If-else conditions

→ so Let's start by using radare2 using command `r2 -d if1`

→ and we have to run this command also : `e asm.syntax=att` to set the syntax for assembly

→ now let's analyse the program using `aaa`

→ after that we will search for main function using `afl | grep main` command

→ and we got the main function

```
[0x7f603fb23090]> afl | grep main
0x55b4e4400fe0    1 4124         reloc.__libc_start_main
0x55b4e42005fa    4 43           main
```

→ so Let's read that function using command `pdf@main`

```
[0x7f603fb23090]> pdf@main
            ; DATA XREF from entry0 @ 0x55b4e420050d
┌ 43: int main (int argc, char **argv, char **envp);
            ; var int64_t var_8h @ rbp-0x8
            ; var int64_t var_4h @ rbp-0x4
            0x55b4e42005fa      55             pushq %rbp
            0x55b4e42005fb      4889e5         movq %rsp, %rbp
            0x55b4e42005fe      c745f8030000.  movl $3, var_8h
            0x55b4e4200605      c745fc040000.  movl $4, var_4h
            0x55b4e420060c      8b45f8         movl var_8h, %eax
            0x55b4e420060f      3b45fc         cmpl var_4h, %eax
        ┌─< 0x55b4e4200612      7d06           jge 0x55b4e420061a
        │   0x55b4e4200614      8345f805       addl $5, var_8h
      ┌─< 0x55b4e4200618      eb04           jmp 0x55b4e420061e
      └─> 0x55b4e420061a      8345fc03       addl $3, var_4h
            ; CODE XREF from main @ 0x55b4e4200618
      └─> 0x55b4e420061e      b800000000     movl $0, %eax
            0x55b4e4200623      5d             popq %rbp
            0x55b4e4200624      c3             retq
```

→ and there are 2 jump instructions `jge` is for jump if value is greater than or equal to given value and `jmp` stands for unconditional jump

→ first of all we will set the breakpoints at `jge` and `jmp`

```
[0x7f603fb23090]> db 0x55b4e4200612
[0x7f603fb23090]> db 0x55b4e4200618
```

→ now execute the program using `dc` and it will hit the breakpoint

```
[0x7f603fb23090]> dc
hit breakpoint at: 0x55b4e4200612
[0x55b4e4200612]> pdf@main
            ; DATA XREF from entry0 @ 0x55b4e420050d
┌ 43: int main (int argc, char **argv, char **envp);
            ; var int64_t var_8h @ rbp-0x8
            ; var int64_t var_4h @ rbp-0x4
            0x55b4e42005fa      55              pushq %rbp
            0x55b4e42005fb      4889e5          movq %rsp, %rbp
            0x55b4e42005fe      c745f8030000.   movl $3, var_8h
            0x55b4e4200605      c745fc040000.   movl $4, var_4h
            0x55b4e420060c      8b45f8          movl var_8h, %eax
            0x55b4e420060f      3b45fc          cmpl var_4h, %eax
            ;-- rip:
        ┌─< 0x55b4e4200612 b    7d06            jge 0x55b4e420061a
        │   0x55b4e4200614      8345f805        addl $5, var_8h
        ┌─< 0x55b4e4200618 b    eb04            jmp 0x55b4e420061e
        │└─> 0x55b4e420061a     8345fc03        addl $3, var_4h
        │   ; CODE XREF from main @ 0x55b4e4200618
        └─> 0x55b4e420061e      b800000000      movl $0, %eax
            0x55b4e4200623      5d              popq %rbp
            0x55b4e4200624      c3              retq
```

→ so we can see the `cmpl` is the comparing the value of `var_4h` and `eax` register so Let's read the value of them

→ to see the value of register we use `dr` command

```
[0x55b4e4200612]> dr
rax = 0x00000003
rbx = 0x00000000
rcx = 0x7f603fb02718
rdx = 0x7ffdb3908ff8
r8 = 0x00000000
r9 = 0x7f603fb321b0
r10 = 0x00000000
r11 = 0x00000000
r12 = 0x55b4e42004f0
r13 = 0x00000000
r14 = 0x00000000
r15 = 0x00000000
rsi = 0x7ffdb3908fe8
rdi = 0x00000001
rsp = 0x7ffdb3908ef0
rbp = 0x7ffdb3908ef0
rip = 0x55b4e4200612
rflags = 0x00000297
orax = 0xffffffffffffffff
```

→ and it contains `3`

→ to see the value of `var_4h` we use `px @rbp-0x4 (which is address of var_4h)`

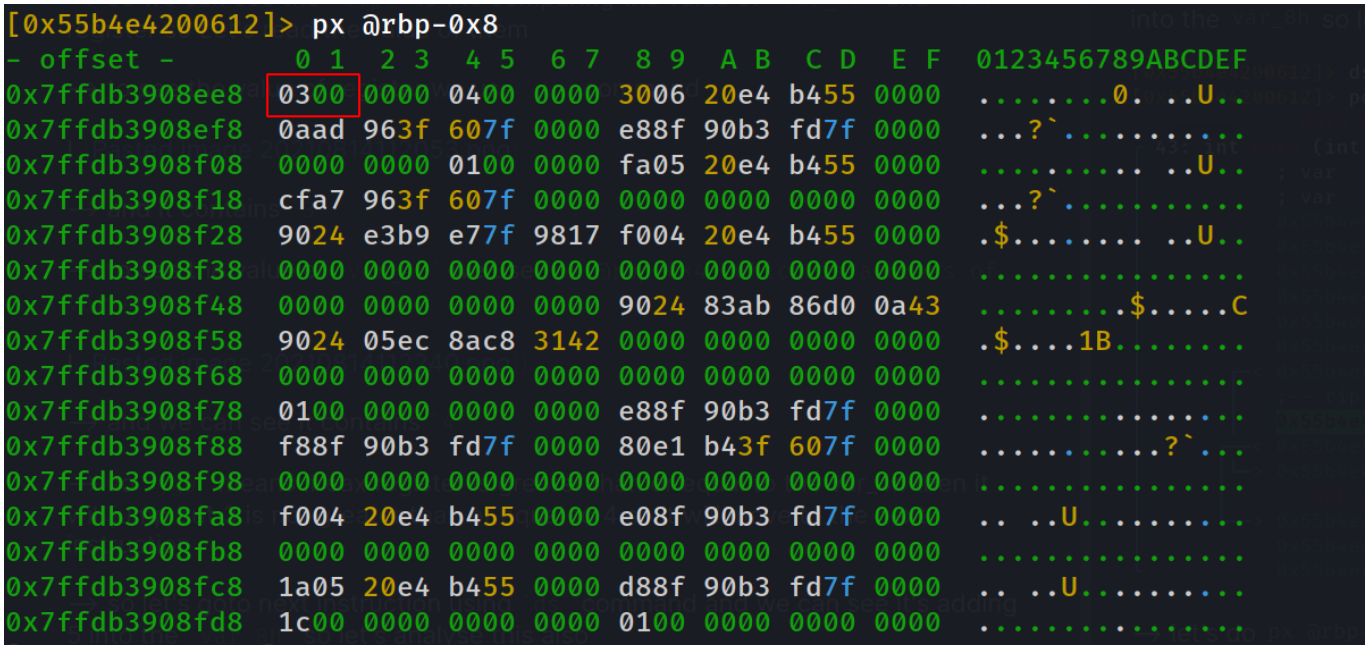
```
[0x55b4e4200612]> px @rbp-0x4
- offset -  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x7ffdb3908eec  0400 0000 3006 20e4 b455 0000 0aad 963f  ....0. ..U.....?
0x7ffdb3908efc  607f 0000 e88f 90b3 fd7f 0000 0000 0000  `...............
0x7ffdb3908f0c  0100 0000 fa05 20e4 b455 0000 cfa7 963f  ...... ..U.....?
0x7ffdb3908f1c  607f 0000 0000 0000 0000 0000 9024 e3b9  `...........$..
0x7ffdb3908f2c  e77f 9817 f004 20e4 b455 0000 0000 0000  ...... ..U......
0x7ffdb3908f3c  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

→ and we can see it contains `4`

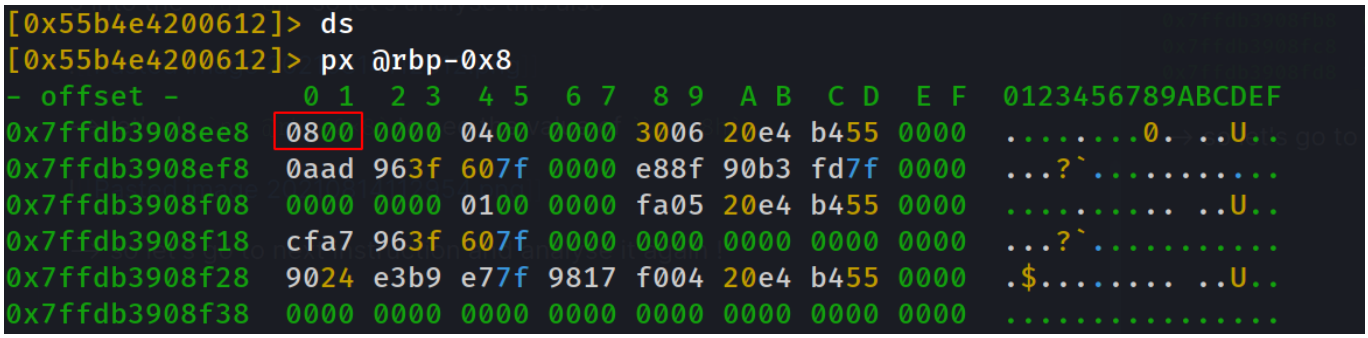
→ so `jge` means if eax register is greater than or equal to the var_4h then it will jump but 3 is not greater than or equal to 4 so it will move to the next instruction

→ so let's goto next instruction using `ds` command and we can see it's adding 5 into the `var_8h` so let's analyse this also

→ let's do `px @rbp-0x8` to see the value of `var_8h`



→ so let's go to next instruction and analyse it again !



→ and we can see the value of `var_8h` changed to 8 means 5 has been added into it

---