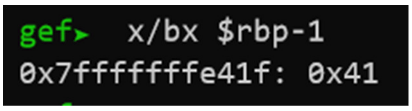
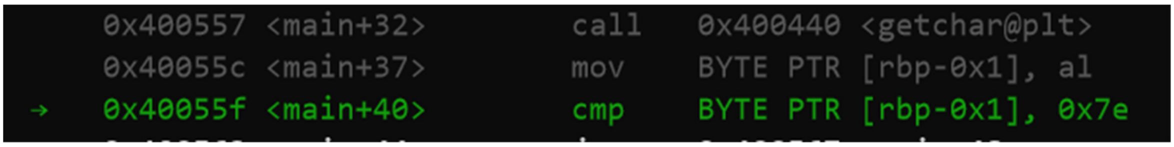
If input is A or x41



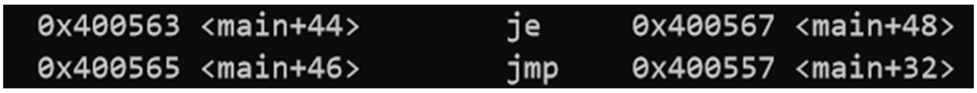
```
gef➤  x/bx $rbp-1
0x7fffffffe41f: 0x41
```

After getting input from user if, input is equals to 0x7e or ~ , then do something



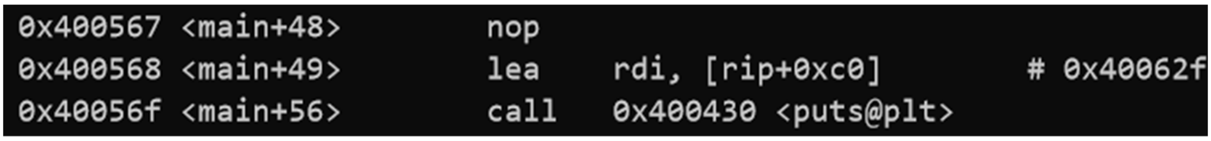
```
     0x400557 <main+32>          call   0x400440 <getchar@plt>
     0x40055c <main+37>          mov    BYTE PTR [rbp-0x1], al
 →   0x40055f <main+40>          cmp    BYTE PTR [rbp-0x1], 0x7e
```

When comparison is done, jump to exit if input is equal to the char being compared, else keep looping



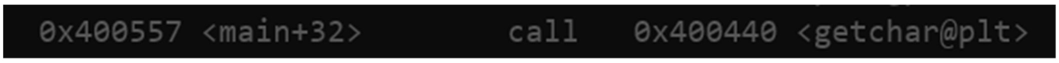
```
  0x400563 <main+44>          je     0x400567 <main+48>
  0x400565 <main+46>          jmp    0x400557 <main+32>
```

Main+48 = exit message - break



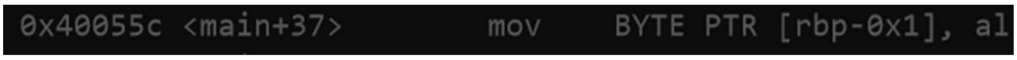
```
0x400567 <main+48>          nop
0x400568 <main+49>          lea    rdi, [rip+0xc0]        # 0x40062f
0x40056f <main+56>          call   0x400430 <puts@plt>
```

Main+32 = get single byte input – keep looping



```
  0x400557 <main+32>          call   0x400440 <getchar@plt>
```

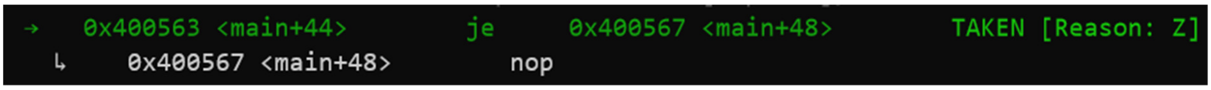
After getchar(), It means only get one byte



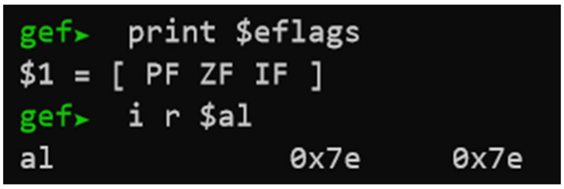
```
0x40055c <main+37>          mov    BYTE PTR [rbp-0x1], al
```

Other notes

If input is equal to char being compared, zero flag is set

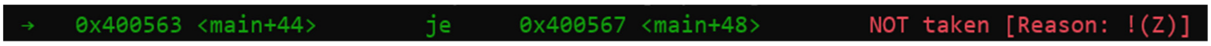


```
 →   0x400563 <main+44>        je     0x400567 <main+48>        TAKEN [Reason: Z]
   ↳     0x400567 <main+48>        nop
```

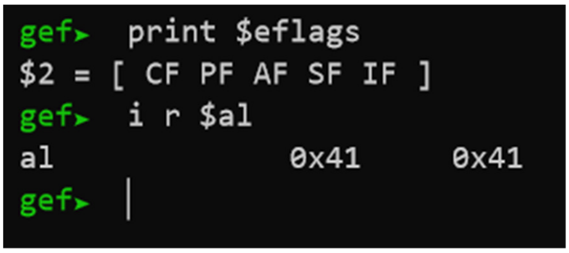


```
gef➤  print $eflags
$1 = [ PF ZF IF ]
gef➤  i r $al
al             0x7e      0x7e
```

If input is not to char being compared, zero flag isnt set



```
→   0x400563 <main+44>       je     0x400567 <main+48>       NOT taken [Reason: !(Z)]
```



```
gef➤  print $eflags
$2 = [ CF PF AF SF IF ]
gef➤  i r $al
al             0x41      0x41
gef➤  |
```