



EDDI

Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

제 4기

2022. 09. 16

유진선

기계어 분석 - 디버깅(if.c) -(1/11)

```
#include <stdio.h>

int main(void)
{
    int num = 3;

    if (num > 7)
    {
        printf("3 > 7\n");
    }
    else
    {
        printf("3 < 7\n");
    }

    return 0;
}
```

if 문 예제 작성

if 문의 어셈블리 동작을 분석하기 위한 코드.

num 의 변수 크기 비교해서 글자를 출력한다.

```
(gdb) disas
Dump of assembler code for function main:
=> 0x000055555555149 <+0>:      endbr64
    0x00005555555514d <+4>:      push   %rbp
    0x00005555555514e <+5>:      mov    %rsp,%rbp
    0x000055555555151 <+8>:      sub    $0x10,%rsp
    0x000055555555155 <+12>:     movl   $0x3,-0x4(%rbp)
    0x00005555555515c <+19>:     cmpl   $0x7,-0x4(%rbp)
    0x000055555555160 <+23>:     jle    0x55555555170 <main+39>
    0x000055555555162 <+25>:     lea    0xe9b(%rip),%rdi    # 0x555555556004
    0x000055555555169 <+32>:     callq  0x55555555050 <puts@plt>
    0x00005555555516e <+37>:     jmp    0x5555555517c <main+51>
    0x000055555555170 <+39>:     lea    0xe93(%rip),%rdi    # 0x55555555600a
    0x000055555555177 <+46>:     callq  0x55555555050 <puts@plt>
    0x00005555555517c <+51>:     mov    $0x0,%eax
    0x000055555555181 <+56>:     leaveq %eax
    0x000055555555182 <+57>:     retq

End of assembler dump.
(gdb)
```

보편적인 if의 형식

1. mov로 비교대상 배치
2. cmp로 실제 비교수행
3. 조건부 jmp로 분기

le : Less Equal

ge : Great Equal

기계어 분석 - 디버깅(if.c)-(2/11)

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555551409 <+0>:   endbr64
```

```
(gdb) info registers
rax      0x5555555555149  93824992235849
rbx      0x5555555555190  93824992235920
rcx      0x5555555555190  93824992235920
rdx      0x7fffffffdf28    140737488346920
rsi      0x7fffffffdf18    140737488346904
rdi      0x1               1
rbp      0x0               0x0
rsp      0x7fffffffde28    0x7fffffffde28
r8       0x0               0
r9       0x7ffff7fe0d60    140737354009952
r10      0xf               15
r11      0x2               2
r12      0x555555555060    93824992235616
r13      0x7fffffffdf10    140737488346896
r14      0x0               0
r15      0x0               0
rip      0x5555555555149  0x5555555555149 <main>
eflags   0x246             [ PF ZF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0               0
es       0x0               0
fs       0x0               0
gs       0x0               0
(gdb)
```

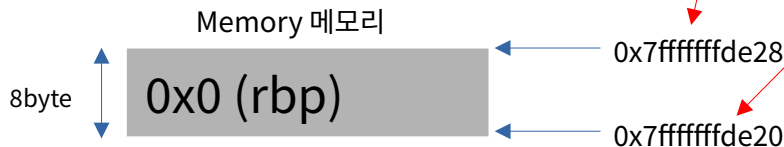
```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555551410 <+0>:   endbr64
=> 0x000055555555141d <+4>:   push  %rbp
```

```
(gdb) info registers
rax      0x5555555555149  93824992235849
rbx      0x5555555555190  93824992235920
rcx      0x5555555555190  93824992235920
rdx      0x7fffffffdf28    140737488346920
rsi      0x7fffffffdf18    140737488346904
rdi      0x1               1
rbp      0x0               0x0
rsp      0x7fffffffde28    0x7fffffffde28
r8       0x0               0
r9       0x7ffff7fe0d60    140737354009952
r10      0xf               15
r11      0x2               2
r12      0x555555555060    93824992235616
r13      0x7fffffffdf10    140737488346896
r14      0x0               0
r15      0x0               0
rip      0x555555555514d  0x555555555514d <main+4>
eflags   0x246             [ PF ZF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0               0
es       0x0               0
fs       0x0               0
gs       0x0               0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555551419 <+0>:   endbr64
=> 0x000055555555141d <+4>:   push  %rbp
=> 0x000055555555141e <+5>:   mov  %rsp,%rbp
=> 0x000055555555141f <+6>:   sub  $0x10,%rsp
```

```
(gdb) info registers
rax      0x5555555555149  93824992235849
rbx      0x5555555555190  93824992235920
rcx      0x5555555555190  93824992235920
rdx      0x7fffffffdf28    140737488346920
rsi      0x7fffffffdf18    140737488346904
rdi      0x1               1
rbp      0x0               0x0
rsp      0x7fffffffde20    0x7fffffffde20
r8       0x0               0
r9       0x7ffff7fe0d60    140737354009952
r10      0xf               15
r11      0x2               2
r12      0x555555555060    93824992235616
r13      0x7fffffffdf10    140737488346896
r14      0x0               0
r15      0x0               0
rip      0x555555555514e  0x555555555514e <main+5>
eflags   0x246             [ PF ZF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0               0
es       0x0               0
fs       0x0               0
gs       0x0               0
(gdb)
```

push 명령어는 현재 스택의 최상위 메모리 (rsp)에 값을 저장하는 명령어
즉, 현재 스택의 최상위 메모리(rsp)에 **rbp**값을 저장하라는 의미



기계어 분석 - 디버깅(if.c)-(3/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
=> 0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x0                0x0
rsp      0x7fffffffde20     0x7fffffffde20
r8       0x0                0
r9       0x7fffffffde06     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x5555555514e      0x5555555514e <main+5>
eflags   0x246             [ PF ZF IF ]
cs       0x33             51
ss       0x2b             43
ds       0x0              0
es       0x0              0
fs       0x0              0
gs       0x0              0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
0x00005555555514e <+5>:    mov     %rsp,%rbp
=> 0x000055555555151 <+8>:    sub     $0x10,%rsp
0x000055555555155 <+12>:   movl    $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde20     0x7fffffffde20
r8       0x0                0
r9       0x7fffffffde06     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555151      0x55555555151 <main+8>
eflags   0x246             [ PF ZF IF ]
cs       0x33             51
ss       0x2b             43
ds       0x0              0
es       0x0              0
fs       0x0              0
gs       0x0              0
(gdb)
```

Memory 메모리

0x0 (rbp)

0x7fffffffde28

8byte

0x7fffffffde20

mov 명령어는 내용을 복사한다.

mov %rsp, %rbp 는 rbp에 rsp값을 복사한다.

즉, rbp 값에 rsp값을 넣으면서 rsp,rbp 주소값이 서로 같아 지면서 스택의 경계선이 사라진다.

→ 새로운 스택을 생성할 준비를 하는 과정

기계어 분석 - 디버깅(if.c)-(4/11)

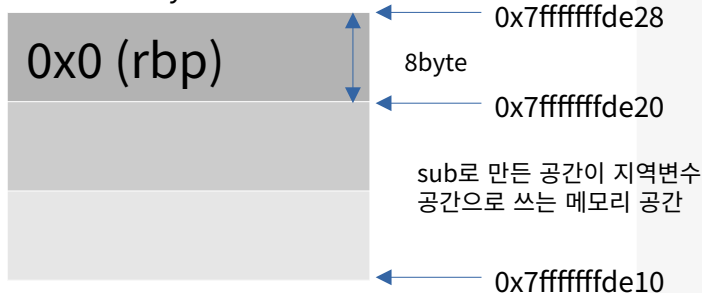
```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>: endbr64
0x00005555555514d <+4>: push %rbp
0x00005555555514e <+5>: mov %rsp,%rbp
=> 0x000055555555151 <+8>: sub $0x10,%rsp
0x000055555555155 <+12>: movl $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149  93824992235849
rbx      0x55555555190  93824992235920
rcx      0x55555555190  93824992235920
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1             1
rbp      0x7fffffffde20  0x7fffffffde20
rsp      0x7fffffffde20  0x7fffffffde20
r8       0x0             0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf             15
r11      0x2             2
r12      0x55555555060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0             0
r15      0x0             0
rip      0x55555555151  <main+8>
eflags   0x246          [ PF ZF IF ]
cs       0x33           51
ss       0x2b           43
ds       0x0            0
es       0x0            0
fs       0x0            0
gs       0x0            0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>: endbr64
0x00005555555514d <+4>: push %rbp
0x00005555555514e <+5>: mov %rsp,%rbp
0x000055555555151 <+8>: sub $0x10,%rsp
=> 0x000055555555155 <+12>: movl $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149  93824992235849
rbx      0x55555555190  93824992235920
rcx      0x55555555190  93824992235920
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1             1
rbp      0x7fffffffde20  0x7fffffffde20
rsp      0x7fffffffde10  0x7fffffffde10
r8       0x0             0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf             15
r11      0x2             2
r12      0x55555555060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0             0
r15      0x0             0
rip      0x55555555155  <main+12>
eflags   0x202          [ IF ]
cs       0x33           51
ss       0x2b           43
ds       0x0            0
es       0x0            0
fs       0x0            0
gs       0x0            0
(gdb)
```

Memory 메모리



sub 명령어는 뺄셈 명령.

sub %0x10, %rsp 는 rsp에 에서 16바이트를 뺀다.

(0x10 → 0001 0000 : $2^4 = 16$ 바이트 가상 주소공간에서는 바이트 단위로 움직인다.

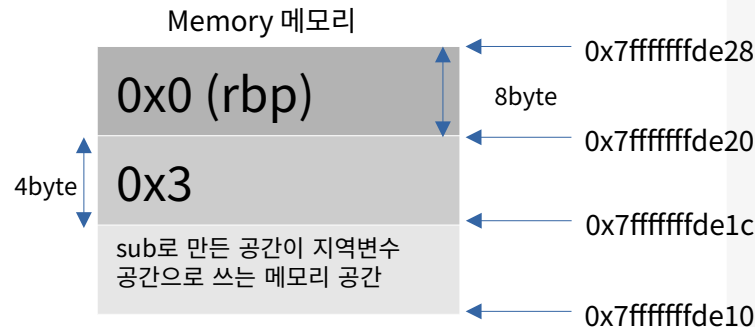
기계어 분석 - 디버깅(if.c)-(5/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555140 <+0>:  endbr64
0x000055555555144 <+4>:  push %rbp
0x00005555555514e <+6>:  mov %rsp,%rbp
0x000055555555151 <+9>:  sub $0x10,%rsp
=> 0x000055555555155 <+12>: movl $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555155      0x55555555155 <main+12>
eflags   [ IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) □
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:  endbr64
0x00005555555514d <+4>:  push %rbp
0x00005555555514e <+6>:  mov %rsp,%rbp
0x000055555555151 <+9>:  sub $0x10,%rsp
0x000055555555155 <+12>: movl $0x3,-0x4(%rbp)
0x00005555555515c <+19>: cmpl $0x7,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x5555555515c      0x5555555515c <main+19>
eflags   [ IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) □
```



movl 명령어는 내용을 복사한다.

l 이 들어가면 4바이트 처리를 하겠다는 의미

movl %0x3, -0x4(%rbp) 는 rbp를 기준으로4바이트 뻥자리에 0x3값을 복사한다.

de20 - 4 → de1c

```
(gdb) x $rbp-4
0x7fffffffde1c: 0x00000003
```

```
#include <stdio.h>

int main(void)
{
    int num = 3;
```

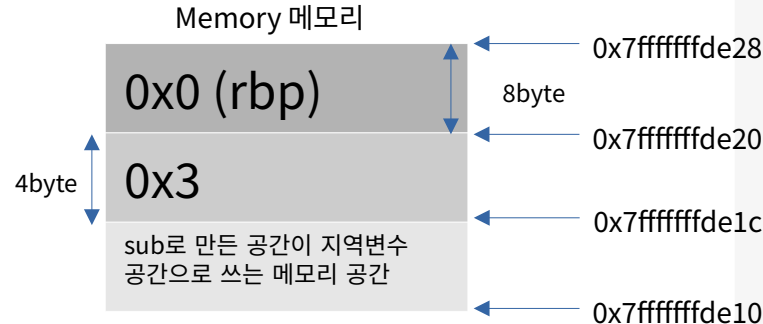
기계어 분석 - 디버깅(if.c)-(6/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555140 <+0>:   endbr64
0x000055555555144 <+4>:   push  %rbp
0x000055555555146 <+6>:   mov   %rsp,%rbp
0x000055555555151 <+11>:  sub   $0x10,%rsp
0x000055555555155 <+12>:  movl  $0x3,-0x4(%rbp)
=> 0x00005555555515c <+19>:  cmpl  $0x7,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rsp      0x7fffffffde20     0x7fffffffde20
rbp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x5555555515c      0x5555555515c <main+19>
eflags   0x202              [ IF ]
cs       0x33              51
ds       0x2b              43
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555140 <+0>:   endbr64
0x000055555555144 <+4>:   push  %rbp
0x000055555555146 <+6>:   mov   %rsp,%rbp
0x000055555555151 <+11>:  sub   $0x10,%rsp
0x000055555555155 <+12>:  movl  $0x3,-0x4(%rbp)
0x00005555555515c <+19>:  cmpl  $0x7,-0x4(%rbp)
0x000055555555160 <+23>:  jle    0x55555555176 <main+39>
0x000055555555161 <+25>:  lea    0xe9b(%rip),%rdi    # 0x555555556004
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rsp      0x7fffffffde20     0x7fffffffde20
rbp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555160      0x55555555160 <main+23>
eflags   0x297              [ CF PF AF SF IF ]
cs       0x33              51
ds       0x2b              43
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb)
```



cmpl 명령어는 내용을 비교한다.

| 이 들어가면 4바이트 처리를 하겠다는 의미

cmpl \$0x7, -0x4(%rbp) 는 0x7값과 rbp를 기준으로 4바이트 뻥자리의 값 (0x3)을 비교한다.

cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,

jle = jump less or equal 로 작거나 같을때 점프한다는 의미

0x5555 5555 5170 주소로 점프 한다.

```
#include <stdio.h>

int main(void)
{
    int num = 3;

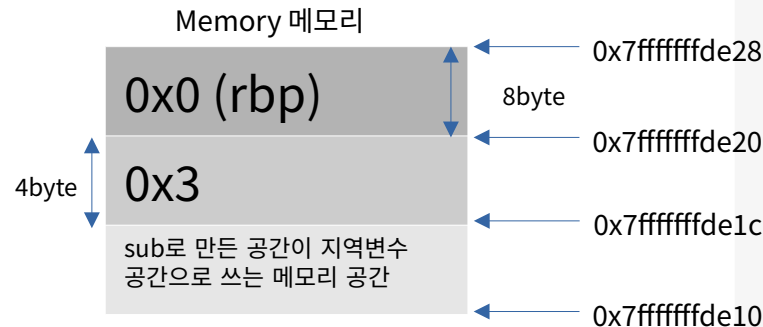
    if (num > 7)
    {
        printf("3 > 7\n");
    }
    else
    {
        printf("3 < 7\n");
    }

    return 0;
}
```

기계어 분석 - 디버깅(if.c)-(7/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149: <4b>: endbr4
0x00005555555514d: <4>: push %rbp
0x00005555555514e: <5>: mov %rsp,%rbp
0x000055555555151: <8>: sub $0x10,%rsp
0x000055555555155: <12>: movl $0x3,-0x4(%rbp)
0x00005555555515c: <19>: cmpl $0x7,-0x4(%rbp)
0x000055555555160: <23>: jle 0x55555555170 <main+39>
0x000055555555162: <25>: lea 0xe9b(%rip),%rdi # 0x555555550000
0x000055555555169: <32>: callq 0x555555550050 <puts@plt>
0x00005555555516e: <37>: jmp 0x5555555517c <main+51>
=> 0x000055555555170: <39>: lea 0xe93(%rip),%rdi # 0x555555550000
0x000055555555177: <46>: callq 0x555555550050 <puts@plt>
0x00005555555517c: <51>: mov $0x0,%eax
0x000055555555181: <56>: leaveq %eax
0x000055555555182: <57>: retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8        0x0                0
r9        0x7ffff7fe0d60    140737354009952
r10       0xf                15
r11       0x2                2
r12       0x55555555060     93824992235616
r13       0x7fffffffdf10    140737488346896
r14       0x0                0
r15       0x0                0
rip       0x55555555170     0x55555555170 <main+39>
eflags    0x297             [ CF PF AF SF IF ]
cs        0x33             51
ss        0x2b             43
ds        0x0                0
es        0x0                0
fs        0x0                0
gs        0x0                0
(gdb) █
```



0x5555 5555 5170 주소로 점프 했기 때문에 해당 부분은 수행하지 않고 넘어간다.

기계어 분석 - 디버깅(if.c)-(7/11)

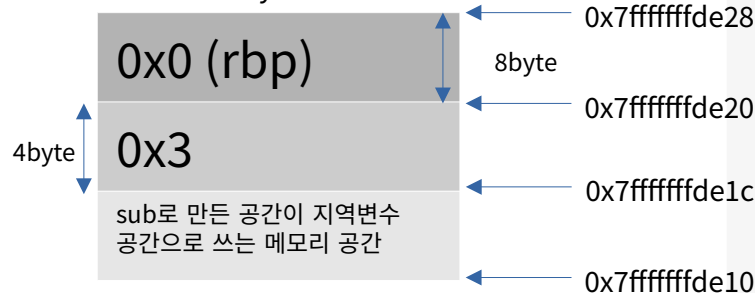
```
(gdb) disas
Dump of assembler code for function main:
0x00005555555149 <+0>:    endbr64
0x0000555555514e <+4>:    push    %rbp
0x0000555555514e <+5>:    mov     %rsp,%rbp
0x00005555555151 <+8>:    sub     $0x10,%rsp
0x00005555555151 <+12>:   movl    $0x3,%eax(%rbp)
0x00005555555151 <+13>:   cmpl    $0x7,%eax(%rbp)
0x00005555555151 <+19>:   cmpl    $0x7,%eax(%rbp)
0x00005555555151 <+23>:   jle     0x5555555170 <main+39>
0x00005555555151 <+25>:   lea     0xe9b(%rip),%rdi    # 0x5555555600a
0x00005555555151 <+32>:   callq   0x5555555177 <puts@plt>
0x00005555555151 <+37>:   jmp     0x555555517c <main+51>
=> 0x00005555555170 <+39>:   lea     0xe93(%rip),%rdi    # 0x5555555600a
0x00005555555170 <+46>:   callq   0x5555555177 <puts@plt>
0x00005555555170 <+51>:   mov     $0x0,%eax
0x00005555555181 <+56>:   leaveq  0(%rip),%rsp
0x00005555555182 <+57>:   retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555170      0x55555555170 <main+39>
eflags   0x297             [ CF PF AF SF IF ]
cs       0x33             51
ss       0x2b             43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) []
```

```
(gdb) disas
Dump of assembler code for function main:
0x00005555555149 <+0>:    endbr64
0x0000555555514e <+4>:    push    %rbp
0x0000555555514e <+5>:    mov     %rsp,%rbp
0x00005555555151 <+8>:    sub     $0x10,%rsp
0x00005555555151 <+12>:   movl    $0x3,%eax(%rbp)
0x00005555555151 <+13>:   cmpl    $0x7,%eax(%rbp)
0x00005555555151 <+19>:   cmpl    $0x7,%eax(%rbp)
0x00005555555151 <+23>:   jle     0x5555555170 <main+39>
0x00005555555151 <+25>:   lea     0xe9b(%rip),%rdi    # 0x5555555600a
0x00005555555151 <+32>:   callq   0x5555555177 <puts@plt>
0x00005555555151 <+37>:   jmp     0x555555517c <main+51>
=> 0x00005555555170 <+39>:   lea     0xe93(%rip),%rdi    # 0x5555555600a
0x00005555555170 <+46>:   callq   0x5555555177 <puts@plt>
0x00005555555170 <+51>:   mov     $0x0,%eax
0x00005555555181 <+56>:   leaveq  0(%rip),%rsp
0x00005555555182 <+57>:   retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x55555555190      93824992235920
rcx      0x55555555190      93824992235920
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x5555555600a      93824992239626
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555177      0x55555555177 <main+46>
eflags   0x297             [ CF PF AF SF IF ]
cs       0x33             51
ss       0x2b             43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) []
```

Memory 메모리



lea 는 배열 명령어 이다.

lea 0xe93(%rip),%rdi 는 0xe93(%rip)값이 rdi에 배치

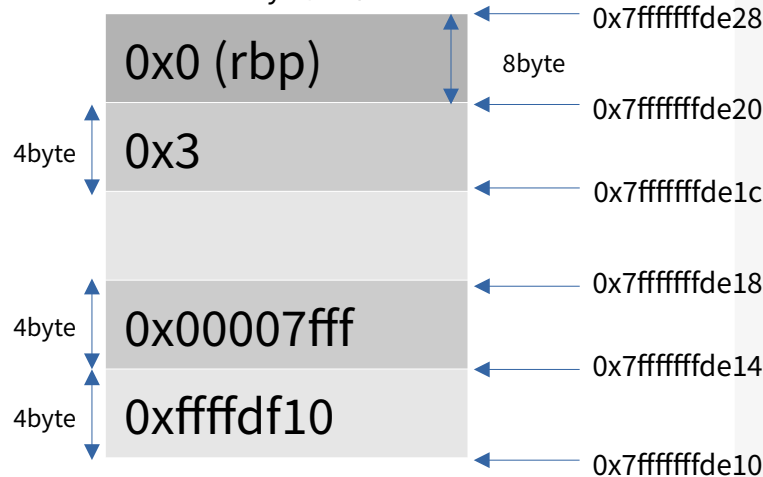
기계어 분석 - 디버깅(if.c)-(8/11)

```
00000555555510e <+23>: jle 0x000005555555117c <main+39>
000005555555110e <+25>: lea 0xe9b(&rtp),&rtdi # 0x555555550004
000005555555110e <+32>: callq 0x55555555517c <puts@plt>
000005555555110e <+37>: jmp 0x55555555517c <main+51>
0000055555551177 <+40>: callq 0xe9b(&rtp),&rtdi # 0x555555550004
0000055555551177 <+41>: mov 0xe9b(&rtp),&rtdi # 0x555555550004
0000055555551181 <+56>: leaveq %rax,%rax
0000055555551181 <+57>: retq
End of assembler dump.
```

```
(gdb) info registers
rax 0x555555555149 93824992235849
rbx 0x555555555190 93824992235920
rcx 0x555555555190 93824992235920
rdx 0x7fffffffdf28 140737488346920
rsi 0x7fffffffdf18 140737488346904
rdi 0x55555555600a 93824992239626
rbp 0x7fffffffde20 0x7fffffffde20
rsp 0x7fffffffde10 0x7fffffffde10
r8 0x0 0
r9 0x7ffff7fe0d0 140737354009952
r10 0xf 15
r11 0x2 2
r12 0x55555555060 93824992235616
r13 0x7fffffffdf10 140737488346896
r14 0x0 0
r15 0x0 0
r16 0x55555555177 0x55555555177 <main+46>
eflags 0x297 [ CF PF AF SF IF ]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb)
```

```
(gdb) info registers
rax 0x0 0
rbx 0x555555555190 93824992235920
rcx 0x7ffff7ecf077 140737352888439
rdx 0x0 0
rsi 0x5555555592a0 93824992252576
rdi 0x7ffffffaf7e0 140737353807840
rbp 0x7fffffffde20 0x7fffffffde20
rsp 0x7fffffffde10 0x7fffffffde10
r8 0x6 6
r9 0x7c 124
r10 0x7ffff7adbe0 140737353800672
r11 0x246 582
r12 0x55555555060 93824992235616
r13 0x7fffffffdf10 140737488346896
r14 0x0 0
r15 0x0 0
r16 0x5555555517c 0x5555555517c <main+51>
eflags 0x246 [ PF ZF IF ]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb)
```

Memory 메모리



callq 명령어
call은 기본적으로 push+jmp로 구성되어 있다.
함수호출이 끝난 이후에 실행해야할 어셈블리 명령어의 주소값을 push로 저장한다.

cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,
jle = jump less or equal 로 작거나 같을때 점프한다는 의미
0x5555 5555 5170 주소로 점프 한다.

```
(gdb) x $rsp+4
0x7fffffffde14: 0x00007fff
(gdb) x $rsp
0x7fffffffde10: 0xffffdf10
```

printf 함수는 n 명령어로 실행하여 넘긴다.
si 명령어로 실행시 아래의 내부 어셈블리 명령어로 이동하여 실행하게 된다.

```
(gdb) disas
Dump of assembler code for function puts@plt:
=> 0x000055555555050 <+0>: endbr64
0x000055555555054 <+4>: bnd jmpq *0x2f75(&rtp) # 0x555555557fd0 <puts@got.plt>
0x00005555555505b <+11>: nopl 0x0(%rax,%rax,1)
End of assembler dump.
```

기계어 분석 - 디버깅(if.c)-(9/11)

```
0x00005555555100 <+23>: jle 0x555555517c <main+39>
0x00005555555102 <+25>: lea 0xe9b(%rip),rdi # 0x5555555000a
0x00005555555109 <+32>: callq 0x5555555058 <puts@plt>
0x0000555555510e <+37>: jmp 0x555555517c <main+51>
0x00005555555116 <+39>: lea 0xe93(%rip),rdi # 0x5555555000a
0x00005555555117 <+40>: callq 0x5555555058 <puts@plt>
0x0000555555511c <+51>: mov $0x0,%eax
0x0000555555511e <+56>: leaqeq %eax,%eax
0x00005555555122 <+57>: retq
End of assembler dump.
```

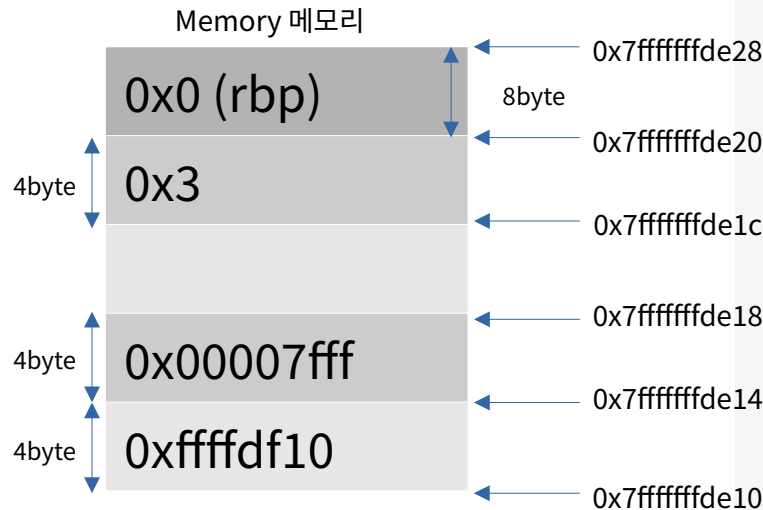
```
0x00005555555130 <+19>: cmpl $0x7,-0x4(%rbp)
0x00005555555136 <+23>: jle 0x555555517c <main+39>
0x00005555555138 <+25>: lea 0xe9b(%rip),rdi # 0x5555555000a
0x0000555555513d <+32>: callq 0x5555555058 <puts@plt>
0x00005555555142 <+37>: jmp 0x555555517c <main+51>
0x00005555555148 <+39>: lea 0xe93(%rip),rdi # 0x5555555000a
0x00005555555149 <+40>: callq 0x5555555058 <puts@plt>
0x0000555555514e <+51>: mov $0x0,%eax
0x00005555555150 <+56>: leaqeq %eax,%eax
0x00005555555156 <+57>: retq
End of assembler dump.
```

```
(gdb) info registers
rax 0x555555555149 93824992235849
rbx 0x555555555190 93824992235920
rcx 0x555555555190 93824992235920
rdx 0x7fffffffdf28 140737488346920
rsi 0x7fffffffdf18 140737488346904
rdi 0x55555555500a 93824992239620
rbp 0x7fffffffde20 0x7fffffffde20
rsp 0x7fffffffde10 0x7fffffffde10
r8 0x0 0
r9 0x7ffff7fe0d60 140737354009952
r10 0xf 15
r11 0x2 2
r12 0x555555555060 93824992235616
r13 0x7fffffffdf10 140737488346896
r14 0x0 0
r15 0x0 0
rip 0x555555555177 0x555555555177 <main+46>
eflags 0x297 [ CF PF AF SF IF ]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb) █
```

```
(gdb) info registers
rax 0x0 6
rbx 0x555555555190 93824992235920
rcx 0x7fffffffec077 140737352880439
rdx 0x0 0
rsi 0x5555555552a0 93824992252576
rdi 0x7fffffffaf7e0 140737353807840
rbp 0x7fffffffde20 0x7fffffffde20
rsp 0x7fffffffde10 0x7fffffffde10
r8 0x6 6
r9 0x7c 124
r10 0x7ffff7fadbe0 140737353800672
r11 0x246 582
r12 0x555555555060 93824992235616
r13 0x7fffffffdf10 140737488346896
r14 0x0 0
r15 0x0 0
rip 0x55555555517c 0x55555555517c <main+51>
eflags 0x246 [ PF IF ]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb) █
```

mov \$0x0, %eax 는 %eax에 0x0을 넣는다.

```
(gdb) x $eax
0x0: Cannot access memory at address 0x0
```



기계어 분석 - 디버깅(if.c)-(10/11)

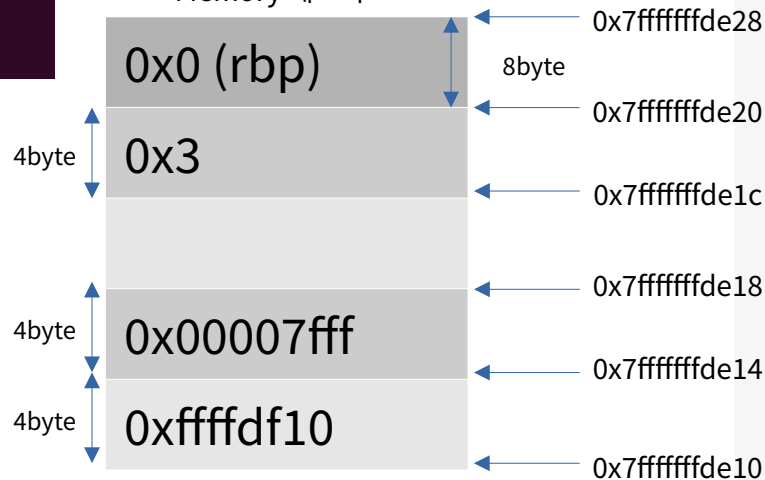
```
0x0000555555510c <+19>:  cmpi    $0x7,0x4(%rbp)
0x00005555555110 <+23>:  jle     0x5555555139 <main+39> # 0x5555555600a
0x000055555551102 <+25>:  lea     0x0(%rip),%rdi # 0x5555555600a
0x000055555551104 <+32>:  callq   0x5555555600 <put@plt>
0x000055555551106 <+37>:  jmp     0x555555517c <main+51> # 0x5555555600a
0x000055555551108 <+39>:  lea     0x0(%rip),%rdi # 0x5555555600a
0x00005555555110a <+40>:  callq   0x5555555600 <put@plt>
0x00005555555110c <+51>:  mov     $0x0,%eax
0x00005555555110e <+57>:  retq
End of assembler dump.
```

```
0x00005555555110e <+37>:  jmp     0x555555517c <main+51>
0x000055555551170 <+39>:  lea     0xe93(%rip),%rdi # 0x5555555600a
0x000055555551177 <+46>:  callq   0x5555555600 <puts@plt>
0x00005555555117c <+51>:  mov     $0x0,%eax
0x000055555551181 <+56>:  leaveq  %eax
0x000055555551182 <+57>:  retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x6      6
rbx      0x5555555190 93824992235920
rcx      0x7ffff7ecf077 140737352888439
rdx      0x0      0
rsi      0x5555555592a0 93824992252576
rdi      0x7ffff7faf7e0 140737353807840
rbp      0x7ffff7fde20 0x7ffff7fde20
rsp      0x7ffff7fde10 0x7ffff7fde10
r8       0x6      6
r9       0x7c     124
r10      0x7ffff7fadbe0 140737353800672
r11      0x246    582
r12      0x55555555060 93824992235616
r13      0x7ffff7fdf10 1407373488346896
r14      0x0      0
r15      0x0      0
rip      0x55555555517c 0x55555555517c <main+51>
eflags   0x246    [ PF ZF IF ]
cs       0x33     51
ss       0x2b     43
ds       0x0      0
es       0x0      0
fs       0x0      0
gs       0x0      0
(gdb) █
```

```
(gdb) info registers
rax      0x0      0
rbx      0x5555555190 93824992235920
rcx      0x7ffff7ecf077 140737352888439
rdx      0x0      0
rsi      0x5555555592a0 93824992252576
rdi      0x7ffff7faf7e0 140737353807840
rbp      0x7ffff7fde20 0x7ffff7fde20
rsp      0x7ffff7fde10 0x7ffff7fde10
r8       0x6      6
r9       0x7c     124
r10      0x7ffff7fadbe0 140737353800672
r11      0x246    582
r12      0x55555555060 93824992235616
r13      0x7ffff7fdf10 1407373488346896
r14      0x0      0
r15      0x0      0
rip      0x555555555181 0x555555555181 <main+56>
eflags   0x246    [ PF ZF IF ]
cs       0x33     51
ss       0x2b     43
ds       0x0      0
es       0x0      0
fs       0x0      0
gs       0x0      0
```

Memory 메모리



mov \$0x0, %eax 는 %eax에 0x0을 넣는다.

기계어 분석 - 디버깅(if.c)-(11/11)

```
00000555555510e <+37>: jmp 0x5555555517c <main+51>
000005555555170 <+39>: lea 0xe93(Nrip),rdi # 0x5555555000a
000005555555177 <+40>: callq 0x5555555000a <puts@plt>
00000555555517c <+51>: mov $0x0,%eax
000005555555181 <+56>: leaveq
000005555555182 <+57>: retq
End of assembler dump.
```

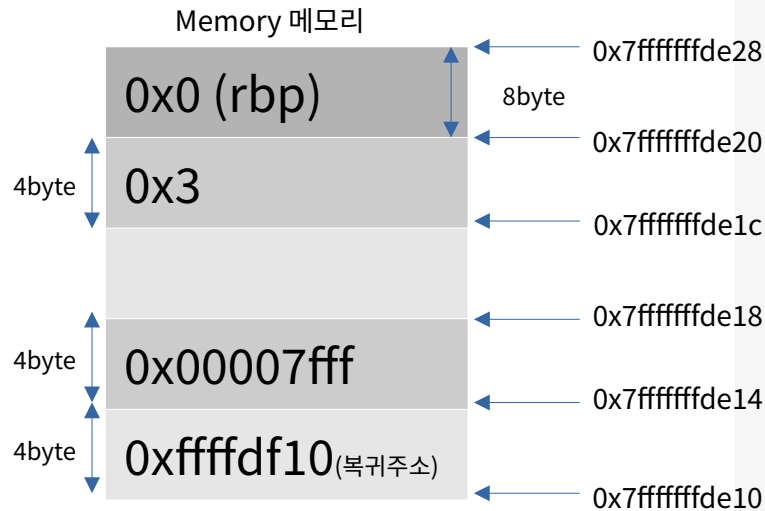
```
(gdb) info registers
rax      0x0      0
rbx      0x55555555190 93824992235920
rcx      0x7ffff7ecf077 140737352888439
rdx      0x0      0
rsi      0x5555555592a0 93824992252576
rdi      0x7ffff7faf7e0 140737353807840
rbp      0x7ffffffffffde20 0x7ffffffffffde20
rsp      0x7ffffffffffde10 0x7ffffffffffde10
r8       0x6      6
r9       0x7c     124
r10      0x7ffff7fadbe0 140737353800672
r11      0x246    582
r12      0x555555555060 93824992235616
r13      0x7ffffffffffdf10 140737488346896
r14      0x0      0
r15      0x0      0
rip      0x555555555181 0x555555555181 <main+56>
eflags   0x246    [ PF ZF IF ]
cs       0x33     51
ss       0x2b     43
ds       0x0      0
es       0x0      0
fs       0x0      0
gs       0x0      0
```

```
(gdb) info registers
rax      0x0      0
rbx      0x555555555190 93824992235920
rcx      0x7ffff7ecf077 140737352888439
rdx      0x0      0
rsi      0x5555555592a0 93824992252576
rdi      0x7ffff7faf7e0 140737353807840
rbp      0x0      0
rsp      0x7ffffffffffde28 0x7ffffffffffde28
r8       0x6      6
r9       0x7c     124
r10      0x7ffff7fadbe0 140737353800672
r11      0x246    582
r12      0x555555555060 93824992235616
r13      0x7ffffffffffdf10 140737488346896
r14      0x0      0
r15      0x0      0
rip      0x555555555182 0x555555555182 <main+57>
eflags   0x246    [ PF ZF IF ]
cs       0x33     51
ss       0x2b     43
ds       0x0      0
es       0x0      0
fs       0x0      0
gs       0x0      0
(gdb)
```

leaveq 명령어는 스택 해제 명령어 이다.

```
(gdb) x $rsp
0x7ffffffffffde10: 0xffffdf10
```

```
(gdb) x $rsp
0x7ffffffffffde28: 0xf7de5083
```



기계어 분석 - 디버깅(for.c)-(1/11)

```
#include <stdio.h>

int main(void)
{
    int i = 0;
    char ch = 'A';

    for (; i < 26; i++, ch++)
    {
        printf("%c, %c\n", ch, ch ^ 0x20);
    }

    return 0;
}
```

```
(gdb) disas
Dump of assembler code for function main:
=> 0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
0x000055555555155 <+12>:   movl    $0x0,-0x4(%rbp)
0x00005555555515c <+19>:   movb    $0x41,-0x5(%rbp)
0x000055555555160 <+23>:   jmp     0x55555555191 <main+72>
0x000055555555162 <+25>:   movzbl  -0x5(%rbp),%eax
0x000055555555166 <+29>:   xor     $0x20,%eax
0x000055555555169 <+32>:   movsbl  %al,%edx
0x00005555555516c <+35>:   movsbl  -0x5(%rbp),%eax
0x000055555555170 <+39>:   mov     %eax,%esi
0x000055555555172 <+41>:   lea     0xe8b(%rip),%rdi    # 0x555555556004
0x000055555555179 <+48>:   mov     $0x0,%eax
0x00005555555517e <+53>:   callq   0x55555555050 <printf@plt>
0x000055555555183 <+58>:   addl    $0x1,-0x4(%rbp)
0x000055555555187 <+62>:   movzbl  -0x5(%rbp),%eax
0x00005555555518b <+66>:   add     $0x1,%eax
0x00005555555518e <+69>:   mov     %al,-0x5(%rbp)
0x000055555555191 <+72>:   cmpl    $0x19,-0x4(%rbp)
0x000055555555195 <+76>:   jle     0x55555555162 <main+25>
0x000055555555197 <+78>:   mov     $0x0,%eax
0x00005555555519c <+83>:   leaveq  %eax
0x00005555555519d <+84>:   retq

End of assembler dump.
(gdb)
```

for 문 예제 작성

for 문의 어셈블리 동작을 분석하기 위한 코드.

for문을 통해 I 값을 출력 한다. (영문 대문자 → 소문자, 소문자 → 대문자 자동 출력)

기계어 분석 - 디버깅(for.c)-(2/11)

```
(gdb) disas
Dump of assembler code for function main:
=> 0x000055555555149 <+0>:    endbr64
   0x00005555555514d <+4>:    push   %rbp
   0x00005555555514e <+5>:    mov    %rsp,%rbp
```

```
(gdb) info registers
rax      0x55555555149  93824992235849
rbx      0x555555551a0  93824992235936
rcx      0x555555551a0  93824992235936
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1            1
rbp      0x0            0
rsp      0x7fffffffde28  0x7fffffffde28
r8       0x0            0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf            15
r11      0x2            2
r12      0x55555555060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0            0
r15      0x0            0
rip      0x5555555514d  0x5555555514d <main+4>
eflags   0x246          [ PF ZF IF ]
cs       0x33          51
ss       0x2b          43
ds       0x0            0
es       0x0            0
fs       0x0            0
gs       0x0            0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
   0x000055555555149 <+0>:    endbr64
   0x00005555555514d <+4>:    push   %rbp
=> 0x00005555555514e <+5>:    mov    %rsp,%rbp
   0x000055555555151 <+8>:    sub    $0x10,%rsp
```

```
(gdb) info registers
rax      0x55555555149  93824992235849
rbx      0x555555551a0  93824992235936
rcx      0x555555551a0  93824992235936
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1            1
rbp      0x0            0
rsp      0x7fffffffde20  0x7fffffffde20
r8       0x0            0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf            15
r11      0x2            2
r12      0x55555555060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0            0
r15      0x0            0
rip      0x5555555514e  0x5555555514e <main+5>
eflags   0x246          [ PF ZF IF ]
cs       0x33          51
ss       0x2b          43
ds       0x0            0
es       0x0            0
fs       0x0            0
gs       0x0            0
(gdb)
```

Memory 메모리

0x0 (rbp)

0x7fffffffde28

8byte

0x7fffffffde20

push 명령어는 현재 스택의 최상위 메모리 (rsp)에 값을 저장하는 명령어
즉, 현재 스택의 최상위 메모리(rsp)에 rbp값을 저장하라는 의미

기계어 분석 - 디버깅(for.c)-(3/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
=> 0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x0                0
rsp      0x7fffffffde20     0
r8        0x0                0
r9        0x7ffff7fe0d60    140737354009952
r10       0xf                15
r11       0x2                2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10    140737488346896
r14       0x0                0
r15       0x0                0
r16       0x5555555514e      0x5555555514e <main+5>
eflags   0x246              [ PF ZF IF ]
cs        0x33              51
ss        0x2b              43
ds        0x0                0
es        0x0                0
fs        0x0                0
gs        0x0                0
(gdb) █
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
=> 0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
0x000055555555155 <+12>:   movl    $0x0,%rax
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde20     0x7fffffffde20
r8        0x0                0
r9        0x7ffff7fe0d60    140737354009952
r10       0xf                15
r11       0x2                2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10    140737488346896
r14       0x0                0
r15       0x0                0
r16       0x55555555151      0x55555555151 <main+8>
eflags   0x246              [ PF ZF IF ]
cs        0x33              51
ss        0x2b              43
ds        0x0                0
es        0x0                0
fs        0x0                0
gs        0x0                0
(gdb) █
```

Memory 메모리

0x0 (rbp)

0x7fffffffde28

8byte

0x7fffffffde20

mov 명령어는 내용을 복사한다.

mov %rsp, %rbp 는 rbp에 rsp값을 복사한다.

즉, rbp 값에 rsp값을 넣으면서 rsp,rbp 주소값이 서로 같아 지면서 스택의 경계선이 사라진다.

→ 새로운 스택을 생성할 준비를 하는 과정

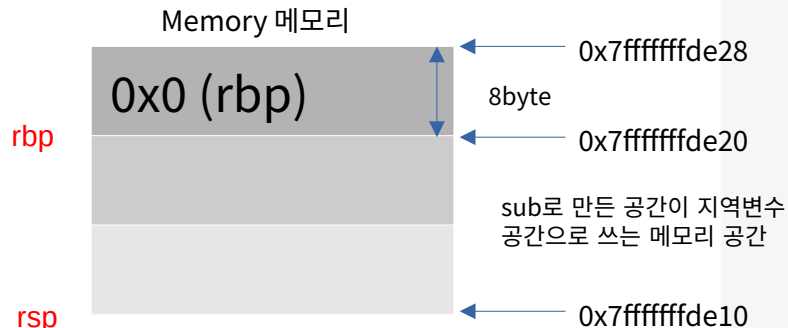
기계어 분석 - 디버깅(for.c)-(4/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:   endbr64
0x00005555555514d <+4>:   push    %rbp
0x00005555555514e <+5>:   mov     %rsp,%rbp
=> 0x000055555555151 <+8>:   sub     $0x10,%rsp
0x000055555555155 <+12>:  movl    $0x0,-0x4(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde20     0x7fffffffde20
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555151      0x55555555151 <main+8>
eflags   0x246             [ PF ZF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0               0
es       0x0               0
fs       0x0               0
gs       0x0               0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:   endbr64
0x00005555555514d <+4>:   push    %rbp
0x00005555555514e <+5>:   mov     %rsp,%rbp
0x000055555555151 <+8>:   sub     $0x10,%rsp
=> 0x000055555555155 <+12>:  movl    $0x0,-0x4(%rbp)
0x00005555555515c <+19>:  movb    $0x41,-0x5(%rbp)
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555155      0x55555555155 <main+12>
eflags   0x202             [ IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0               0
es       0x0               0
fs       0x0               0
gs       0x0               0
(gdb)
```



sub 명령어는 뺄셈 명령

sub %0x10, %rsp 는 rsp에 에서 16바이트를 뺀다.

(0x10 → 0001 0000 : $2^4 = 16$ 바이트 가상 주소공간에서는 바이트 단위로 움직인다.

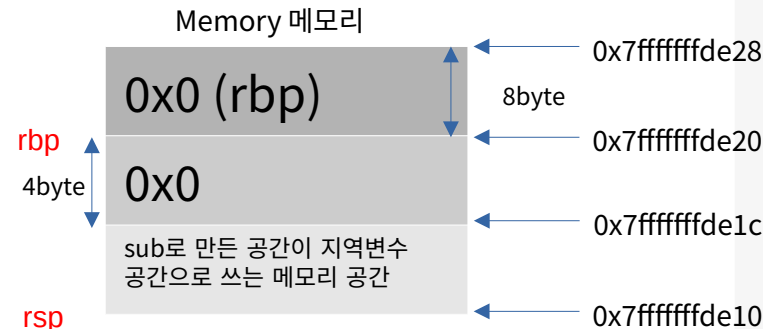
기계어 분석 - 디버깅(for.c)-(5/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
=> 0x000055555555155 <+12>:   movl    $0x0,-0x4(%rbp)
0x00005555555515c <+19>:   movb    $0x41,-0x5(%rbp)
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:    endbr64
0x00005555555514d <+4>:    push    %rbp
0x00005555555514e <+5>:    mov     %rsp,%rbp
0x000055555555151 <+8>:    sub     $0x10,%rsp
0x000055555555155 <+12>:   movl    $0x0,-0x4(%rbp)
0x00005555555515c <+19>:   movb    $0x41,-0x5(%rbp)
0x000055555555160 <+23>:   jmp     0x55555555191 <main+72>
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d60     140737354009952
r10       0xf
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10     140737488346896
r14       0x0
r15       0x0
rip       0x55555555155      0x55555555155 <main+12>
eflags    0x202             [ IF ]
cs        0x33             51
ss        0x2b             43
ds        0x0
es        0x0
fs        0x0
gs        0x0
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d60     140737354009952
r10       0xf
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10     140737488346896
r14       0x0
r15       0x0
rip       0x5555555515c      0x5555555515c <main+19>
eflags    0x202             [ IF ]
cs        0x33             51
ss        0x2b             43
ds        0x0
es        0x0
fs        0x0
gs        0x0
```



movl 명령어는 내용을 복사한다.
L 이 들어가면 4바이트 처리를 하겠다는 의미
movl %0x0, -0x4(%rbp) 는 rbp를 기준으로4바이트 뻥자리에 0x0값을 복사한다.
de20 - 4 → de1c

```
(gdb) x $rbp-4
0x7fffffffde1c: 0x00000000
```

```
#include <stdio.h>
int main(void)
{
    int i = 0;
    char ch = 'A';
```

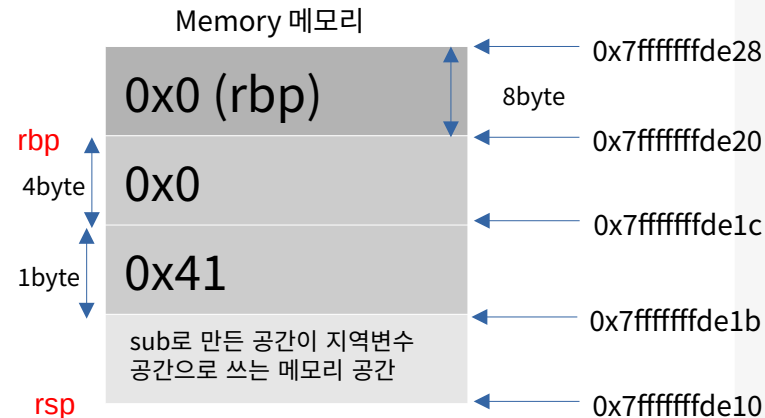
기계어 분석 - 디버깅(for.c)-(6/11)

```
(gdb) disas
Dump of assembler code for function main:
0x0000555555551149 <+0>:    endbr64
0x000055555555114d <+4>:    push    %rbp
0x000055555555114e <+5>:    mov     %rsp,%rbp
0x0000555555551151 <+8>:    sub     $0x10,%rsp
0x0000555555551155 <+12>:   movl    $0x0,-0x4(%rbp)
=> 0x000055555555115c <+19>:   movb    $0x41,-0x5(%rbp)
0x0000555555551160 <+23>:   jmp     0x555555551191 <main+72>
```

```
(gdb) info registers
rax      0x555555551149  93824992235849
rbx      0x5555555511a0  93824992235936
rcx      0x5555555511a0  93824992235936
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1
rbp      0x7fffffffde20  0x7fffffffde20
rsp      0x7fffffffde10  0x7fffffffde10
r8       0x0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf
r11      0x2
r12      0x555555550060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0
r15      0x0
rip      0x55555555115c  0x55555555115c <main+19>
eflags   0x202          [ IF ]
cs       0x33          51
ss       0x2b          43
ds       0x0
es       0x0
fs       0x0
gs       0x0
(gdb)
```

```
(gdb) disas
Dump of assembler code for function main:
0x0000555555551149 <+0>:    endbr64
0x000055555555114d <+4>:    push    %rbp
0x000055555555114e <+5>:    mov     %rsp,%rbp
0x0000555555551151 <+8>:    sub     $0x10,%rsp
0x0000555555551155 <+12>:   movl    $0x0,-0x4(%rbp)
0x000055555555115c <+19>:   movb    $0x41,-0x5(%rbp)
=> 0x0000555555551160 <+23>:   jmp     0x555555551191 <main+72>
0x0000555555551162 <+25>:   movzbl  -0x5(%rbp),%eax
```

```
(gdb) info registers
rax      0x555555551149  93824992235849
rbx      0x5555555511a0  93824992235936
rcx      0x5555555511a0  93824992235936
rdx      0x7fffffffdf28  140737488346920
rsi      0x7fffffffdf18  140737488346904
rdi      0x1
rbp      0x7fffffffde20  0x7fffffffde20
rsp      0x7fffffffde10  0x7fffffffde10
r8       0x0
r9       0x7ffff7fe0d60  140737354009952
r10      0xf
r11      0x2
r12      0x555555550060  93824992235616
r13      0x7fffffffdf10  140737488346896
r14      0x0
r15      0x0
rip      0x555555551160  0x555555551160 <main+23>
eflags   0x202          [ IF ]
cs       0x33          51
ss       0x2b          43
ds       0x0
es       0x0
fs       0x0
gs       0x0
(gdb)
```



mov 명령어는 내용을 복사한다.

b 이 들어가면 1바이트 처리를 하겠다는 의미

movl %0x41, -0x5(%rbp) 는 rbp를 기준으로 1바이트 뺀자리에 0x41값을 복사한다.

'A' = 16 * 4 + 1 * 1 = 64 + 1 십진수로 65

de20 - 5 → de1b

```
(gdb) x $rbp-5
0x7fffffffde1b: 0x00000041
```

```
#include <stdio.h>

int main(void)
{
    int i = 0;
    char ch = 'A';
}
```

기계어 분석 - 디버깅(for.c)-(7/11)

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:  endbr64
0x00005555555514a <+4>:  push  %rbp
0x00005555555514b <+5>:  mov   %rsp,%rbp
0x00005555555514c <+8>:  sub   $0x10,%rsp
0x00005555555514d <+12>: movl  $0x0,-0x4(%rbp)
0x00005555555514e <+19>: movl  $0x41,-0x5(%rbp)
=> 0x00005555555514f <+23>: jmp    0x55555555191 <main+72>
0x000055555555150 <+25>:  movzbl -0x5(%rbp),%eax
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28      140737488346920
rsi      0x7fffffffdf18      140737488346904
rdi      0x1
rbp      0x7fffffffde20      0x7fffffffde20
rsp      0x7fffffffde10      0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d0      140737354009952
r10       0xf15
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10      140737488346896
r14       0x0
r15       0x0
rip      0x55555555160      0x55555555160 <main+23>
eflags    [ IF ]
cs        0x33
ss        0x2b
ds        0x0
es        0x0
fs        0x0
gs        0x0
```

```
0x000055555555160 <+23>:  jmp    0x55555555191 <main+72>
0x000055555555162 <+25>:  movzbl -0x5(%rbp),%eax
0x000055555555164 <+29>:  xor    $0x0,%eax
0x000055555555166 <+32>:  movsbl %eax,%edx
0x000055555555168 <+35>:  movsbl -0x5(%rbp),%eax
0x000055555555170 <+39>:  mov    %eax,%esi
0x000055555555172 <+41>:  lea    0x6b0(%rip),%rdi # 0x55555555060
0x000055555555174 <+48>:  mov    $0x0,%eax
0x000055555555176 <+53>:  callq  0x55555555080 <printf@plt>
0x000055555555183 <+58>:  addl   $0x1,-0x4(%rbp)
0x000055555555187 <+62>:  movzbl -0x5(%rbp),%eax
0x00005555555518b <+66>:  add    $0x1,%eax
0x00005555555518e <+69>:  mov    %eax,-0x5(%rbp)
=> 0x000055555555191 <+72>:  jle     0x55555555102 <main+25>
0x000055555555195 <+76>:  jle     0x55555555102 <main+25>
0x000055555555197 <+78>:  mov    $0x0,%eax
0x00005555555519c <+83>:  leaveq  $0x0,%eax
0x00005555555519d <+84>:  retq
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28      140737488346920
rsi      0x7fffffffdf18      140737488346904
rdi      0x1
rbp      0x7fffffffde20      0x7fffffffde20
rsp      0x7fffffffde10      0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d0      140737354009952
r10       0xf15
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10      140737488346896
r14       0x0
r15       0x0
rip      0x55555555191      0x55555555191 <main+72>
eflags    [ IF ]
cs        0x33
ss        0x2b
ds        0x0
es        0x0
fs        0x0
gs        0x0
```

jmp 명령어는 해당 주소로 이동한다.
0x5555 5555 5191 주소로 점프 한다.



기계어 분석 - 디버깅(for.c)-(8/11)

```
0x00005555555510e <+69>: mov    %al,-0x5(%rbp)
=> 0x000055555555191 <+72>: cmpl   $0x19,-0x4(%rbp)
0x000055555555195 <+76>: jle     0x55555555162 <main+25>
0x000055555555197 <+78>: mov     $0x0,%eax
0x00005555555519c <+83>: leaveq  %eax
0x00005555555519d <+84>: retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28      140737488346920
rsi      0x7fffffffdf18      140737488346904
rdi      0x1
rbp      0x7fffffffde20      0x7fffffffde20
rsp      0x7fffffffde10      0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d60      140737354009952
r10       0xf
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10      140737488346896
r14       0x0
r15       0x0
r16       0x0
r17       0x0
r18       0x0
r19       0x0
r20       0x0
r21       0x0
r22       0x0
r23       0x0
r24       0x0
r25       0x0
r26       0x0
r27       0x0
r28       0x0
r29       0x0
r30       0x0
r31       0x0
eflags    0x202             [ IF ]
cs        0x33
ss        0x2b
ds        0x0
es        0x0
fs        0x0
gs        0x0
(gdb) □
```

```
0x00005555555510e <+25>: movzbl -0x5(%rbp),%eax
0x00005555555510f <+29>: xor     $0x0,%eax
0x000055555555110 <+32>: movsbl  %al,%edx
0x000055555555111 <+35>: movsbl  -0x5(%rbp),%eax
0x000055555555112 <+39>: mov     %eax,%esi
0x000055555555113 <+41>: lea     0xe8b(%rip),%rdi
0x000055555555114 <+48>: mov     $0x0,%eax
0x000055555555115 <+53>: callq   0x55555555050 <printf@plt>
0x000055555555116 <+58>: addl    $0x1,-0x4(%rbp)
0x000055555555117 <+62>: movzbl  -0x5(%rbp),%eax
0x000055555555118 <+66>: add     $0x1,%eax
0x000055555555119 <+69>: mov     %al,-0x5(%rbp)
0x00005555555511a <+72>: cmpl    $0x19,-0x4(%rbp)
=> 0x00005555555511b <+76>: jle     0x55555555162 <main+25>
0x00005555555511c <+78>: mov     $0x0,%eax
0x00005555555511d <+83>: leaveq  %eax
0x00005555555511e <+84>: retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28      140737488346920
rsi      0x7fffffffdf18      140737488346904
rdi      0x1
rbp      0x7fffffffde20      0x7fffffffde20
rsp      0x7fffffffde10      0x7fffffffde10
r8        0x0
r9        0x7ffff7fe0d60      140737354009952
r10       0xf
r11       0x2
r12       0x55555555060      93824992235616
r13       0x7fffffffdf10      140737488346896
r14       0x0
r15       0x0
r16       0x55555555195      0x55555555195 <main+76>
eflags    0x297             [ CF PF AF SF IF ]
cs        0x33
ss        0x2b
ds        0x0
es        0x0
fs        0x0
gs        0x0
(gdb) □
```

cmpl 명령어는 내용을 비교한다.

| 이 들어가면 4바이트 처리를 하겠다는 의미

cmpl \$0x19, -0x4(%rbp) 는 0x19값과 rbp를 기준으로 4바이트 뺀자리의 값 (0x3)을 비교한다.

cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,

jle = jump less or equal 로 작거나 같을때 점프한다는 의미

0x5555 5555 5162 주소로 점프 한다.

기계어 분석 - 디버깅(for.c)-(1/11)

```
0x000055555555162 <+25>: movzbl -0x5(%rbp),%eax
0x000055555555166 <+29>: xor $0x20,%eax
0x000055555555169 <+32>: movsbl %al,%edx
0x00005555555516c <+35>: movsbl -0x5(%rbp),%eax
0x000055555555170 <+39>: mov %eax,%esi
0x000055555555172 <+41>: lea 0x6b(%rip),%rdi #
0x000055555555179 <+48>: mov $0x0,%eax
0x00005555555517e <+53>: callq 0x55555555050 <printf@plt>
0x000055555555183 <+58>: addl $0x1,-0x4(%rbp)
0x000055555555187 <+62>: movzbl -0x5(%rbp),%eax
0x00005555555518b <+66>: add $0x1,%eax
0x00005555555518e <+69>: mov %al,-0x5(%rbp)
0x000055555555191 <+72>: cmpl $0x19,-0x4(%rbp)
=> 0x000055555555195 <+76>: jle 0x55555555162 <main+25>
0x000055555555197 <+78>: mov $0x0,%eax
0x00005555555519c <+83>: leaveq
0x00005555555519d <+84>: retq
End of assembler dump.
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555195      0x55555555195 <main+76>
eflags   0x297             [ CF PF AF SF IF ]
cs       0x33              51
ds       0x2b              43
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) █
```

```
(gdb) disas
Dump of assembler code for function main:
0x000055555555149 <+0>:     endbr64
0x00005555555514d <+4>:     push %rbp
0x00005555555514e <+5>:     mov %rsp,%rbp
0x000055555555151 <+8>:     sub $0x10,%rsp
0x000055555555155 <+12>:    movl $0x0,-0x4(%rbp)
0x00005555555515c <+19>:    movb $0x41,-0x5(%rbp)
0x000055555555160 <+23>:    jmp 0x55555555191 <main+72>
=> 0x000055555555162 <+25>:    movzbl -0x5(%rbp),%eax
0x000055555555166 <+29>:    xor $0x20,%eax
```

```
(gdb) info registers
rax      0x55555555149      93824992235849
rbx      0x555555551a0      93824992235936
rcx      0x555555551a0      93824992235936
rdx      0x7fffffffdf28     140737488346920
rsi      0x7fffffffdf18     140737488346904
rdi      0x1                1
rbp      0x7fffffffde20     0x7fffffffde20
rsp      0x7fffffffde10     0x7fffffffde10
r8       0x0                0
r9       0x7ffff7fe0d60     140737354009952
r10      0xf                15
r11      0x2                2
r12      0x55555555060      93824992235616
r13      0x7fffffffdf10     140737488346896
r14      0x0                0
r15      0x0                0
rip      0x55555555162      0x55555555162 <main+25>
eflags   0x297             [ CF PF AF SF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
(gdb) █
```