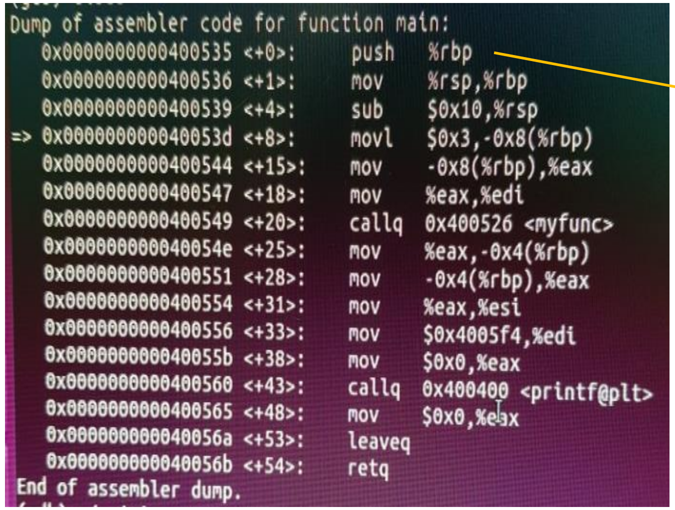
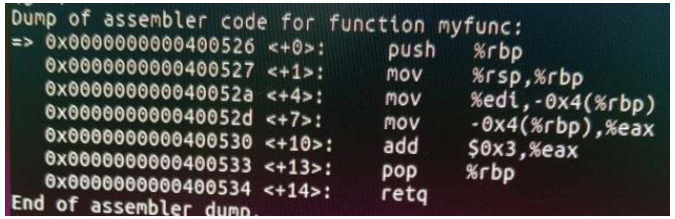
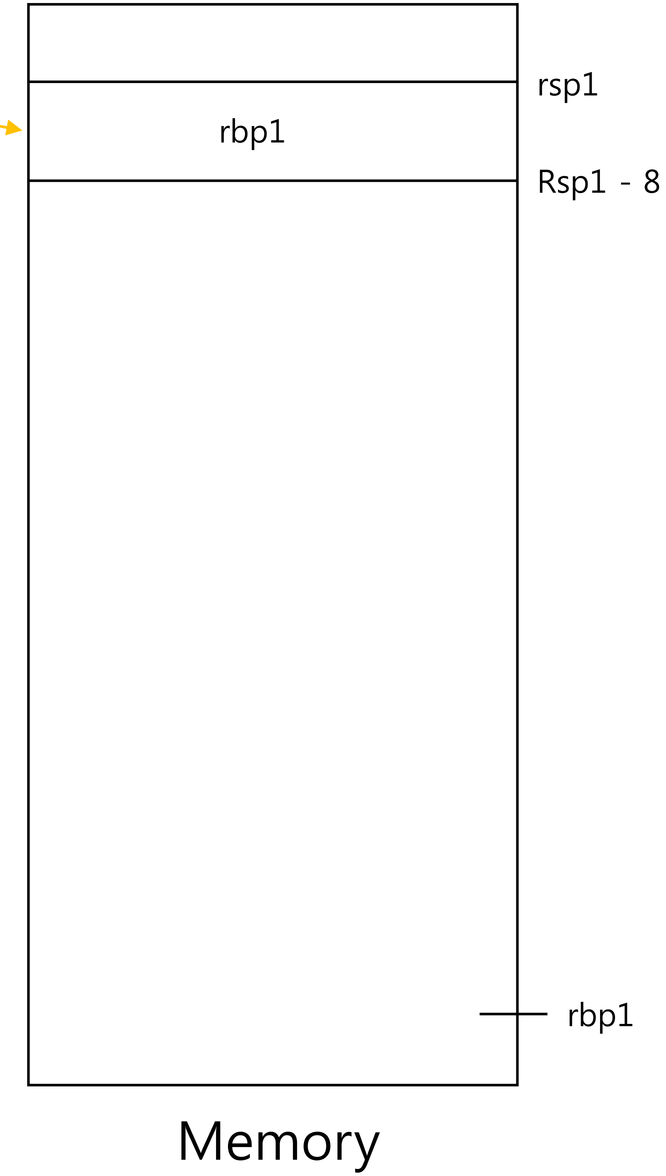
1. 기계어 분석 – (1)



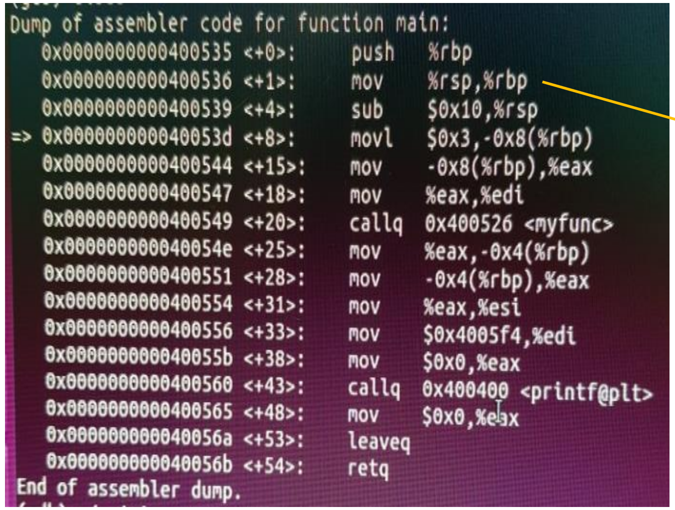
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



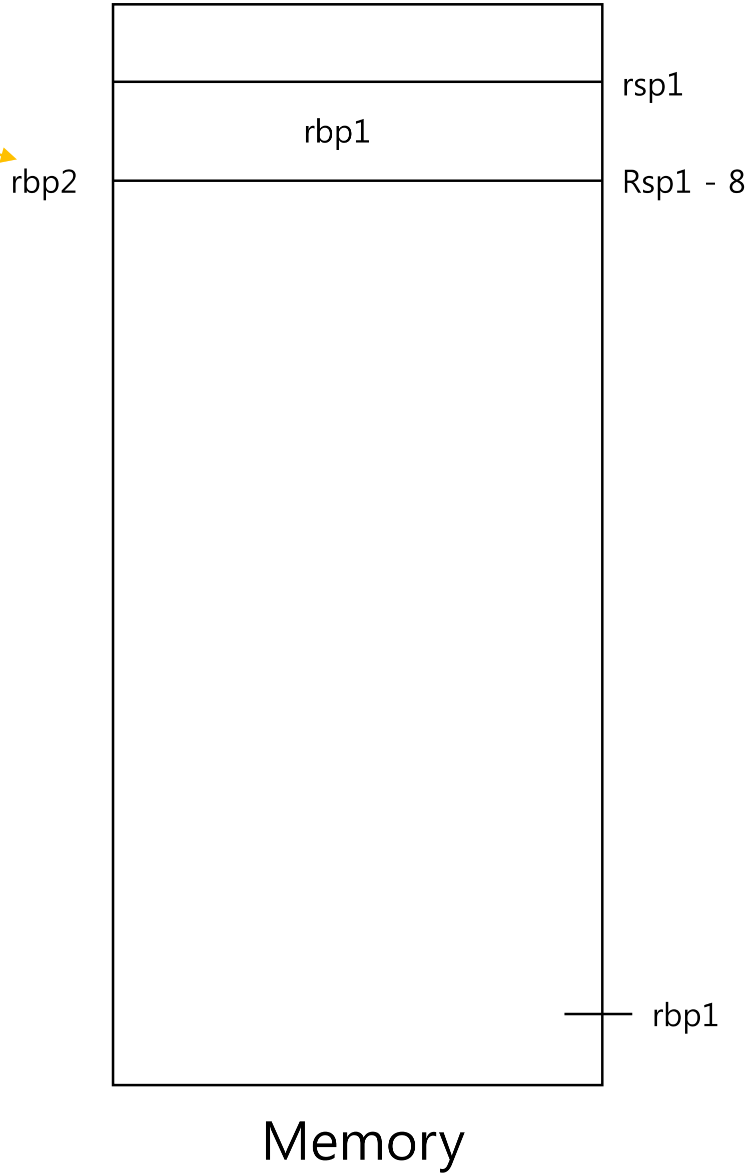
```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```



1. 기계어 분석 – (2)



```
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.
```



rsp1

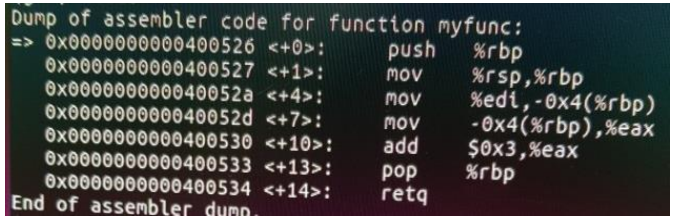
rbp1

rbp2

Rsp1 - 8

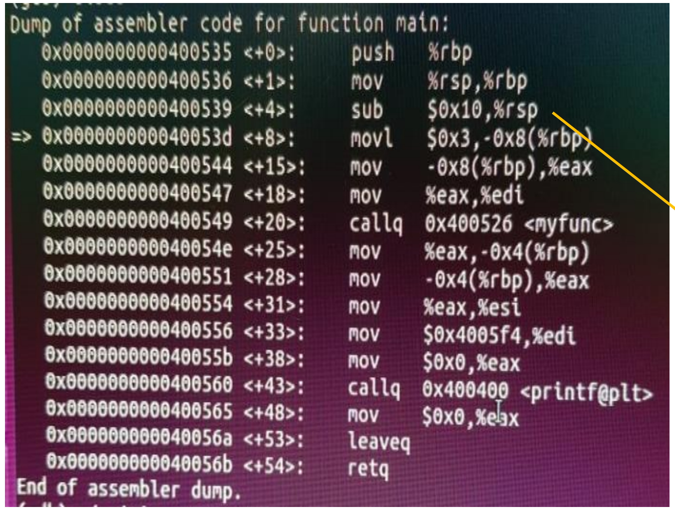
rbp1

Memory

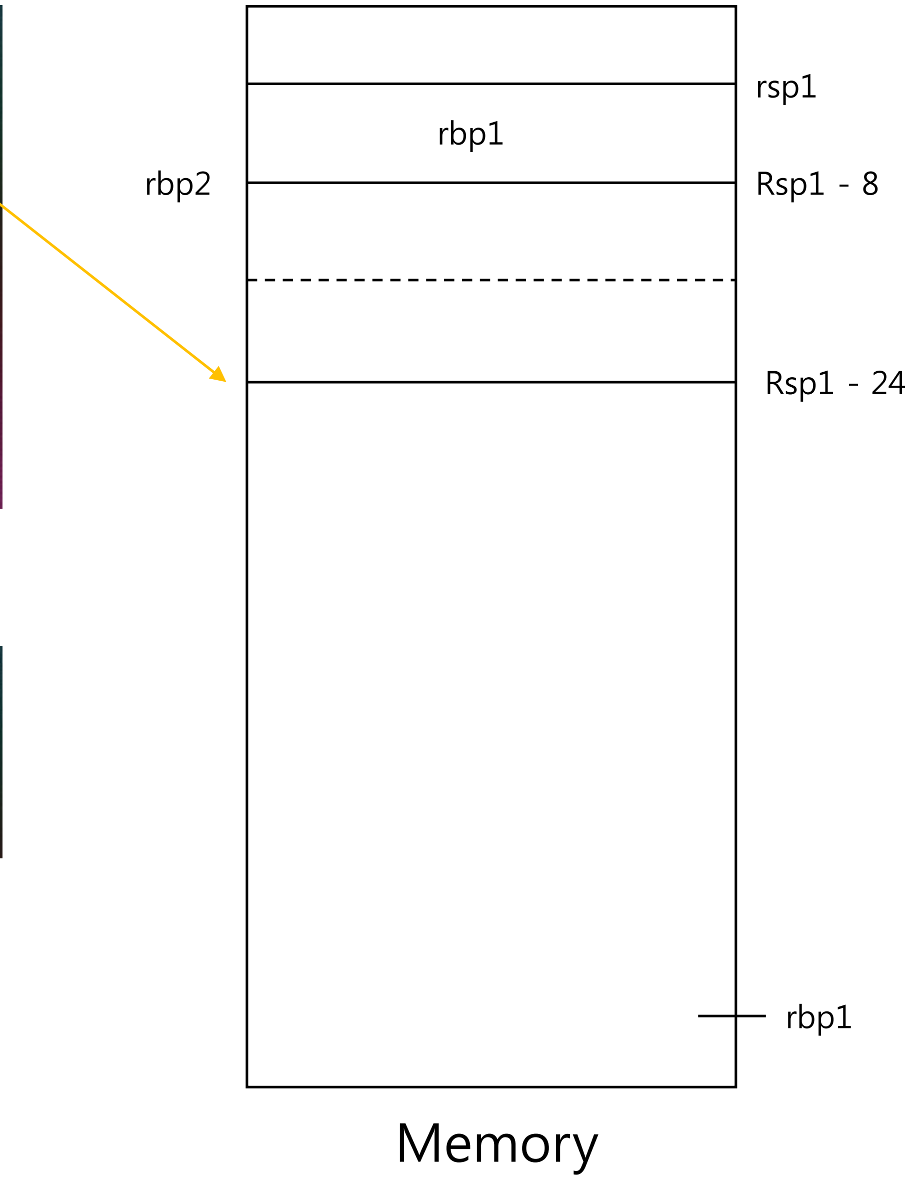


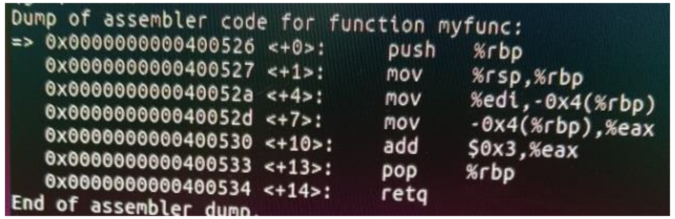
```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```

1. 기계어 분석 – (3)



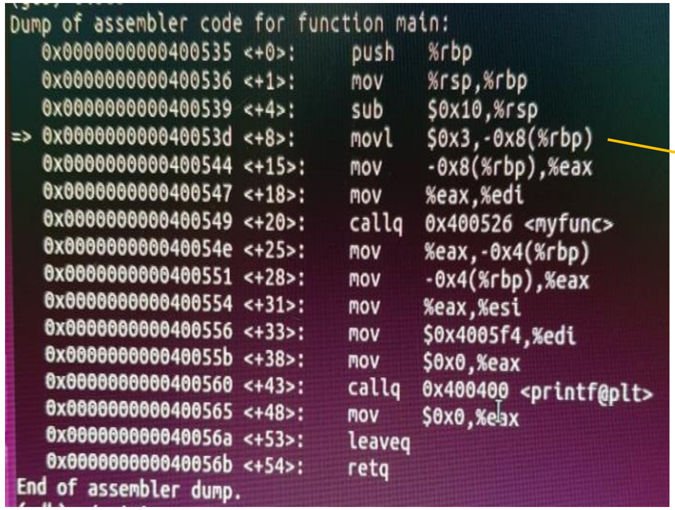
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



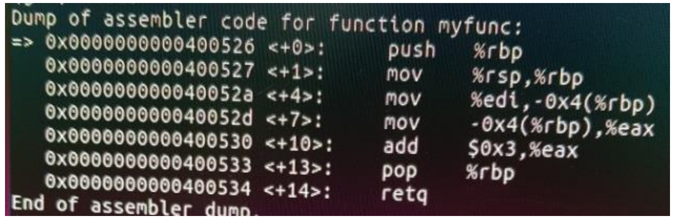


```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```

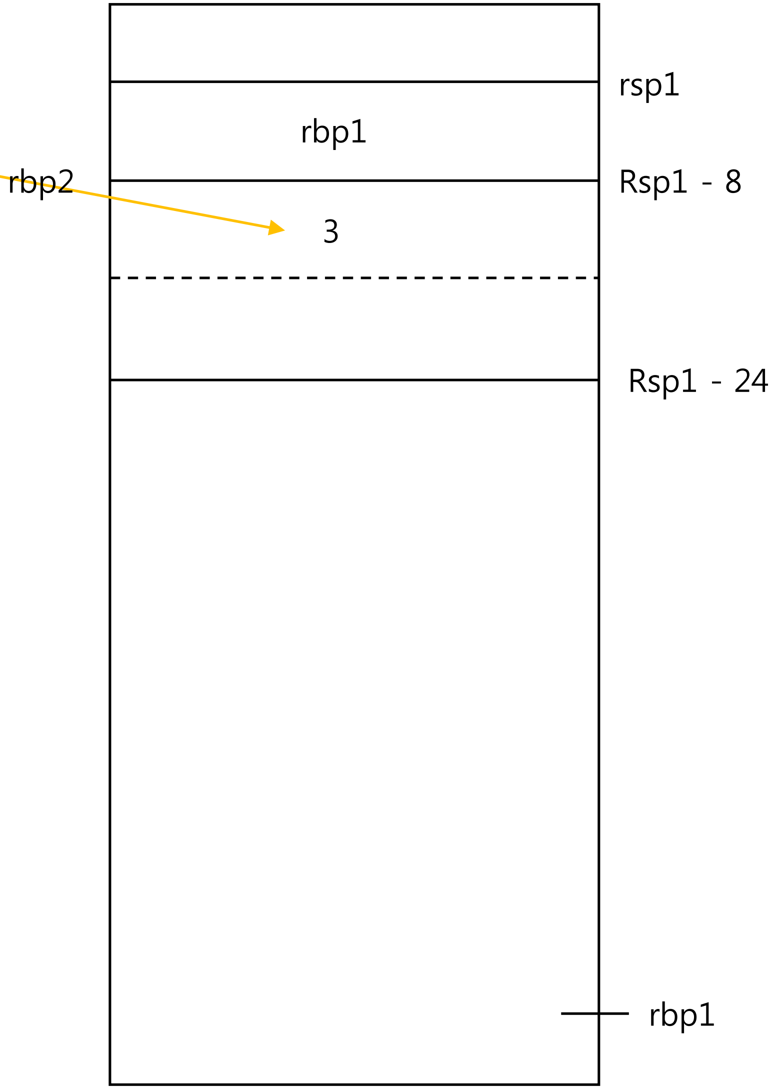
1. 기계어 분석 – (4)



```
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.
```

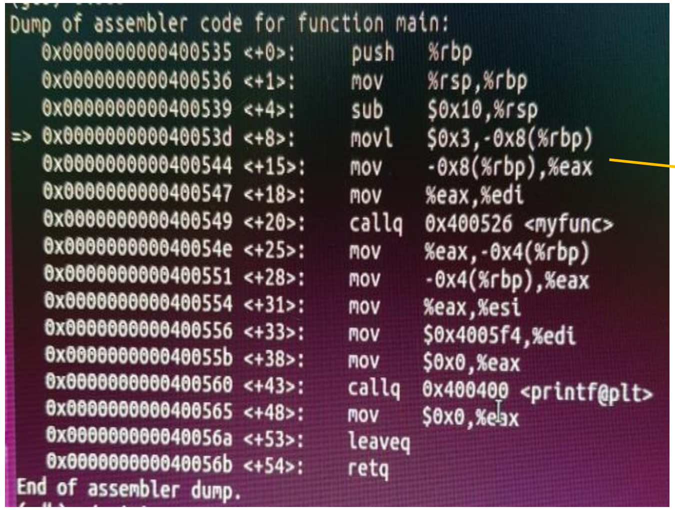


```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```

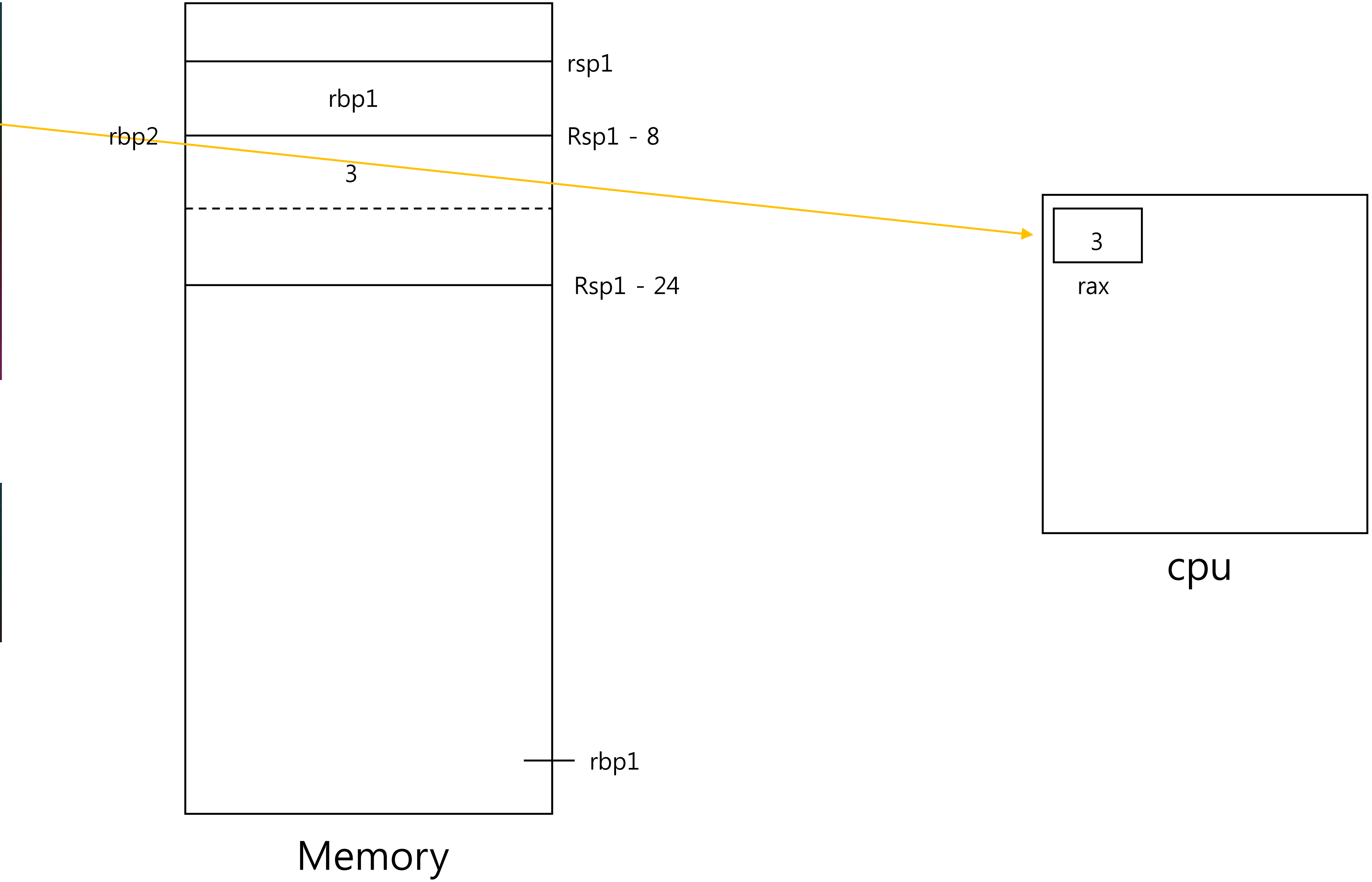


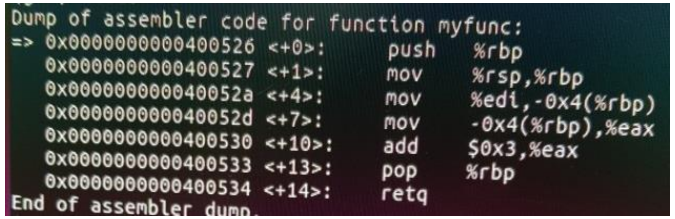
Memory

1. 기계어 분석 – (5)



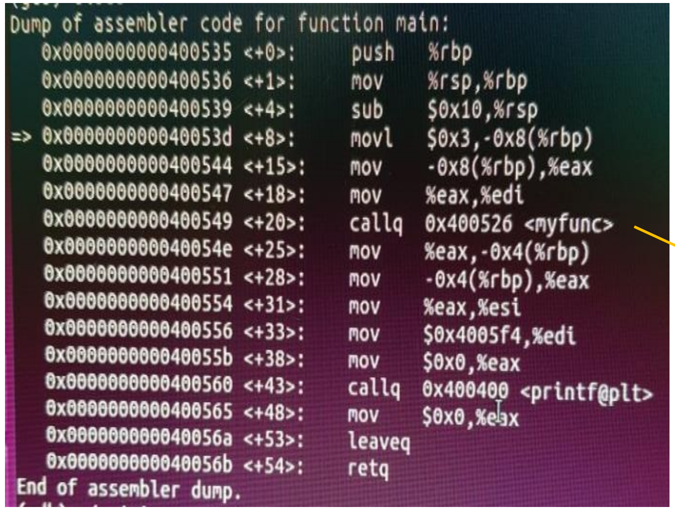
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



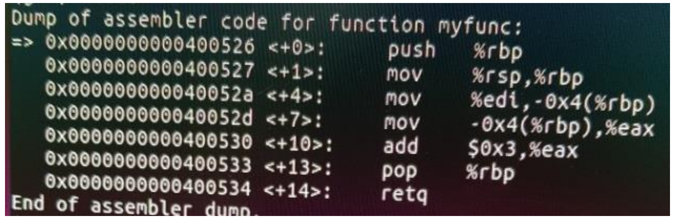


```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```

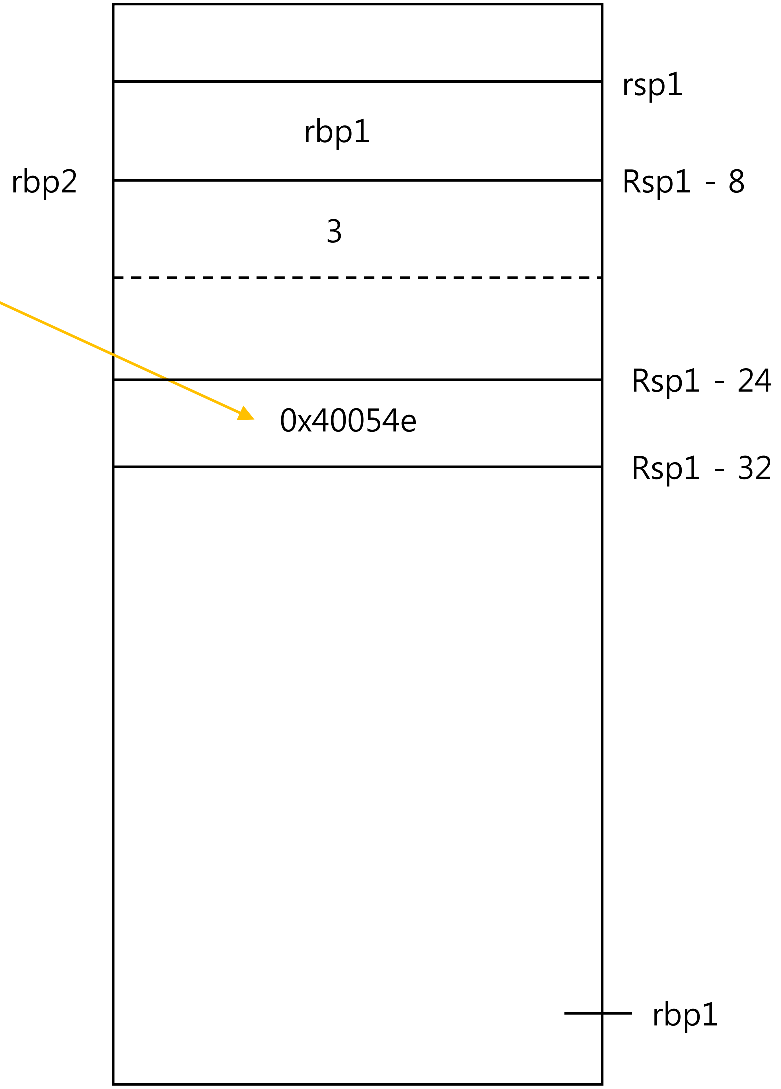
1. 기계어 분석 – (6)



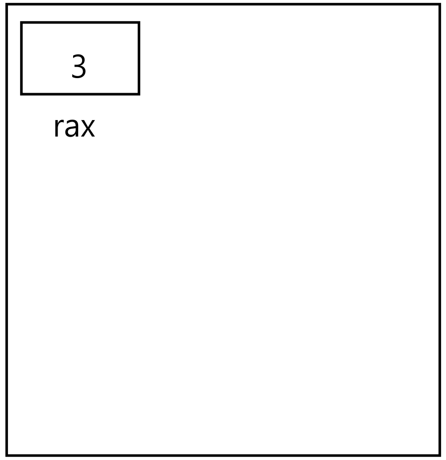
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



```
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.
```

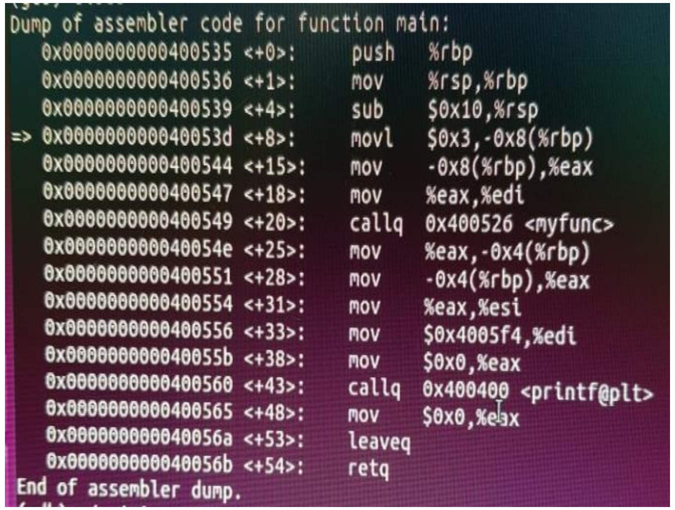


Memory

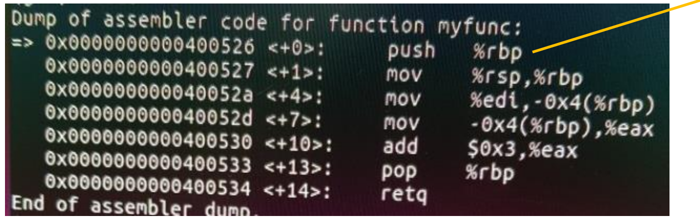


cpu

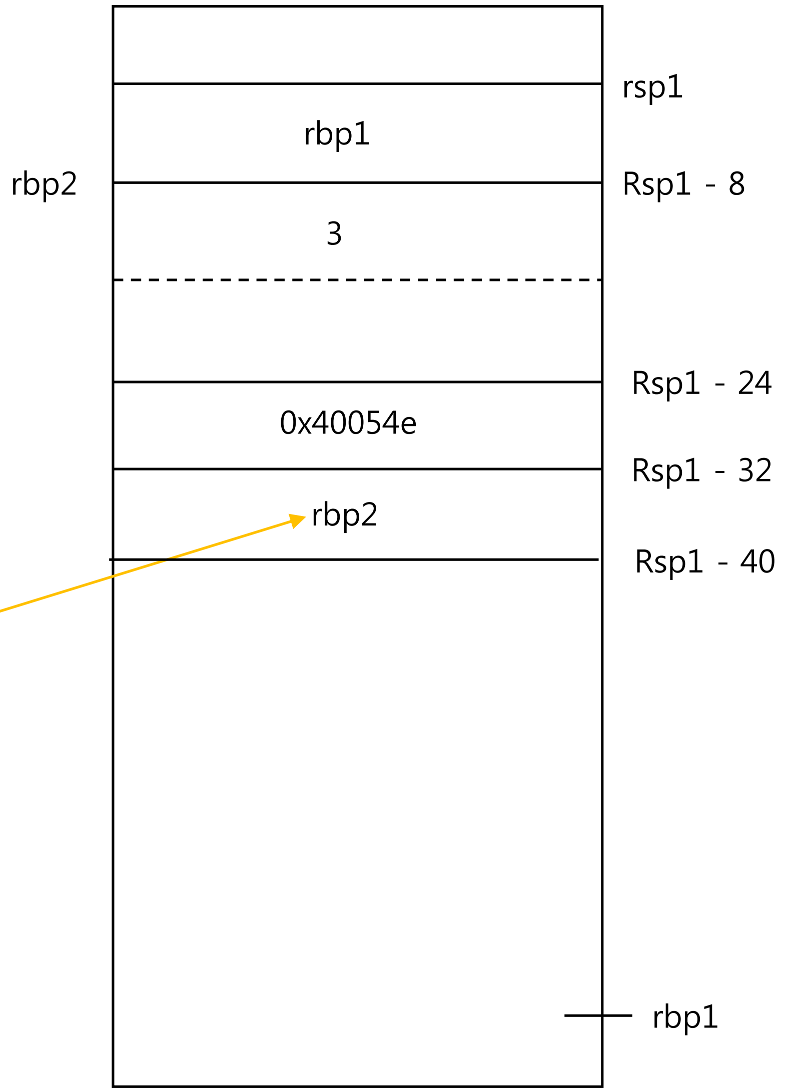
1. 기계어 분석 – (7)



```
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.
```



```
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.
```



rsp1

rbp1

rbp2

Rsp1 - 8

3

Rsp1 - 24

0x40054e

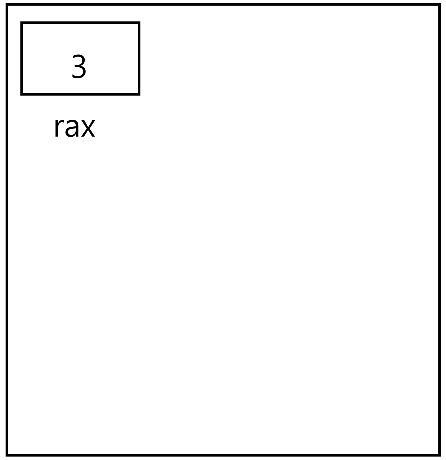
Rsp1 - 32

rbp2

Rsp1 - 40

rbp1

Memory

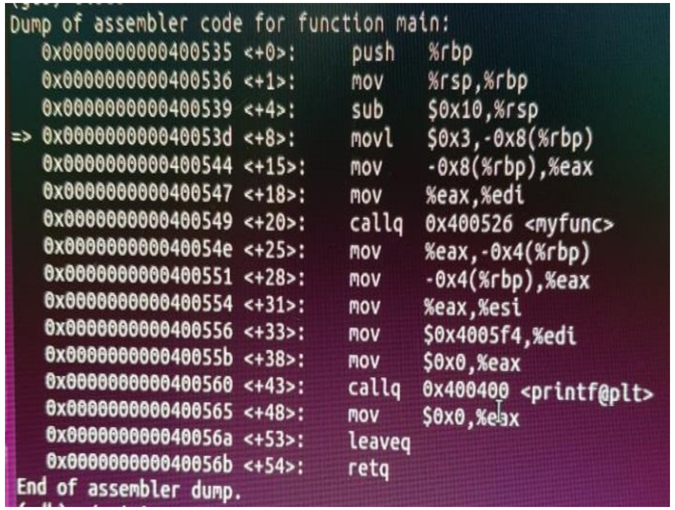


3

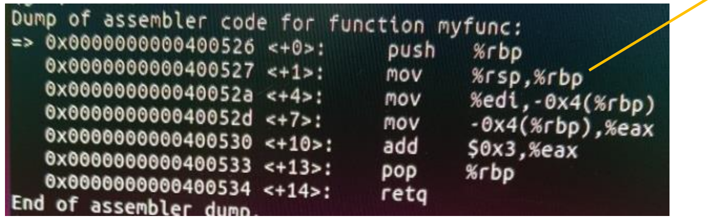
rax

cpu

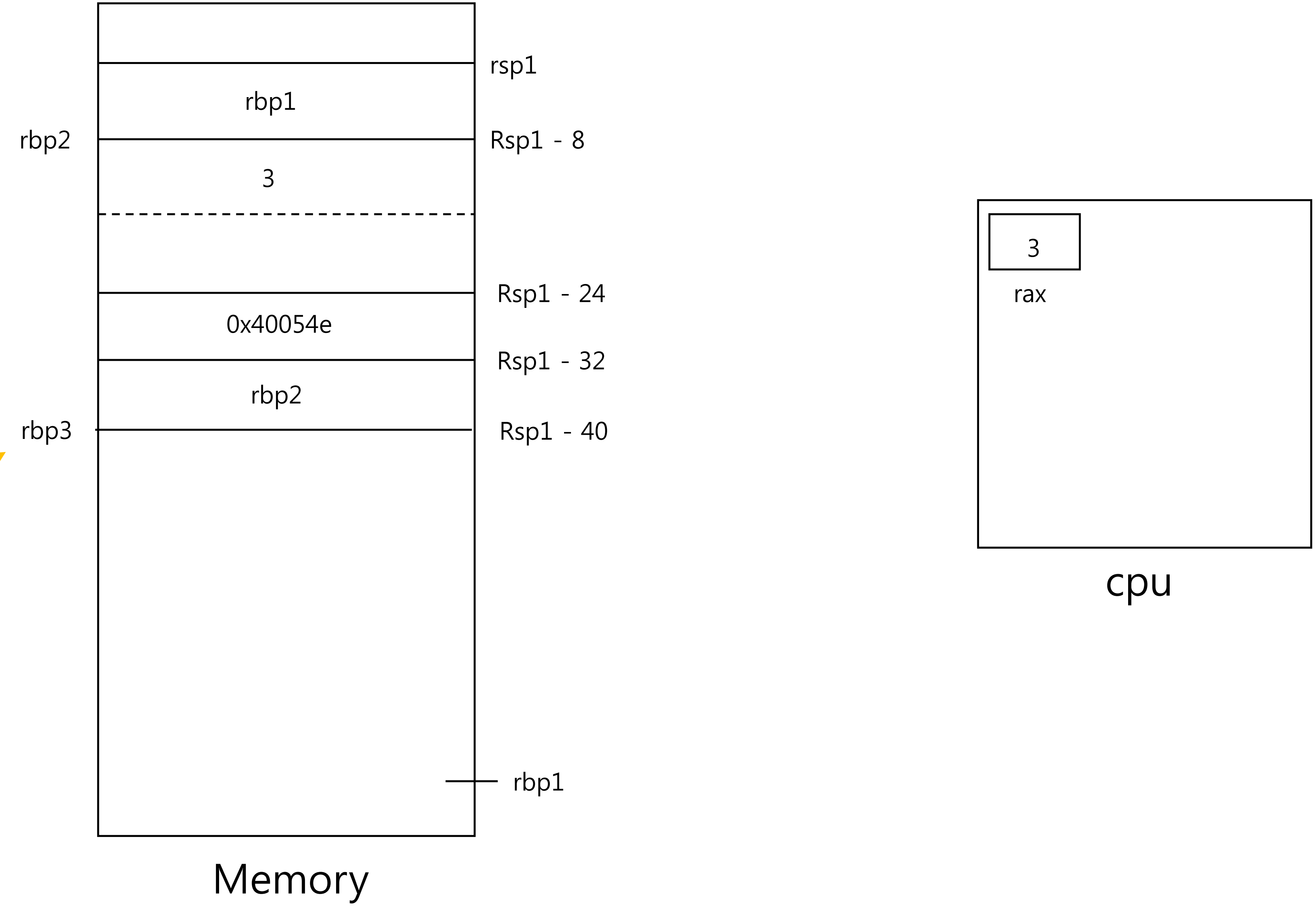
1. 기계어 분석 – (8)



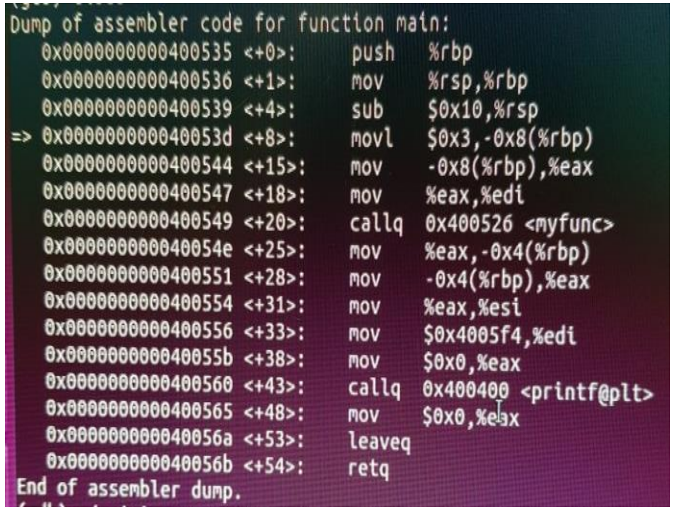
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



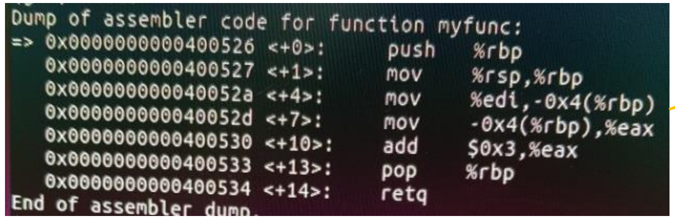
```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```



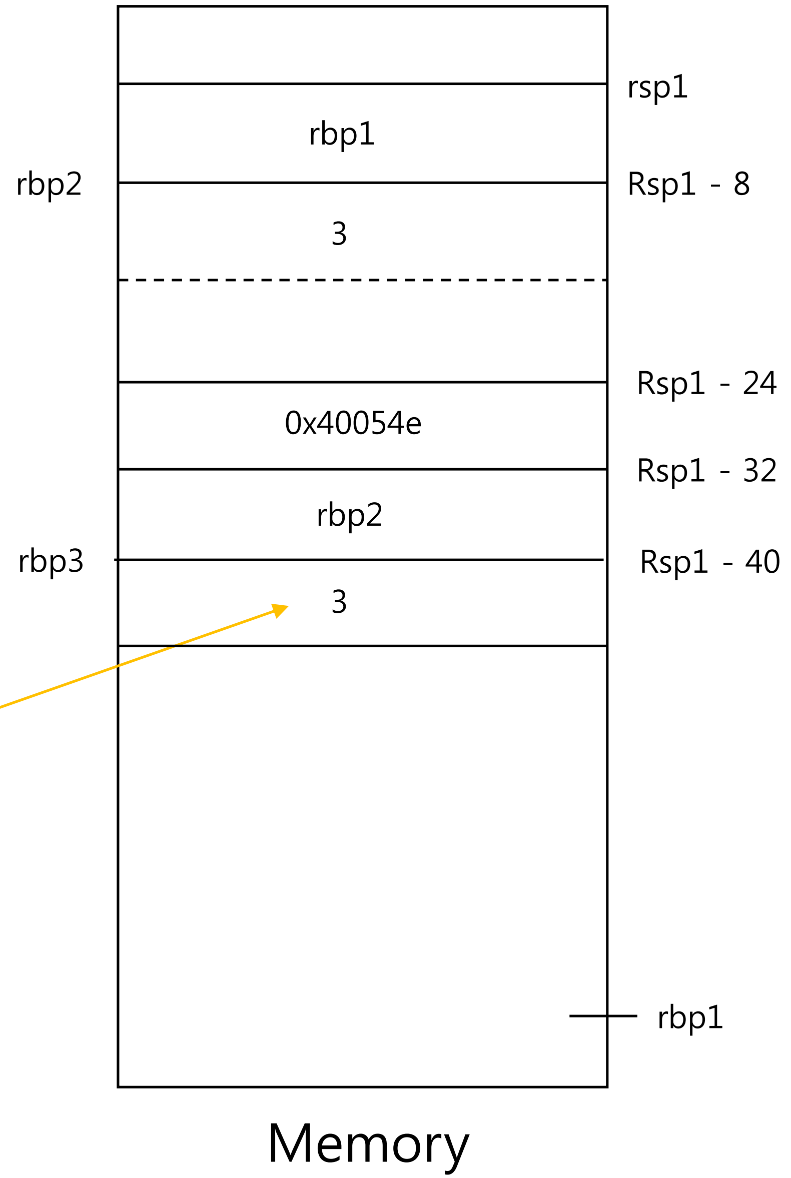
1. 기계어 분석 – (9)



```
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.
```



```
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.
```



rsp1

rbp1

rbp2

Rsp1 - 8

3

Rsp1 - 24

0x40054e

Rsp1 - 32

rbp2

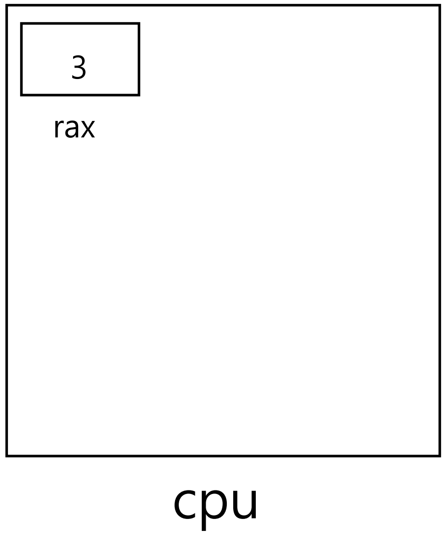
rbp3

Rsp1 - 40

3

rbp1

Memory

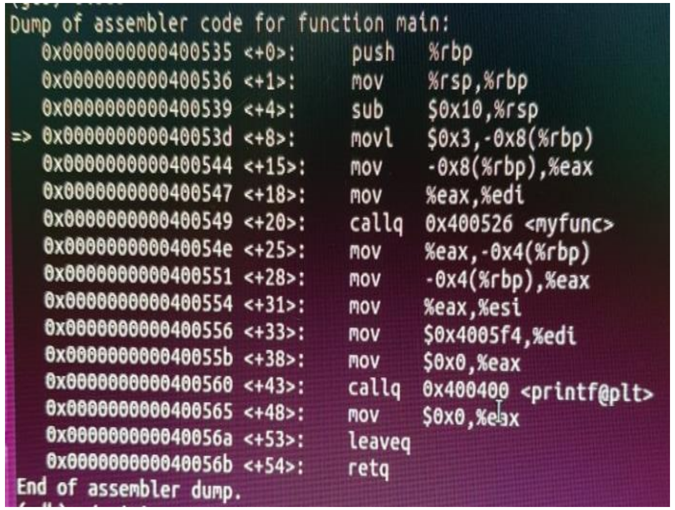


3

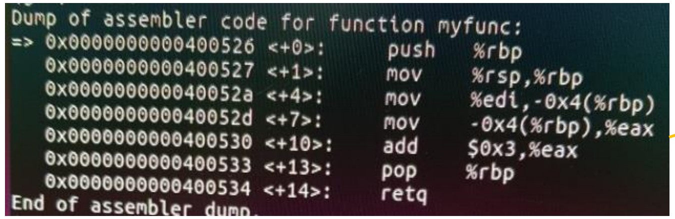
rax

cpu

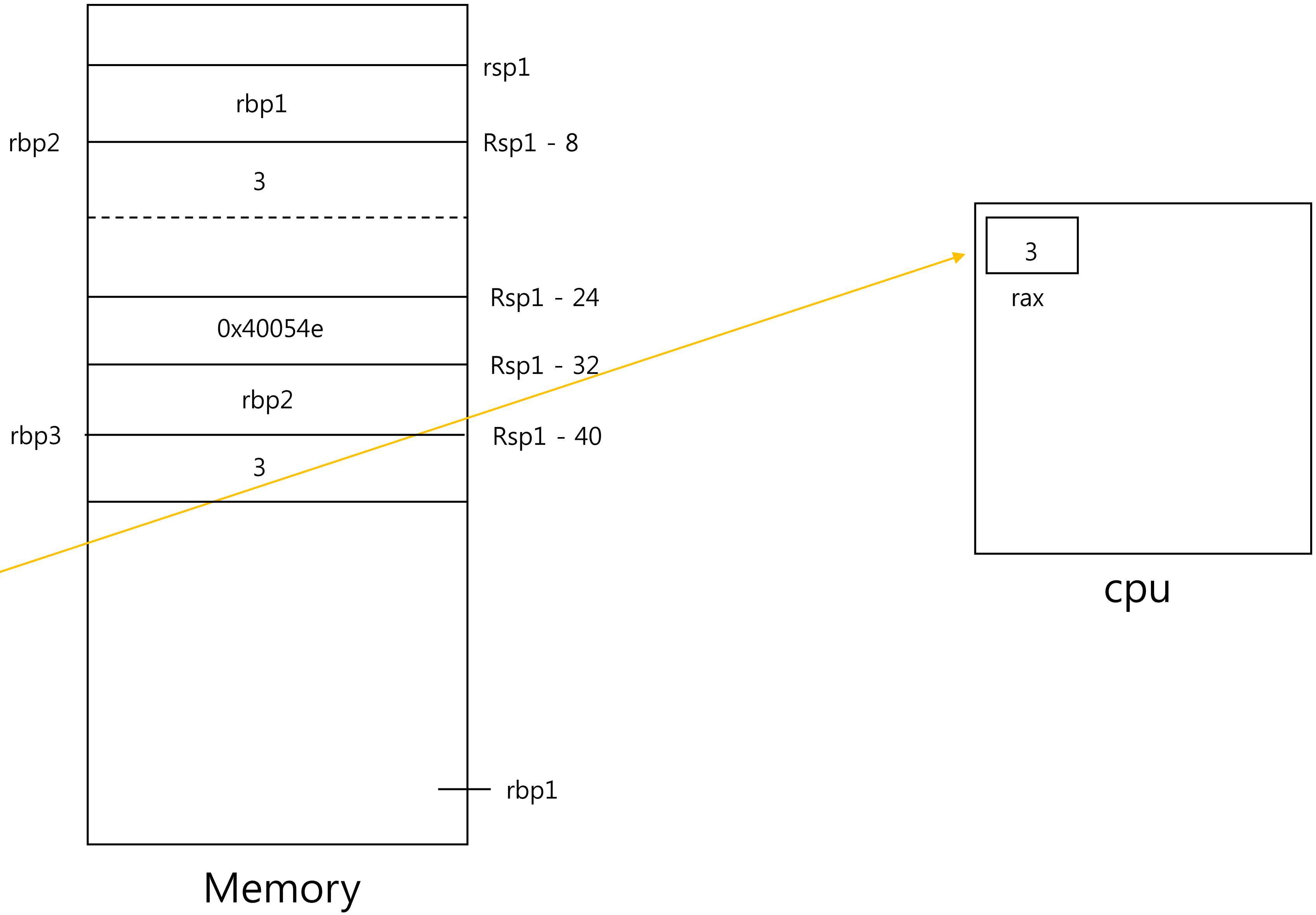
1. 기계어 분석 – (10)



```
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.
```



```
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.
```



rsp1

rbp1

rbp2

Rsp1 - 8

3

Rsp1 - 24

0x40054e

Rsp1 - 32

rbp2

rbp3

Rsp1 - 40

3

rbp1

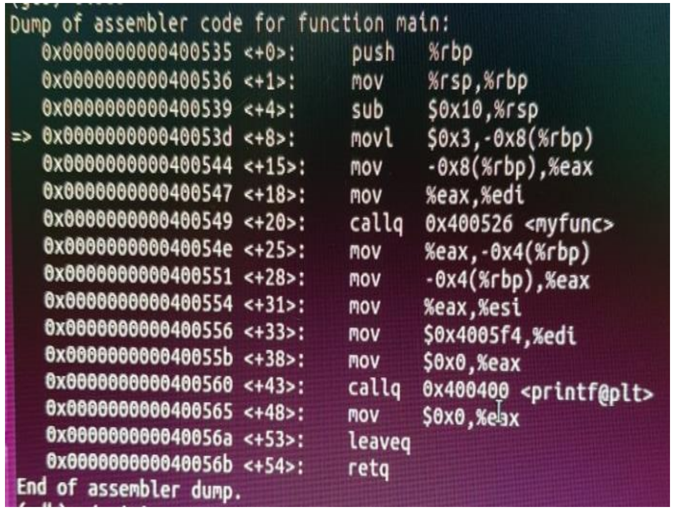
Memory

3

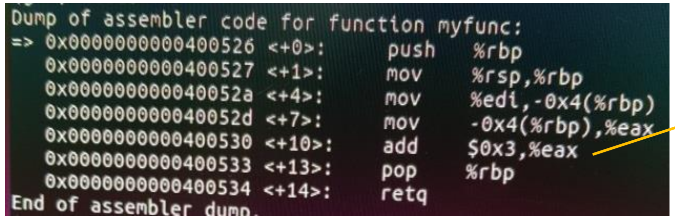
rax

cpu

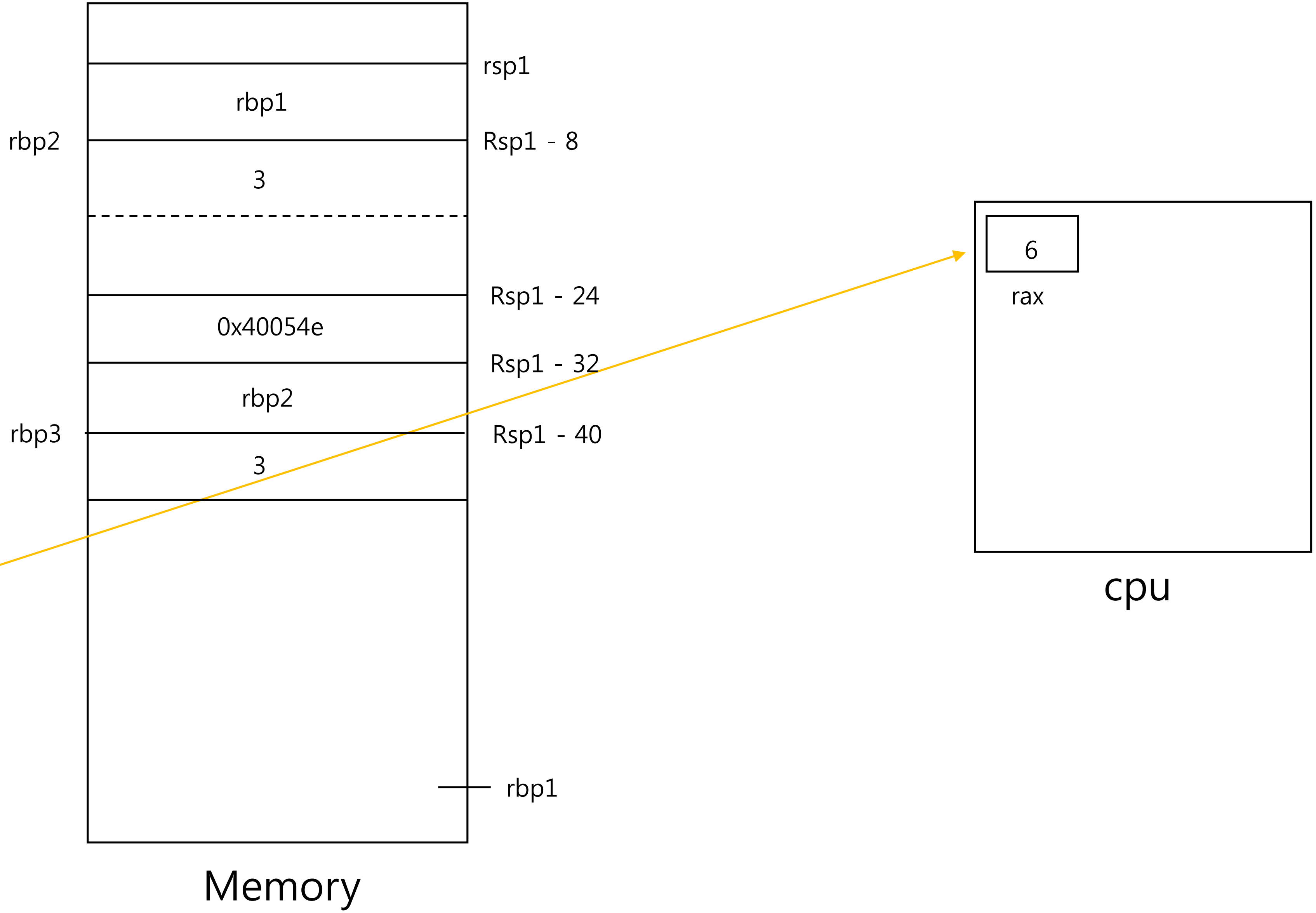
1. 기계어 분석 – (11)



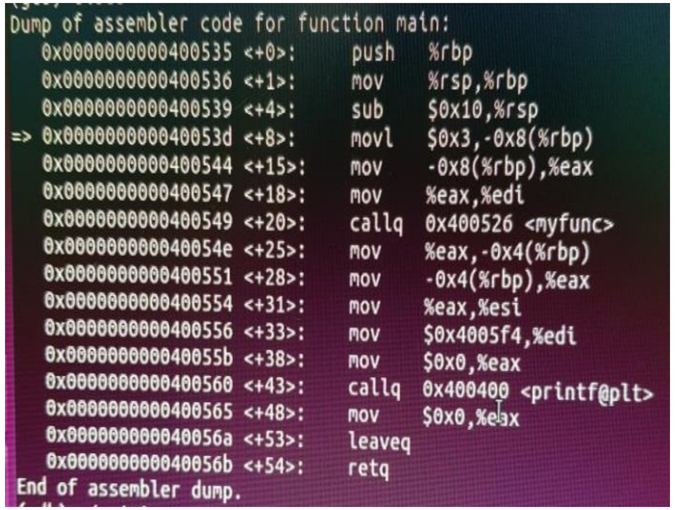
```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



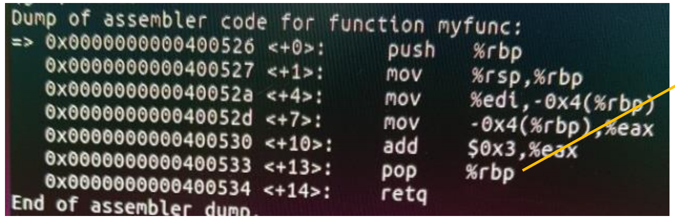
```
Dump of assembler code for function myfunc:
=> 0x0000000000400526 <+0>:     push   %rbp
   0x0000000000400527 <+1>:     mov    %rsp,%rbp
   0x000000000040052a <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040052d <+7>:     mov    -0x4(%rbp),%eax
   0x0000000000400530 <+10>:    add    $0x3,%eax
   0x0000000000400533 <+13>:    pop    %rbp
   0x0000000000400534 <+14>:    retq
End of assembler dump.
```



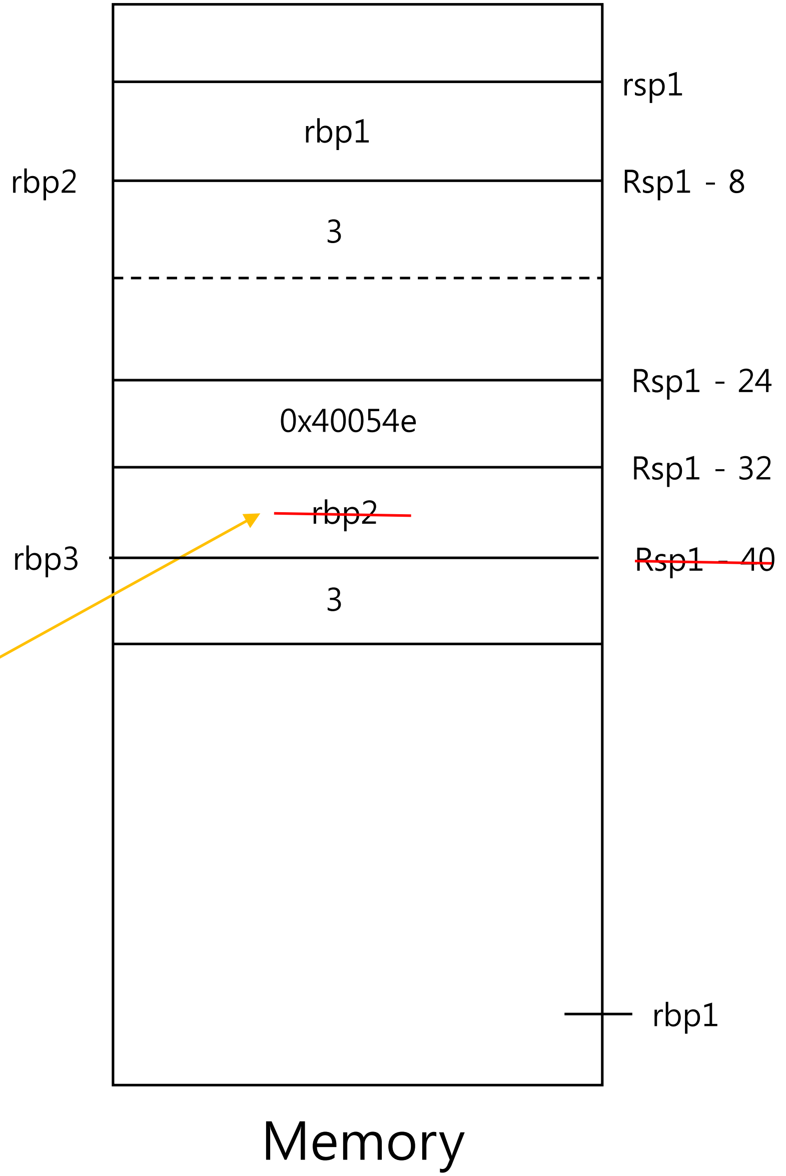
1. 기계어 분석 – (12)



```
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.
```



```
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.
```



rsp1

rbp1

rbp2

Rsp1 - 8

3

Rsp1 - 24

0x40054e

Rsp1 - 32

~~rbp2~~

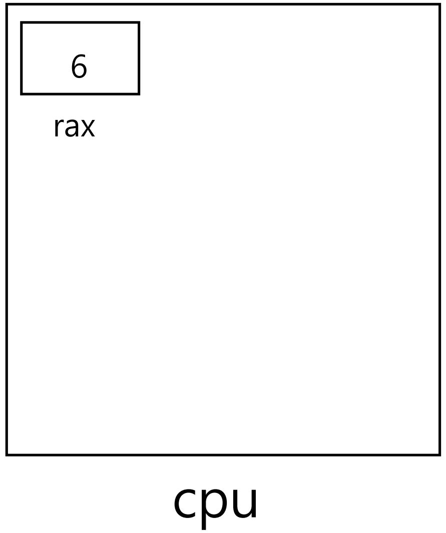
rbp3

~~Rsp1 - 40~~

3

rbp1

Memory

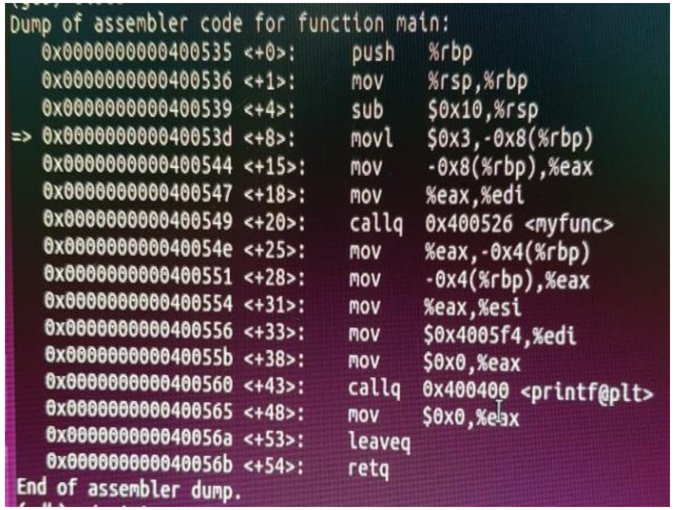


6

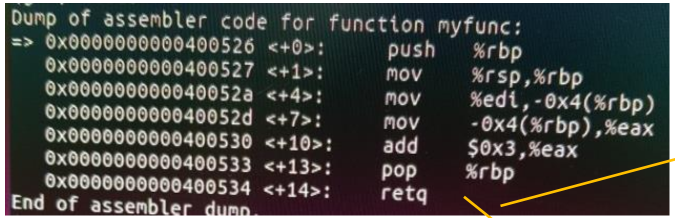
rax

cpu

1. 기계어 분석 – (13)

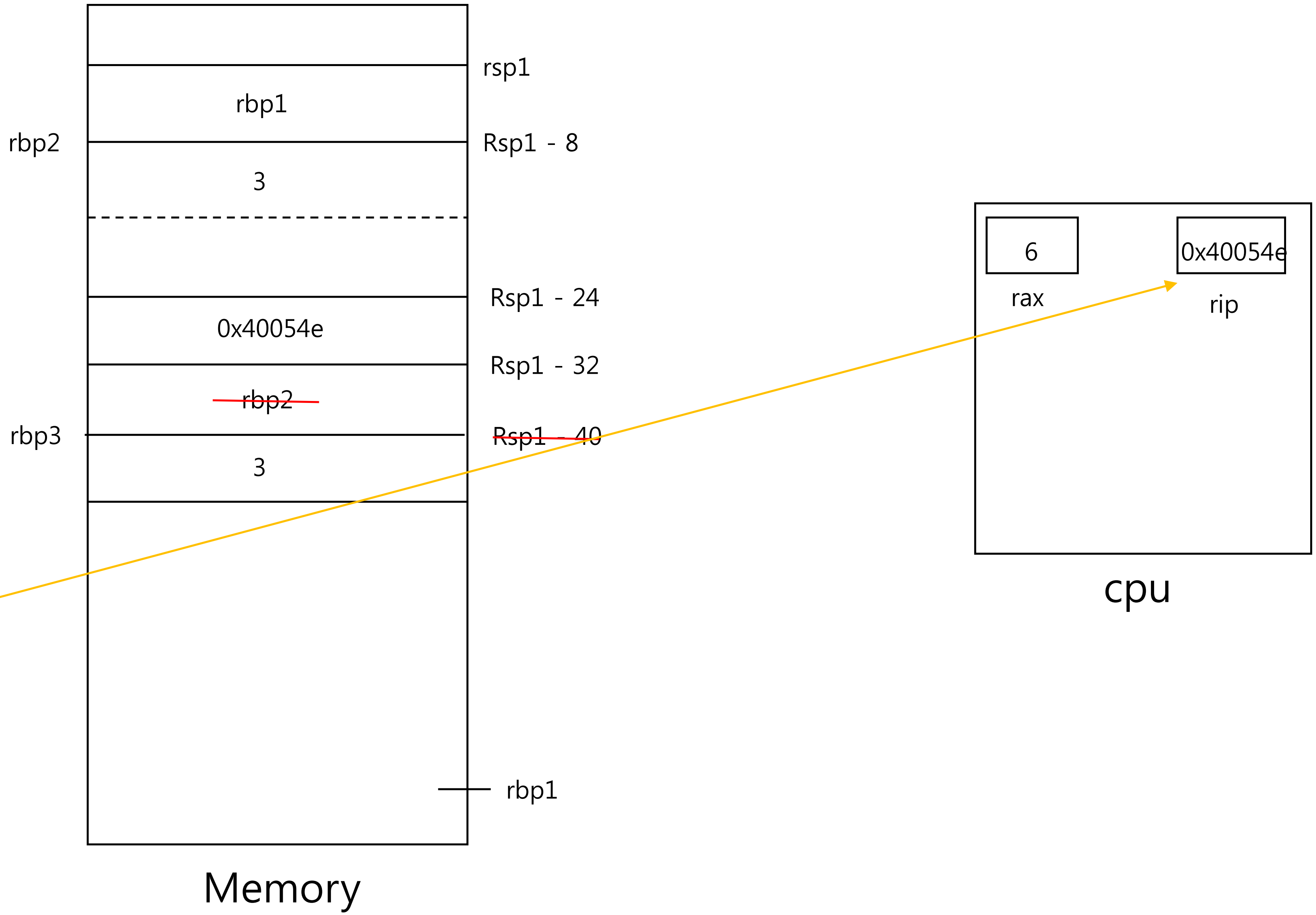


```
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
=> 0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   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.
```



```
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.
```

Pop rip



2. 포인터 크기 내용 정리

- 8 비트 시스템 – 1byte
- 16 비트 -> 2byte
- 32 비트 -> 4byte
- 64 비트 -> 8byte

Why?

컴퓨터의 산술 연산은 ALU에 의존적임.
ALU의 연산은 범용 레지스터에 종속적이고 컴퓨터가 64비트라는 의미는
레지스터들이 64bit으로 구성 되었음을 의미함.

변수의 정의는 메모리에 정보를 저장하는 공간.
포인터의 정의는 메모리에 주소를 저장하는 공간이다.
그렇다면, 64bit으로 표현할 수 있는 최대값 또한 저장할 수 있어야 함.

즉, 포인터의 크기가 작다면 64bit 주소를 표현할 방법이 없기 때문에
최대치인 64bit(8byte)가 포인터의 크기가 된 것임.

3. 2진수 16진수 변환 정리

- 2진수 1자리
  0 1 -> 2개
- 2진수 2자리
  00 01 10 11 -> 4개
- 2진수 3자리
  000 001 010 011 100 101 110 111 -> 8개
- 2진수 4자리 ---> 16개

- 16진수 1자리
  0 ~ 15 16개
- 16진수 2자리
  --- 256개

3. 2진수 16진수 변환 정리

- 16진수 표기법

1.	0	0
2.	1	1
3.	2	2
4.	3	3
5.	4	4
6.	5	5
7.	6	6
8.	7	7
9.	8	8
10.	9	9
11.	10	a
12.	11	b
13.	12	c
14.	13	d
15.	14	e
16.	15	f

10진수 33을 2진수 및 16진수로 표기법

33 = 32 + 1

32	16	8	4	2	1
1	0	0	0	0	1

10 0001

8421	8421	(4개씩 끊어 쓰면 편함)
0010	0001	

0x2	1	

0x21 => 2 x 16^1 + 1 x 16^0 = 33