Embedded Class

Homework#2

-목차

김 시 윤

1. 2진수 16진수 변환정리.

2진수,16진수란?
0과 1 이라는 두가지로 모든숫자를 표현하는 방식이다.
우리가 일상생활에서 사용하는 숫자는 10진수로 0~9 까지의 숫자로 표현한다.
10진수는 한 비트에서 표현할 수 있는 최대값인 9가 넘어가게 되면 2비트 10으로 표현한다.
마찬가지로 2진수도 한비트 표현할수 있는 최대숫자인 1일 넘어가게 되면 10 으로 표현하며, 10진수로 2라 읽는다.

십진수에서 129는 백의자리 1 , 십의자리 2, 일의자리 9 라해서 백이십구 라고읽는다.
이걸 10으로 표현하면 $100\times1 + 10\times2 + 1\times9 \rightarrow 1\times10^2 + 2\times10^1 + 9\times10^0$ 으로 표현할 수 있다.
마찬가지로 2진수 1010 은 $1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$ 으로 표현한다.

16진수도 위와 같다. 16진수는 표현할수 있는 숫자의 개수가 총 16개로 0~F로 표현한다.
9 이상의 숫자를 한비트에 표현해야하기 때문에 영어를 빌려와서 표현한다.
예를들어 9를 초과했을 때 10진수에서 10이라면 비트수가 2개이기 때문에 A를 빌려와서 표현한다.
10~15까지의 숫자를 A~F 까지의 영어 단어로 대체한다.

16진수 0x2F를 10진수로 바꿔보면 $2\times16^1 + 15\times16^0 = 32 + 15 = 47$ 이 된다.

2진수 -> 16진수 , 16진수 -> 2진수

2진수에서 16진수는 간단하다.
16 = 2^4 으로 2진수 4비트가 16진수 1비트와 동일하다.
2진수 10011010을 16진수로 바꾸자할 때 4비트로 나눠서 보면 쉽게 알 수 있다.
1001 1010 이 숫자에서 앞의 4비트인 1001을 10진수로 바꿔보면
$1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$ 이되며 9가된다.
뒤의 4비트인 1010도 마저 10진수로 박궈보면
1001+1이기 때문에 1010은 10이 된다.

앞에 4비트를 16진수로 표현하면 10진수와 마찬가지로 9라는 숫자로 표현 가능하고
뒤의 4비트를 16진수로 표현하면 A로 대체가 된다.
각 각 두비트를 16진수로 표현하면 0x9A 가 된다.

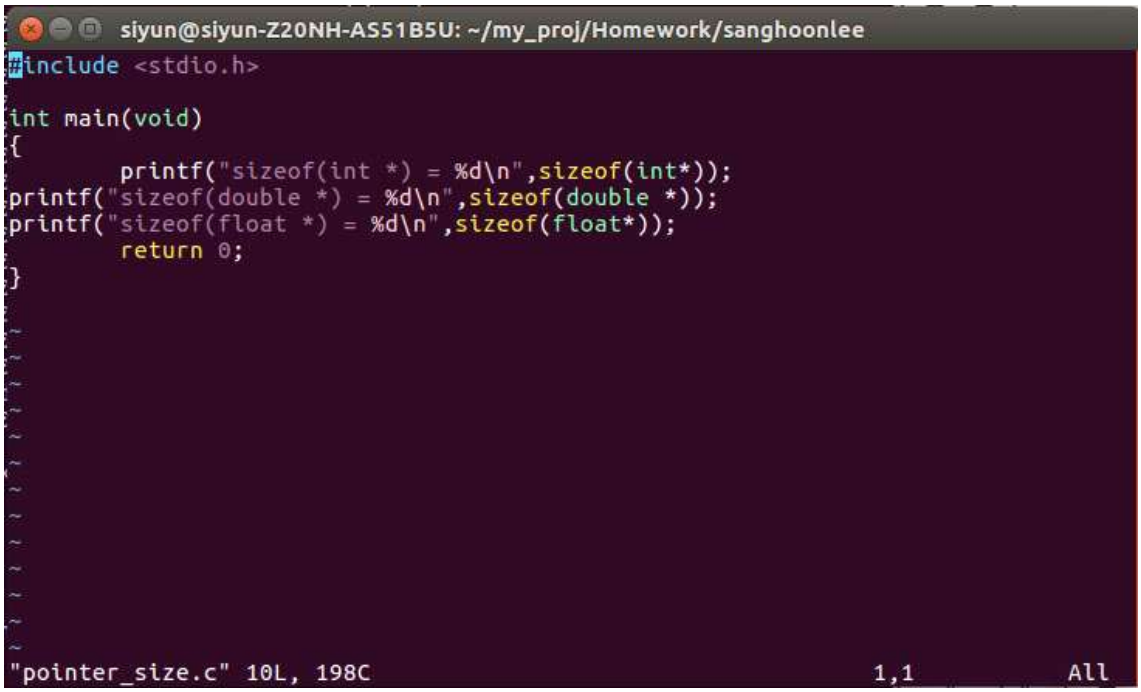
2. 포인터 크기 내용정리.

포인터를 알기전에 우선 비트를 알아야한다.
데이터가 저장할 수 있는 최소 단위를 bit(비트) 라고한다.
이 bit 가 모여서 byte 가 되거 메가바이트 기가바이트가 된다.

여기서 byte를 알아야하는데 1byte는 8개의 비트들의 모임이다.
따라서 1byte = 8bit 가 된다.

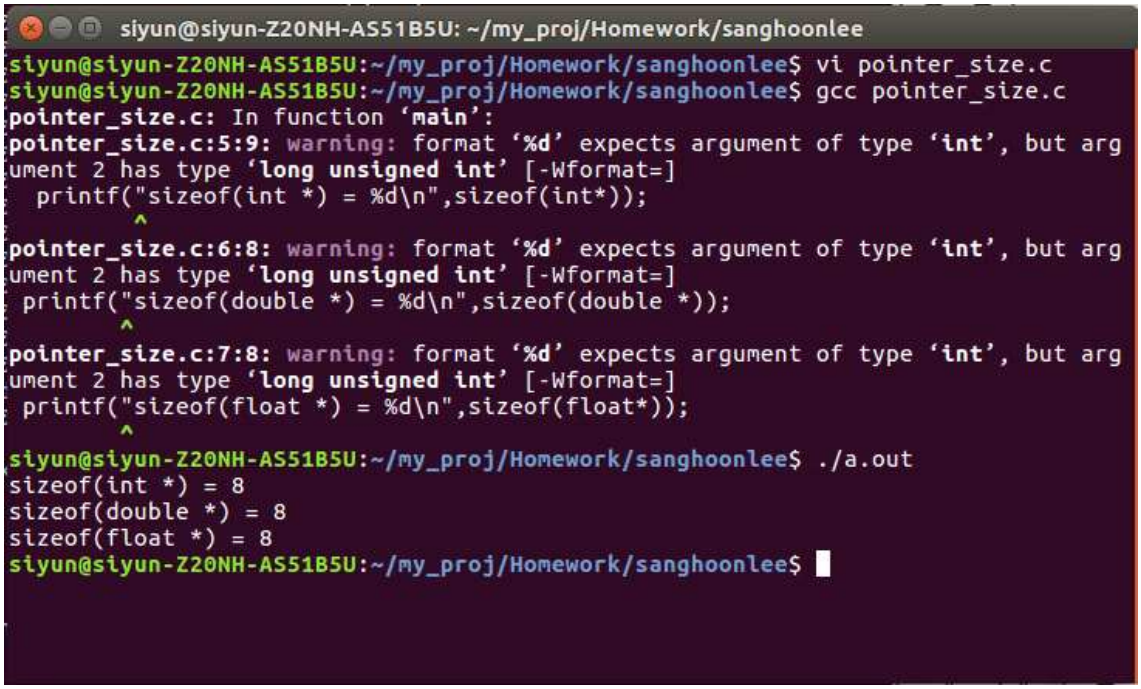
여기서 포인터는 레지스터의 비트수로 포인터 크기가 결정 되는데,
16bit 일 경우에 포인터 크기는 2byte 가 되고
32bit 일 경우에 포인터 크기는 4byte
64bit 일 경우에 포인터 크기는 8byte 가 된다.

레지스터의 비트수가 64bit 인 컴퓨터에서 위의 사실을 증명해 보았다.
우선 linux 터미널을 털어 pointer_size.c 파일을 생성하였다.



[그림.1] pointer size source code

레지스터의 비트수에 따라 결정되는 포인터 크기를 알아보기위해
각 변수의 포인터의 크기를 재보는 [그림.1]의 소스코드를 작성하였다.
이 소스코드를 write(저장) 하고
컴파일 후 출력해보면 아래 [그림.2]와 같다.



[그림.2] pointer size source code out

확인결과 int * , double * ,float * 모두 8byte 크기를 갖는 것을 확인 할 수 있었다.

여기서 warning을 없애기 위해서는 %d를 %lu로 바꿔주면 된다.

3. Assembly language (기계어 정리)

(실습)

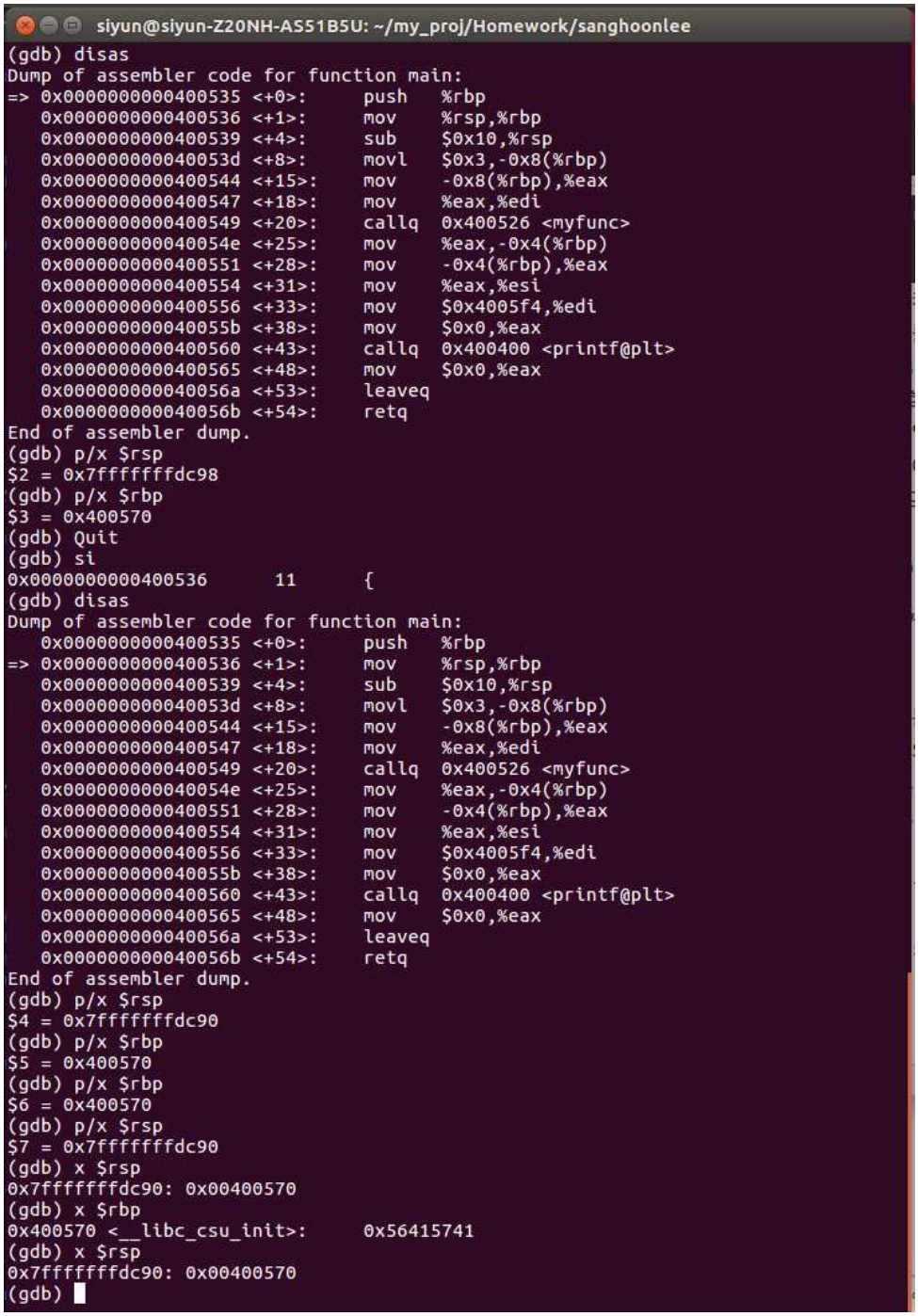
```
 #include <stdio.h>


 int myfunc(int num)
 {
        // return num * 2;
        return num + 3;
        // return num << 1;
 }


int main(void)
{
        int num = 3, res;
        res = myfunc(num);
        printf("res = %d\n", res);


        return 0;
 }
```

위의 함수 c코드를 기계어로 바꿔보는 과정을 실습하였다.



처음에 stack frame을 만들어준다.

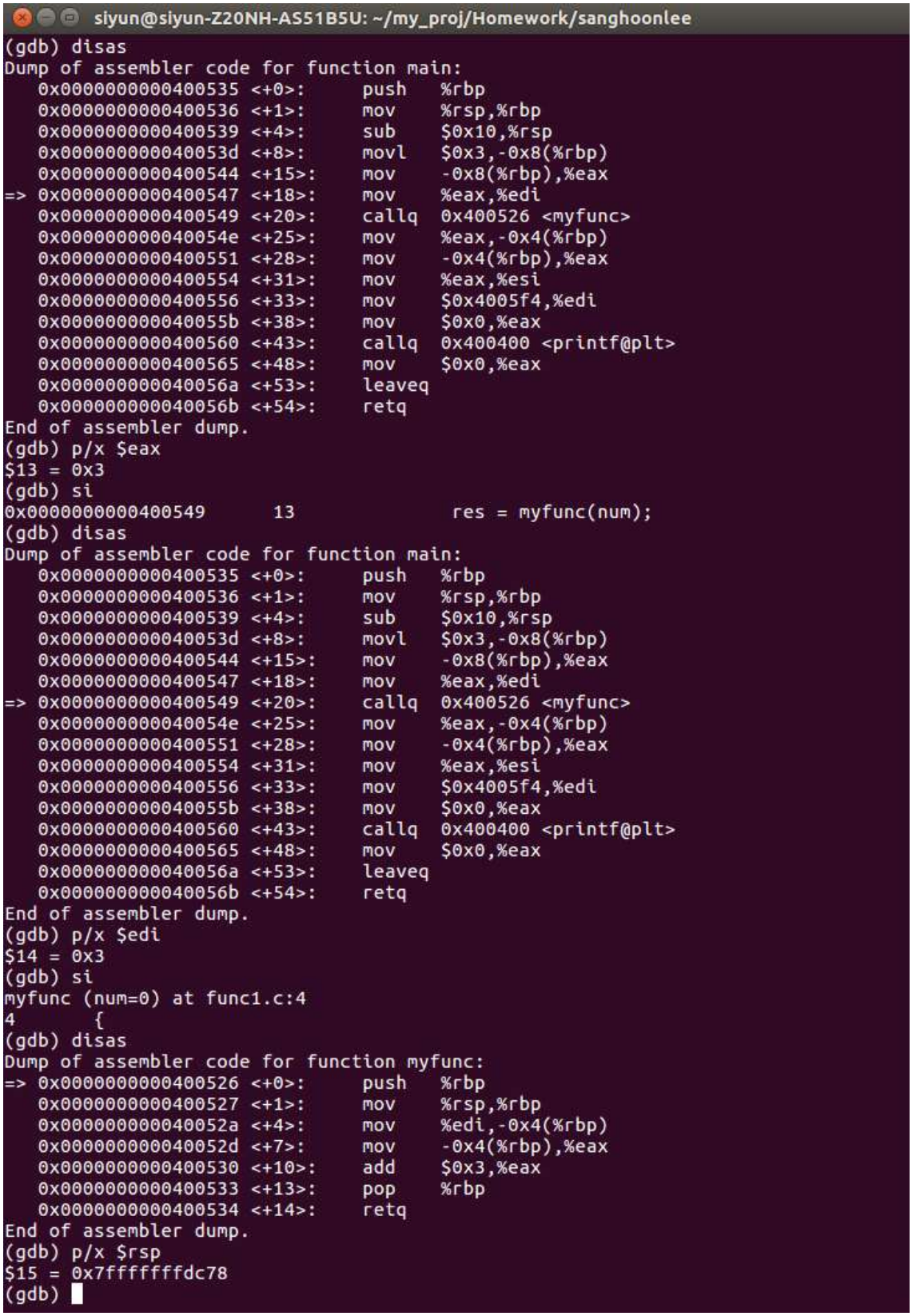
다은 16byte 공간에서 8byte에 3을 저장하고 rax 레지스터 반인 eax에다 3을 이동시킨다.

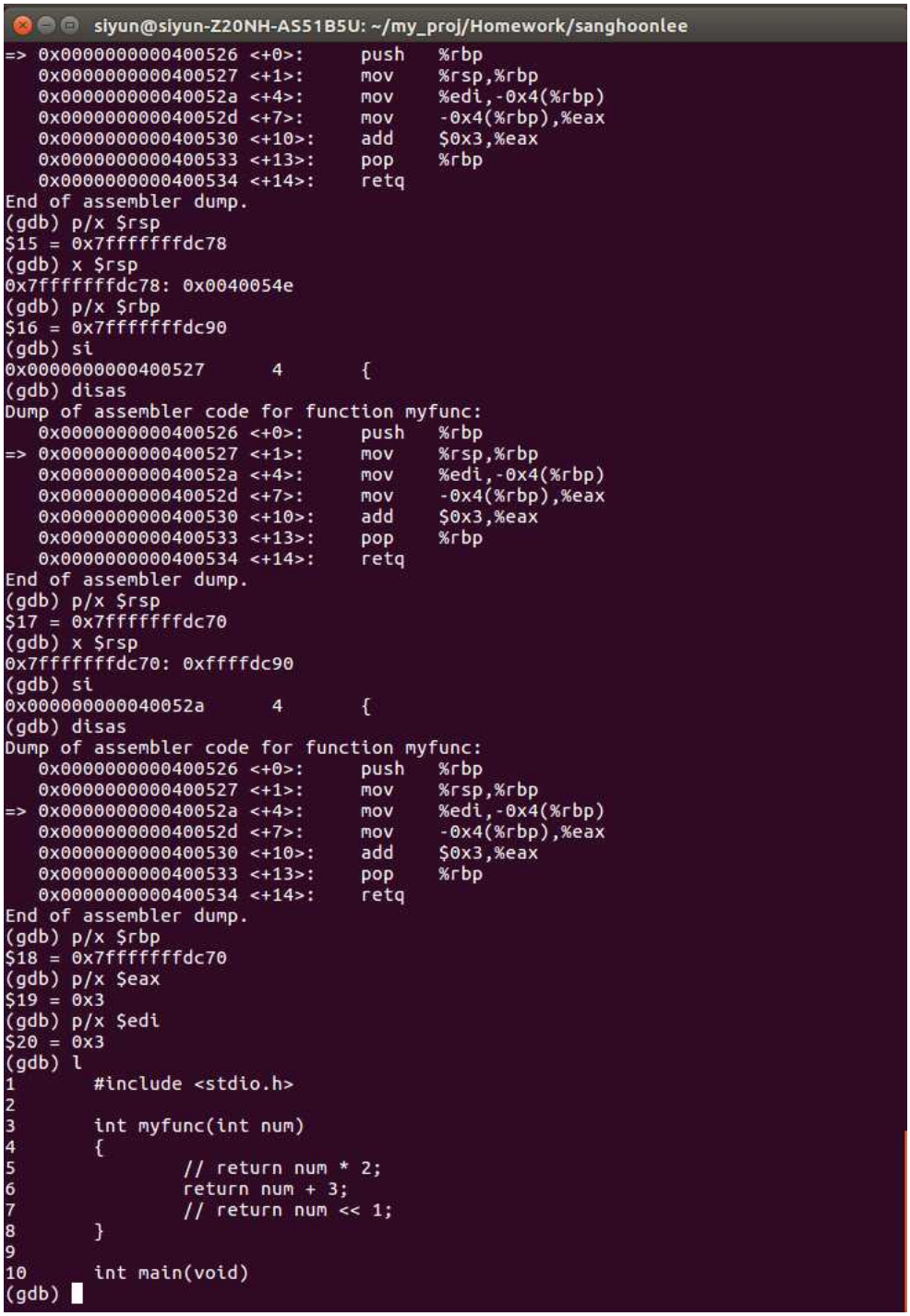
eax 값인 3을 edi 에 이동시킨다.

함수의 stack으로 점프한다.

```
siyun@siyun-Z20NH-AS51B5U: ~/my_proj/Homework/sanghoonlee
0x7fffffffdc90: 0x00400570
(gdb) si

Breakpoint 1, main () at func1.c:12
12              int num = 3, res;
(gdb) disas
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   0x0000000000400547 <+18>:    mov    %eax,%edi
   0x0000000000400549 <+20>:    callq  0x400526 <myfunc>
   0x000000000040054e <+25>:    mov    %eax,-0x4(%rbp)
   0x0000000000400551 <+28>:    mov    -0x4(%rbp),%eax
   0x0000000000400554 <+31>:    mov    %eax,%esi
   0x0000000000400556 <+33>:    mov    $0x4005f4,%edi
   0x000000000040055b <+38>:    mov    $0x0,%eax
   0x0000000000400560 <+43>:    callq  0x400400 <printf@plt>
   0x0000000000400565 <+48>:    mov    $0x0,%eax
   0x000000000040056a <+53>:    leaveq
   0x000000000040056b <+54>:    retq
End of assembler dump.
(gdb) p/x $rsp
$9 = 0x7fffffffdc80
(gdb) p/x $rbp
$10 = 0x7fffffffdc90
(gdb) si
13              res = myfunc(num);
(gdb) disas
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
   0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
=> 0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   0x0000000000400547 <+18>:    mov    %eax,%edi
   0x0000000000400549 <+20>:    callq  0x400526 <myfunc>
   0x000000000040054e <+25>:    mov    %eax,-0x4(%rbp)
   0x0000000000400551 <+28>:    mov    -0x4(%rbp),%eax
   0x0000000000400554 <+31>:    mov    %eax,%esi
   0x0000000000400556 <+33>:    mov    $0x4005f4,%edi
   0x000000000040055b <+38>:    mov    $0x0,%eax
   0x0000000000400560 <+43>:    callq  0x400400 <printf@plt>
   0x0000000000400565 <+48>:    mov    $0x0,%eax
   0x000000000040056a <+53>:    leaveq
   0x000000000040056b <+54>:    retq
End of assembler dump.
(gdb) x $rbp - 8
0x7fffffffdc88: 0x00000003
(gdb) p/x &num1
No symbol "num1" in current context.
(gdb) p/x &num1
No symbol "num1" in current context.
(gdb) p/x &num1
No symbol "num1" in current context.
(gdb) p/x &num
$11 = 0x7fffffffdc88
(gdb)
```


siyun@siyun-Z20NH-AS51B5U: ~/my_proj/Homework/sanghoonlee
(gdb) disas
Dump of assembler code for function main:
0x0000000000400535 <+0>: push %rbp
0x0000000000400536 <+1>: mov %rsp,%rbp
0x0000000000400539 <+4>: sub $0x10,%rsp
0x000000000040053d <+8>: movl $0x3,-0x8(%rbp)
0x0000000000400544 <+15>: mov -0x8(%rbp),%eax
=> 0x0000000000400547 <+18>: mov %eax,%edi
0x0000000000400549 <+20>: callq 0x400526 <myfunc>
0x000000000040054e <+25>: mov %eax,-0x4(%rbp)
0x0000000000400551 <+28>: mov -0x4(%rbp),%eax
0x0000000000400554 <+31>: mov %eax,%esi
0x0000000000400556 <+33>: mov $0x4005f4,%edi
0x000000000040055b <+38>: mov $0x0,%eax
0x0000000000400560 <+43>: callq 0x400400 <printf@plt>
0x0000000000400565 <+48>: mov $0x0,%eax
0x000000000040056a <+53>: leaveq
0x000000000040056b <+54>: retq
End of assembler dump.
(gdb) p/x $eax
$13 = 0x3
(gdb) si
0x0000000000400549 13 res = myfunc(num);
(gdb) disas
Dump of assembler code for function main:
0x0000000000400535 <+0>: push %rbp
0x0000000000400536 <+1>: mov %rsp,%rbp
0x0000000000400539 <+4>: sub $0x10,%rsp
0x000000000040053d <+8>: movl $0x3,-0x8(%rbp)
0x0000000000400544 <+15>: mov -0x8(%rbp),%eax
0x0000000000400547 <+18>: mov %eax,%edi
=> 0x0000000000400549 <+20>: callq 0x400526 <myfunc>
0x000000000040054e <+25>: mov %eax,-0x4(%rbp)
0x0000000000400551 <+28>: mov -0x4(%rbp),%eax
0x0000000000400554 <+31>: mov %eax,%esi
0x0000000000400556 <+33>: mov $0x4005f4,%edi
0x000000000040055b <+38>: mov $0x0,%eax
0x0000000000400560 <+43>: callq 0x400400 <printf@plt>
0x0000000000400565 <+48>: mov $0x0,%eax
0x000000000040056a <+53>: leaveq
0x000000000040056b <+54>: retq
End of assembler dump.
(gdb) p/x $edi
$14 = 0x3
(gdb) si
myfunc (num=0) at func1.c:4
4 {
(gdb) disas
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>: push %rbp
0x0000000000400527 <+1>: mov %rsp,%rbp
0x000000000040052a <+4>: mov %edi,-0x4(%rbp)
0x000000000040052d <+7>: mov -0x4(%rbp),%eax
0x0000000000400530 <+10>: add $0x3,%eax
0x0000000000400533 <+13>: pop %rbp
0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) p/x $rsp
$15 = 0x7fffffffdc78
(gdb)



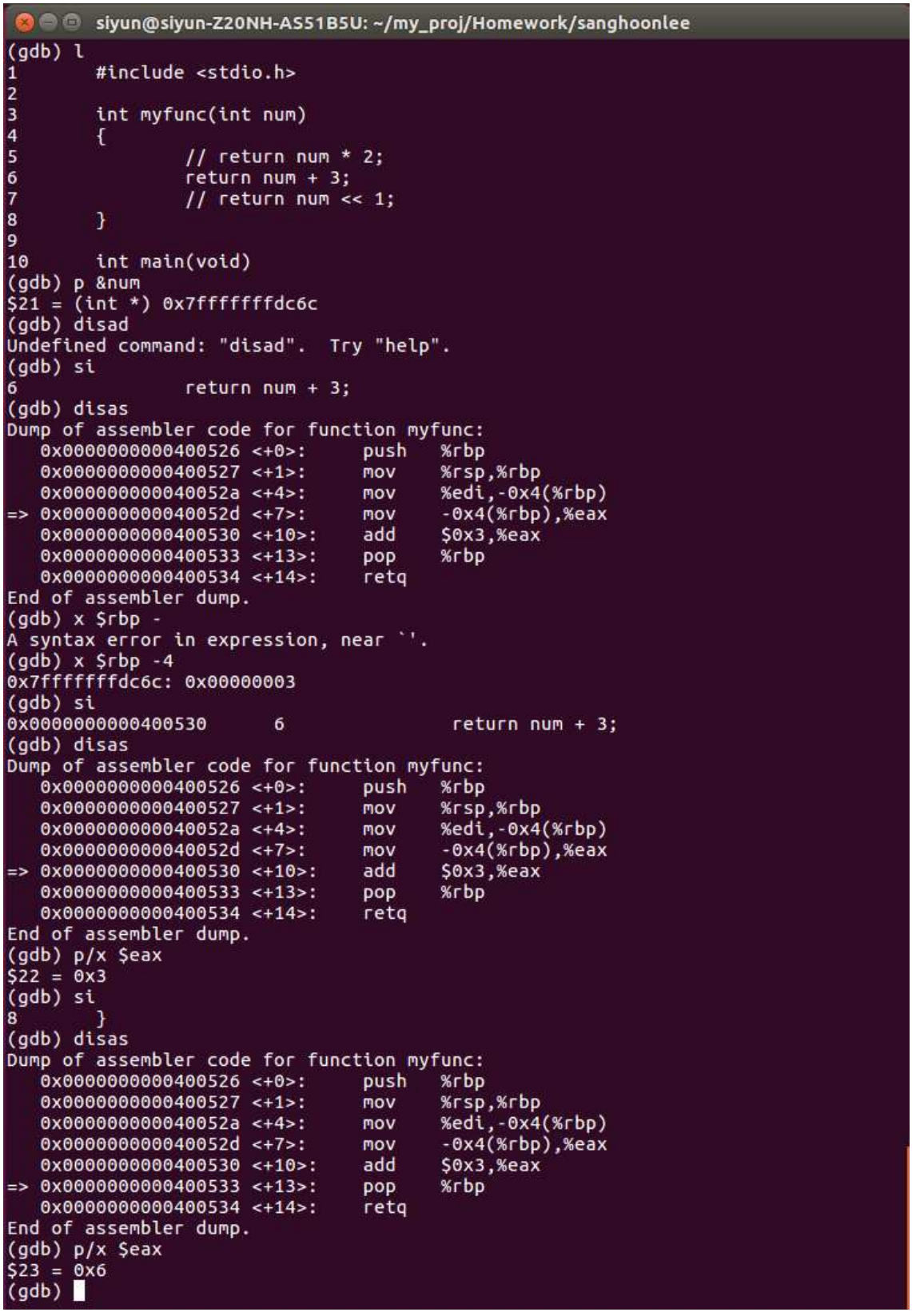
함수도 똑같이 stack frame을 만든다.

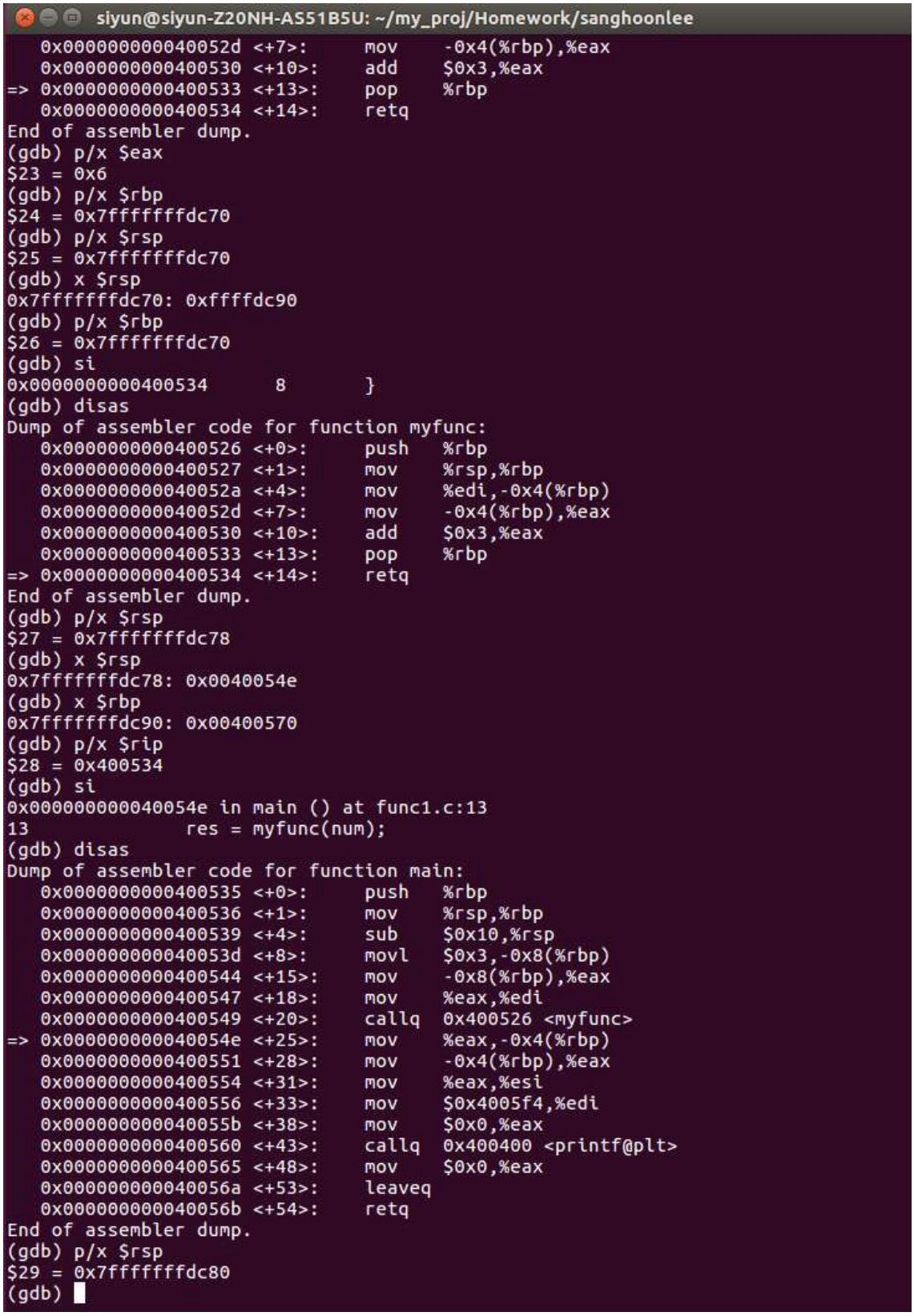
8byte에서 4byte 에 edi 저장한 3을 옮긴다.

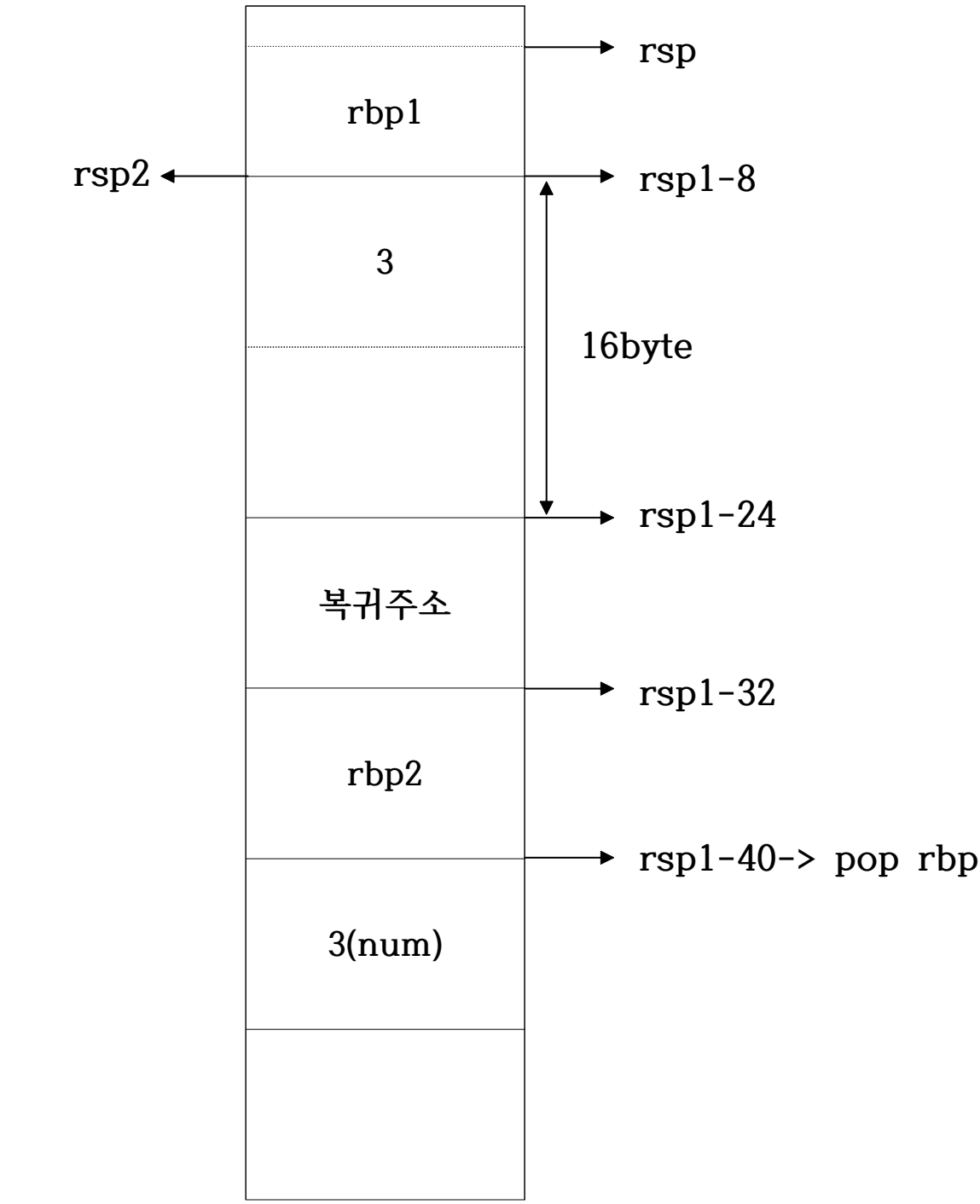
나머지 3은 eax 레지스터에 옮긴다.

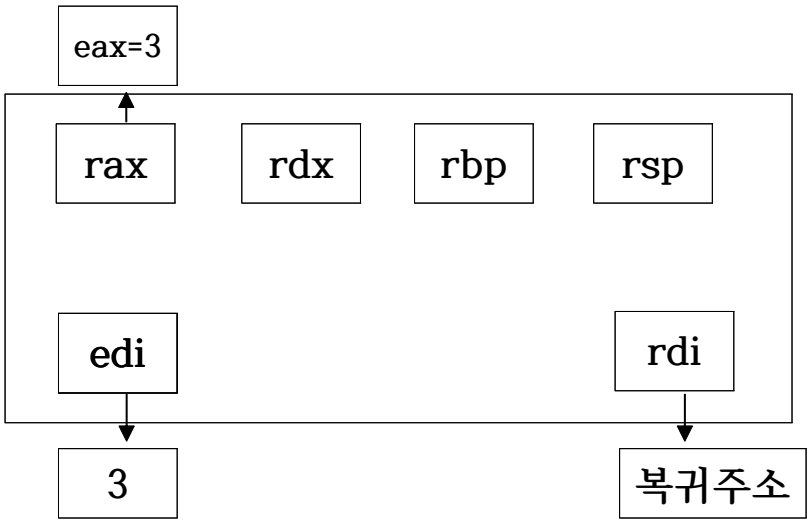
3+ eax 를한다 (eax 는 num)

6인상태로 함수를 빠져나온다.


siyun@siyun-Z20NH-AS51B5U: ~/my_proj/Homework/sanghoonlee
(gdb) l
1 #include <stdio.h>
2
3 int myfunc(int num)
4 {
5 // return num * 2;
6 return num + 3;
7 // return num << 1;
8 }
9
10 int main(void)
(gdb) p &num
$21 = (int *) 0x7fffffffdc6c
(gdb) disad
Undefined command: "disad". Try "help".
(gdb) si
6 return num + 3;
(gdb) disas
Dump of assembler code for function myfunc:
0x0000000000400526 <+0>: push %rbp
0x0000000000400527 <+1>: mov %rsp,%rbp
0x000000000040052a <+4>: mov %edi,-0x4(%rbp)
=> 0x000000000040052d <+7>: mov -0x4(%rbp),%eax
0x0000000000400530 <+10>: add $0x3,%eax
0x0000000000400533 <+13>: pop %rbp
0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) x $rbp -
A syntax error in expression, near `'.
(gdb) x $rbp -4
0x7fffffffdc6c: 0x00000003
(gdb) si
0x0000000000400530 6 return num + 3;
(gdb) disas
Dump of assembler code for function myfunc:
0x0000000000400526 <+0>: push %rbp
0x0000000000400527 <+1>: mov %rsp,%rbp
0x000000000040052a <+4>: mov %edi,-0x4(%rbp)
0x000000000040052d <+7>: mov -0x4(%rbp),%eax
=> 0x0000000000400530 <+10>: add $0x3,%eax
0x0000000000400533 <+13>: pop %rbp
0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) p/x $eax
$22 = 0x3
(gdb) si
8 }
(gdb) disas
Dump of assembler code for function myfunc:
0x0000000000400526 <+0>: push %rbp
0x0000000000400527 <+1>: mov %rsp,%rbp
0x000000000040052a <+4>: mov %edi,-0x4(%rbp)
0x000000000040052d <+7>: mov -0x4(%rbp),%eax
0x0000000000400530 <+10>: add $0x3,%eax
=> 0x0000000000400533 <+13>: pop %rbp
0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) p/x $eax
$23 = 0x6
(gdb)


siyun@siyun-Z20NH-AS51B5U: ~/my_proj/Homework/sanghoonlee
0x000000000040052d <+7>: mov -0x4(%rbp),%eax
0x0000000000400530 <+10>: add $0x3,%eax
=> 0x0000000000400533 <+13>: pop %rbp
0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) p/x $eax
$23 = 0x6
(gdb) p/x $rbp
$24 = 0x7fffffffdc70
(gdb) p/x $rsp
$25 = 0x7fffffffdc70
(gdb) x $rsp
0x7fffffffdc70: 0xffffdc90
(gdb) p/x $rbp
$26 = 0x7fffffffdc70
(gdb) si
0x0000000000400534 8 }
(gdb) disas
Dump of assembler code for function myfunc:
0x0000000000400526 <+0>: push %rbp
0x0000000000400527 <+1>: mov %rsp,%rbp
0x000000000040052a <+4>: mov %edi,-0x4(%rbp)
0x000000000040052d <+7>: mov -0x4(%rbp),%eax
0x0000000000400530 <+10>: add $0x3,%eax
0x0000000000400533 <+13>: pop %rbp
=> 0x0000000000400534 <+14>: retq
End of assembler dump.
(gdb) p/x $rsp
$27 = 0x7fffffffdc78
(gdb) x $rsp
0x7fffffffdc78: 0x0040054e
(gdb) x $rbp
0x7fffffffdc90: 0x00400570
(gdb) p/x $rip
$28 = 0x400534
(gdb) si
0x000000000040054e in main () at func1.c:13
13 res = myfunc(num);
(gdb) disas
Dump of assembler code for function main:
0x0000000000400535 <+0>: push %rbp
0x0000000000400536 <+1>: mov %rsp,%rbp
0x0000000000400539 <+4>: sub $0x10,%rsp
0x000000000040053d <+8>: movl $0x3,-0x8(%rbp)
0x0000000000400544 <+15>: mov -0x8(%rbp),%eax
0x0000000000400547 <+18>: mov %eax,%edi
0x0000000000400549 <+20>: callq 0x400526 <myfunc>
=> 0x000000000040054e <+25>: mov %eax,-0x4(%rbp)
0x0000000000400551 <+28>: mov -0x4(%rbp),%eax
0x0000000000400554 <+31>: mov %eax,%esi
0x0000000000400556 <+33>: mov $0x4005f4,%edi
0x000000000040055b <+38>: mov $0x0,%eax
0x0000000000400560 <+43>: callq 0x400400 <printf@plt>
0x0000000000400565 <+48>: mov $0x0,%eax
0x000000000040056a <+53>: leaveq
0x000000000040056b <+54>: retq
End of assembler dump.
(gdb) p/x $rsp
$29 = 0x7fffffffdc80
(gdb)


rsp
rbp1
rsp2
rsp1-8
3
16byte
rsp1-24
복귀주소
rsp1-32
rbp2
rsp1-40-> pop rbp
3(num)


eax=3
rax
rdx
rbp
rsp
edi
rdi
3
복귀주소