EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

제 4기

2022. 09. 16

유진선

기계어 분석 – 디버깅(if.c) -(1/11)

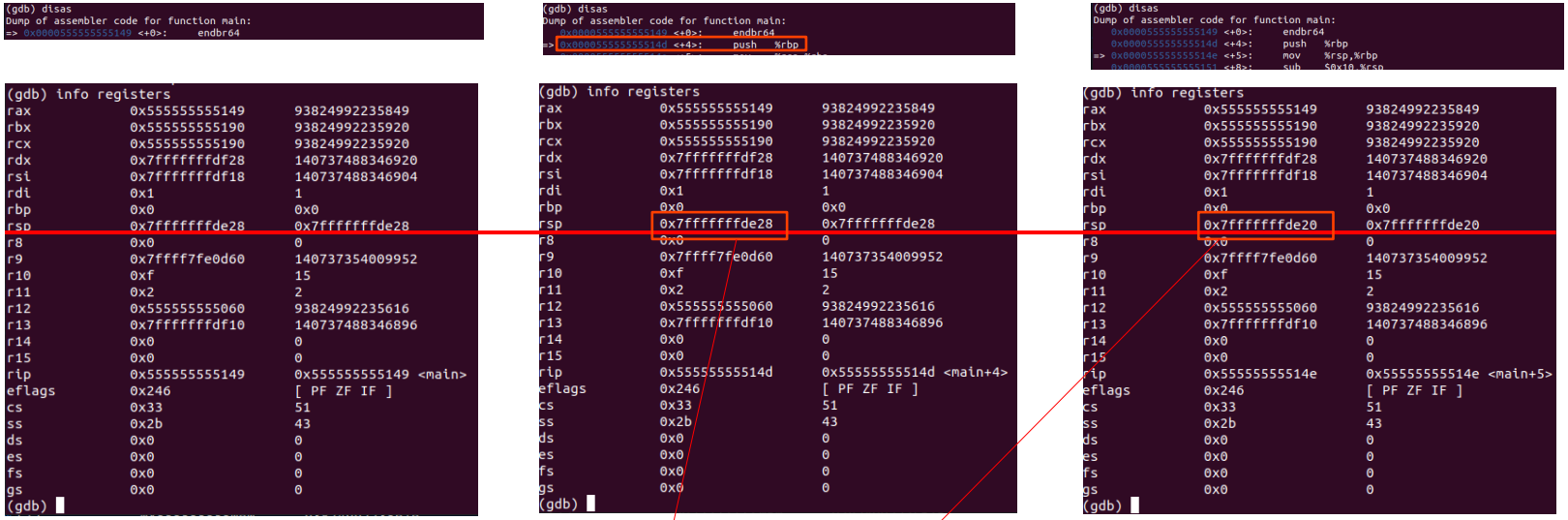
```c
#include <stdio.h>

int main(void)
{
        int num = 3;

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

        return 0;

}
```

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
   0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
   0x0000555555555160 <+23>:    jle    0x555555555170 <main+39>
   0x0000555555555162 <+25>:    lea    0xe9b(%rip),%rdi        # 0x555555556004
   0x0000555555555169 <+32>:    callq  0x555555555050 <puts@plt>
   0x000055555555516e <+37>:    jmp    0x55555555517c <main+51>
   0x0000555555555170 <+39>:    lea    0xe93(%rip),%rdi        # 0x55555555600a
   0x0000555555555177 <+46>:    callq  0x555555555050 <puts@plt>
   0x000055555555517c <+51>:    mov    $0x0,%eax
   0x0000555555555181 <+56>:    leaveq
   0x0000555555555182 <+57>:    retq
End of assembler dump.
(gdb)
```

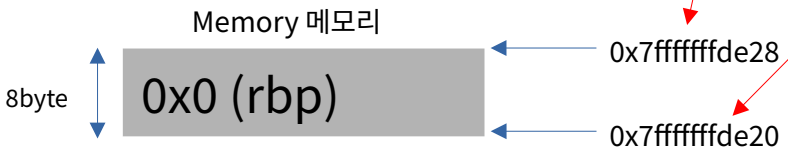
if 문 예제 작성
if 문의 어셈블리 동작을 분석하기 위한 코드.
num 의 변수 크기 비교해서 글자를 출력한다.

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)



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



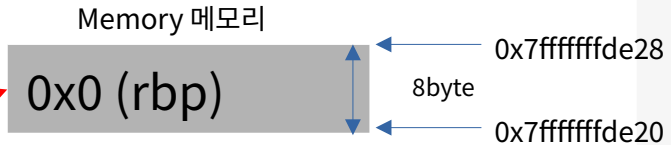
기계어 분석 – 디버깅(if.c)-(3/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x000055555555514a <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
=> 0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x55555555514e      0x55555555514e <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:
   0x000055555555514a <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
=> 0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555151      0x555555555151 <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)
```



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

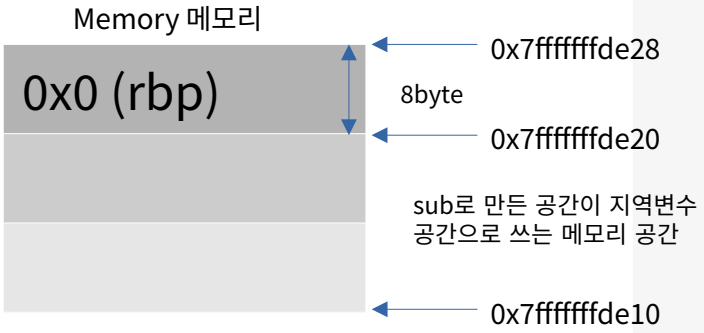
기계어 분석 – 디버깅(if.c)-(4/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
=> 0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555151      0x555555555151 <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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
=> 0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555155      0x555555555155 <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 = 16바이트 가상 주소공간에서는 바이트 단위로 움직인다.

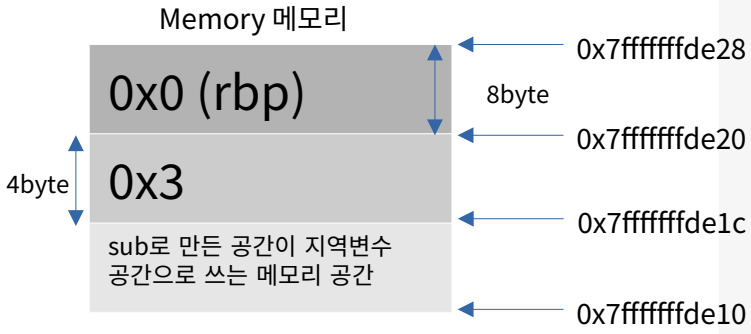
기계어 분석 – 디버깅(if.c)-(5/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
=> 0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555155      0x555555555155 <main+12>
eflags         0x202               [ 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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
=> 0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515c      0x55555555515c <main+19>
eflags         0x202               [ 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 뺀자리에 0x3값을 복사한다.
de20 – 4 → de1c

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

```c
#include <stdio.h>

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

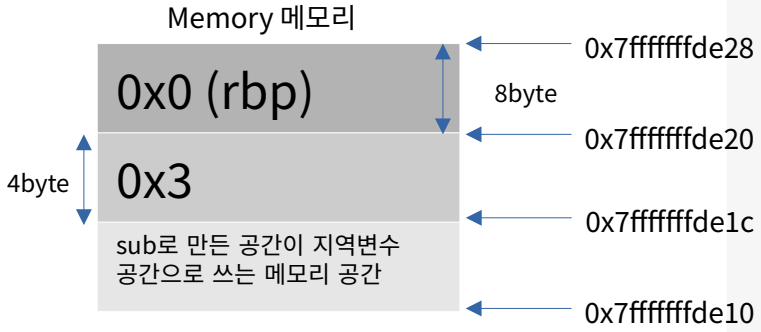
기계어 분석 – 디버깅(if.c)-(6/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
=> 0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515c      0x55555555515c <main+19>
eflags         0x202               [ 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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
   0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
=> 0x0000555555555160 <+23>:    jle    0x555555555170 <main+39>
   0x0000555555555162 <+25>:    lea    0xe9b(%rip),%rdi        # 0x555555556004
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555160      0x555555555160 <main+23>
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)
```



cmpl 명령어는 내용을 비교한다.
l 이 들어가면 4바이트 처리를 하겠다는 의미
cmpl $0x7, -0x4(%rbp) 는 0x7값과 rbp를 기준으로4바이트 뺀자리의 값 (0x3)을 비교한다.

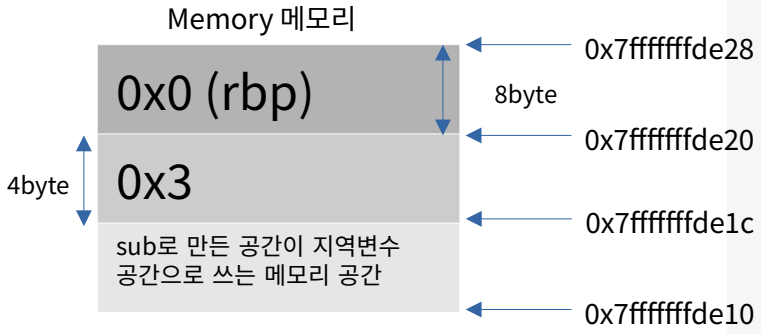
cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,
jle = jump less or equal 로 작거나 같을때 점프한하는 의미
0x5555 5555 5170 주소로 점프 한다.

```c
#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)

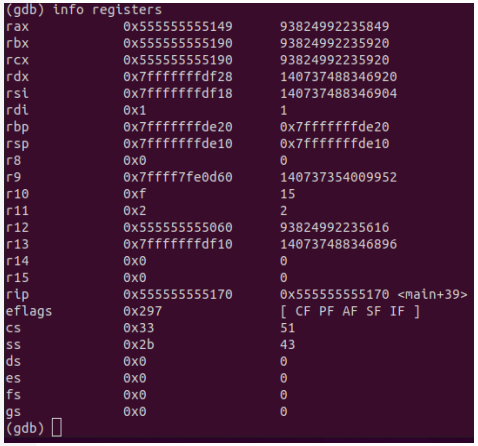
```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x3,-0x4(%rbp)
   0x000055555555515c <+19>:    cmpl   $0x7,-0x4(%rbp)
   0x0000555555555160 <+23>:    jle    0x555555555170 <main+39>
   0x0000555555555162 <+25>:    lea    0xe9b(%rip),%rdi        # 0x555555556004
   0x0000555555555169 <+32>:    callq  0x555555555050 <puts@plt>
   0x000055555555516e <+37>:    jmp    0x55555555517c <main+51>
=> 0x0000555555555170 <+39>:    lea    0xe93(%rip),%rdi        # 0x55555555600a
   0x0000555555555177 <+46>:    callq  0x555555555050 <puts@plt>
   0x000055555555517c <+51>:    mov    $0x0,%eax
   0x0000555555555181 <+56>:    leaveq
   0x0000555555555182 <+57>:    retq
End of assembler dump.
```

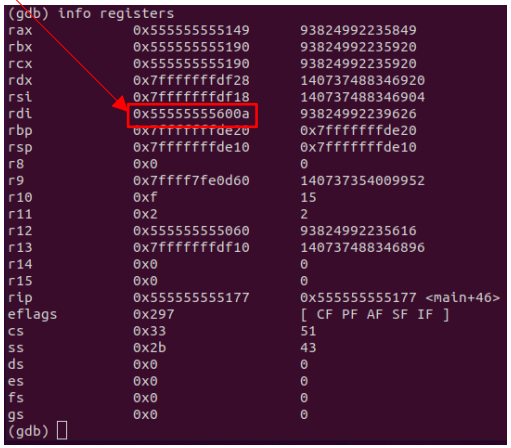
```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      93824992235920
rdx            0x7fffffffdf28      140737488346920
rsi            0x7fffffffdf18      140737488346904
rdi            0x1                 1
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            0x555555555170      0x555555555170 <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)
```

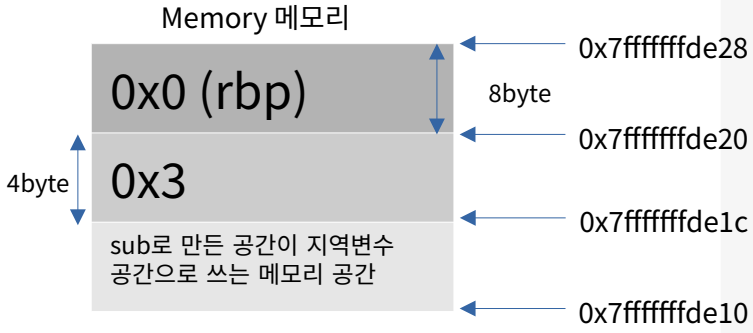


0x5555 5555 5170 주소로 점프 했기 때문에 해당 부분은 수행하지 않고 넘어간다.

기계어 분석 – 디버깅(if.c)-(7/11)

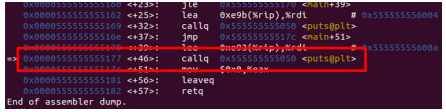


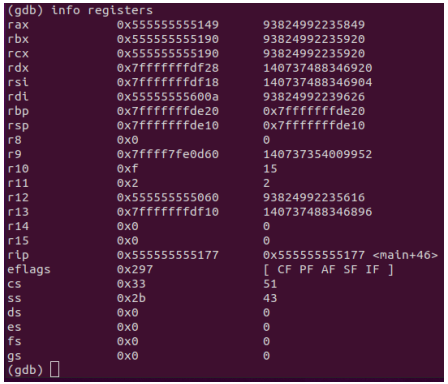


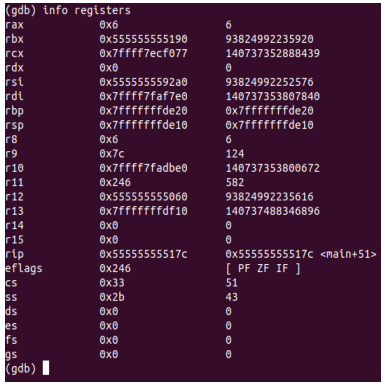


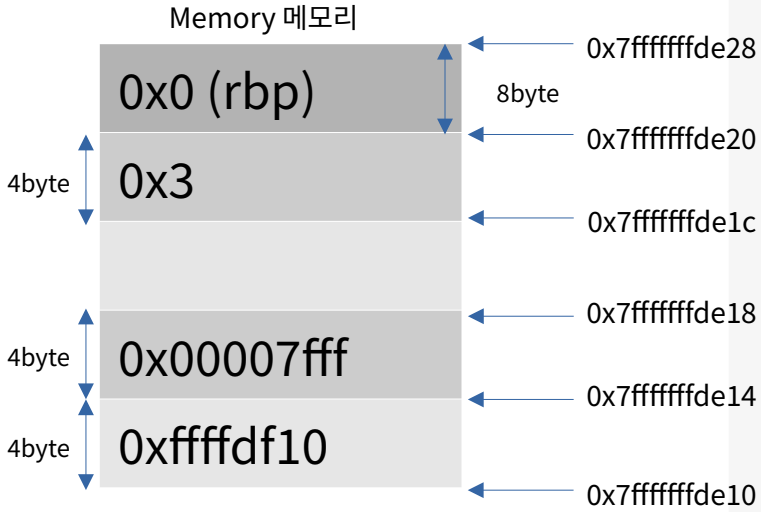
lea 는 배열 명령어 이다.
lea 0xe93(%rip),%rdi 는 0xe93(%rip)값이 rdi에 배치

기계어 분석 – 디버깅(if.c)-(8/11)









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

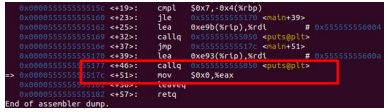
cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,
jle = jump less or equal 로 작거나 같을때 점프한하는 의미
0x5555 5555 5170 주소로 점프 한다.

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

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

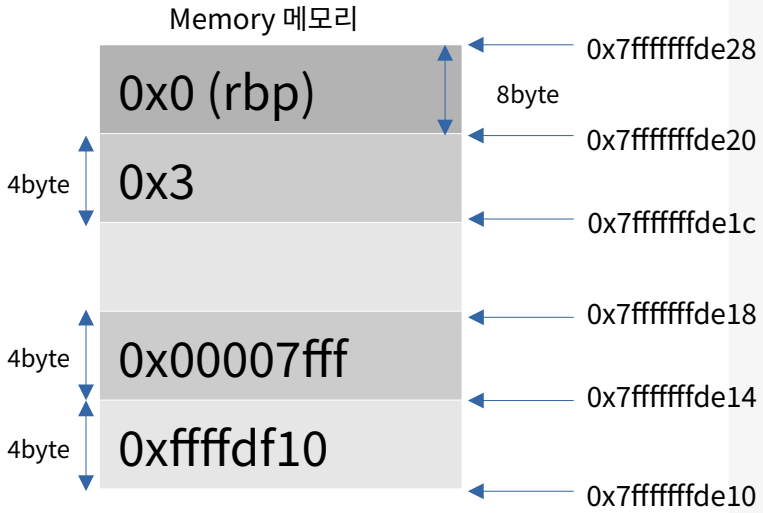
```
(gdb) disas
Dump of assembler code for function puts@plt:
=> 0x00005555555550b0 <+0>:     endbr64
   0x00005555555550b4 <+4>:     bnd jmpq *0x2f75(%rip)        # 0x555555557fd0 <puts@got.plt>
   0x00005555555550bb <+11>:    nopl   0x0(%rax,%rax,1)
End of assembler dump.
```

기계어 분석 – 디버깅(if.c)-(9/11)




```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x555555555190      93824992235920
rcx            0x555555555190      93824992235920
rdx            0x7fffffffdf28      140737488346920
rsi            0x7fffffffdf18      140737488346904
rdi            0x555555556000a     93824992239626
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)
```

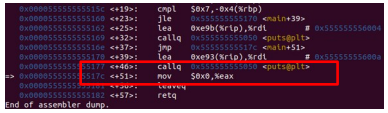
```
(gdb) info registers
rax            0x6                 6
rbx            0x555555555190      93824992235920
rcx            0x7ffff7ecf077      140737352888439
rdx            0x0                 0
rsi            0x5555555592a0      93824992252576
rdi            0x7ffff7faf7e0      140737353807840
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 ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

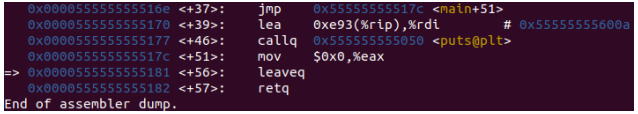


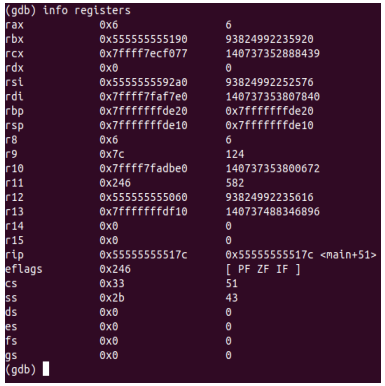
mov $0x0, %eax 는 %eax에 0x0을 넣는다.

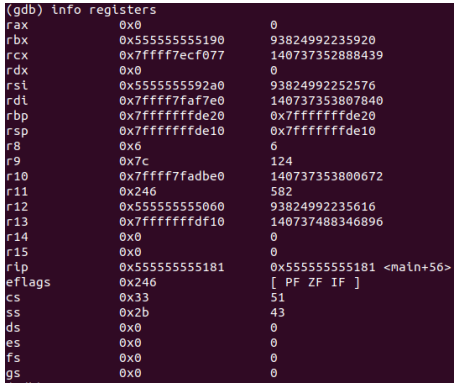
```
(gdb) x $eax
0x0:    Cannot access memory at address 0x0
```

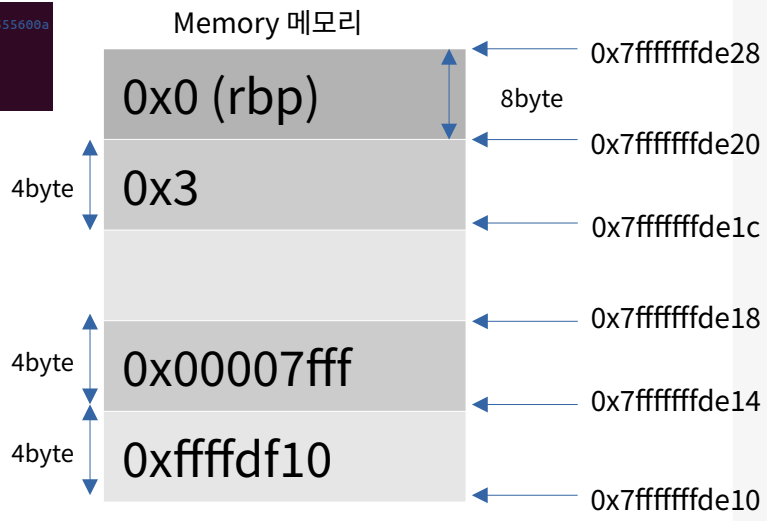
기계어 분석 – 디버깅(if.c)-(10/11)





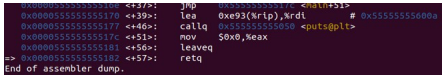


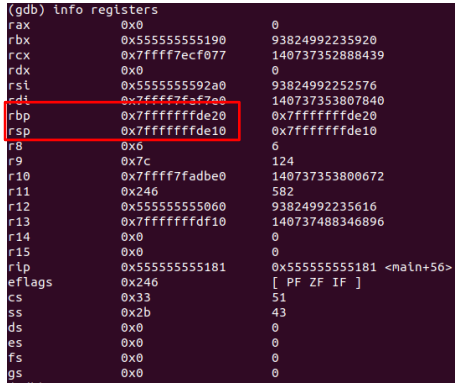


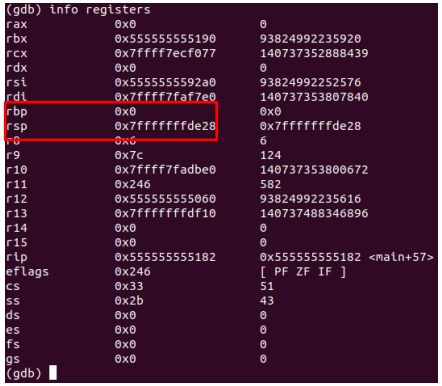


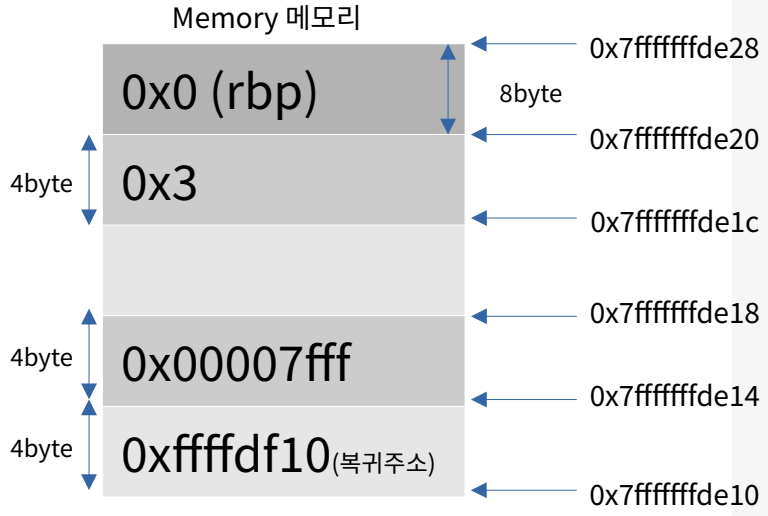
mov $0x0, %eax 는 %eax에 0x0을 넣는다.

기계어 분석 – 디버깅(if.c)-(11/11)

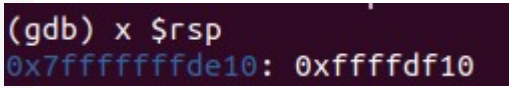








leaveq 명령어는 스택 해제 명령어 이다.



기계어 분석 – 디버깅(for.c)-(1/11)

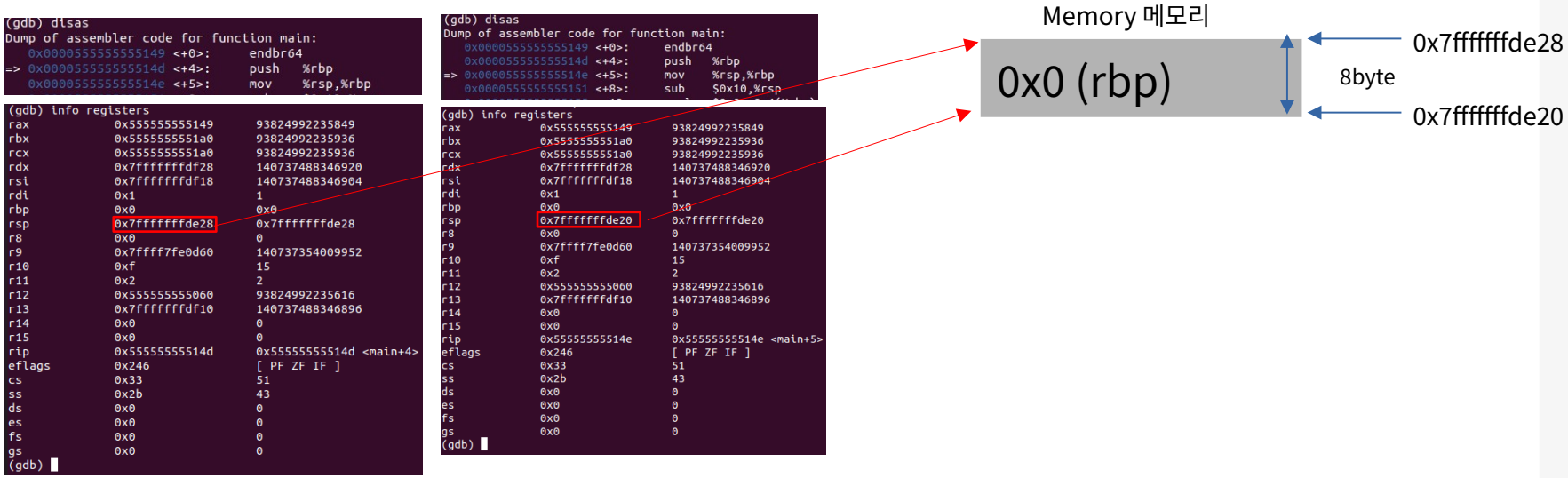
EDDI
Electronic Design Development Institute

```c
#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:
=> 0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
   0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
   0x0000555555555160 <+23>:    jmp    0x555555555191 <main+72>
   0x0000555555555162 <+25>:    movzbl -0x5(%rbp),%eax
   0x0000555555555166 <+29>:    xor    $0x20,%eax
   0x0000555555555169 <+32>:    movsbl %al,%edx
   0x000055555555516c <+35>:    movsbl -0x5(%rbp),%eax
   0x0000555555555170 <+39>:    mov    %eax,%esi
   0x0000555555555172 <+41>:    lea    0xe8b(%rip),%rdi        # 0x555555556004
   0x0000555555555179 <+48>:    mov    $0x0,%eax
   0x000055555555517e <+53>:    callq  0x555555555050 <printf@plt>
   0x0000555555555183 <+58>:    addl   $0x1,-0x4(%rbp)
   0x0000555555555187 <+62>:    movzbl -0x5(%rbp),%eax
   0x000055555555518b <+66>:    add    $0x1,%eax
   0x000055555555518e <+69>:    mov    %al,-0x5(%rbp)
   0x0000555555555191 <+72>:    cmpl   $0x19,-0x4(%rbp)
   0x0000555555555195 <+76>:    jle    0x555555555162 <main+25>
   0x0000555555555197 <+78>:    mov    $0x0,%eax
   0x000055555555519c <+83>:    leaveq
   0x000055555555519d <+84>:    retq
End of assembler dump.
(gdb)
```

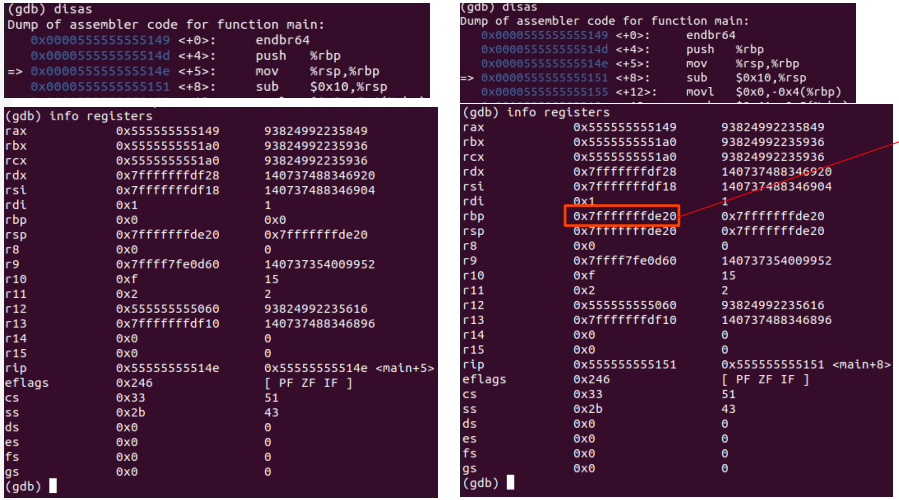
for 문 예제 작성
for 문의 어셈블리 동작을 분석하기 위한 코드.
for문을 통해 I 값을 출력 한다. (영문 대문자 →소문자, 소문자 → 대문자 자동 출력)

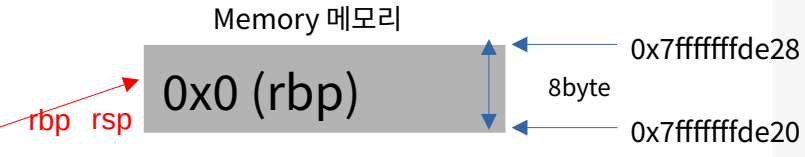
기계어 분석 – 디버깅(for.c)-(2/11)



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)





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

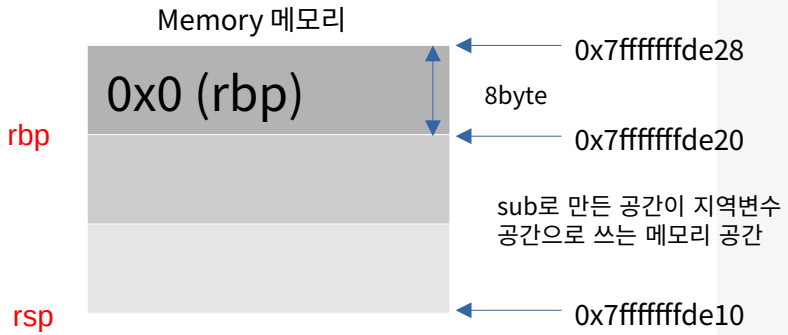
기계어 분석 – 디버깅(for.c)-(4/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
=> 0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555151      0x555555555151 <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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
=> 0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
   0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555155      0x555555555155 <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 = 16바이트 가상 주소공간에서는 바이트 단위로 움직인다.

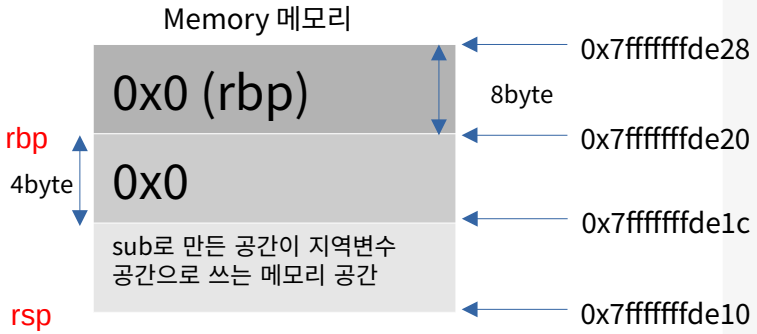
기계어 분석 – 디버깅(for.c)-(5/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
=> 0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
   0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
=> 0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
   0x0000555555555160 <+23>:    jmp    0x555555555191 <main+72>
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555155      0x555555555155 <main+12>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515c      0x55555555515c <main+19>
eflags         0x202               [ 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 %0x0, -0x4(%rbp) 는 rbp를 기준으로4바이트 밴자리에 0x0값을 복사한다.
de20 – 4 → de1c

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

```c
#include <stdio.h>

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

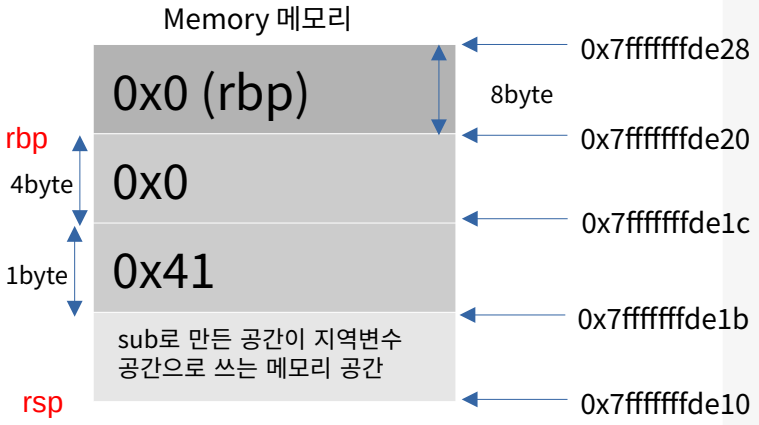
기계어 분석 – 디버깅(for.c)-(6/11)

```
(gdb) disas
Dump of assembler code for function main:
   0x000055555555514d <+0>:     endbr64
   0x0000555555555151 <+4>:     push   %rbp
   0x0000555555555152 <+5>:     mov    %rsp,%rbp
   0x0000555555555155 <+8>:     sub    $0x10,%rsp
   0x0000555555555159 <+12>:    movl   $0x0,-0x4(%rbp)
=> 0x0000555555555160 <+19>:    movb   $0x41,-0x5(%rbp)
   0x0000555555555164 <+23>:    jmp    0x555555555191 <main+72>
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515c      0x55555555515c <main+19>
eflags         0x202               [ 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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
   0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
=> 0x0000555555555160 <+23>:    jmp    0x555555555191 <main+72>
   0x0000555555555162 <+25>:    movzbl -0x5(%rbp),%eax
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555160      0x555555555160 <main+23>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb)
```

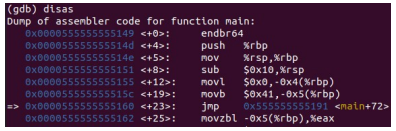


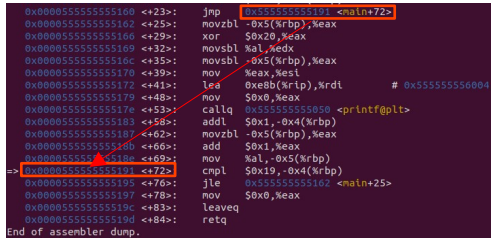
mov 명령어는 내용을 복사한다.
b 이 들어가면 1바이트 처리를 하겠다는 의미
movl %0x41, -0x5(%rbp) 는 rbp를 기준으로1바이트 뺀자리에 0x41값을 복사한다.
'A' = 16 *4 + 1*1 = 64 + 1 십진수로 65
de20 – 5 → de1b

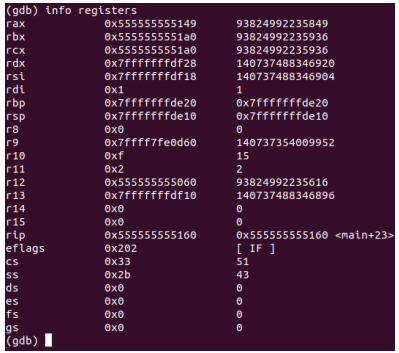
```
(gdb) x $rbp-5
0x7fffffffde1b: 0x00000041
```

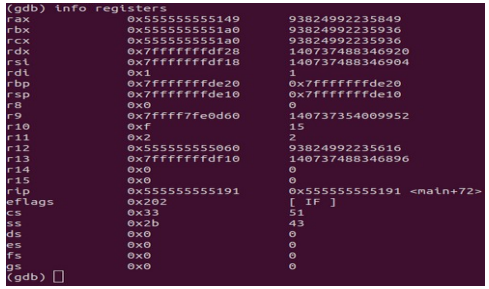
```c
#include <stdio.h>

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

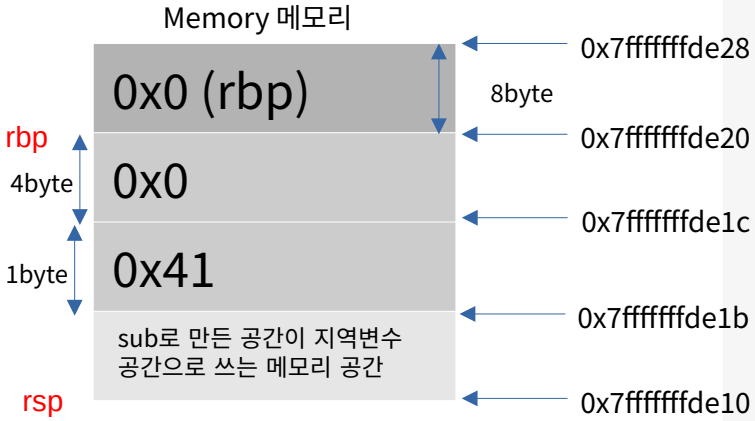
기계어 분석 – 디버깅(for.c)-(7/11)





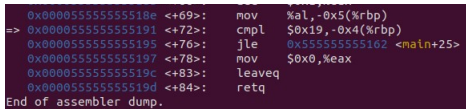


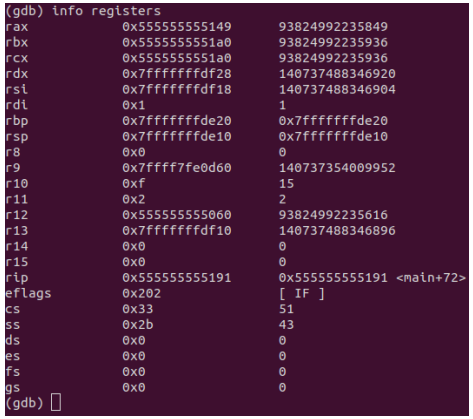


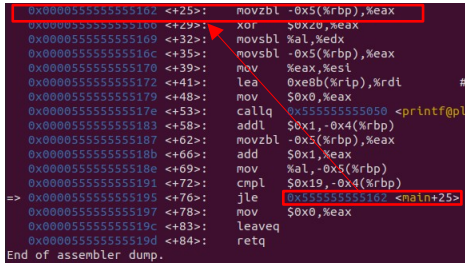


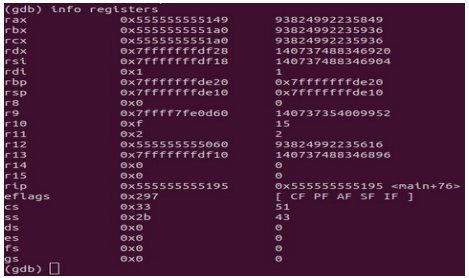
jmp 명령어는 해당 주소로 이동한다.
0x5555 5555 5191 주소로 점프 한다.

기계어 분석 – 디버깅(for.c)-(8/11)









cmpl 명령어는 내용을 비교한다.
l 이 들어가면 4바이트 처리를 하겠다는 의미
cmpl $0x19, -0x4(%rbp) 는 0x19값과 rbp를 기준으로4바이트 뺀자리의 값 (0x3)을 비교한다.

cmp의 경우 보통 jmp명령어와 같이 사용하게 되는데,
jle = jump less or equal 로 작거나 같을때 점프한하는 의미
0x5555 5555 5162 주소로 점프 한다.

기계어 분석 – 디버깅(for.c)-(1/11)

EDDI
Electronic Design Development Institute

```
   0x0000555555555162 <+25>:    movzbl -0x5(%rbp),%eax
   0x0000555555555166 <+29>:    xor    $0x20,%eax
   0x0000555555555169 <+32>:    movsbl %al,%edx
   0x000055555555516c <+35>:    movsbl -0x5(%rbp),%eax
   0x0000555555555170 <+39>:    mov    %eax,%esi
   0x0000555555555172 <+41>:    lea    0xe8b(%rip),%rdi        #
   0x0000555555555179 <+48>:    mov    $0x0,%eax
   0x000055555555517e <+53>:    callq  0x555555555050 <printf@pl
   0x0000555555555183 <+58>:    addl   $0x1,-0x4(%rbp)
   0x0000555555555187 <+62>:    movzbl -0x5(%rbp),%eax
   0x000055555555518b <+66>:    add    $0x1,%eax
   0x000055555555518e <+69>:    mov    %al,-0x5(%rbp)
   0x0000555555555191 <+72>:    cmpl   $0x19,-0x4(%rbp)
=> 0x0000555555555195 <+76>:    jle    0x555555555162 <main+25>
   0x0000555555555197 <+78>:    mov    $0x0,%eax
   0x000055555555519c <+83>:    leaveq
   0x000055555555519d <+84>:    retq
End of assembler dump.
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555195      0x555555555195 <main+76>
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:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     sub    $0x10,%rsp
   0x0000555555555155 <+12>:    movl   $0x0,-0x4(%rbp)
   0x000055555555515c <+19>:    movb   $0x41,-0x5(%rbp)
   0x0000555555555160 <+23>:    jmp    0x555555555191 <main+72>
=> 0x0000555555555162 <+25>:    movzbl -0x5(%rbp),%eax
   0x0000555555555166 <+29>:    xor    $0x20,%eax
```

```
(gdb) info registers
rax            0x555555555149      93824992235849
rbx            0x5555555551a0      93824992235936
rcx            0x5555555551a0      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            0x555555555060      93824992235616
r13            0x7fffffffdf10      140737488346896
r14            0x0                 0
r15            0x0                 0
rip            0x555555555162      0x555555555162 <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)
```