에디 로봇 아카데미

임베디드 마스터 Lv2 과정

제 2 기

2021.10.08

김진호

```
(gdb) l mult
        # include <stdio.h>

        int mult(int n1, int n2)
        {
                return n1 * n2;
        }

        int main(void)
        {
        int num = 3, num2 = 2;
            int res;

      res = mult(num, num2);
        printf("res = %d₩n", res);

        return 0;
   }
```

간단한 C 코드를 disassemble 하여 C 코드가 동작하는 원리와 CPU 내부 register 와 stack 에 대한 이해를 합니다.

```
(gdb) disas
Dump of assembler code for function main:
=> 0x0000555555555160 <+0>:    endbr64
   0x0000555555555164 <+4>:    push   %rbp
   0x0000555555555165 <+5>:    mov    %rsp,%rbp
   0x0000555555555168 <+8>:    sub    $0x10,%rsp
   0x000055555555516c <+12>:   movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:   movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:   mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:   mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:   mov    %edx,%esi
   0x0000555555555182 <+34>:   mov    %eax,%edi
   0x0000555555555184 <+36>:   callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:   mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:   mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:   mov    %eax,%esi
   0x0000555555555191 <+49>:   lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:   mov    $0x0,%eax
   0x000055555555519d <+61>:   callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:   mov    $0x0,%eax
   0x00005555555551a7 <+71>:   leaveq
   0x00005555555551a8 <+72>:   retq
End of assembler dump.
```

---

gdb 에서 disas 명령어를 통해 out file에 기계어를 어셈블러로 disassemble 하여 볼수있습니다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x7fffffffe068     140737488347240
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x0                0x0
rsp            0x7fffffffdf68     0x7fffffffdf68
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x555555555160     0x555555555160 <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) x $rsp
0x7fffffffdf68:    0xf7deb0b3
```

---

info register 명령어를 통해 전체 CPU 레지스터를 볼수있고, 'x $<register>' 명령어를 통해 해당 레지스터 주소의 값을 HEX 단위로 볼수있습니다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
=> 0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

disas 명령어 실행시 '=>'기호를 통해 현재 실행하기 바로 전 위치를 알 수 있습니다.

(gdb) x $rsp

0x7fffffffdf68: 0xf7deb0b3

현재 스택 포인터의 주소는 0x7fffffffdf68이며, 스택포인터 주소안에 값은 0xf7deb0b3임을 알수있습니다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
=> 0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

disas 명령어를 통해 'push %rbp' 어셈블러 코드가 실행되었음을 알수있습니다.

push 명령어는 push <reg> 해당 레지스터의 값을 스택으로 복사(이동)시키는 명령어 입니다.
레지스터 앞에 %<reg> 기호가 붙는다면 레지스터 주소를 의미합니다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x7fffffffe068     140737488347240
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x0                0x0
rsp            0x7fffffffdf60     0x7fffffffdf60
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x555555555165     0x555555555165 <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) x $rsp
0x7fffffffdf60:    0x00000000
(gdb) x $rbp
0x0:    Cannot access memory at address 0x0
```

$rbp 레지스터 데이터 값이 0x0 이므로 $r네 레지스터 데이터 값이 0x00000000으로 변경된 것을 확인할 수 있습니다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
=> 0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

disas 명령어를 통해 'mov %rsp,%rbp' 어셈블러 코드가 실행되었음을 알수있습니다.

mov 명령어는 mov <reg1>,<reg2> 형식으로 <reg1> 레지스터의 주소를 <reg2> 레지스터 주소로 변경시키는 명령어 입니다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x7fffffffe068     140737488347240
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x7fffffffdf60     0x7fffffffdf60
rsp            0x7fffffffdf60     0x7fffffffdf60
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x555555555168     0x555555555168 <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) x $rsp
0x7fffffffdf60:    0x00000000
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

실행결과 rsp와 rbp의 주소 값이 같아진 것을 확인 할 수 있습니다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
=> 0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

sub 명령어는 마이너스 기능을 수행합니다.

sub a1, a2 형식으로 a1 의 값을 a2 만큼 감소한 후, 결과값을 a1 에 저장합니다.

프로그램이 실행되고 메모리에 로드될때 프로그램의 구조는 "코드-데이터-힙-스택" 순으로 쌓이게 됩니다. 이때 힙영역과 스택영역을 가변적으로 설정 하기 위해서 힙영역은 양수방향으로 자라나고, 스택영역은 음수방향으로 자라나게 됩니다.

때문에, 스택 포인터의 주소값이 증가한게 아닌 감소하여 음수방향으로 스택 포인터가 증가하게 됩니다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x7fffffffe068     140737488347240
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x7fffffffdf60     0x7fffffffdf60
rsp            0x7fffffffdf50     0x7fffffffdf50
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x55555555516c     0x55555555516c <main+12>
eflags         0x206              [ PF IF ]
cs             0x33               51
ss             0x2b               43
ds             0x0                0
es             0x0                0
fs             0x0                0
gs             0x0                0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

베이스 포인터는 스택 프레임의 시작(기준점)이 되고, 스택 포인터는 스택의 최상단을 가르키게 됩니다.

베이스 포인터(rbp)와 스택 포인터(rsp) 사이에 0x10 의 메모리 공간(스택 프레임)안에 데이터가 쌓이면서 프로그램이 수행됩니다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
=>0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

movl 명령어는 32bit(4Byte) 크기를 다루는 명령어이다.

$rbp – 0xc 에 0x00000003 을 저장하고

$rbp – 0x8 에 0x00000002 를 저장한다.

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
=>0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

movl 명령어는 32bit(4Byte) 크기를 다루는 명령어이다.

$rbp – 0xc 에 0x00000003 을 저장하고

$rbp – 0x8 에 0x00000002 를 저장한다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x7fffffffe068     140737488347240
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x7fffffffdf60     0x7fffffffdf60
rsp            0x7fffffffdf50     0x7fffffffdf50
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x55555555517a     0x55555555517a <main+26>
eflags         0x206              [ PF IF ]
cs             0x33               51
ss             0x2b               43
ds             0x0                0
es             0x0                0
fs             0x0                0
gs             0x0                0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
=> 0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

컴파일 시점에서 compile optimizer 를 통해 컴파일러의 최적화 옵션을 변경할수있습니다.

optimizer 를 변경하면 이 부분의 코드는 불필요 하기 때문에 최적화되어 더 이상 발생하지 않습니다.

```
(gdb) info register
rax            0x555555555160     93824992235872
rbx            0x5555555551b0     93824992235952
rcx            0x5555555551b0     93824992235952
rdx            0x2                2
rsi            0x7fffffffe058     140737488347224
rdi            0x1                1
rbp            0x7fffffffdf60     0x7fffffffdf60
rsp            0x7fffffffdf50     0x7fffffffdf50
r8             0x0                0
r9             0x7ffff7fe0d50     140737354009936
r10            0xb                11
r11            0x2                2
r12            0x555555555060     93824992235616
r13            0x7fffffffe050     140737488347216
r14            0x0                0
r15            0x0                0
rip            0x55555555517d     0x55555555517d <main+29>
eflags         0x206              [ PF IF ]
cs             0x33               51
ss             0x2b               43
ds             0x0                0
es             0x0                0
fs             0x0                0
gs             0x0                0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
=> 0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

```
(gdb) info register
rax            0x3                 3
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x7fffffffe058      140737488347224
rdi            0x1                 1
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555180      0x555555555180 <main+32>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
=> 0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

mov 명령어로 edx 의 값을 esi 로 복사한다.

```
(gdb) info register
rax            0x3                 3
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x1                 1
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555182      0x555555555182 <main+34>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
=> 0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

mov 명령어로 edx 의 값을 esi 로 복사한다.

```
(gdb) info register
rax            0x3                 3
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555184      0x555555555184 <main+36>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

---

mov 명령어로 edx 의 값을 esi 로 복사한다.

(gdb) disas

Dump of assembler code for function mult:

```
=> 0x0000555555555149 <+0>:    endbr64
   0x000055555555514d <+4>:    push   %rbp
   0x000055555555514e <+5>:    mov    %rsp,%rbp
   0x0000555555555151 <+8>:    mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:   mov    %esi,-0x8(%rbp)
   0x0000555555555157 <+14>:   mov    -0x4(%rbp),%eax
   0x000055555555515a <+17>:   imul   -0x8(%rbp),%eax
   0x000055555555515e <+21>:   pop    %rbp
   0x000055555555515f <+22>:   retq
```

End of assembler dump.

---

mult 함수 메모리 영역에서 새롭게 스택프레임을 생성하고 새롭게 생성된 스택프레임 안에서 코드가 동작하는 것을 확인할수있다.

스택프레임이 나눠지기 때문에 C 언어에서 다른 함수에 사용된 변수를 임의로 접근하는 것이 불가능하고 만약 임의로 다른 함수의 변수에 접근하려면 포인터(메모리에 직접 접근)를 사용하여 처리한다.

```
(gdb) info register
rax            0x3                 3
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf48      0x7fffffffdf48
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555149      0x555555555149 <mult>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf48:    0x55555189
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function mult:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    %esi,-0x8(%rbp)
   0x0000555555555157 <+14>:    mov    -0x4(%rbp),%eax
   0x000055555555515a <+17>:    imul   -0x8(%rbp),%eax
=> 0x000055555555515e <+21>:    pop    %rbp
   0x000055555555515f <+22>:    retq
End of assembler dump.
```

---

imul 명령어는 곱셈 명령어이다.

eax 레지스터와 곱셈이 가능하며, 레지스터와 메모리만 대상이 될수있다.

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf40      0x7fffffffdf40
rsp            0x7fffffffdf40      0x7fffffffdf40
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515e      0x55555555515e <mult+21>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf40:    0xffffdf60
(gdb) x $rbp
0x7fffffffdf40:    0xffffdf60
```

```
(gdb) disas
Dump of assembler code for function mult:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    %esi,-0x8(%rbp)
   0x0000555555555157 <+14>:    mov    -0x4(%rbp),%eax
   0x000055555555515a <+17>:    imul   -0x8(%rbp),%eax
   0x000055555555515e <+21>:    pop    %rbp
=> 0x000055555555515f <+22>:    retq
```

---

pop 명령어는 현재 rsp 레지스터가 가지고 있는 스택 메모리 주소에 있는 데이터를 인자로 복사하고, rsp 포인터의 값을 +0x4 한다. (이 때, 무조건 4바이트 단위로 복사하게 된다.) rsp 포인터는 pop 메모리에 의해 커진다.

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf48      0x7fffffffdf48
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x55555555515f      0x55555555515f <mult+22>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf48:    0x55555189
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
=> 0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555189      0x555555555189 <main+41>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
=> 0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x55555555518c      0x55555555518c <main+44>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x2                 2
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x55555555518f      0x55555555518f <main+47>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
=> 0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

---

lea 명령어는 lea a1, a2 형식이며, a1 레지스터에 a2의 주소 값을 복사한다.

rip 명령어는 현재 실행중인 명령어의 주소 값을 가지고 있다.

따라서 lea 명령어는 rdi에 저장되어 있는 값을 rip로 복사한 뒤 현재 사용중인 명령어의 주소 값을 바꿔주는 역할을 한다.

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x6                 6
rdi            0x3                 3
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555191      0x555555555191 <main+49>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
=> 0x0000555555555198 <+56>:    mov    $0x0,%eax
   0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

```
(gdb) info register
rax            0x6                 6
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x6                 6
rdi            0x555555556004      93824992239620
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x555555555198      0x555555555198 <main+56>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```

```
(gdb) disas
Dump of assembler code for function main:
   0x0000555555555160 <+0>:     endbr64
   0x0000555555555164 <+4>:     push   %rbp
   0x0000555555555165 <+5>:     mov    %rsp,%rbp
   0x0000555555555168 <+8>:     sub    $0x10,%rsp
   0x000055555555516c <+12>:    movl   $0x3,-0xc(%rbp)
   0x0000555555555173 <+19>:    movl   $0x2,-0x8(%rbp)
   0x000055555555517a <+26>:    mov    -0x8(%rbp),%edx
   0x000055555555517d <+29>:    mov    -0xc(%rbp),%eax
   0x0000555555555180 <+32>:    mov    %edx,%esi
   0x0000555555555182 <+34>:    mov    %eax,%edi
   0x0000555555555184 <+36>:    callq  0x555555555149 <mult>
   0x0000555555555189 <+41>:    mov    %eax,-0x4(%rbp)
   0x000055555555518c <+44>:    mov    -0x4(%rbp),%eax
   0x000055555555518f <+47>:    mov    %eax,%esi
   0x0000555555555191 <+49>:    lea    0xe6c(%rip),%rdi        # 0x555555556004
   0x0000555555555198 <+56>:    mov    $0x0,%eax
=> 0x000055555555519d <+61>:    callq  0x555555555050 <printf@plt>
   0x00005555555551a2 <+66>:    mov    $0x0,%eax
   0x00005555555551a7 <+71>:    leaveq
   0x00005555555551a8 <+72>:    retq
End of assembler dump.
```

```
(gdb) info register
rax            0x0                 0
rbx            0x5555555551b0      93824992235952
rcx            0x5555555551b0      93824992235952
rdx            0x2                 2
rsi            0x6                 6
rdi            0x555555556004      93824992239620
rbp            0x7fffffffdf60      0x7fffffffdf60
rsp            0x7fffffffdf50      0x7fffffffdf50
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0xb                 11
r11            0x2                 2
r12            0x555555555060      93824992235616
r13            0x7fffffffe050      140737488347216
r14            0x0                 0
r15            0x0                 0
rip            0x55555555519d      0x55555555519d <main+61>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) x $rsp
0x7fffffffdf50:    0xffffe050
(gdb) x $rbp
0x7fffffffdf60:    0x00000000
```