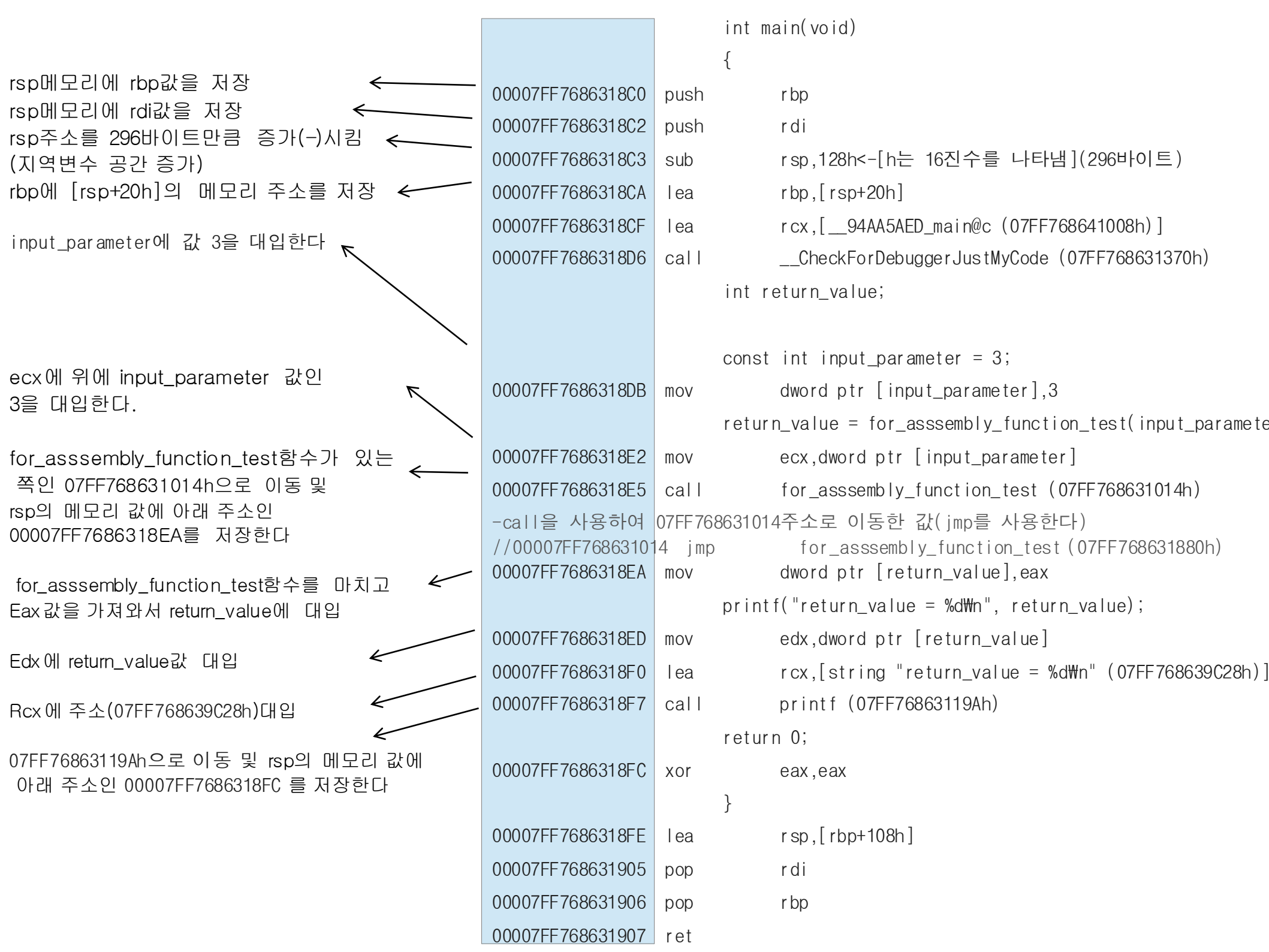
★필수적인 Register의 목적들 (sp, bp, ip, ax)
⇒ ax: 함수의 출력(리턴) 값이 배치됨
⇒ bp: Stack의 기준점 역할을 해줌
⇒ sp: 현재 Stack의 최상위 위치
⇒ ip: 다음에 실행할 명령의 메모리 주소를 가리킴

- Stack이 무엇인가요 ?
  =>지역 변수가 저장되는 메모리 공간


rsp메모리에 rbp값을 저장
rsp메모리에 rdi값을 저장
rsp주소를 296바이트만큼 증가(-)시킴
(지역변수 공간 증가)
rbp에 [rsp+20h]의 메모리 주소를 저장
input_parameter에 값 3을 대입한다
ecx에 위에 input_parameter 값인
3을 대입한다.
for_asssembly_function_test함수가 있는
쪽인 07FF768631014h으로 이동 및
rsp의 메모리 값에 아래 주소인
00007FF7686318EA를 저장한다
for_asssembly_function_test함수를 마치고
Eax값을 가져와서 return_value에 대입
Edx에 return_value값 대입
Rcx에 주소(07FF768639C28h)대입
07FF76863119Ah으로 이동 및 rsp의 메모리 값에
아래 주소인 00007FF7686318FC 를 저장한다
int main(void)
{
00007FF7686318C0 push rbp
00007FF7686318C2 push rdi
00007FF7686318C3 sub rsp,128h<-[h는 16진수를 나타냄](296바이트)
00007FF7686318CA lea rbp,[rsp+20h]
00007FF7686318CF lea rcx,[__94AA5AED_main@c (07FF768641008h)]
00007FF7686318D6 call __CheckForDebuggerJustMyCode (07FF768631370h)
int return_value;
const int input_parameter = 3;
00007FF7686318DB mov dword ptr [input_parameter],3
return_value = for_asssembly_function_test(input_paramete
00007FF7686318E2 mov ecx,dword ptr [input_parameter]
00007FF7686318E5 call for_asssembly_function_test (07FF768631014h)
-call을 사용하여 07FF768631014주소로 이동한 값(jmp를 사용한다)
//00007FF768631014 jmp for_asssembly_function_test (07FF768631880h)
00007FF7686318EA mov dword ptr [return_value],eax
printf("return_value = %d₩n", return_value);
00007FF7686318ED mov edx,dword ptr [return_value]
00007FF7686318F0 lea rcx,[string "return_value = %d₩n" (07FF768639C28h)]
00007FF7686318F7 call printf (07FF76863119Ah)
return 0;
00007FF7686318FC xor eax,eax
}
00007FF7686318FE lea rsp,[rbp+108h]
00007FF768631905 pop rdi
00007FF768631906 pop rbp
00007FF768631907 ret

00007FF7686318C0 push rbp하기 전의 레지스터 및 메모리 값

RSP = 000000036D6FF960 RBP = 000000036D6FF980

sp의 메모리 값 = 08 10 64 68 f7 7f 00 00 b4 05 67 4f fa 7f 00 00 d5 1e d3 86 f8 39 00 00 02 00 00

sp의 메모리 값을 표시 한 곳을보면 오른쪽에서 왼쪽순으로 rbp값이 저장된 것을 확인 할 수 있다.

00007FF7686318C0 push rbp의 레지스터 및 메모리 값

RSP = 000000036D6FF958 RBP = 000000036D6FF980

sp의 메모리 값 = 80 f9 6f 6d 03 00 00 00 08 10 64 68 f7 7f 00 00 b4 05 67 4f fa 7f 00 00 d5 1e d3

00007FF7686318C2 push rdi의 레지스터 및 메모리 값

RSP = 000000036D6FF950 RDI = 0000000000000000

0x000000036D6FF950 - 0x128 =0x000000036D6FF828

sp의 메모리 값 = 00 00 00 00 00 00 00 00 80 f9 6f 6d 03 00 00 00 08 10 64 68 f7 7f 00 00 b4 05 67

00007FF7686318C3 sub rsp,128h(296바이트)의 레지스터 및 메모리 값

RSP = 000000036D6FF828 RBP = 000000036D6FF980

0x000000036D6FF828 + 0x20 =0x000000036D6FF848

00007FF7686318CA lea rbp,[rsp+20h]의 레지스터 및 메모리 값

RSP = 000000036D6FF828 RBP = 000000036D6FF848

디버깅을 다시했더니 주소 값이 초기화됌

00007FF7686318DB mov dword ptr [input_parameter],3의 레지스터 및 메모리 값

input_parameter의 초기 값 = 0x0000005c6b2ff644 {567} RSP = 0000005C6B2FF600 RBP = 0000005C6B2FF620

input_parameter의 mov 이후 값 = 0x0000005c6b2ff644 {3}

RCX에 input_parameter값인 3h를 넣음

00007FF7686318E2 mov ecx,dword ptr [input_parameter]의 레지스터 및 메모리 값

RCX = 0000000000000003 RSP = 0000005C6B2FF600 RBP = 0000005C6B2FF620

00007FF7686318E5 call for_asssembly_function_test (07FF768631014h)의 레지스터 및 메모리 값

RSP = 0000005C6B2FF5F8 RBP = 0000005C6B2FF620 RSP = 000000F61B6FF848 RBP = 000000F61B6FF870

(집에 와서 해서 sp 및 주소 값이 다름) Sp 메모리 값에 아래 주소를 저장시킴

sp의 메모리 값 = ea 18 63 68 f7 7f 00 00 08 10 64 68 f7 7f 00 00 b4 05 f7 9a fa 7f 00 00 dc 22 6b

sp의 메모리 값 = ea 18 7a 9f f7 7f 00 00 08 10 7b 9f f7 7f 00 00 b4 05 0e 09 f9 7f 00 00 ab 04 87

00007FF79F7A18EA mov dword ptr [return_value],eax의 레지스터 및 메모리 값

0x000000F61B6FF874(return_value) = 00000006 RAX = 0000000000000006

Return_value값인 6을 Rdx에 대입

Rcx에 07FF79F7A9C28h 대입

??

00007FF79F7A18ED mov edx,dword ptr [return_value]의 레지스터 및 메모리 값

RDX = 0000000000000006 RCX = 00007FF79F7B1008

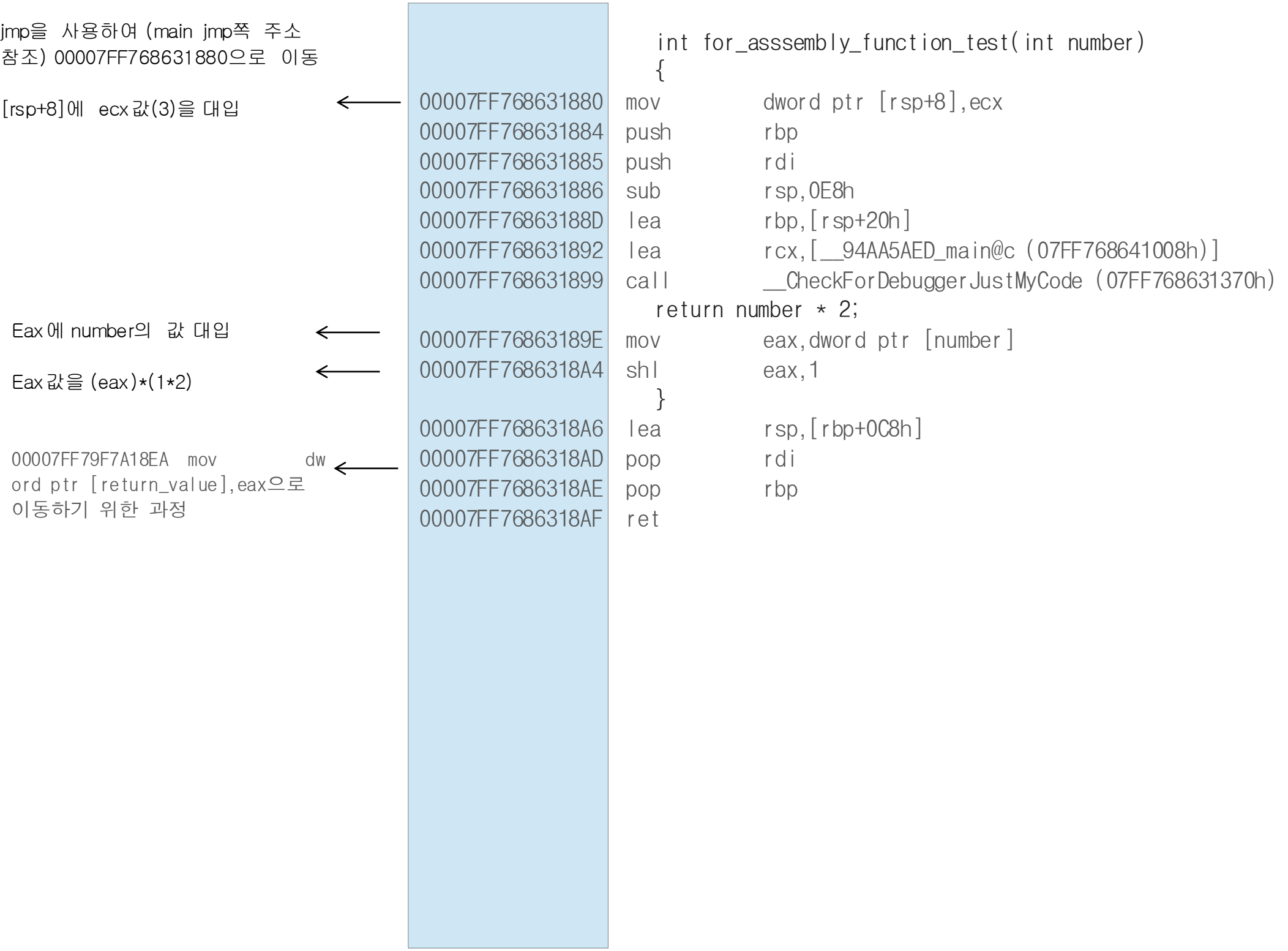
00007FF79F7A18F0 lea rcx,[string "return_value = %dWn" (07FF79F7A9C28h)]의 레지스터 및 메모리 값

RCX = 00007FF79F7A9C28

00007FF7686318F7 call printf (07FF76863119Ah) 의 레지스터 및 메모리 값

00007FF79F7A119A jmp printf (07FF79F7A1920h) 으로 이동 후

다음 sp 값 = fc 18 7a 9f f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 0e 09 f9 7f 00 00 ab 04 87


jmp을 사용하여 (main jmp쪽 주소 참조) 00007FF768631880으로 이동
[rsp+8]에 ecx값(3)을 대입
Eax에 number의 값 대입
Eax값을 (eax)*(1*2)
00007FF79F7A18EA mov dword ptr [return_value],eax으로 이동하기 위한 과정
int for_asssembly_function_test(int number)
{
00007FF768631880 mov dword ptr [rsp+8],ecx
00007FF768631884 push rbp
00007FF768631885 push rdi
00007FF768631886 sub rsp,0E8h
00007FF76863188D lea rbp,[rsp+20h]
00007FF768631892 lea rcx,[__94AA5AED_main@c (07FF768641008h)]
00007FF768631899 call __CheckForDebuggerJustMyCode (07FF768631370h)
return number * 2;
00007FF76863189E mov eax,dword ptr [number]
00007FF7686318A4 shl eax,1
}
00007FF7686318A6 lea rsp,[rbp+0C8h]
00007FF7686318AD pop rdi
00007FF7686318AE pop rbp
00007FF7686318AF ret

00007FF768631880 mov dword ptr [rsp+8],ecx의 레지스터 및 메모리 값

RSP = 0000005C6B2FF5F8 RBP = 0000005C6B2FF620 RSP = 000000F61B6FF848 RBP = 000000F61B6FF870

sp의 메모리 값 = ea 18 63 68 f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 f7 9a fa 7f 00 00 dc 22 6b

[Rsp+8]= 000000F61B6FF750에ecx{3}을 대입 Rsp(000000F61B6FF750){3}

00007FF768631884 push rbp의 레지스터 및 메모리 값

RSP = 000000F61B6FF840 RBP = 000000F61B6FF870

메모리 값 = 70 f8 6f 1b f6 00 00 00 ea 18 7a 9f f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 0e

Sp메모리값에 rbp값 저장

00007FF7686318C2 push rdi의 레지스터 및 메모리 값

RSP = 000000F61B6FF838 RDI = 0000000000000000

sp의 메모리 값 = 00 00 00 00 00 00 00 00 70 f8 6f 1b f6 00 00 00 ea 18 7a 9f f7 7f 00 00 03 00 00

Sp메모리값에 rdi값 저장

00007FF79F7A1886 sub rsp,0E8h의 레지스터 및 메모리 값

RSP = 000000F61B6FF750 RBP = 000000F61B6FF870

위 sp값 000000F61B6FF838-0E8h= 000000F61B6FF750

00007FF79F7A188D lea rbp,[rsp+20h]의 레지스터 및 메모리 값

RSP = 000000F61B6FF750 RBP = 000000F61B6FF770

Rbp에 [rsp+20h] 주소 값 저장

00007FF79F7A189E mov eax,dword ptr [number]의 레지스터 및 메모리 값

RAX = 0000000000000003

?[number]?값을 eax에 대입

00007FF79F7A18A4 shl eax,1의 레지스터 및 메모리 값

RAX = 0000000000000006

Eax*2의 제곱근으로 $3*2^1$으로 값6이 나온다

00007FF79F7A18A6 lea rsp,[rbp+0C8h]의 레지스터 및 메모리 값

RSP = 000000F61B6FF838

[rbp+0C8h]값은 rsp에 저장

00007FF79F7A18AD pop rdi의 레지스터 및 메모리 값

RSP = 000000F61B6FF840 RBP = 000000F61B6FF770 RDI = 0000000000000000

sp의 메모리 값 = 70 f8 6f 1b f6 00 00 00 ea 18 7a 9f f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 0e

Sp메모리값을 rdi에 저장

00007FF79F7A18AD pop rdi의 레지스터 및 메모리 값

Sp메모리값을 rdi에 저장

RSP = 000000F61B6FF840 RBP = 000000F61B6FF770 RDI = 0000000000000000

sp의 메모리 값 = 70 f8 6f 1b f6 00 00 00 ea 18 7a 9f f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 0e

00007FF79F7A18AE pop rbp의 레지스터 및 메모리 값

Sp메모리값을 rbp에 저장

RSP = 000000F61B6FF848 RBP = 000000F61B6FF870

메모리 값 = ea 18 7a 9f f7 7f 00 00 03 00 00 00 f7 7f 00 00 b4 05 0e 09 f9 7f 00 00 ab 04 87

00007FF79F7A18AF ret의 레지스터 및 메모리 값

Sp메모리값을 rip에 저장 즉 0007FF79F7A18EA로 이동

00007FF79F7A18EA mov dword ptr [return_value],eax으로 이동

RSP = 000000F61B6FF850 RBP = 000000F61B6FF870