EDDI
Electronic Design
Development Institute

에디로봇아카데미
임베디드 마스터 Lv1 과정

제 5기

2023. 05. 17

박 상호

gdb

EDDI Electronic Design Development Institute

1. 컴파일

gcc –g –o(File Name) (File Name).c

-g : Debug Information 생성

2. gdb 실행

gdb (File Name) (PID) || (Core File Name)

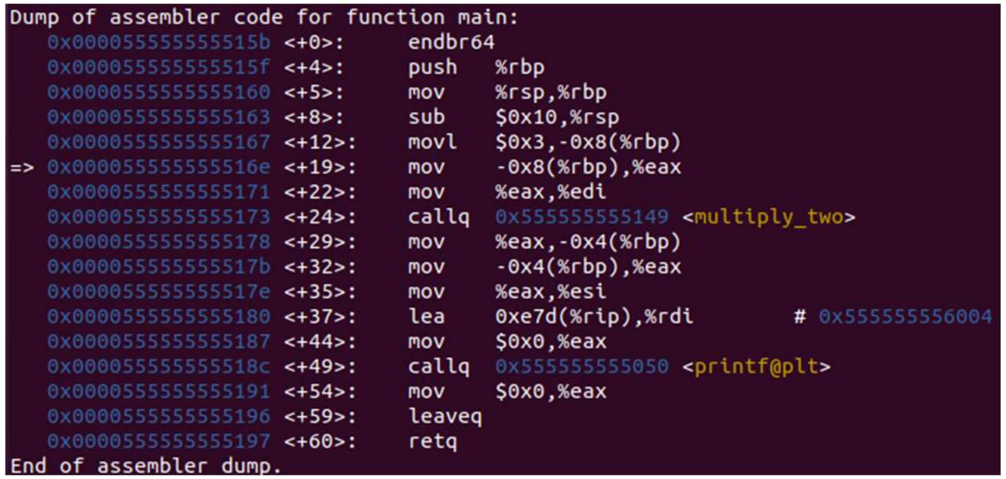
3. gdb 종료

q 입력 or Ctrl + d

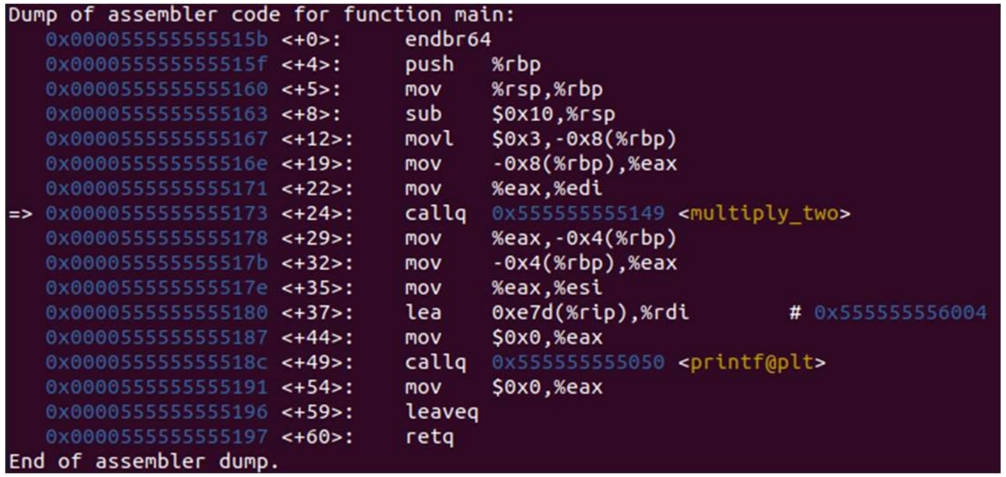
4. gdb Debug 명령어

Debug Command	Options	
r	(arguments)	(arguments를 사용하여)Program 실행
k		Program 종료
b	(Line Number)	(Line Number)에 Break Point 추가
	(Function)	(Function)에 Break Point 추가
	(Address)	(Address)에 Break Point 추가
	*.c:(Function)	*.c File의 (Function)에 Break Point 추가
	(+n) or (-n)	현재 행의 +/- n 행에 Break Point
watch	(Variable)	(Variable) 값 변동 시 Break
enable	(Break Point Number)	(Break Point Number) 활성화
disable	(Break Point Number)	(Break Point Number) 비활성화
cl	(== b)	Break Point 제거
s		Step (Step Into)
si		Step Instruction
n		Next (Step Over)
ni		Next Instruction
p (==p)	/(d, u, t, x, c, f) (Variable)	(출력 형식) 에 맞춰 (Variable) 출력
	(Variable)@(Array Size)	(Variable)을 Array로 (Array Size)만큼 출력
	$(Register)	(Register)의 현재 값을 출력
	(Function)	(Function)의 Address 출력
	(Function)::(Variable)	(Function)의 지역 변수 (Variable)의 값 출력
	'*.c::(Variable)	*.c File의 전역 변수 (Variable)의 값 출력
disas		Disassembly

Stack Frame (x64)



```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
=> 0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```



```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
=> 0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```

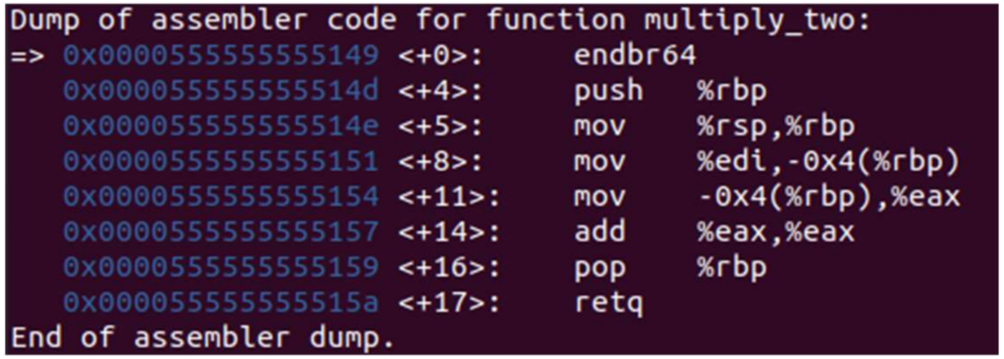
main 함수 호출

Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x03	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

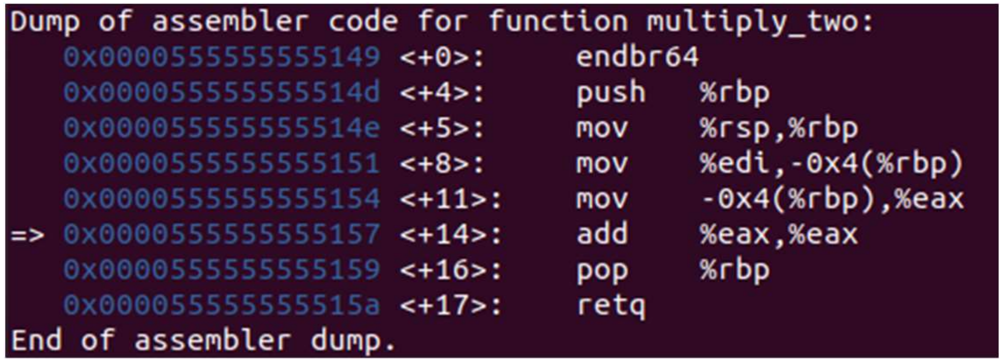
multiply_two 함수 호출 전 Register를 통한 num 인자 전달

eax	0x00000003 (mov -0x8(%rbp),%eax)
…	
rbp	0x7FFFFFFFFFFFDDE0
rsp	rbp – 0x10
edi	0x00000003 (mov %eax,%edi)
rip	0x0000555555555173

Stack Frame (x64)



```
Dump of assembler code for function multiply_two:
=> 0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    -0x4(%rbp),%eax
   0x0000555555555157 <+14>:    add    %eax,%eax
   0x0000555555555159 <+16>:    pop    %rbp
   0x000055555555515a <+17>:    retq
End of assembler dump.
```



```
Dump of assembler code for function multiply_two:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    -0x4(%rbp),%eax
=> 0x0000555555555157 <+14>:    add    %eax,%eax
   0x0000555555555159 <+16>:    pop    %rbp
   0x000055555555515a <+17>:    retq
End of assembler dump.
```

multiply_two 함수 호출

Ret Address	
...	
Ret Address	rbp + 0x08 == rip (main + 29)
rbp	== rbp ==rsp
0x03	rbp – 0x04 (mov -0x4(%rbp),%eax)
	rbp – 0x08

함수 호출 후 Register

eax	0x00000003
...	
rbp	0x7FFFFFFFFFFFDDC0
rsp	rbp
edi	0x00000003
rip	0x0000555555555157

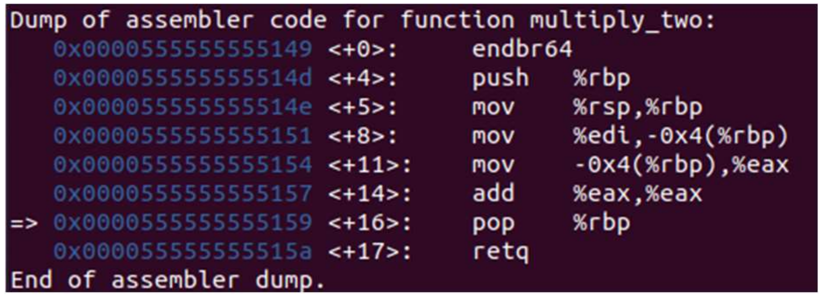
Stack Frame (x64)

multiply_two

Ret Address	
...	
Ret Address	rbp + 0x08 == rip (main + 29)
rbp	== rbp == rsp
0x03	rbp – 0x04 (mov -0x4(%rbp),%eax)
	rbp – 0x08

Add %eax,%eax 수행 후 Register

eax	0x00000003 + 0x00000003(add %eax,%eax)
...	
rbp	0x7FFFFFFFFFFFDDC0
rsp	rbp
edi	0x00000003
rip	0x0000555555555159



```
Dump of assembler code for function multiply_two:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    -0x4(%rbp),%eax
   0x0000555555555157 <+14>:    add    %eax,%eax
=> 0x0000555555555159 <+16>:    pop    %rbp
   0x000055555555515a <+17>:    retq
End of assembler dump.
```

Stack Frame (x64)

multiply_two (Before)

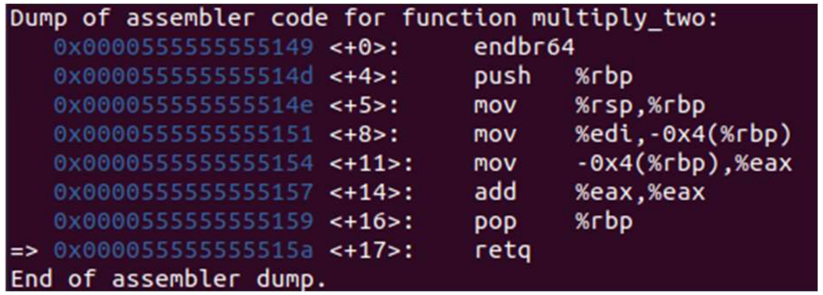
Ret Address	
…	
Ret Address	rbp + 0x08 == rip (main + 29)
rbp	== rbp == rsp
0x03	rbp – 0x04 (mov -0x4(%rbp),%eax)
	rbp – 0x08

multiply_two (After pop %rbp)

Ret Address	
rbp	== rbp
…	
Ret Address	retq == pop rip (main + 29) == rsp

pop %rbp 후 Register

eax	0x00000006
…	
rbp	0x7FFFFFFFFFFFDDE0
rsp	0x0000555555555178 (main + 29)
edi	0x00000003
rip	0x000055555555515A



```
Dump of assembler code for function multiply_two:
   0x0000555555555149 <+0>:     endbr64
   0x000055555555514d <+4>:     push   %rbp
   0x000055555555514e <+5>:     mov    %rsp,%rbp
   0x0000555555555151 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000555555555154 <+11>:    mov    -0x4(%rbp),%eax
   0x0000555555555157 <+14>:    add    %eax,%eax
   0x0000555555555159 <+16>:    pop    %rbp
=> 0x000055555555515a <+17>:    retq
End of assembler dump.
```

Stack Frame (x64)

multiply_two (Before)

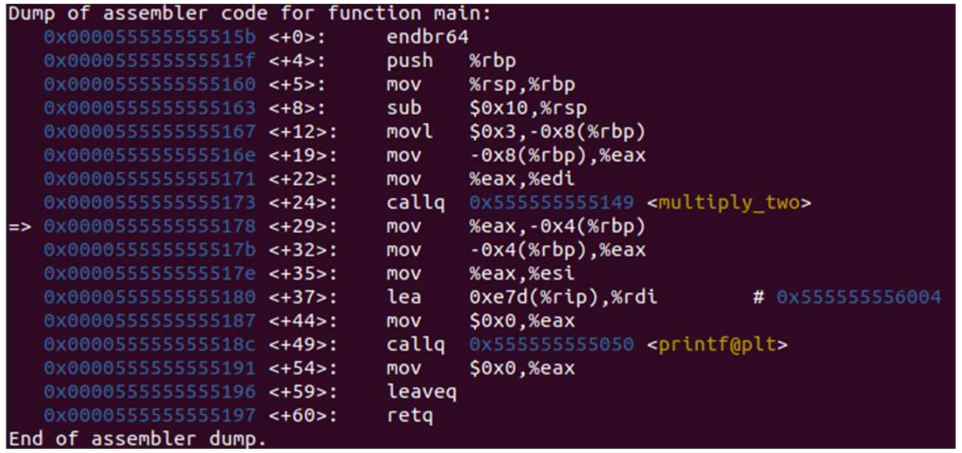
Ret Address	
rbp	== rbp
…	
Ret Address	retq == pop rip (main + 29) == rsp

main 함수 복귀 (After retq)

Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x03	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

retq 후 Register

eax	0x00000006
…	
rbp	0x7FFFFFFFFFFFDDE0
rsp	rbp – 0x10
edi	0x00000003
rip	0x0000555555555178



```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
=> 0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```

Stack Frame (x64)

main (Before)

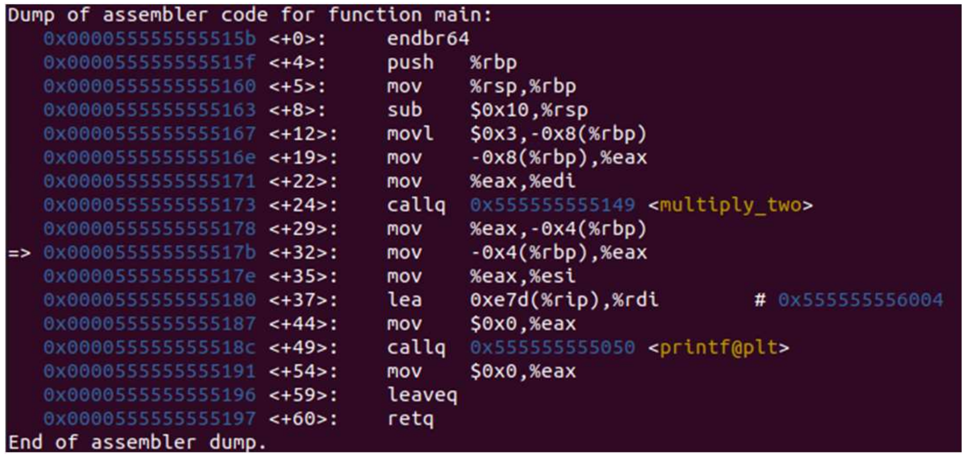
Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x03	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

Register

eax	0x00000006
…	
rbp	0x7FFFFFFFFFFFDDE0
rsp	rbp – 0x10
edi	0x00000003
rip	0x0000555555555178

Multiply_two 반환 값 전달(After mov %eax,-0x4(%rbp))

Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x06	rbp – 0x08 (mov %eax,-0x4(%rbp))
	rbp – 0x0C
	rbp – 0x10 == rsp

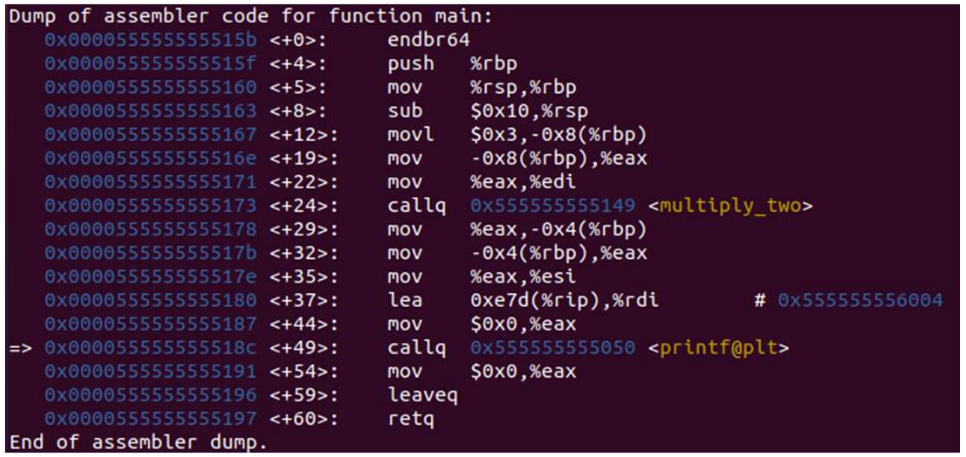


```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
=> 0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```

Stack Frame (x64)

main (Before)

Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x06	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

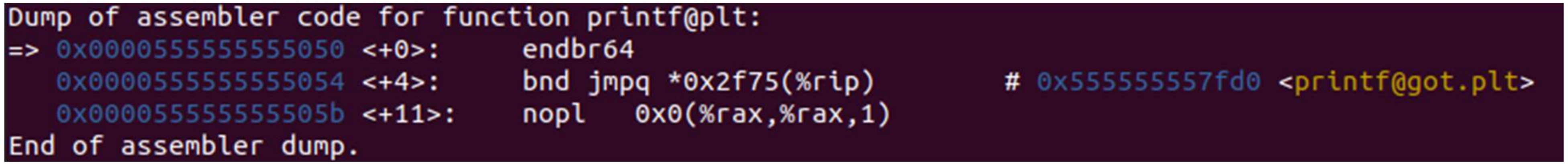


```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
=> 0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```

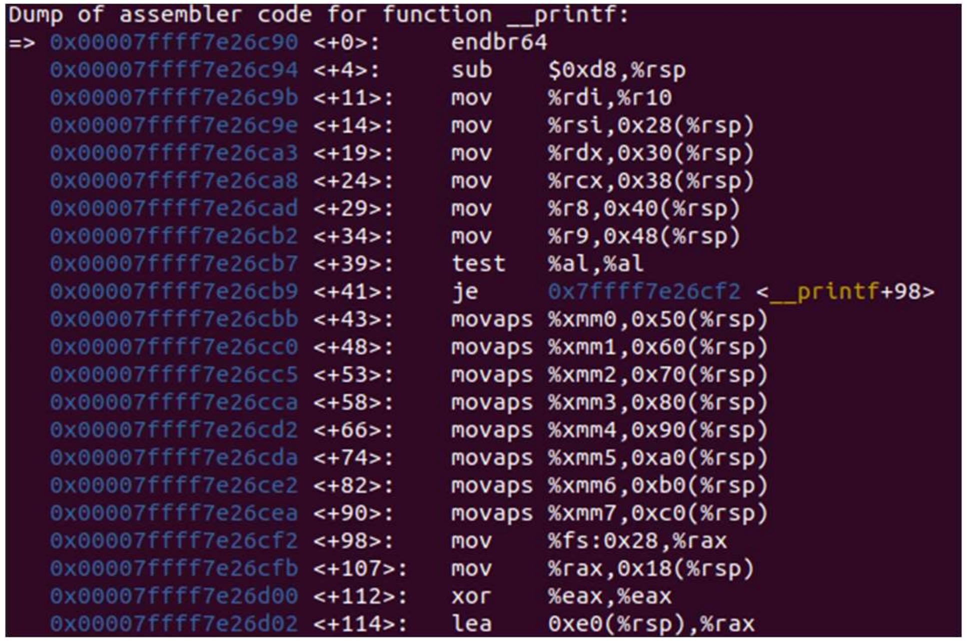
Register의 multiply_two 반환 값 초기화, printf 호출 전 Register를 통한 format과 argument 인자 전달

eax	0x00000000 (mov $0x0,%eax)
…	
rbp	0x7FFFFFFFFFFFDDE0
rsp	rbp – 0x10
esi	0x00000006 (mov -0x4(%rbp),%eax) (mov %eax, %esi)
rdi	0x0000555555556004 (lea 0xe7d(%rip), %rdi)
rip	0x000055555555518C

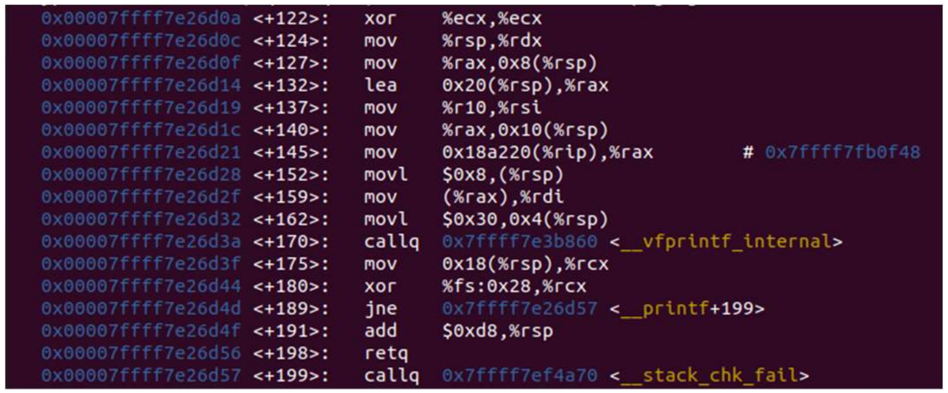
Stack Frame (x64)



```
Dump of assembler code for function printf@plt:
=> 0x0000555555555050 <+0>:     endbr64
   0x0000555555555054 <+4>:     bnd jmpq *0x2f75(%rip)        # 0x555555557fd0 <printf@got.plt>
   0x000055555555505b <+11>:    nopl   0x0(%rax,%rax,1)
End of assembler dump.
```



```
Dump of assembler code for function __printf:
=> 0x00007ffff7e26c90 <+0>:     endbr64
   0x00007ffff7e26c94 <+4>:     sub    $0xd8,%rsp
   0x00007ffff7e26c9b <+11>:    mov    %rdi,%r10
   0x00007ffff7e26c9e <+14>:    mov    %rsi,0x28(%rsp)
   0x00007ffff7e26ca3 <+19>:    mov    %rdx,0x30(%rsp)
   0x00007ffff7e26ca8 <+24>:    mov    %rcx,0x38(%rsp)
   0x00007ffff7e26cad <+29>:    mov    %r8,0x40(%rsp)
   0x00007ffff7e26cb2 <+34>:    mov    %r9,0x48(%rsp)
   0x00007ffff7e26cb7 <+39>:    test   %al,%al
   0x00007ffff7e26cb9 <+41>:    je     0x7ffff7e26cf2 <__printf+98>
   0x00007ffff7e26cbb <+43>:    movaps %xmm0,0x50(%rsp)
   0x00007ffff7e26cc0 <+48>:    movaps %xmm1,0x60(%rsp)
   0x00007ffff7e26cc5 <+53>:    movaps %xmm2,0x70(%rsp)
   0x00007ffff7e26cca <+58>:    movaps %xmm3,0x80(%rsp)
   0x00007ffff7e26cd2 <+66>:    movaps %xmm4,0x90(%rsp)
   0x00007ffff7e26cda <+74>:    movaps %xmm5,0xa0(%rsp)
   0x00007ffff7e26ce2 <+82>:    movaps %xmm6,0xb0(%rsp)
   0x00007ffff7e26cea <+90>:    movaps %xmm7,0xc0(%rsp)
   0x00007ffff7e26cf2 <+98>:    mov    %fs:0x28,%rax
   0x00007ffff7e26cfb <+107>:   mov    %rax,0x18(%rsp)
   0x00007ffff7e26d00 <+112>:   xor    %eax,%eax
   0x00007ffff7e26d02 <+114>:   lea    0xe0(%rsp),%rax
```



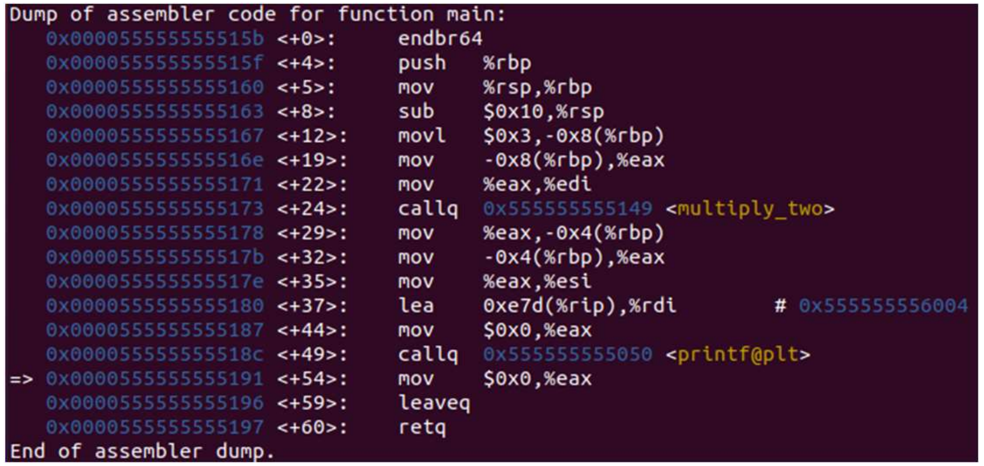
```
   0x00007ffff7e26d0a <+122>:   xor    %ecx,%ecx
   0x00007ffff7e26d0c <+124>:   mov    %rsp,%rdx
   0x00007ffff7e26d0f <+127>:   mov    %rax,0x8(%rsp)
   0x00007ffff7e26d14 <+132>:   lea    0x20(%rsp),%rax
   0x00007ffff7e26d19 <+137>:   mov    %r10,%rsi
   0x00007ffff7e26d1c <+140>:   mov    %rax,0x10(%rsp)
   0x00007ffff7e26d21 <+145>:   mov    0x18a220(%rip),%rax        # 0x7ffff7fb0f48
   0x00007ffff7e26d28 <+152>:   movl   $0x8,(%rsp)
   0x00007ffff7e26d2f <+159>:   mov    (%rax),%rdi
   0x00007ffff7e26d32 <+162>:   movl   $0x30,0x4(%rsp)
   0x00007ffff7e26d3a <+170>:   callq  0x7ffff7e3b860 <__vfprintf_internal>
   0x00007ffff7e26d3f <+175>:   mov    0x18(%rsp),%rcx
   0x00007ffff7e26d44 <+180>:   xor    %fs:0x28,%rcx
   0x00007ffff7e26d4d <+189>:   jne    0x7ffff7e26d57 <__printf+199>
   0x00007ffff7e26d4f <+191>:   add    $0xd8,%rsp
   0x00007ffff7e26d56 <+198>:   retq
   0x00007ffff7e26d57 <+199>:   callq  0x7ffff7ef4a70 <__stack_chk_fail>
```

Dynamic Linking된 Printf 를 PLT, GOT를 참조하여 실행

Stack Frame (x64)

printf 호출 후 Stack Frame

Ret Address	rbp + 0x08
…	
Ret Address	rbp + 0x08 == rip (main + 54)
rbp	
…	
r9	rsp + 0x48
r8	rsp + 0x40
rcx	rsp + 0x38
rdx	rsp + 0x30
rsi	rsp + 0x28
…	
0x34000000340	rbp – 0xD8 == rsp



```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
=> 0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
End of assembler dump.
```

printf 종료 후 Stack Frame, 반환 값은 eax에 저장

Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x03	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

Stack Frame (x64)

leaveq 명령어로 지역 변수 공간 반환 (mov %rbp, %rsp)

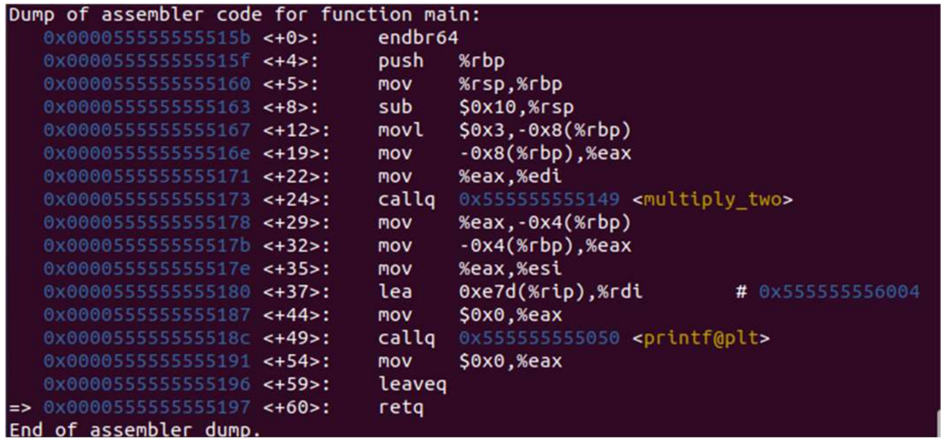
Ret Address	rbp + 0x08
rbp	== rbp
	rbp – 0x00
0x03	rbp – 0x08
	rbp – 0x0C
	rbp – 0x10 == rsp

leaveq 명령어로 function 실행 전 rbp 복구 (pop %rbp)

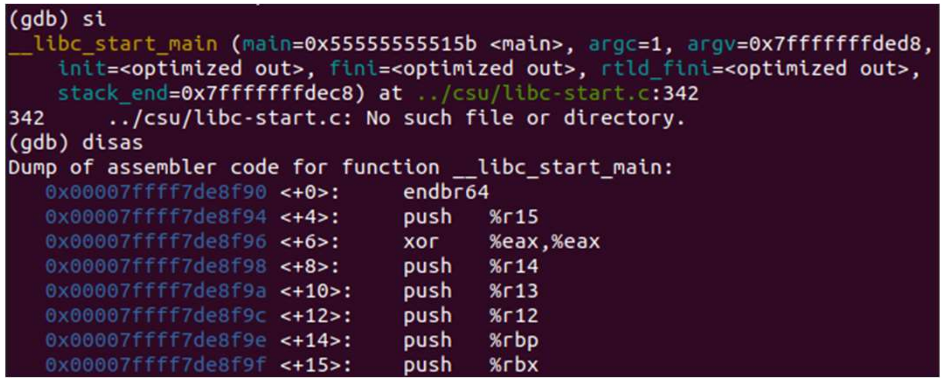
Ret Address	rbp + 0x08
rbp	== rbp == rsp (mov %rbp, %rsp)

retq 명령어로 function 종료 (retq)

Ret Address	== rsp



```
Dump of assembler code for function main:
   0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <multiply_two>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
=> 0x0000555555555197 <+60>:    retq
End of assembler dump.
```



```
(gdb) si
__libc_start_main (main=0x55555555515b <main>, argc=1, argv=0x7fffffffded8,
    init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>,
    stack_end=0x7fffffffdec8) at ../csu/libc-start.c:342
342     ../csu/libc-start.c: No such file or directory.
(gdb) disas
Dump of assembler code for function __libc_start_main:
   0x00007ffff7de8f90 <+0>:     endbr64
   0x00007ffff7de8f94 <+4>:     push   %r15
   0x00007ffff7de8f96 <+6>:     xor    %eax,%eax
   0x00007ffff7de8f98 <+8>:     push   %r14
   0x00007ffff7de8f9a <+10>:    push   %r13
   0x00007ffff7de8f9c <+12>:    push   %r12
   0x00007ffff7de8f9e <+14>:    push   %rbp
   0x00007ffff7de8f9f <+15>:    push   %rbx
```