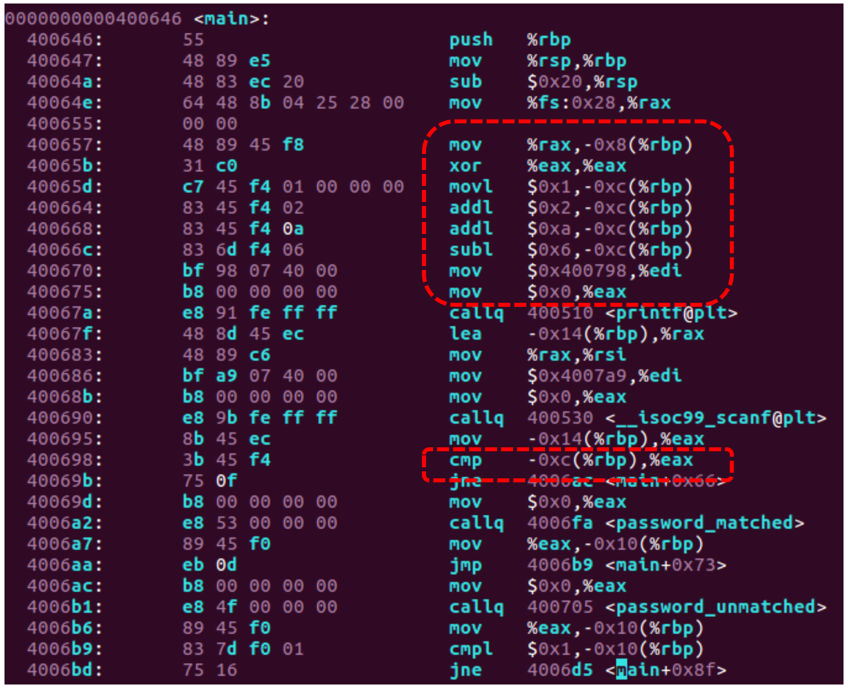
Re-run the modified executable code to bypass password check

- Modified executable (binary format) code,
  - 4) **Retry to execute,**
    - **Still, you don't know password, but enter anything…**



```
Enter password: 33
your password is matched
password check routine bypassed ... Good !!
```

  - 5) (additional credit) Analyze the binary code using gdb
    - ➔ Estimate correct password



```
0000000000400646 <main>:
  400646:       55                      push   %rbp
  400647:       48 89 e5                mov    %rsp,%rbp
  40064a:       48 83 ec 20             sub    $0x20,%rsp
  40064e:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
  400655:       00 00
  400657:       48 89 45 f8             mov    %rax,-0x8(%rbp)
  40065b:       31 c0                   xor    %eax,%eax
  40065d:       c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
  400664:       83 45 f4 02             addl   $0x2,-0xc(%rbp)
  400668:       83 45 f4 0a             addl   $0xa,-0xc(%rbp)
  40066c:       83 6d f4 06             subl   $0x6,-0xc(%rbp)
  400670:       bf 98 07 40 00          mov    $0x400798,%edi
  400675:       b8 00 00 00 00          mov    $0x0,%eax
  40067a:       e8 91 fe ff ff          callq  400510 <printf@plt>
  40067f:       48 8d 45 ec             lea    -0x14(%rbp),%rax
  400683:       48 89 c6                mov    %rax,%rsi
  400686:       bf a9 07 40 00          mov    $0x4007a9,%edi
  40068b:       b8 00 00 00 00          mov    $0x0,%eax
  400690:       e8 9b fe ff ff          callq  400530 <__isoc99_scanf@plt>
  400695:       8b 45 ec                mov    -0x14(%rbp),%eax
  400698:       3b 45 f4                cmp    -0xc(%rbp),%eax
  40069b:       75 0f                   jne    4006ac <main+0x66>
  40069d:       b8 00 00 00 00          mov    $0x0,%eax
  4006a2:       e8 53 00 00 00          callq  4006fa <password_matched>
  4006a7:       89 45 f0                mov    %eax,-0x10(%rbp)
  4006aa:       eb 0d                   jmp    4006b9 <main+0x73>
  4006ac:       b8 00 00 00 00          mov    $0x0,%eax
  4006b1:       e8 4f 00 00 00          callq  400705 <password_unmatched>
  4006b6:       89 45 f0                mov    %eax,-0x10(%rbp)
  4006b9:       83 7d f0 01             cmpl   $0x1,-0x10(%rbp)
  4006bd:       75 16                   jne    4006d5 <main+0x8f>
```

Submission files

- Modified binary (executable) code ‘foo’
- Documentation in details for your efforts to analyze (estimate) the original source and modify the binary code