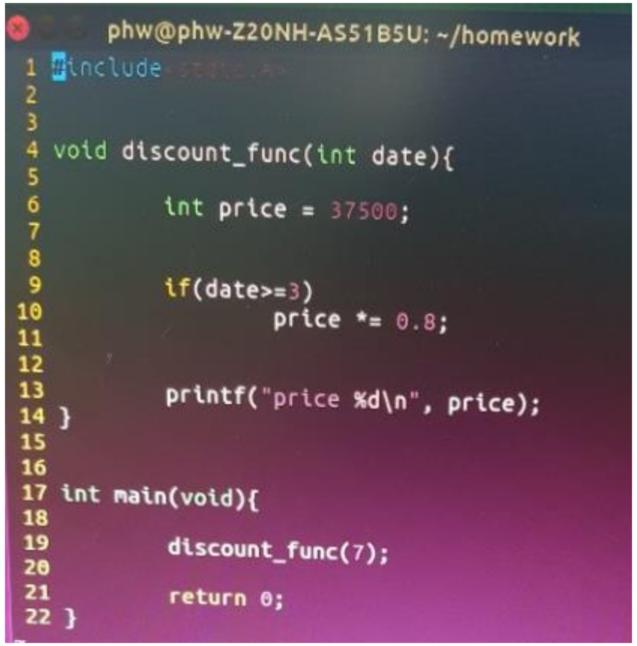
과제 1

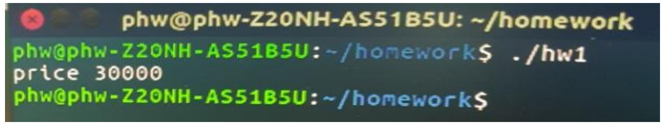
1. 스키장에서 스키 장비를 임대하는데 37500원이 든다.
   또 3일 이상 이용할 경우 20%를 할인 해준다.
   일주일간 이용할 경우 임대 요금은 얼마일까 ?
   (연산 과정은 모두 함수로 돌린다)

Code



```c
#include <stdio.h>


void discount_func(int date){

        int price = 37500;


        if(date>=3)
                price *= 0.8;


        printf("price %d\n", price);
}


int main(void){

        discount_func(7);

        return 0;
}
```

Result

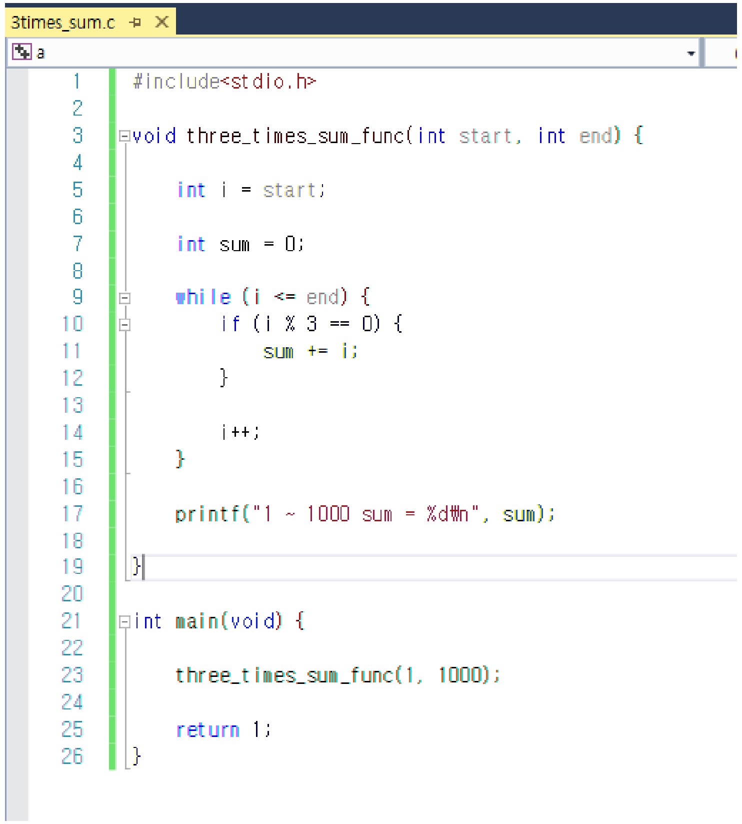


```
phw@phw-Z20NH-AS51B5U:~/homework$ ./hw1
price 30000
phw@phw-Z20NH-AS51B5U:~/homework$
```

과제 3

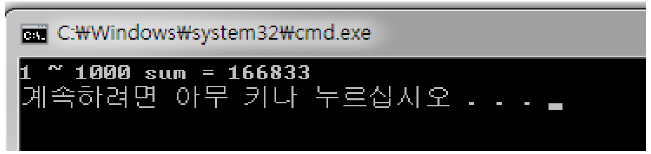
3. 1 ~ 1000사이에 3의 배수의 합을 구하시오.

Code



```
#include<stdio.h>

void three_times_sum_func(int start, int end) {

    int i = start;

    int sum = 0;

    while (i <= end) {
        if (i % 3 == 0) {
            sum += i;
        }

        i++;
    }

    printf("1 ~ 1000 sum = %d₩n", sum);

}

int main(void) {

    three_times_sum_func(1, 1000);

    return 1;
}
```

Result

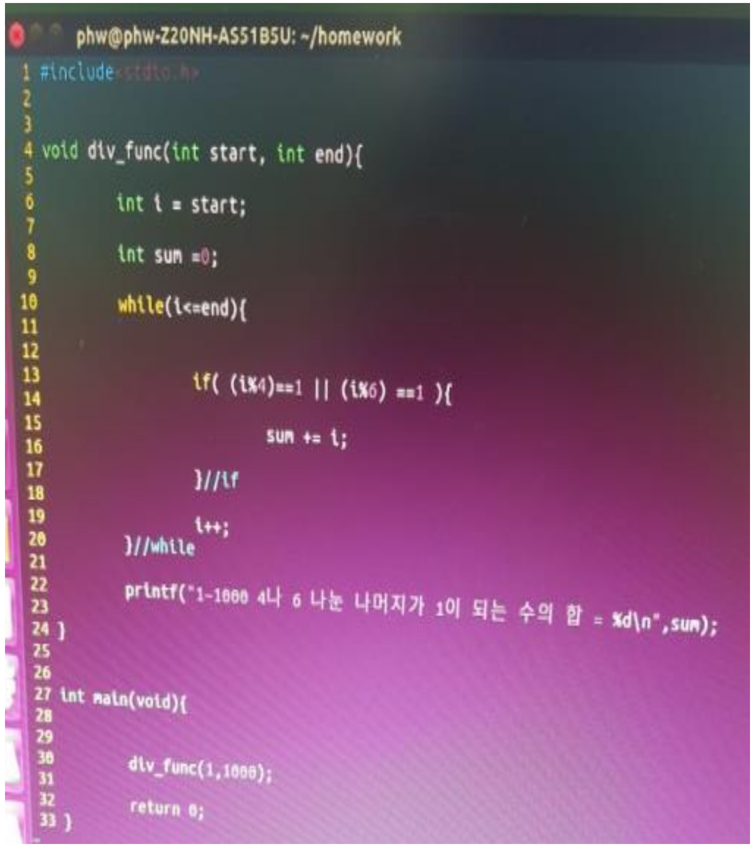


```
1 ~ 1000 sum = 166833
계속하려면 아무 키나 누르십시오 . . .
```

과제 4

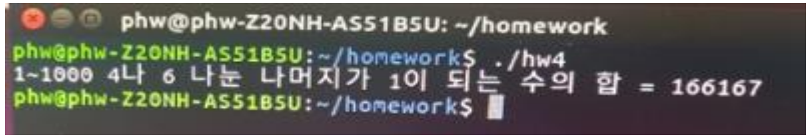
4. 1 ~ 1000사이에 4나 6으로 나눠도 나머지가 1인 수의 합을 출력하라.

Code



```c
#include<stdio.h>


void div_func(int start, int end){

        int i = start;

        int sum =0;

        while(i<=end){


                if( (i%4)==1 || (i%6) ==1 ){

                        sum += i;

                }//if

                i++;
        }//while

        printf("1-1000 4나 6 나눈 나머지가 1이 되는 수의 합 = %d\n",sum);
}


int main(void){


        div_func(1,1000);

        return 0;
}
```

Result

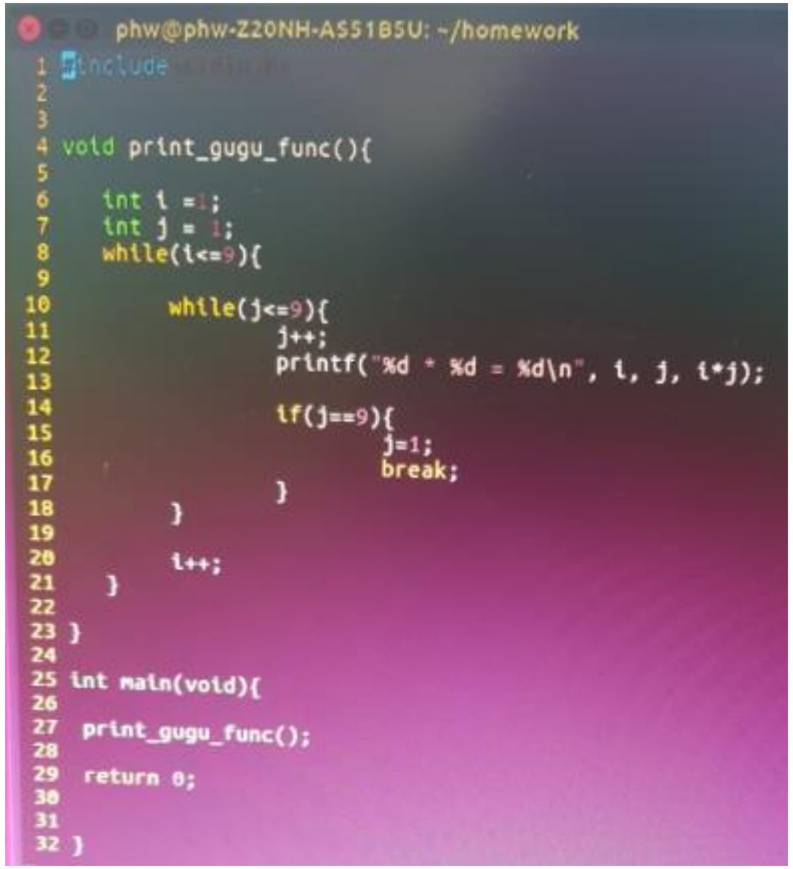


```
phw@phw-Z20NH-AS51B5U:~/homework$ ./hw4
1~1000 4나 6 나눈 나머지가 1이 되는 수의 합 = 166167
phw@phw-Z20NH-AS51B5U:~/homework$ 
```

과제 10

10. 구구단을 만들어보시오.

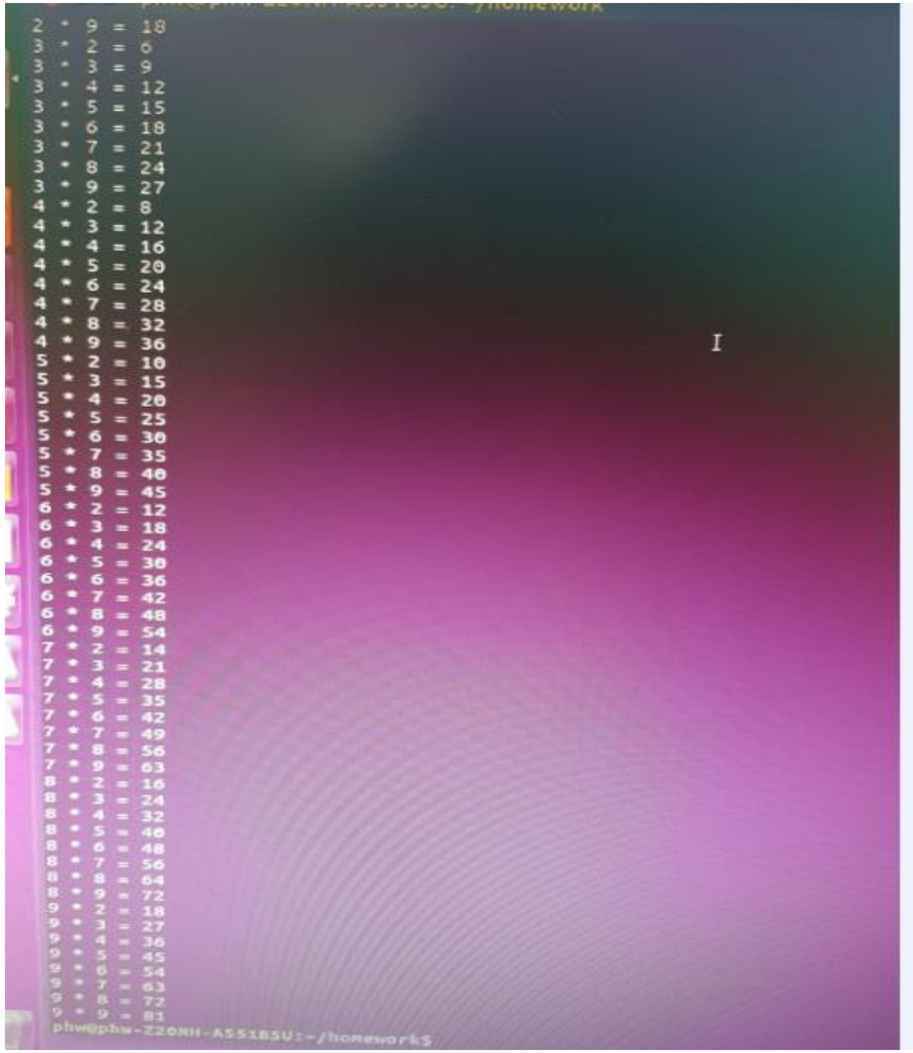
Code



phw@phw-Z20NH-AS51B5U: ~/homework

```c
#include <stdio.h>


void print_gugu_func(){

	int i =1;
	int j = 1;
	while(i<=9){

		while(j<=9){
			j++;
			printf("%d * %d = %d\n", i, j, i*j);

			if(j==9){
				j=1;
				break;
			}
		}

		i++;
	}

}

int main(void){

 print_gugu_func();

 return 0;


}
```

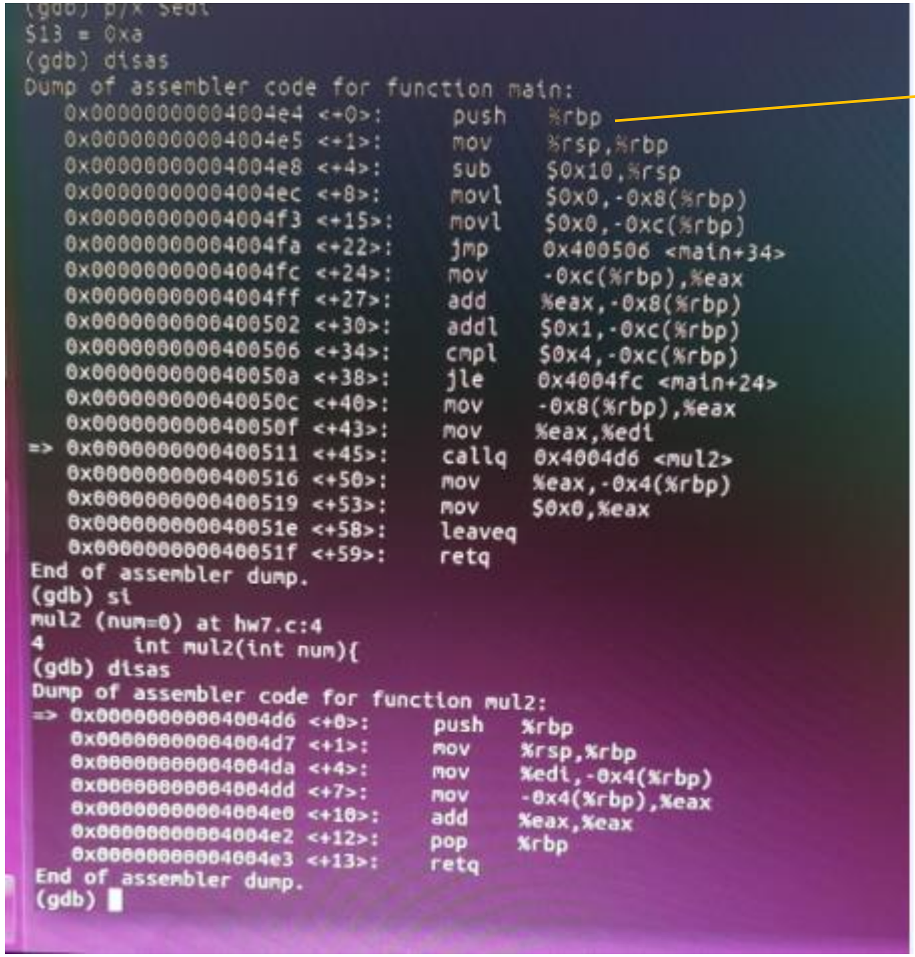
Result



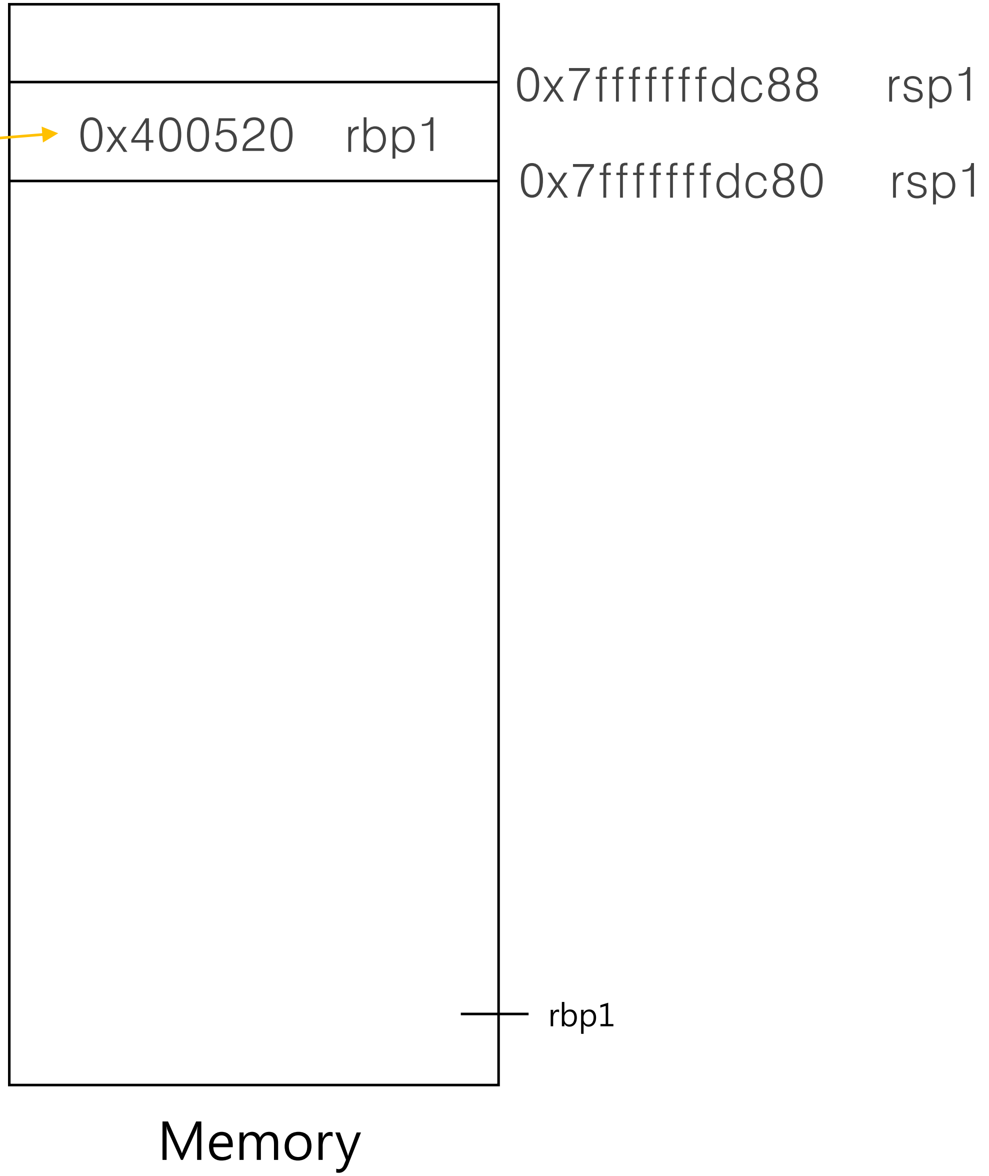
```
2 * 9 = 18
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
phw@phw-Z20NH-AS51B5U:~/homework$
```

과제 7 - 1 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



0x7fffffffdc88   rsp1

0x400520   rbp1

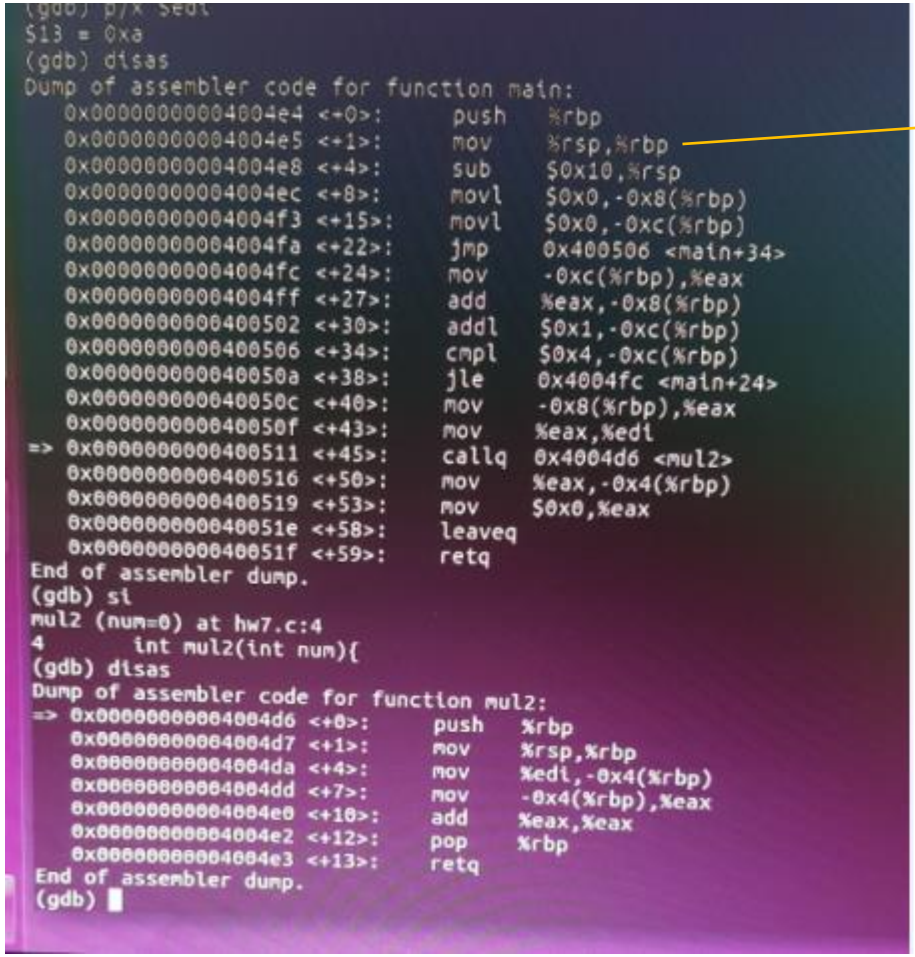
0x7fffffffdc80   rsp1

rbp1

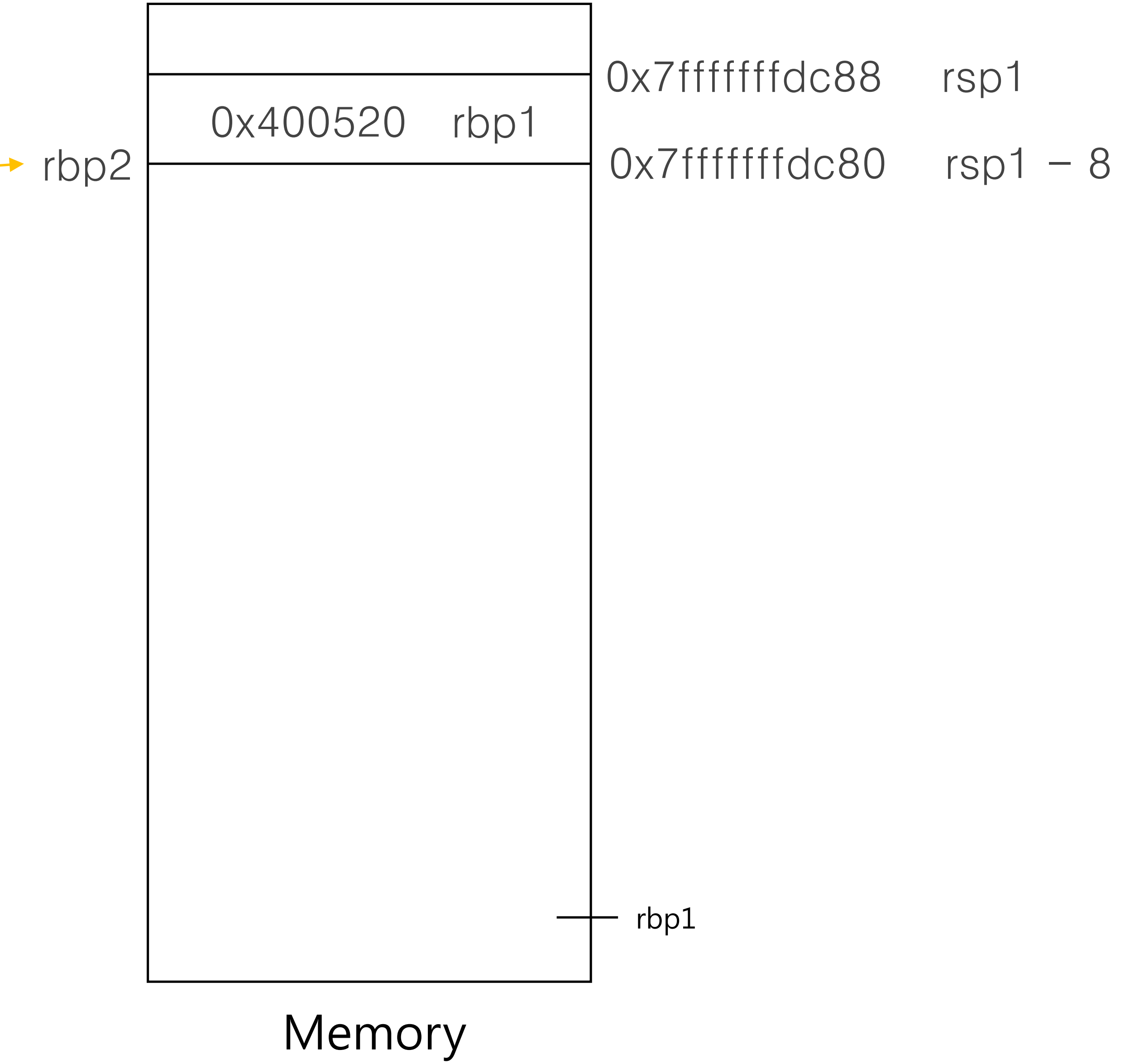
Memory

과제 7 - 2 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



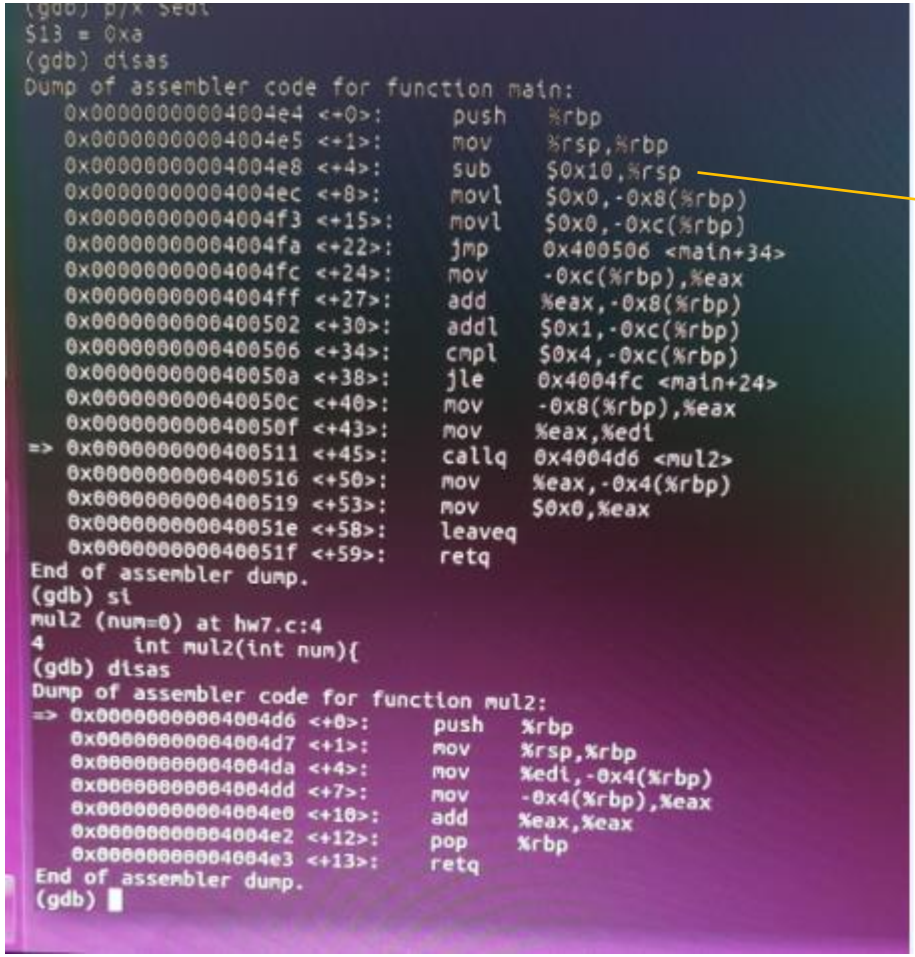
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:	push   %rbp
   0x00000000004004e5 <+1>:	mov    %rsp,%rbp
   0x00000000004004e8 <+4>:	sub    $0x10,%rsp
   0x00000000004004ec <+8>:	movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:	movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:	jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:	mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:	add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:	addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:	cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:	jle    0x4004fc <main+24>
   0x000000000040050c <+40>:	mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:	mov    %eax,%edi
=> 0x0000000000400511 <+45>:	callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:	mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:	mov    $0x0,%eax
   0x000000000040051e <+58>:	leaveq
   0x000000000040051f <+59>:	retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4	  int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:	push   %rbp
   0x00000000004004d7 <+1>:	mov    %rsp,%rbp
   0x00000000004004da <+4>:	mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:	mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:	add    %eax,%eax
   0x00000000004004e2 <+12>:	pop    %rbp
   0x00000000004004e3 <+13>:	retq
End of assembler dump.
(gdb) 
```



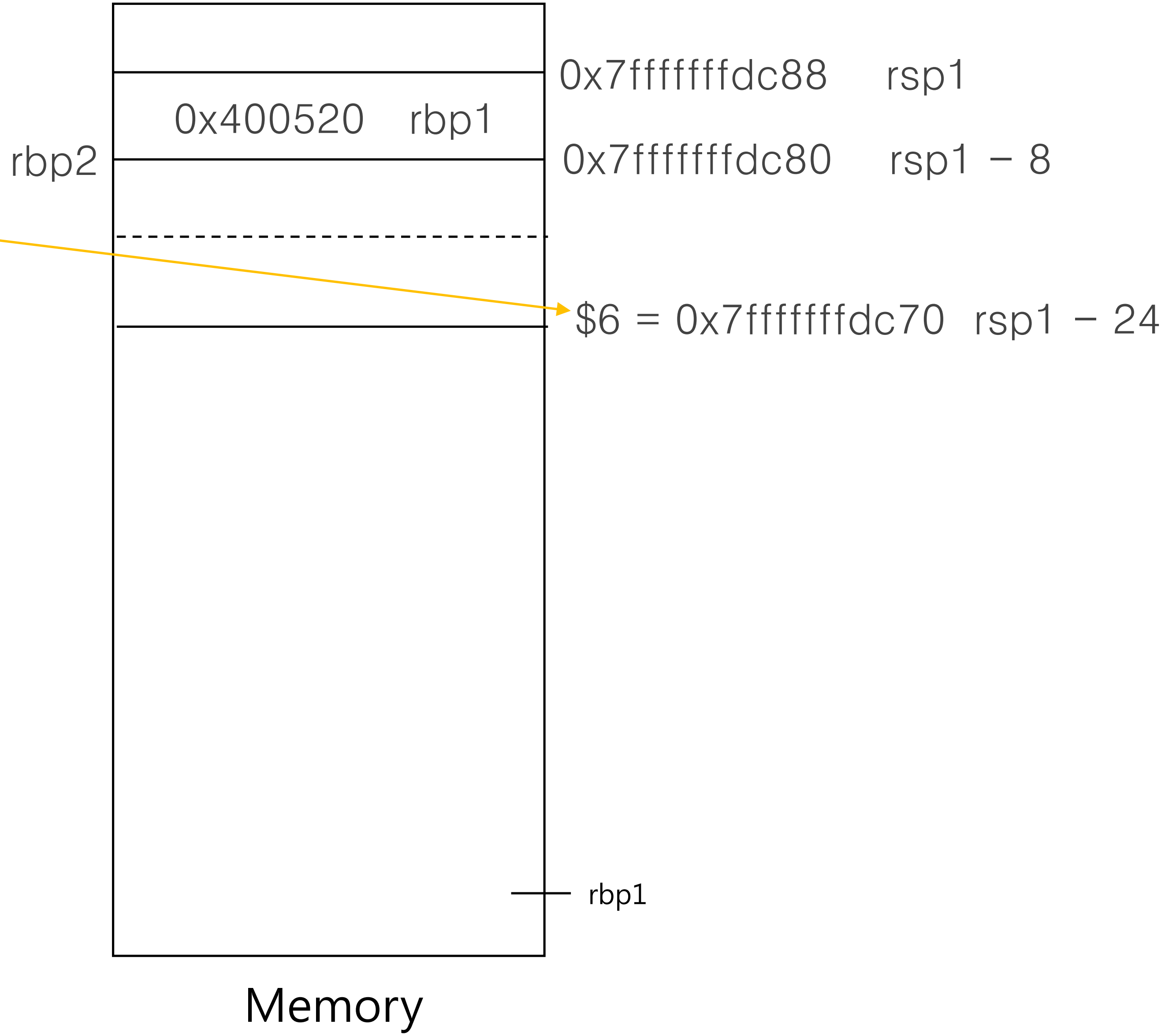
Memory

과제 7 - 3 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

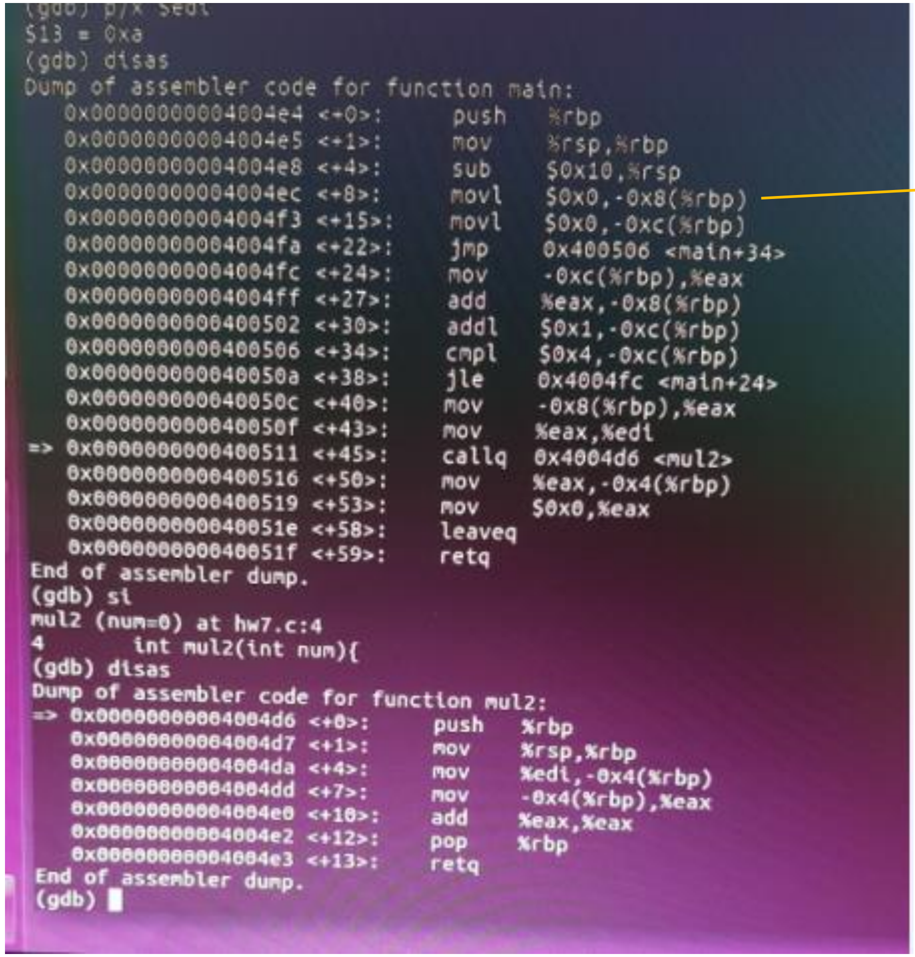


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

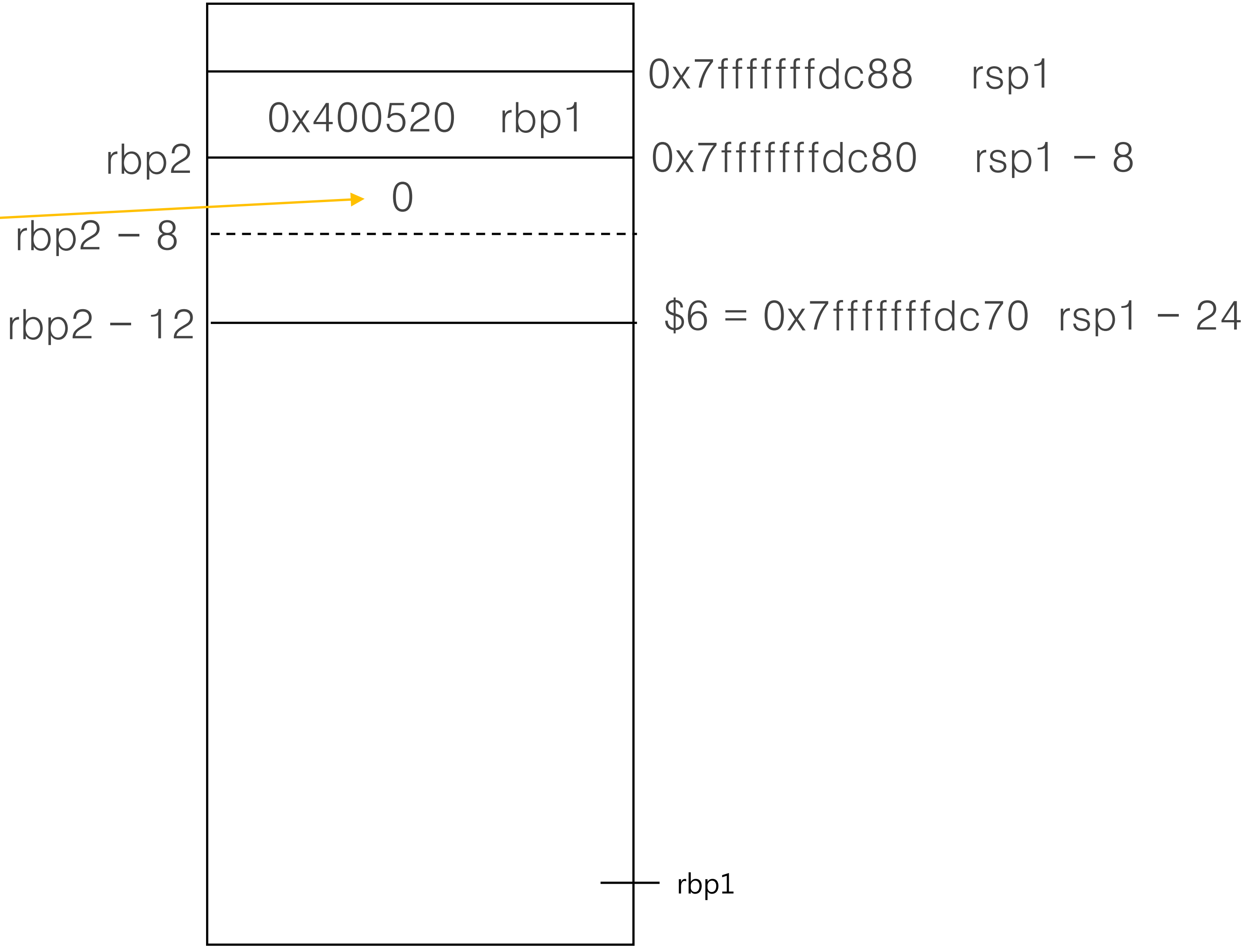


과제 7 - 4 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



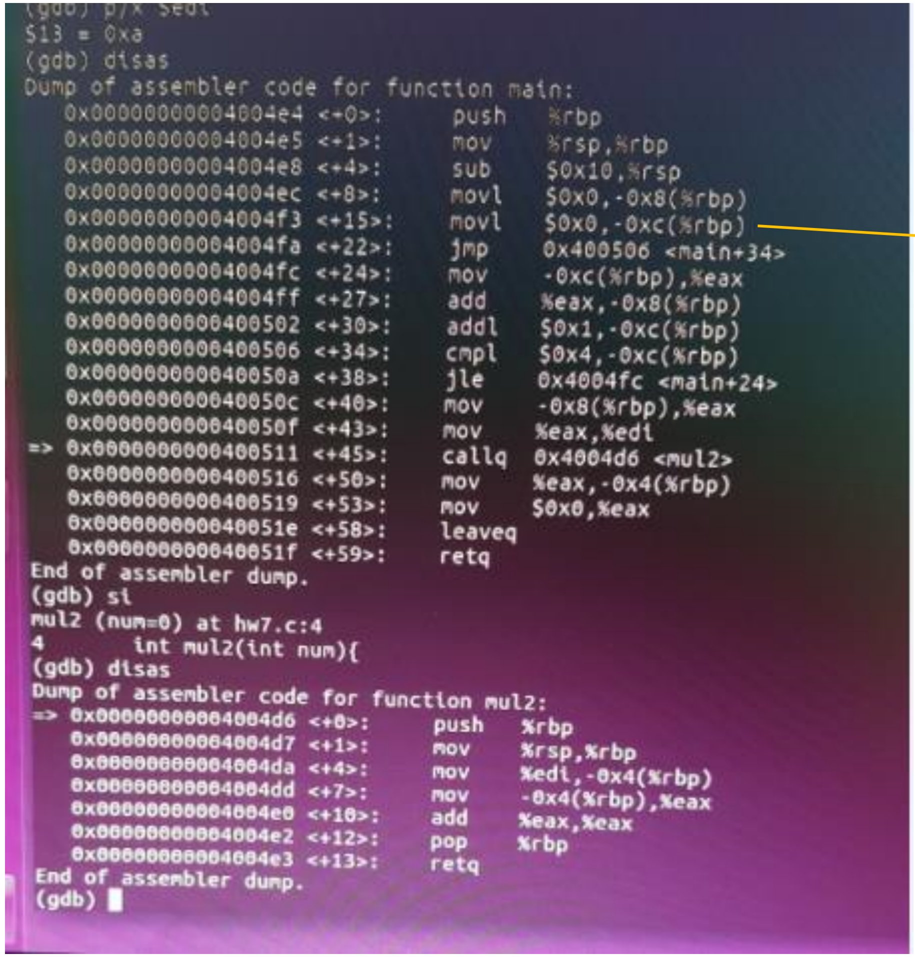
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:	push   %rbp
   0x00000000004004e5 <+1>:	mov    %rsp,%rbp
   0x00000000004004e8 <+4>:	sub    $0x10,%rsp
   0x00000000004004ec <+8>:	movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:	movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:	jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:	mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:	add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:	addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:	cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:	jle    0x4004fc <main+24>
   0x000000000040050c <+40>:	mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:	mov    %eax,%edi
=> 0x0000000000400511 <+45>:	callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:	mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:	mov    $0x0,%eax
   0x000000000040051e <+58>:	leaveq
   0x000000000040051f <+59>:	retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4	int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:	push   %rbp
   0x00000000004004d7 <+1>:	mov    %rsp,%rbp
   0x00000000004004da <+4>:	mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:	mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:	add    %eax,%eax
   0x00000000004004e2 <+12>:	pop    %rbp
   0x00000000004004e3 <+13>:	retq
End of assembler dump.
(gdb) 
```



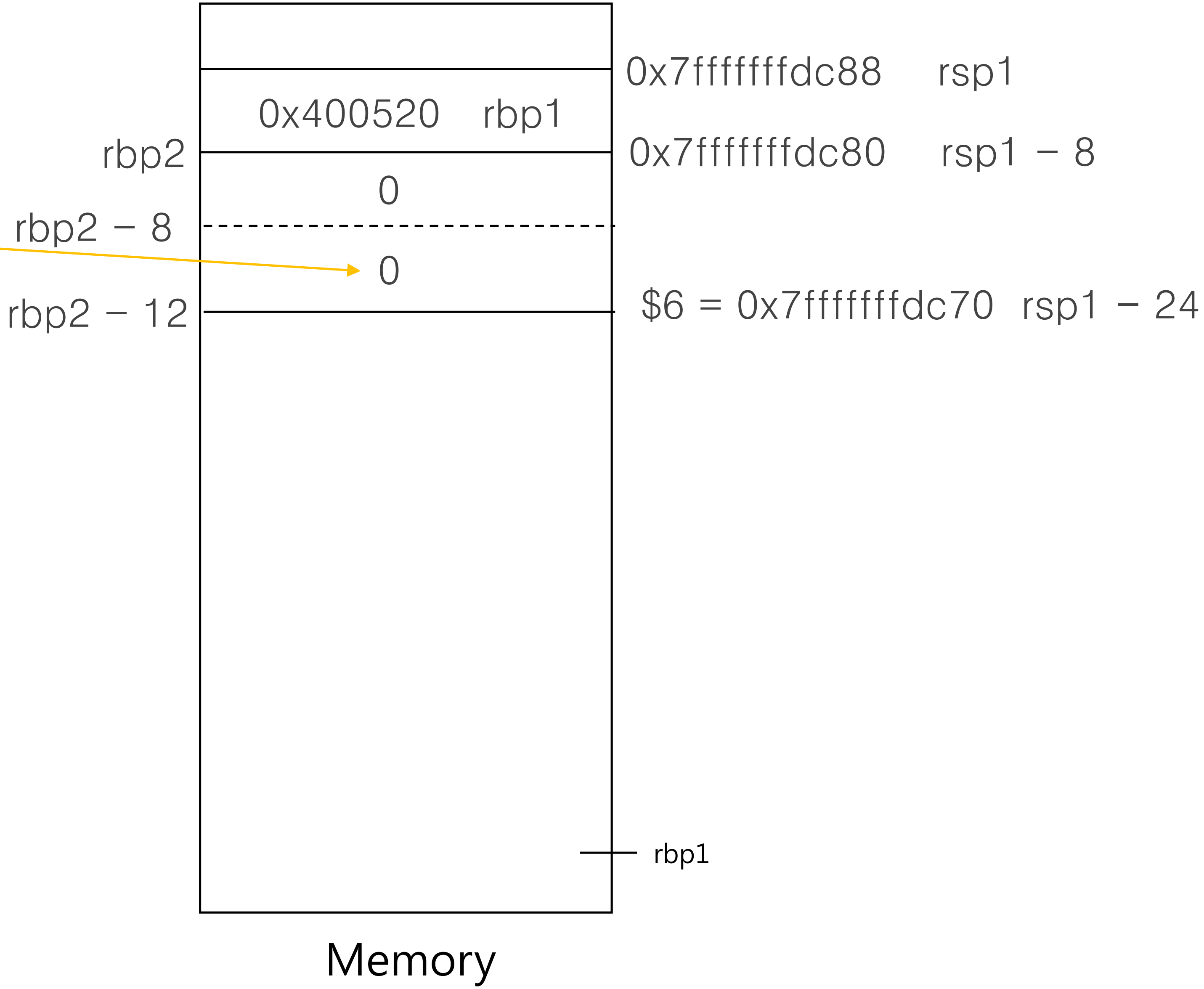
Memory

과제 7 - 5 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

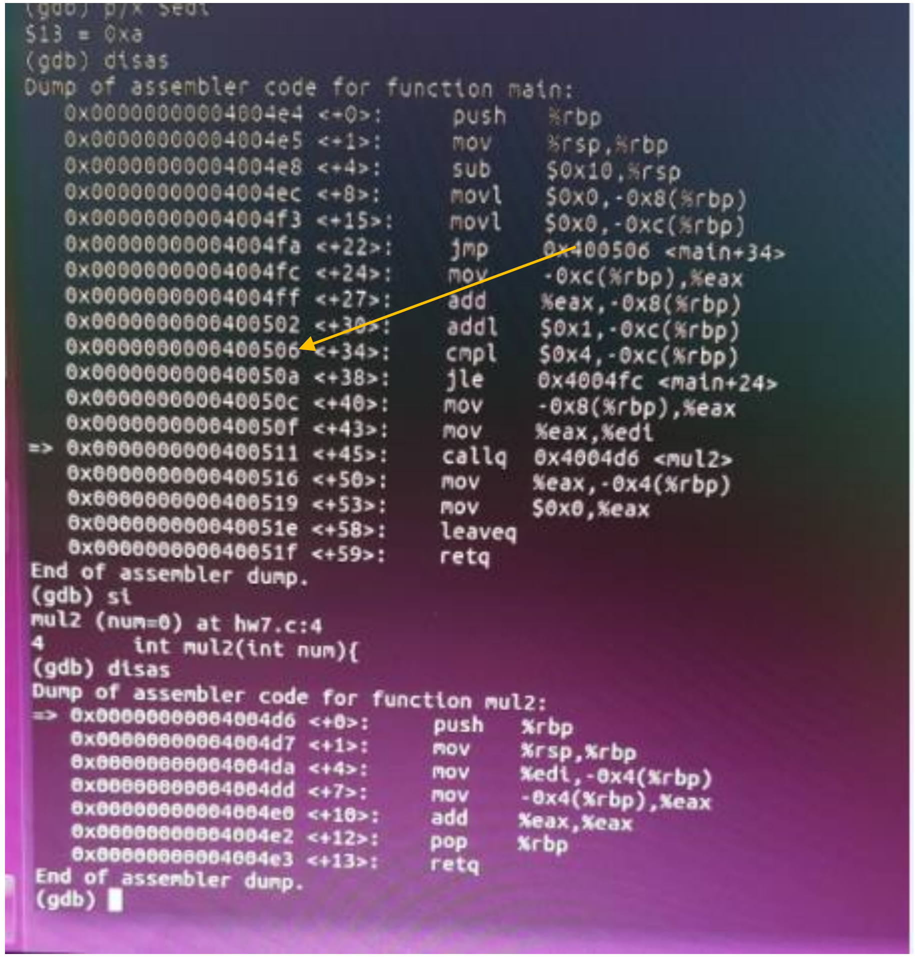


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

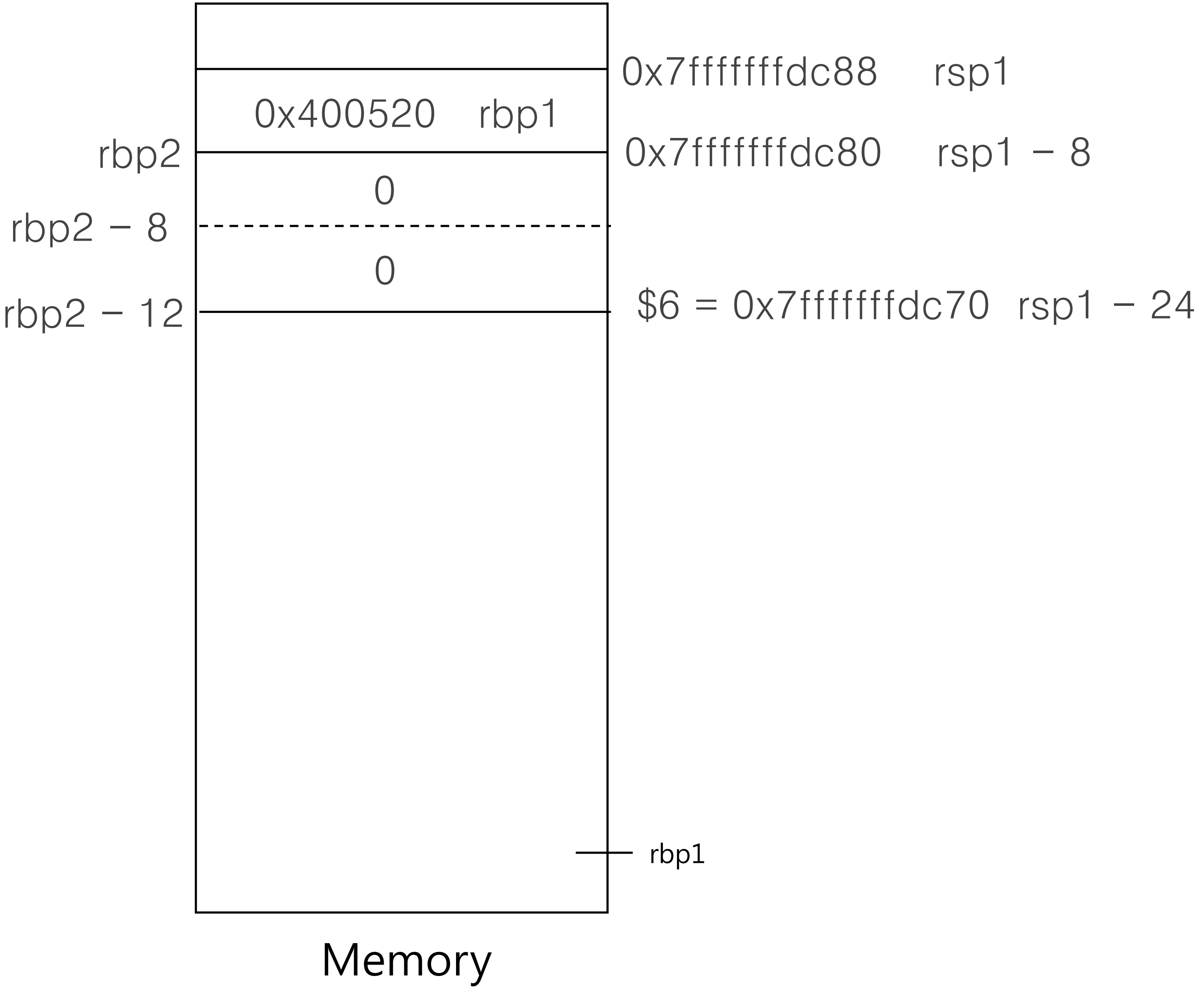


과제 7 - 6 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
    이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
    esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
    메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

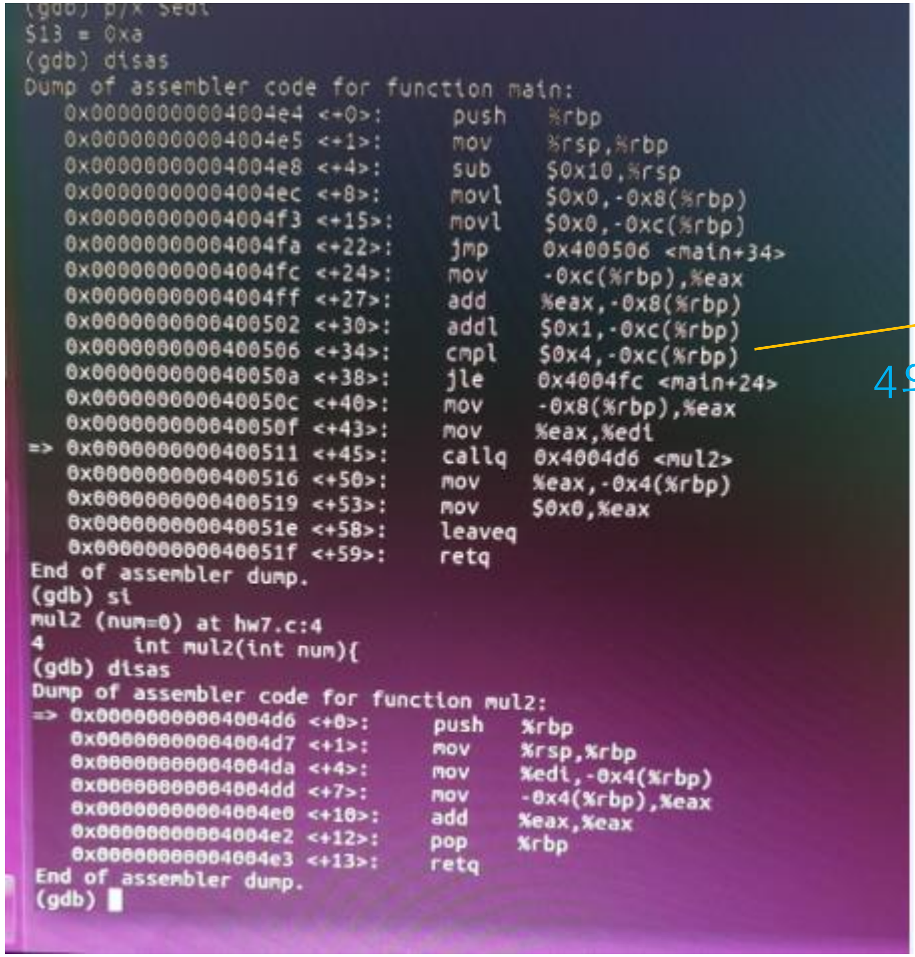


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

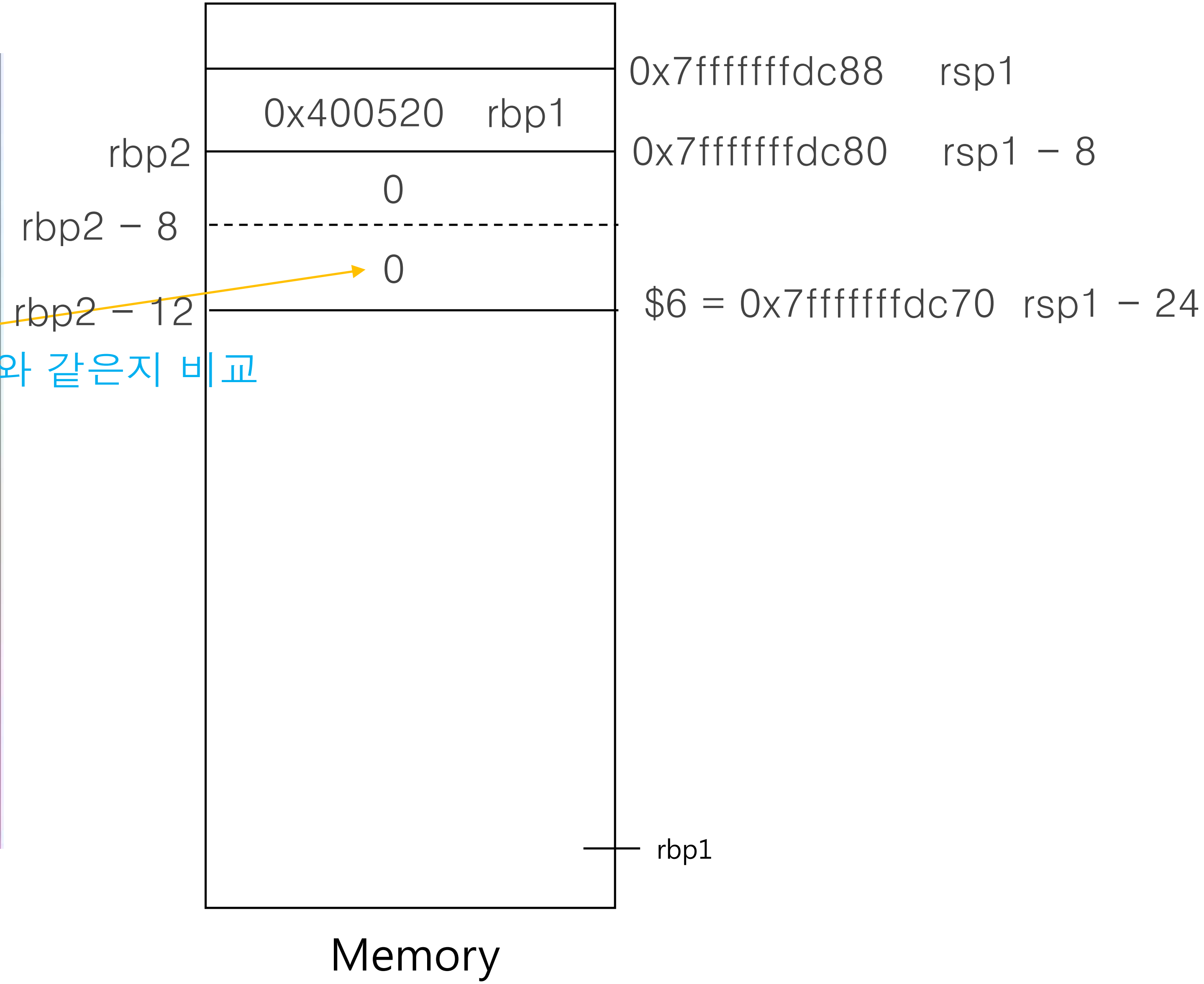


과제 7 - 7 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



0x7fffffffdc88 rsp1

0x400520 rbp1

rbp2

0x7fffffffdc80 rsp1 – 8

0

rbp2 – 8

0

rbp2 – 12

$6 = 0x7fffffffdc70 rsp1 – 24

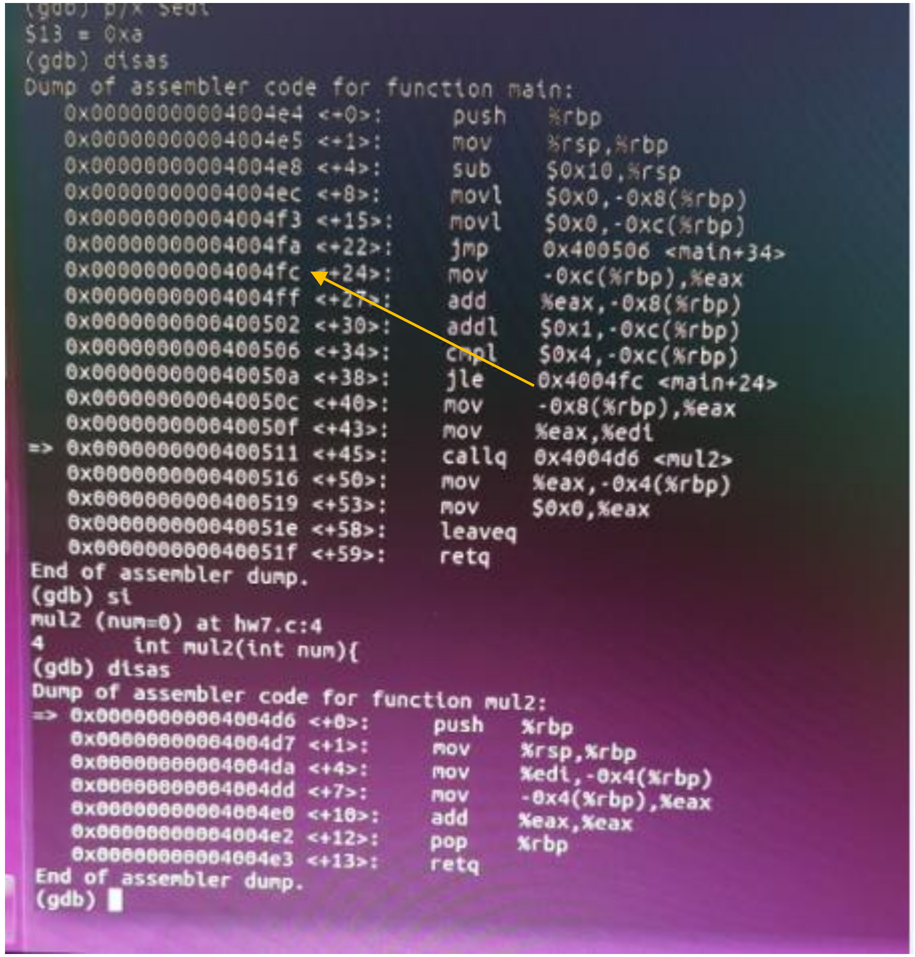
4와 같은지 비교

rbp1

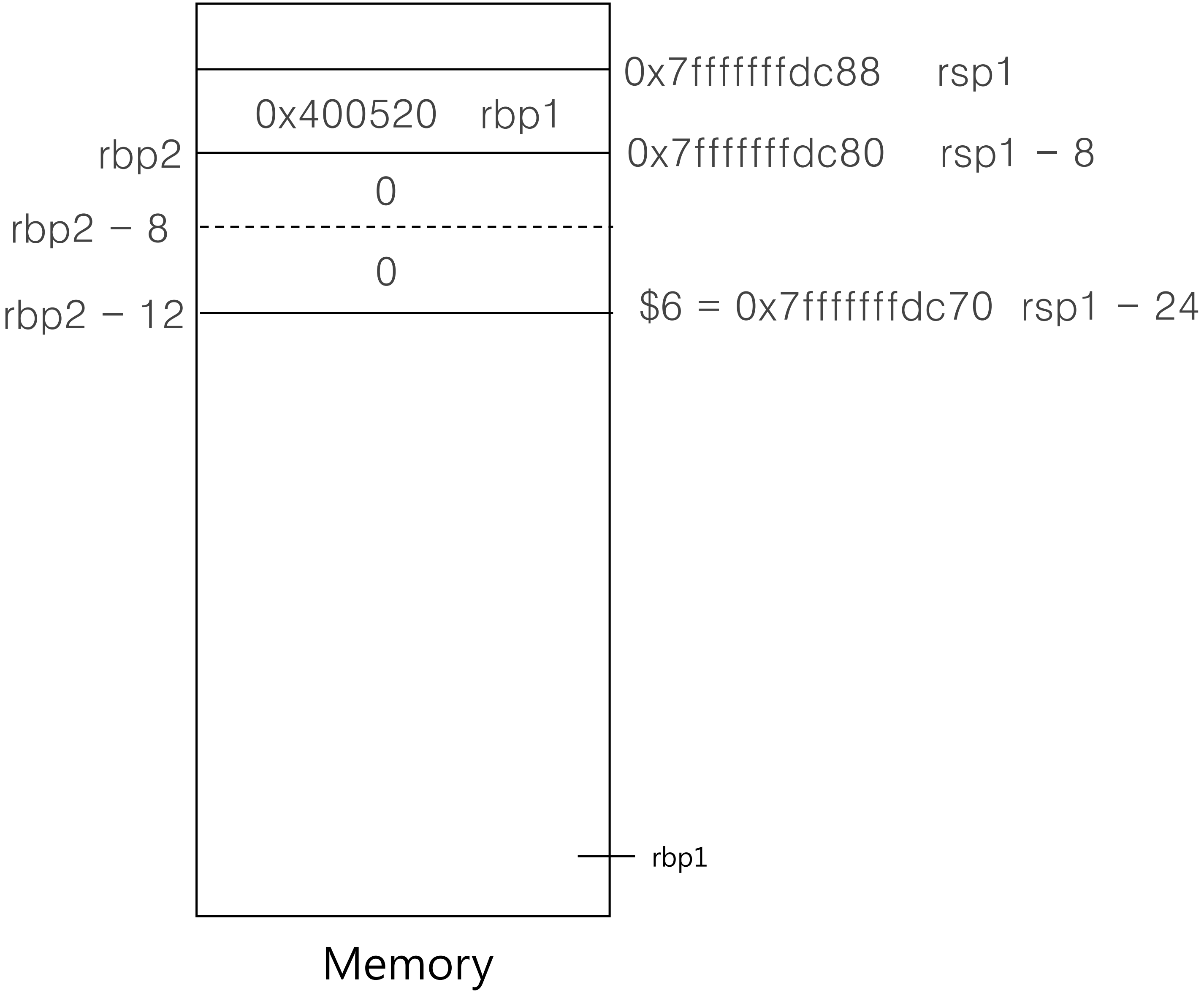
Memory

과제 7 - 8 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

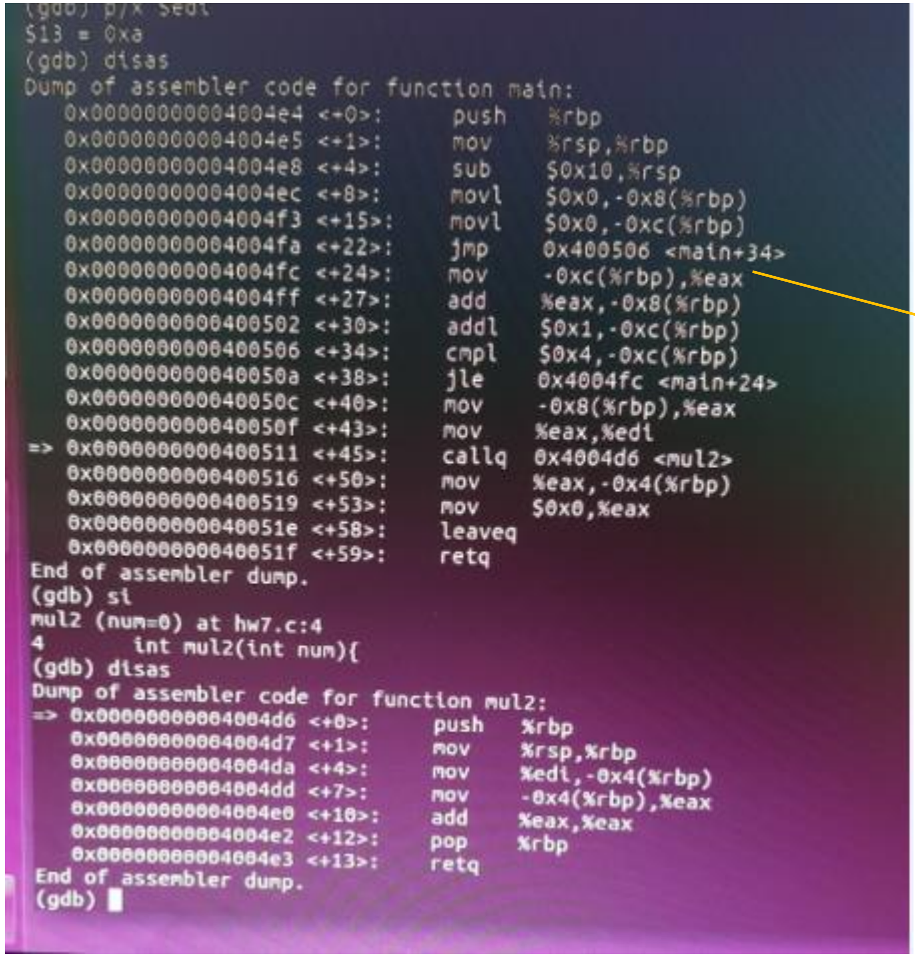


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

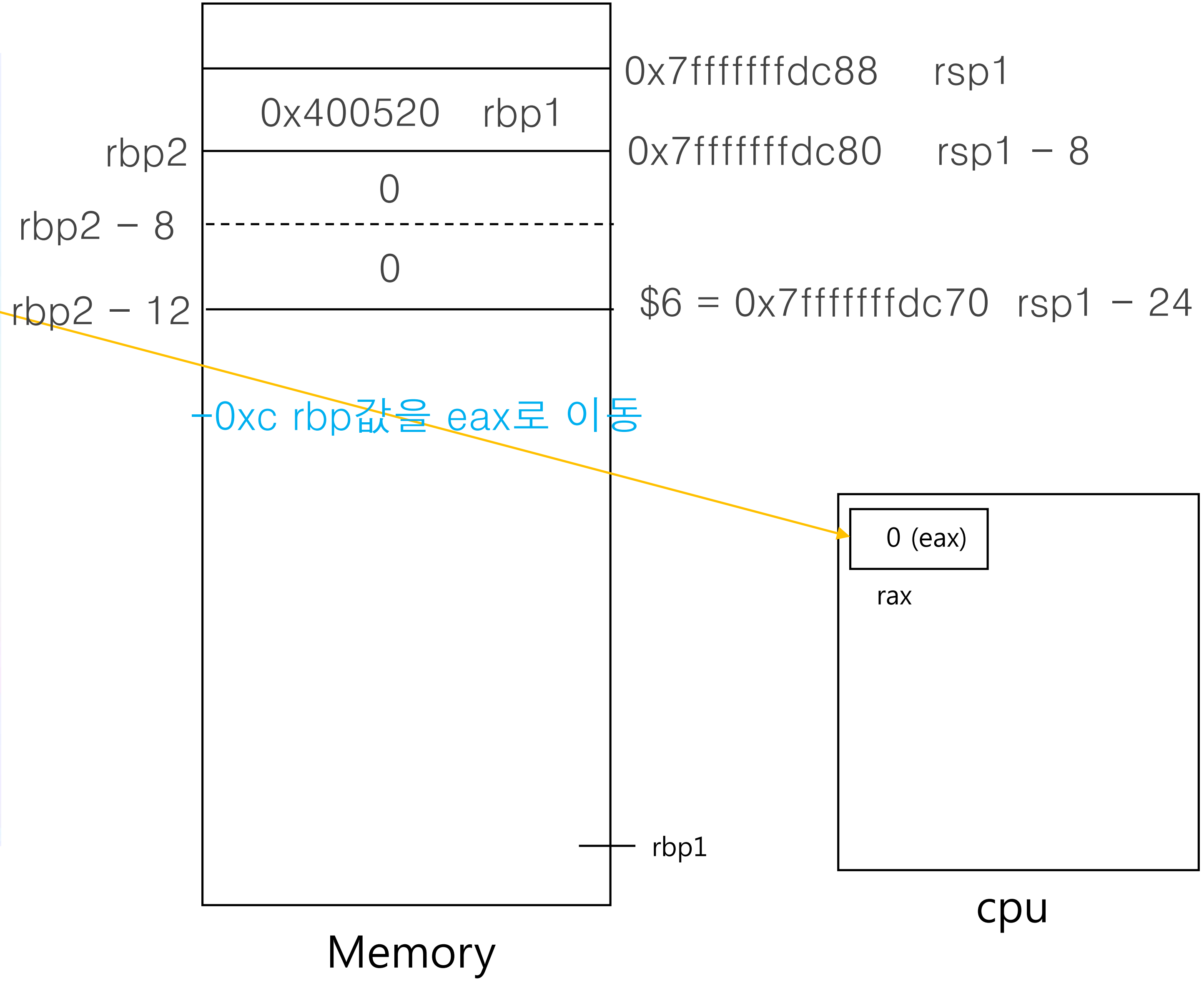


과제 7 - 9 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

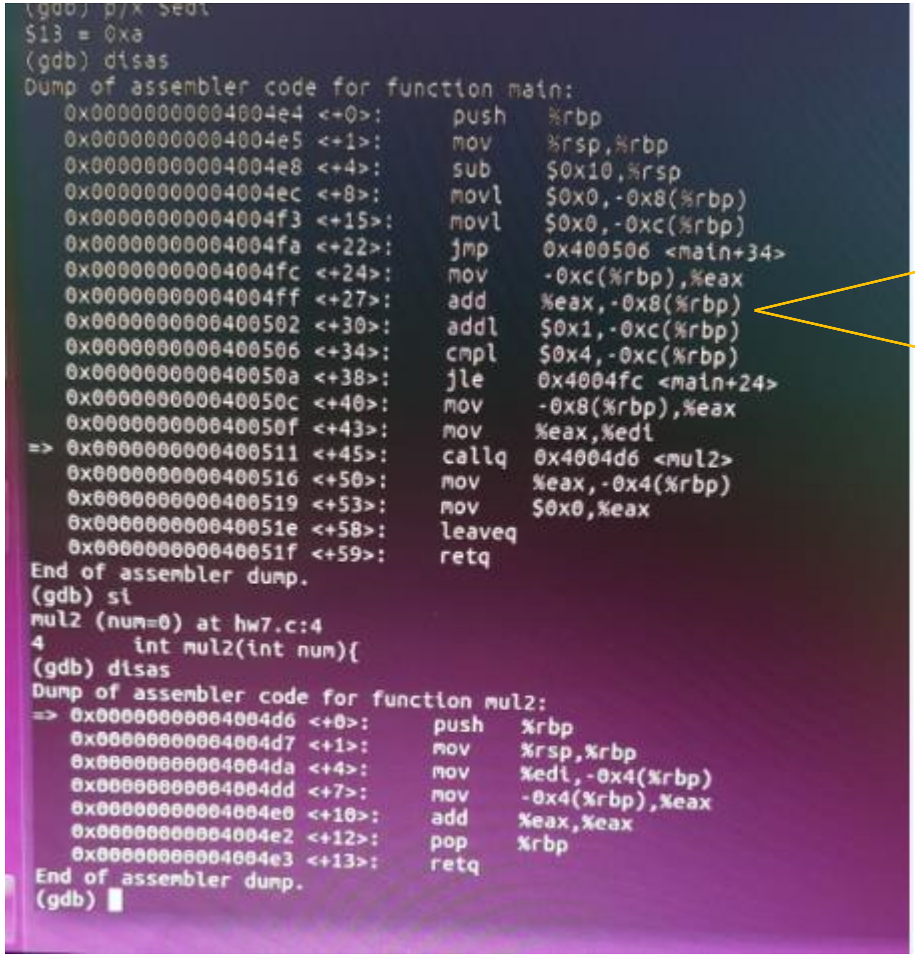


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

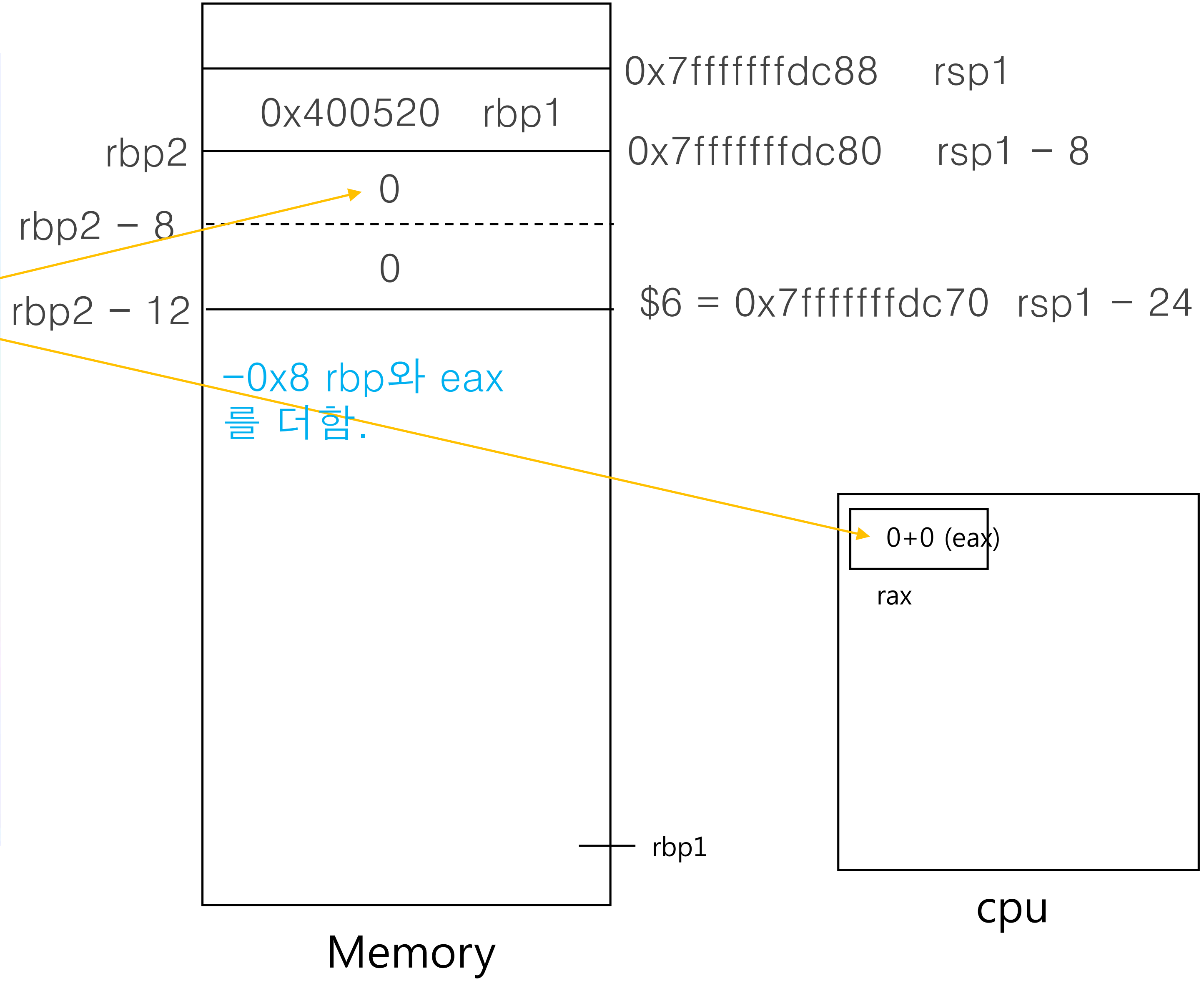


과제 7 - 10 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



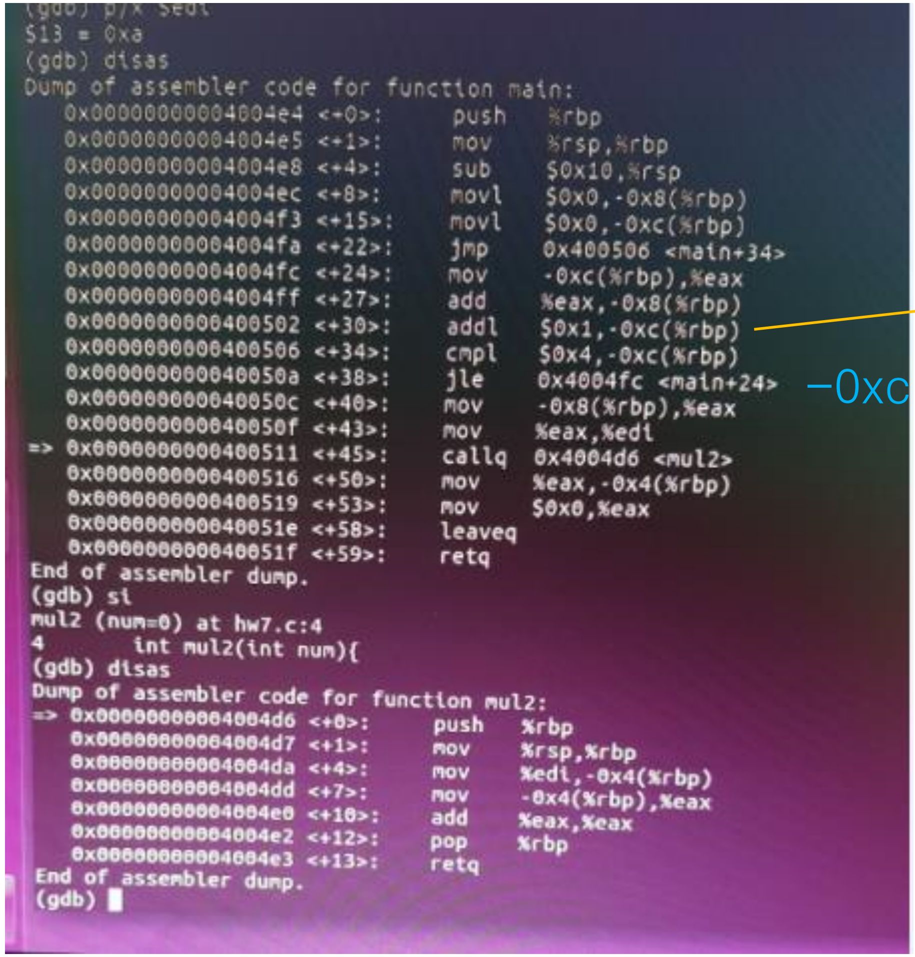
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



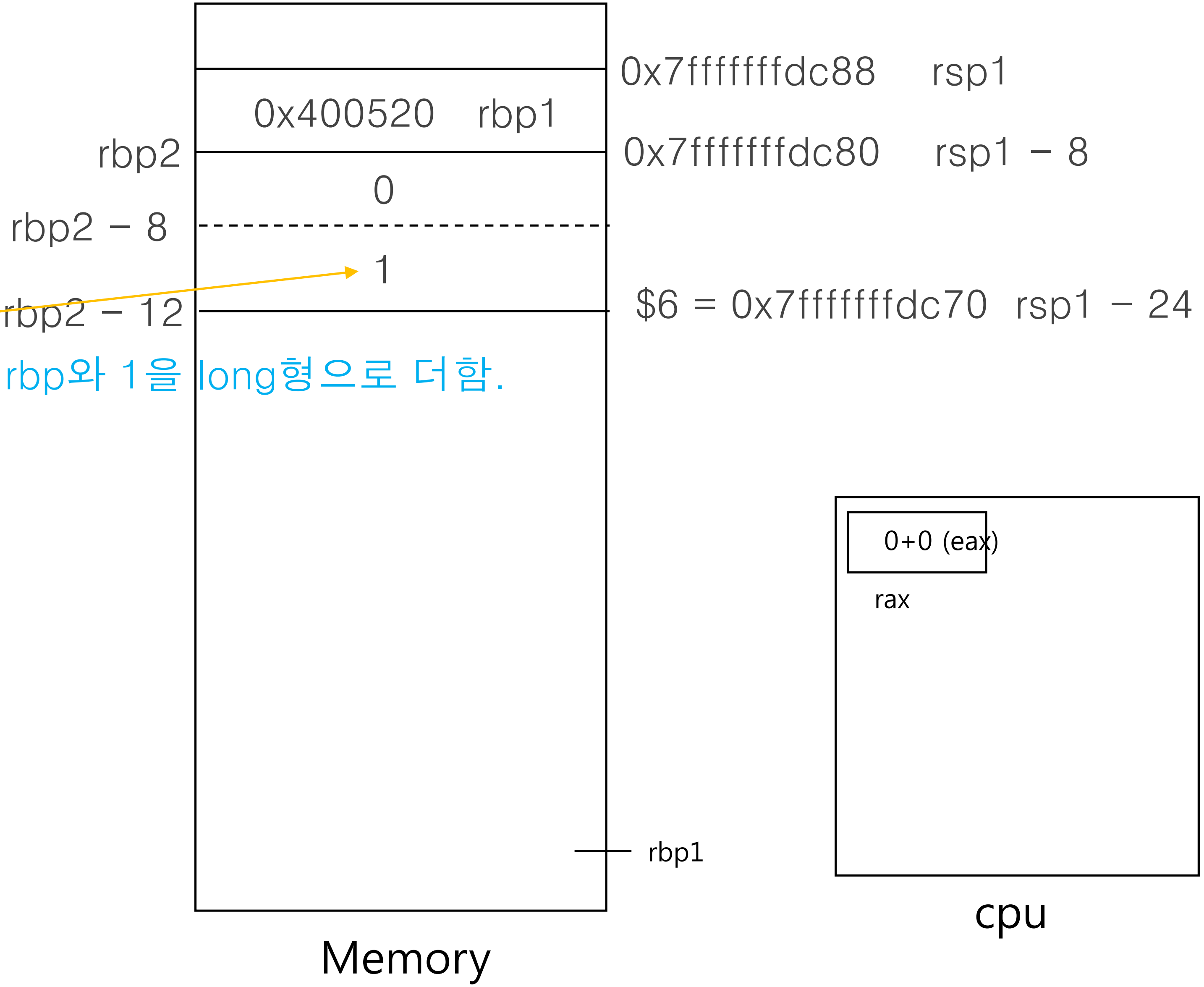
–0x8 rbp와 eax 를 더함.

과제 7 - 11 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



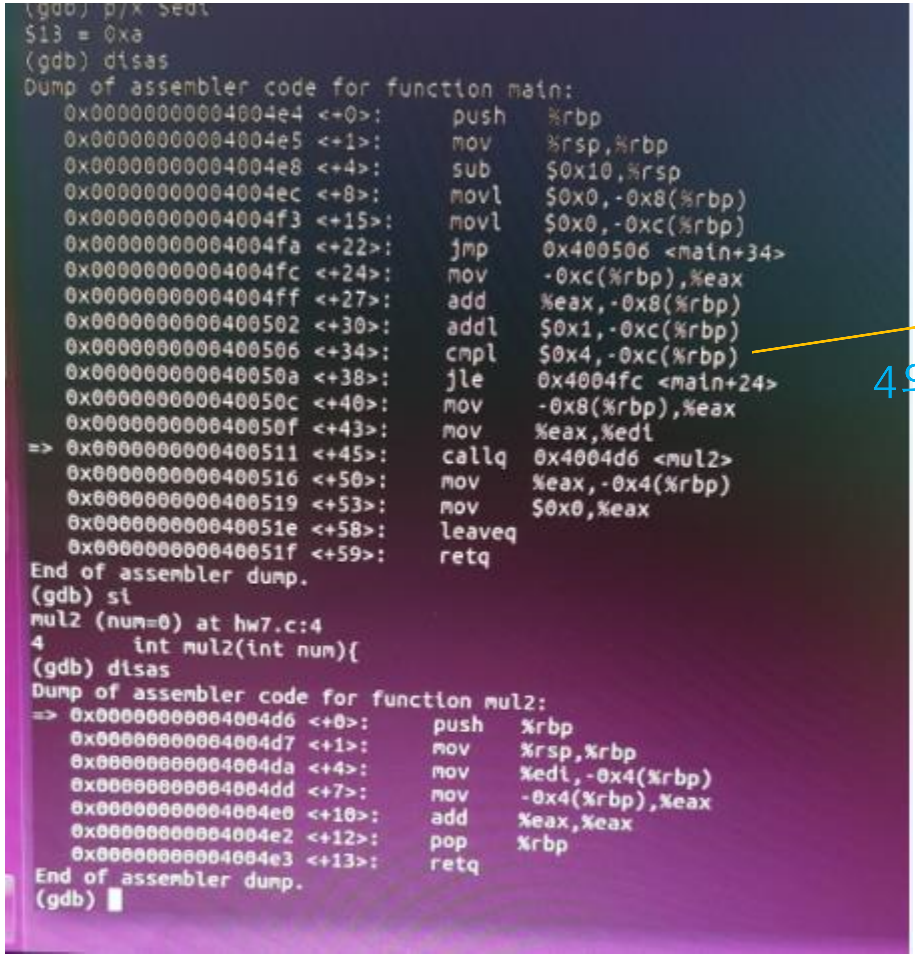
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:	push   %rbp
   0x00000000004004e5 <+1>:	mov    %rsp,%rbp
   0x00000000004004e8 <+4>:	sub    $0x10,%rsp
   0x00000000004004ec <+8>:	movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:	movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:	jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:	mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:	add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:	addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:	cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:	jle    0x4004fc <main+24>
   0x000000000040050c <+40>:	mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:	mov    %eax,%edi
=> 0x0000000000400511 <+45>:	callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:	mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:	mov    $0x0,%eax
   0x000000000040051e <+58>:	leaveq
   0x000000000040051f <+59>:	retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4	    int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:	push   %rbp
   0x00000000004004d7 <+1>:	mov    %rsp,%rbp
   0x00000000004004da <+4>:	mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:	mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:	add    %eax,%eax
   0x00000000004004e2 <+12>:	pop    %rbp
   0x00000000004004e3 <+13>:	retq
End of assembler dump.
(gdb) 
```



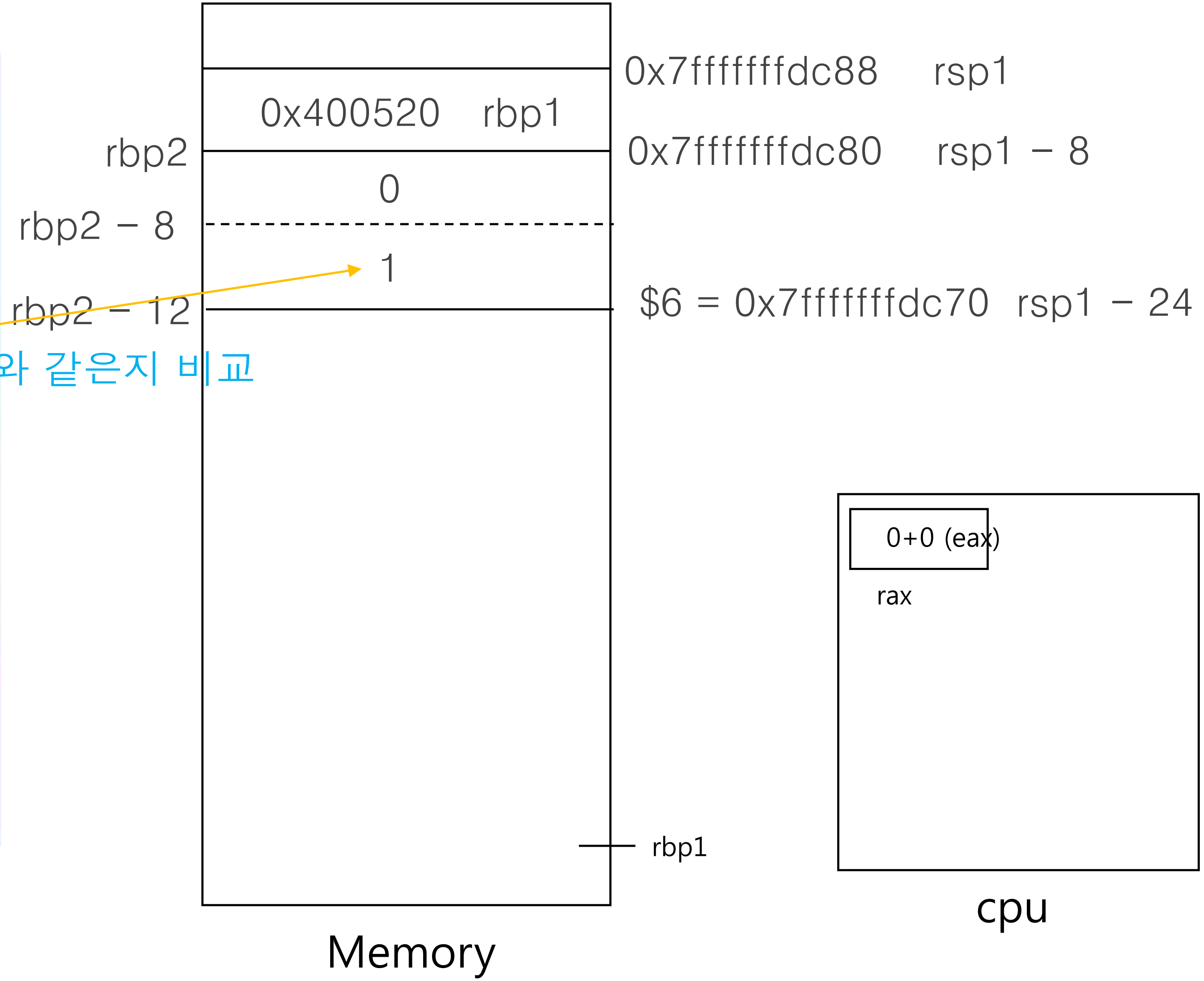
−0xc rbp와 1을 long형으로 더함.

과제 7 - 12 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

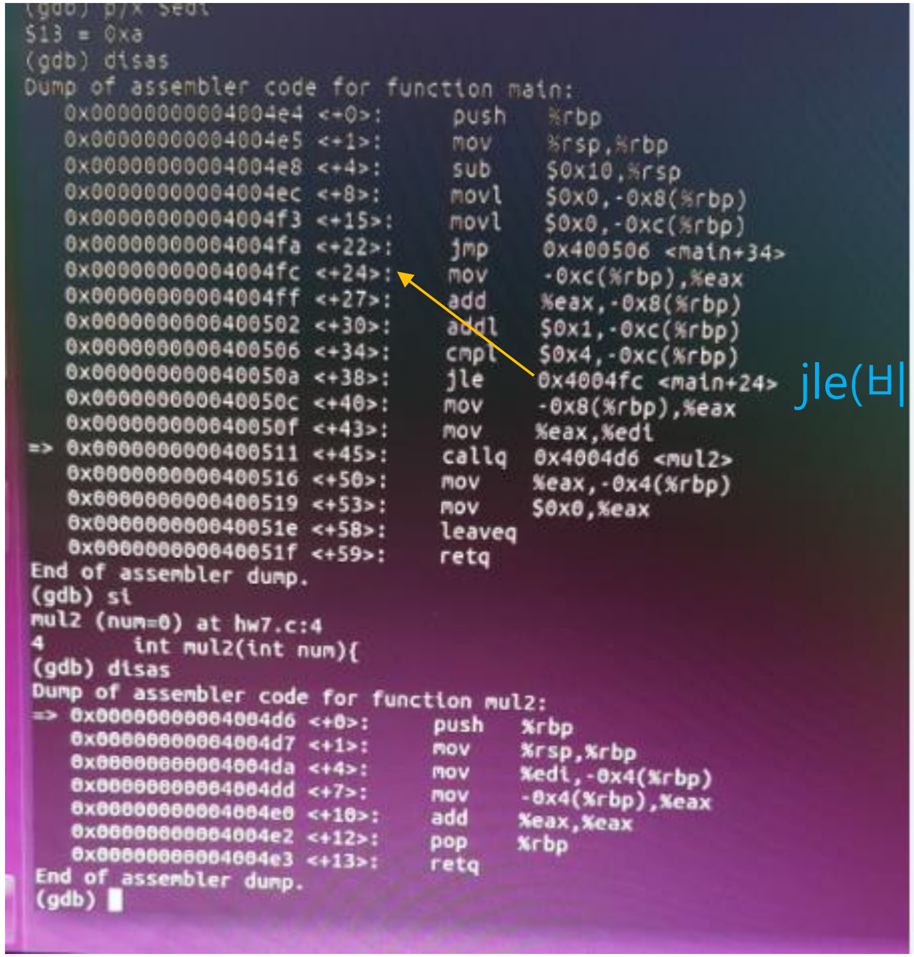


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



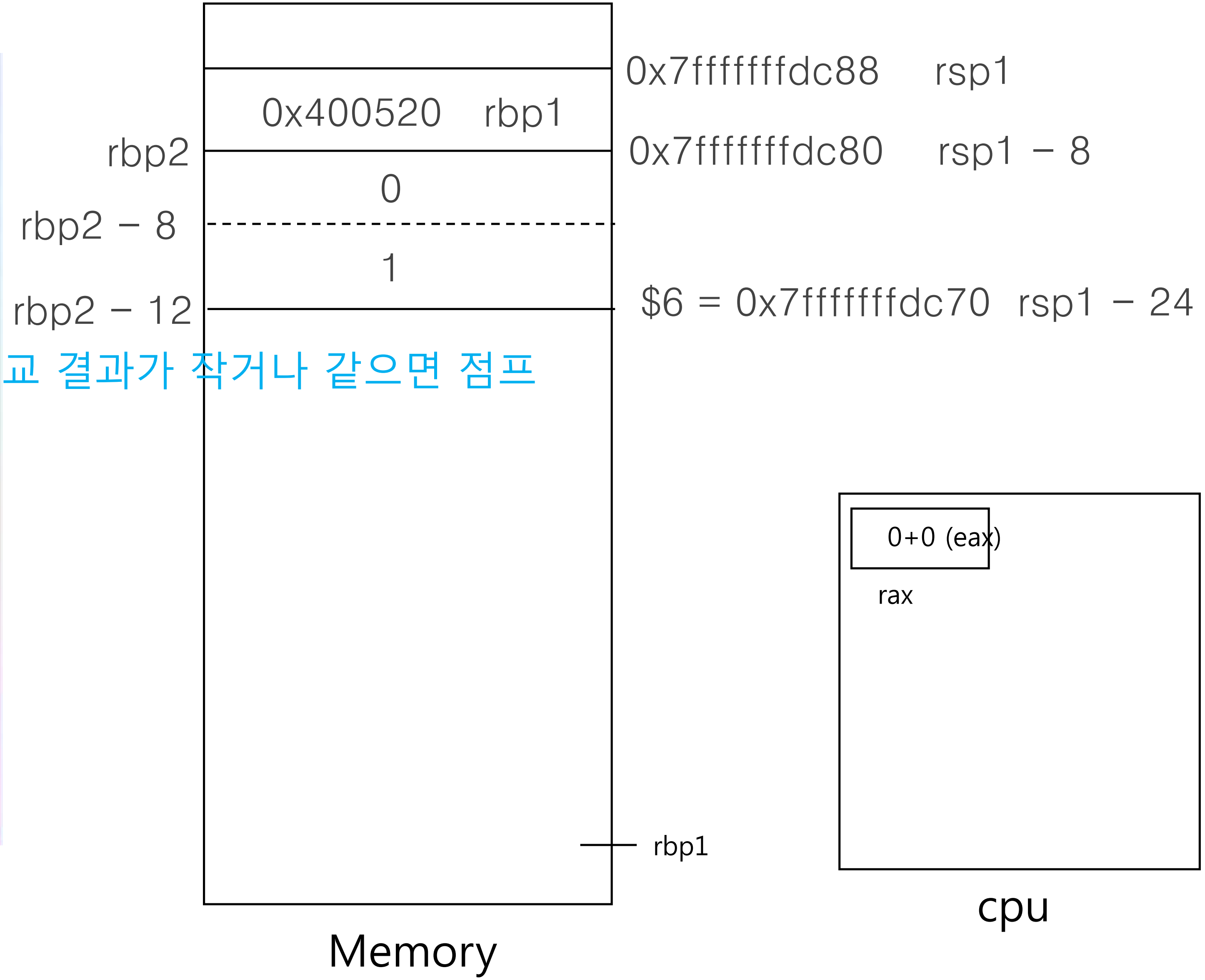
과제 7 - 13 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



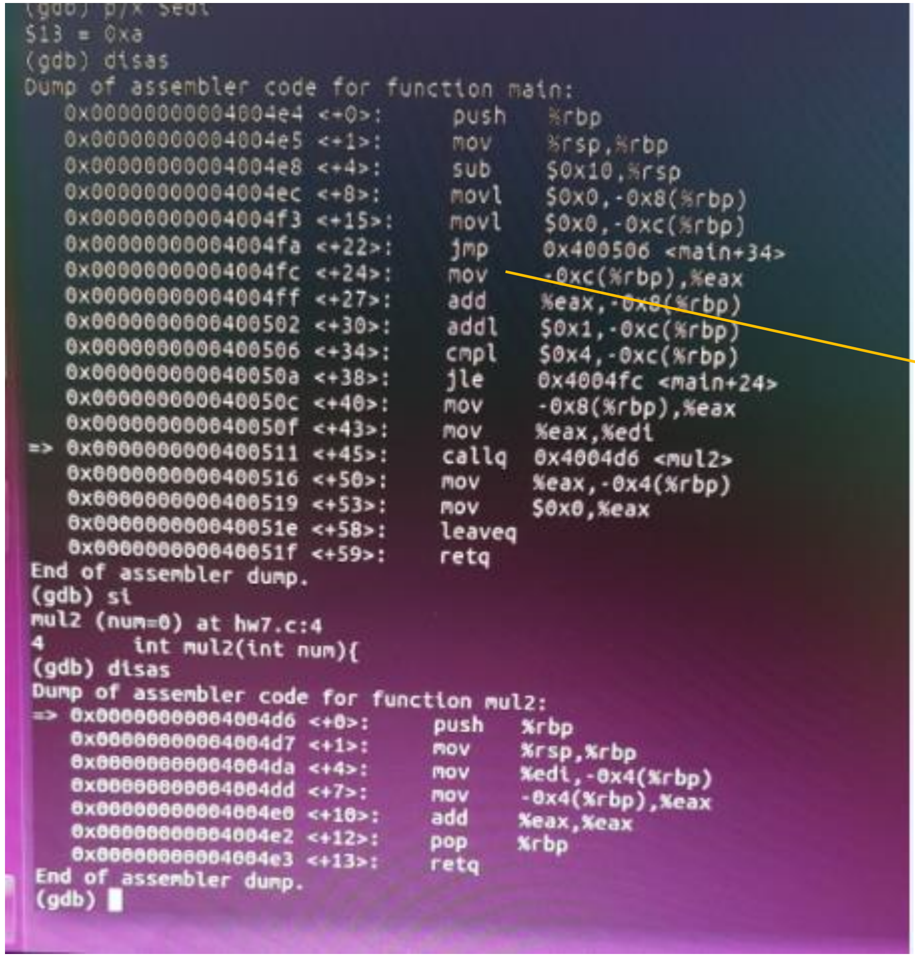
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb)
```

jle(비교 결과가 작거나 같으면 점프

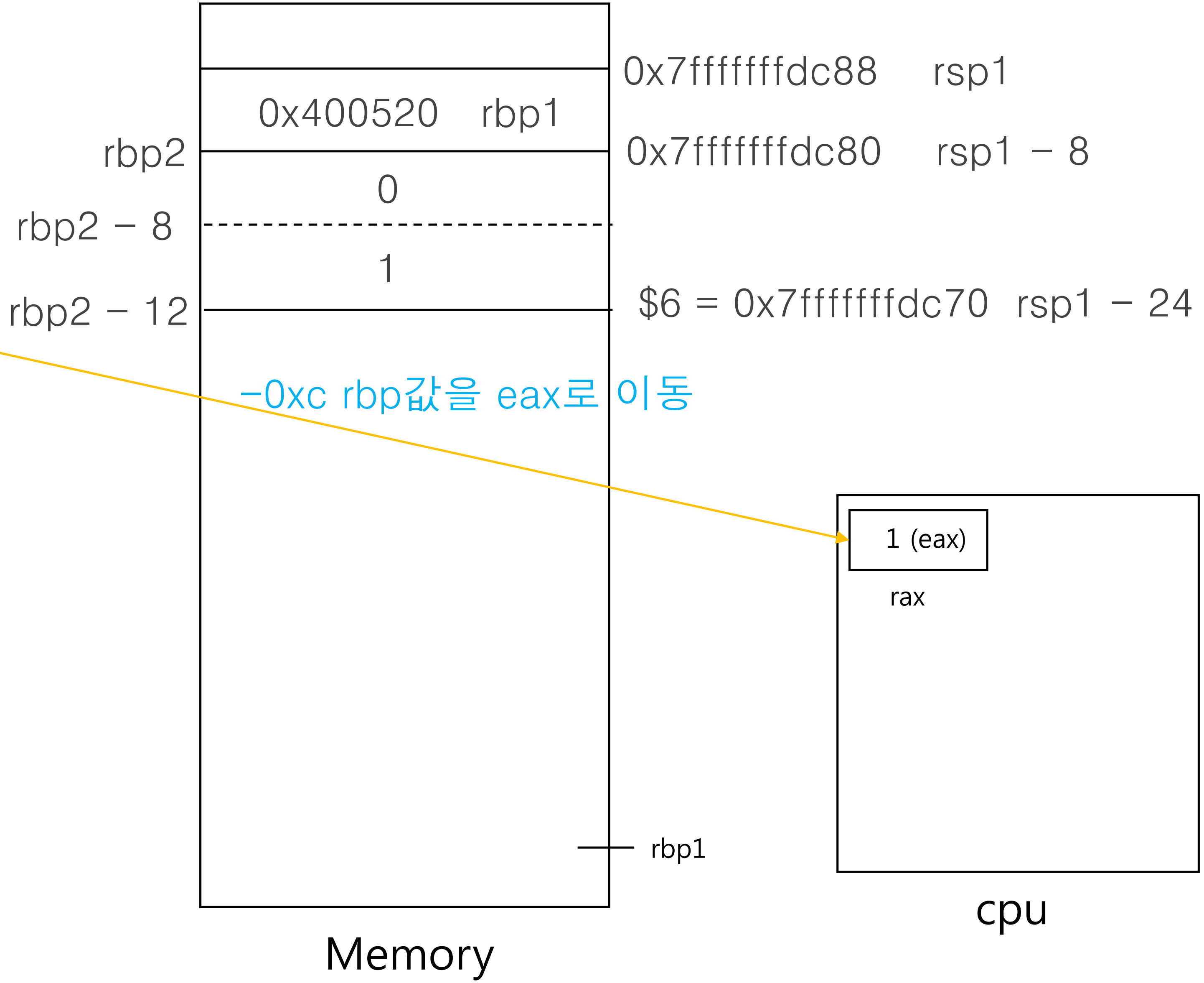


과제 7 - 14 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

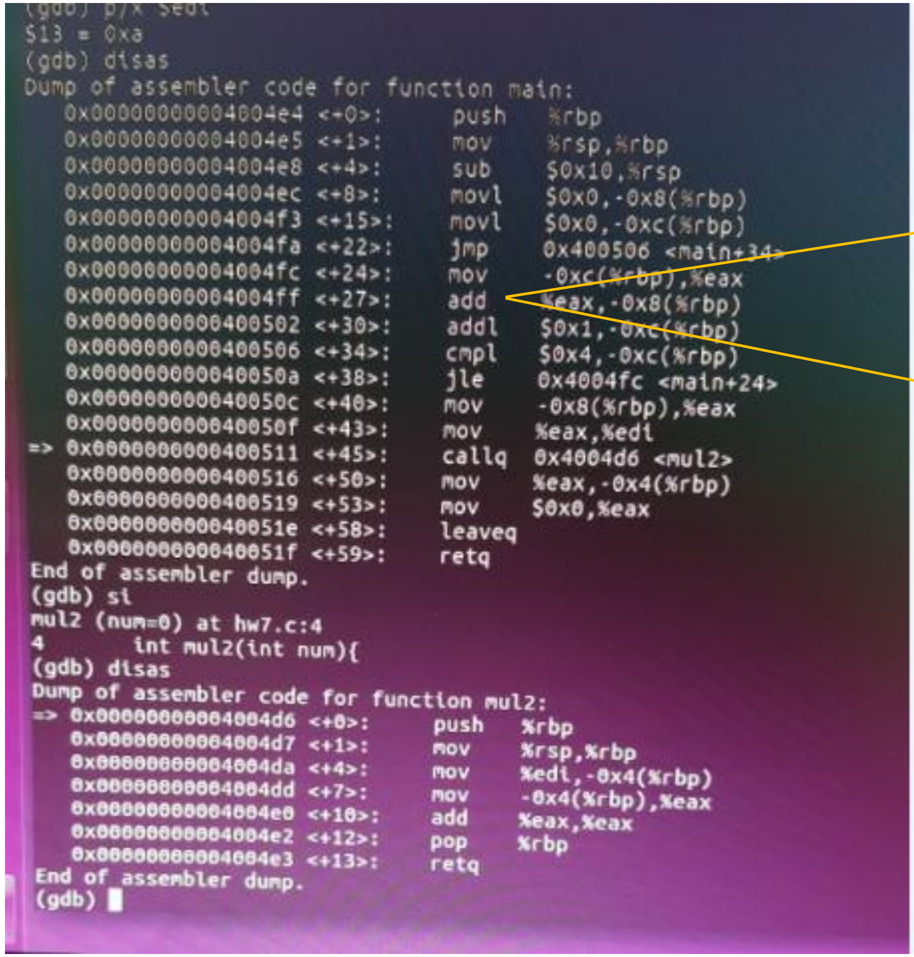


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

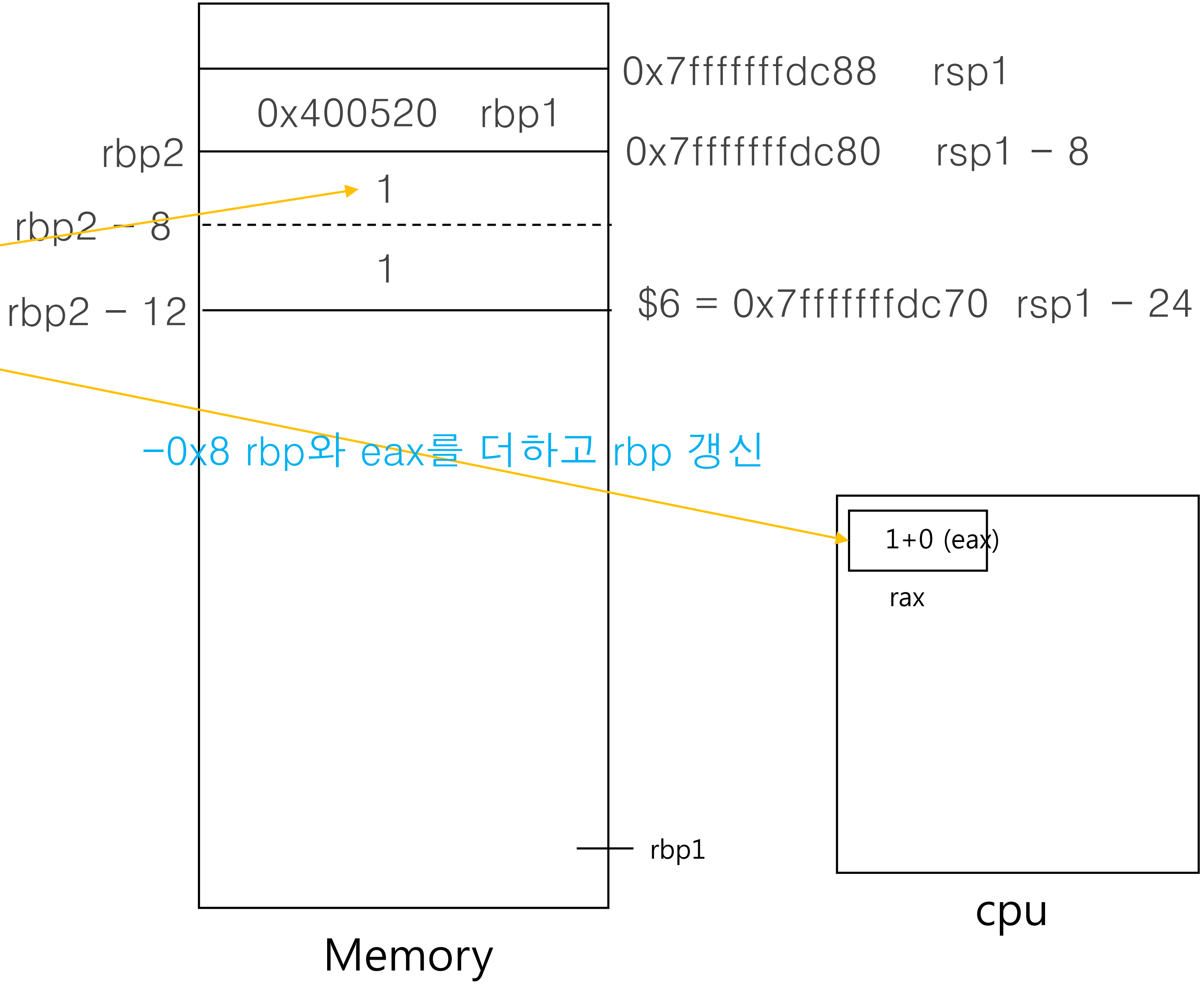


과제 7 - 15 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

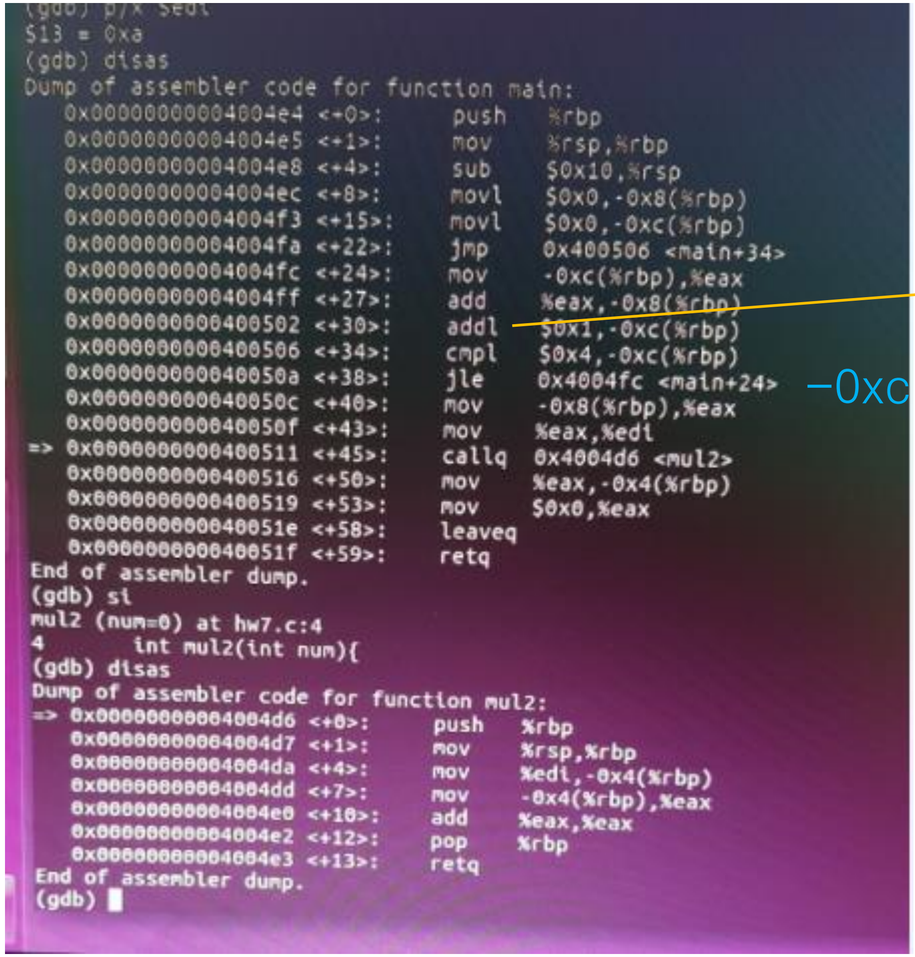


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

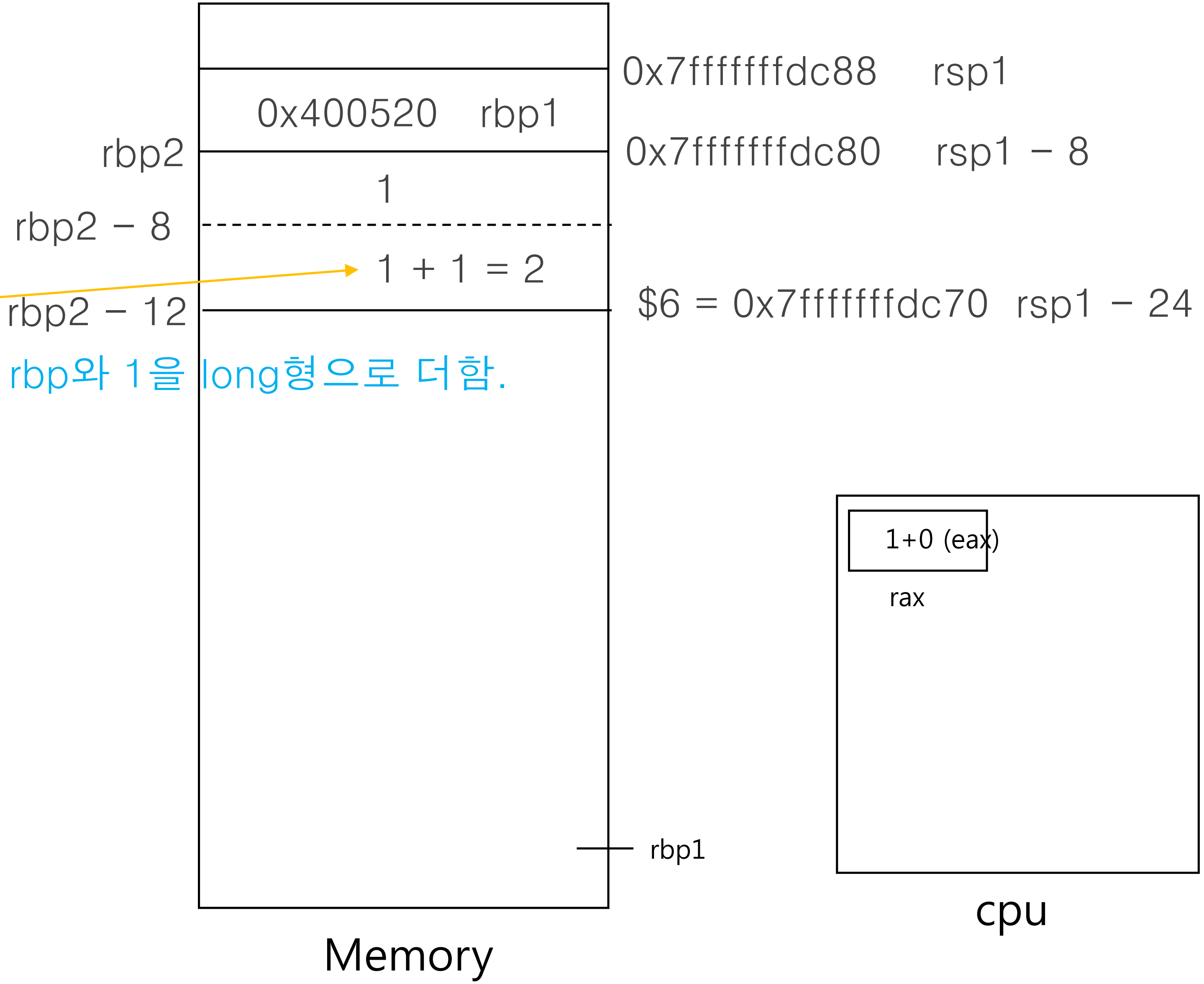


과제 7 - 16 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



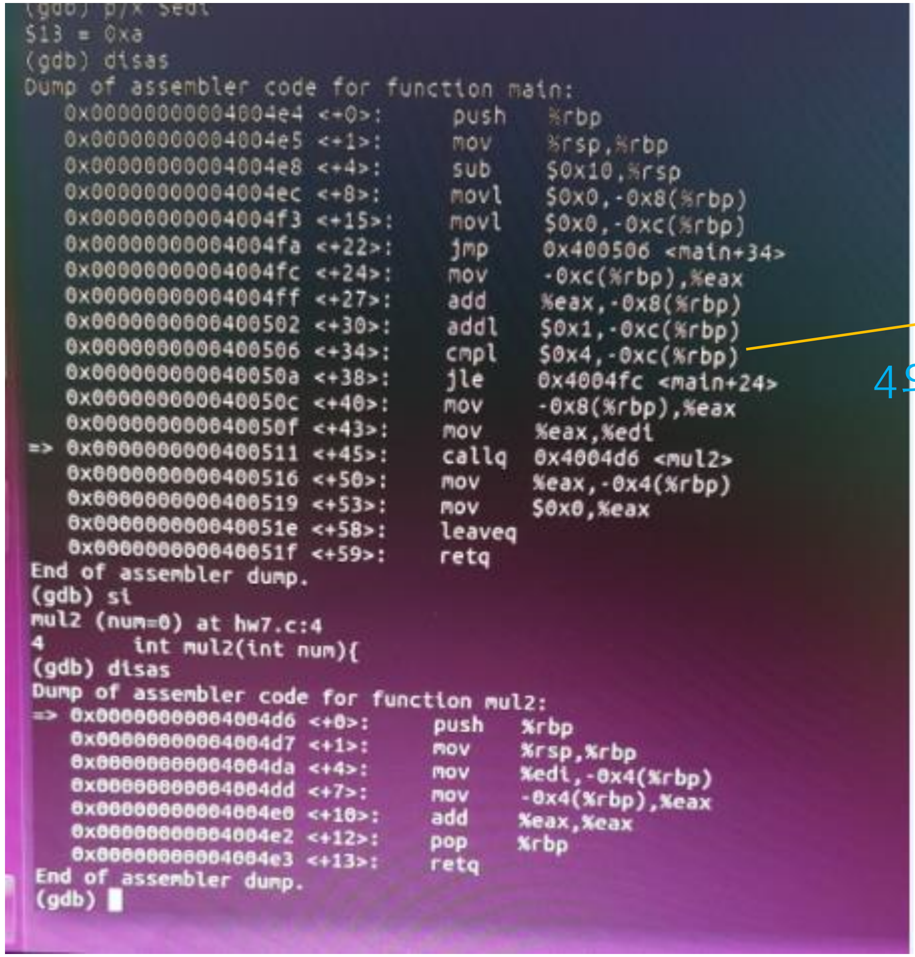
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



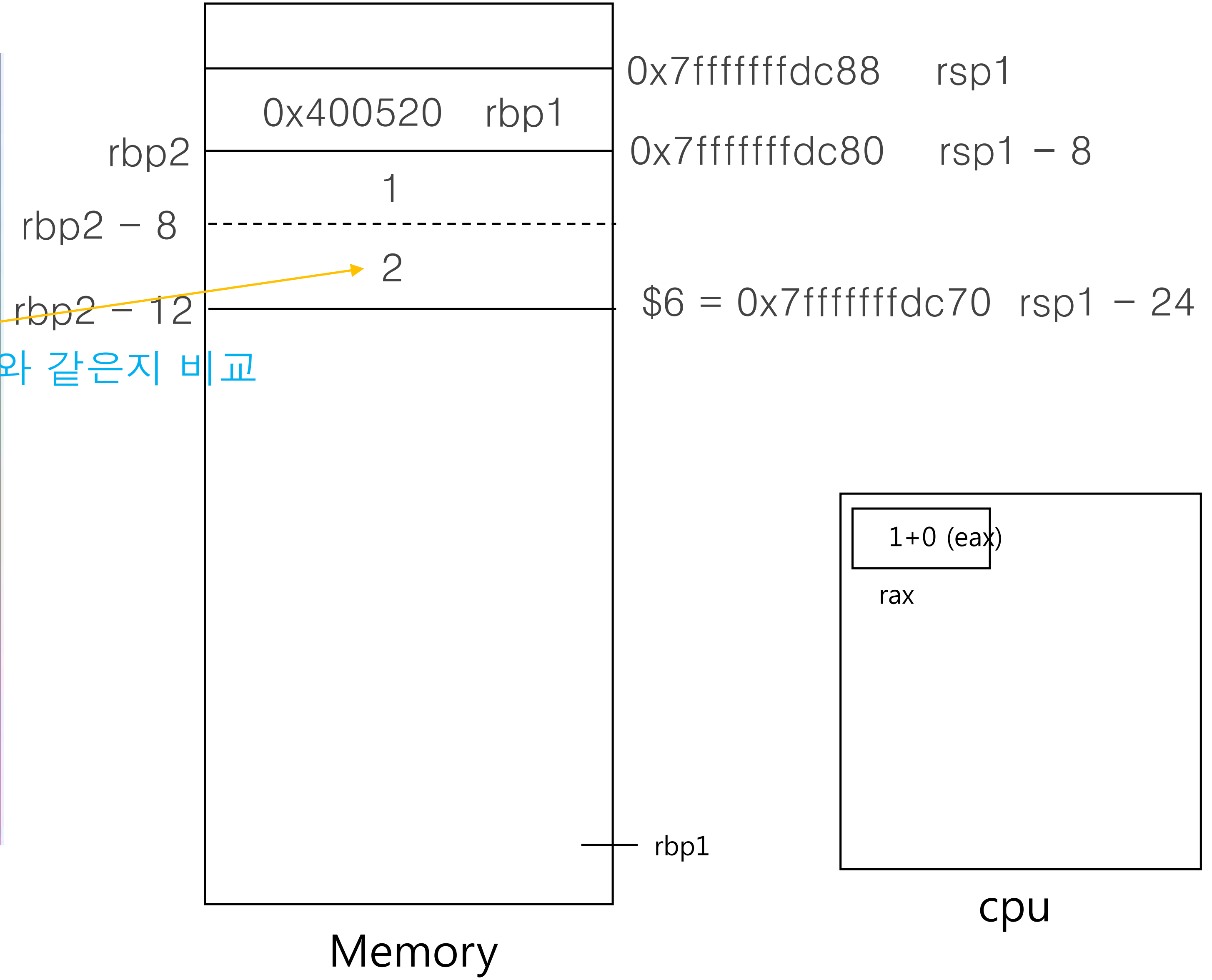
−0xc rbp와 1을 long형으로 더함.

과제 7 - 17 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

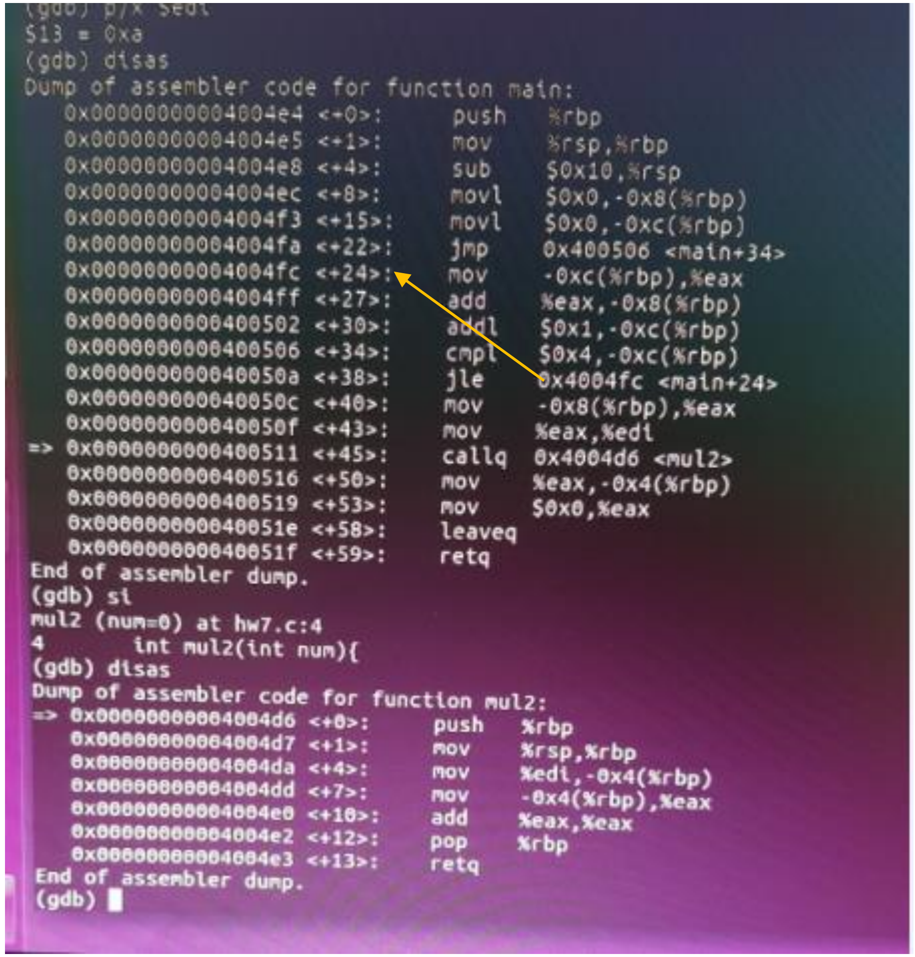


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

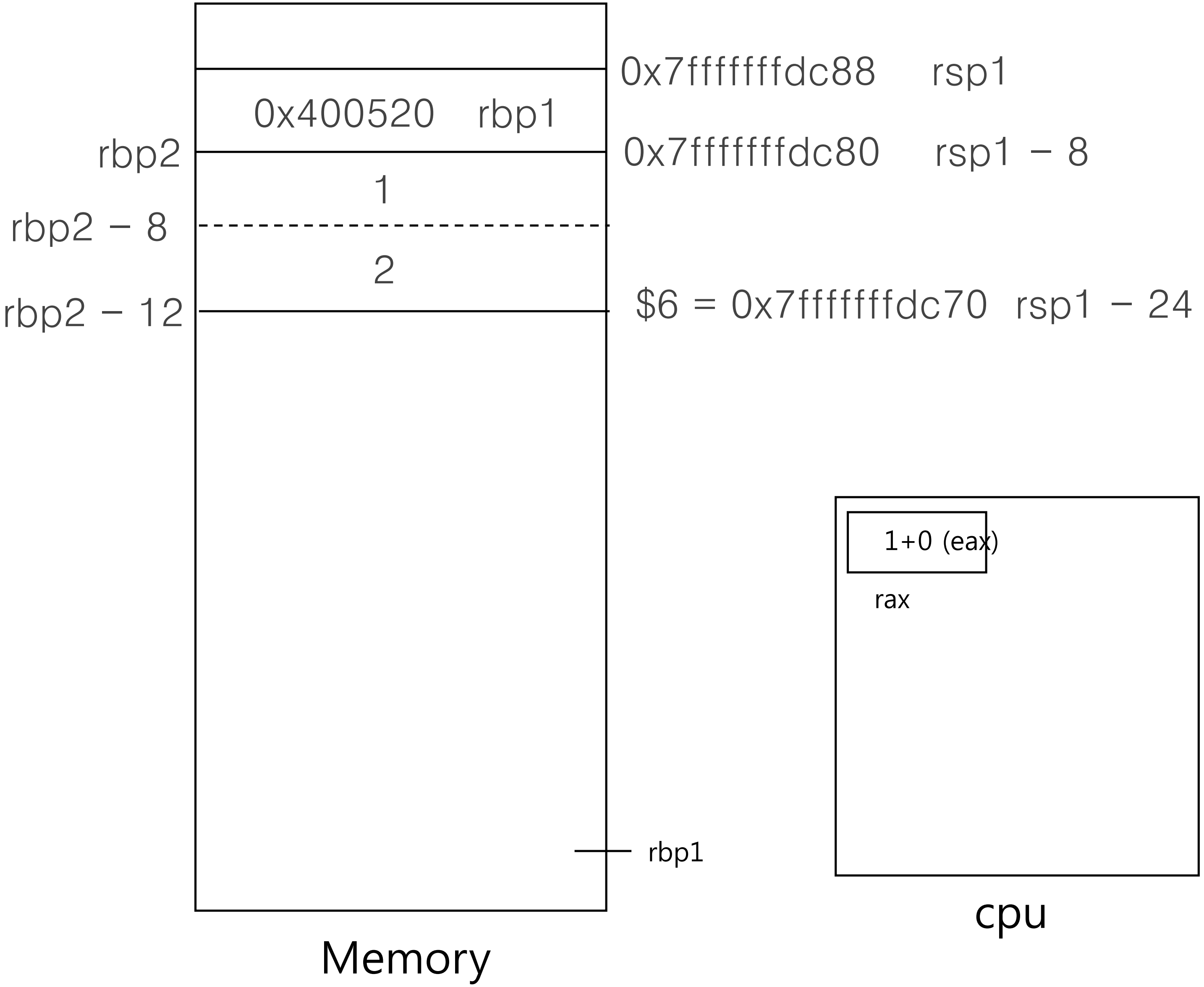


과제 7 - 18 - −0xc rbp가 5가 될 때까지 반복

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

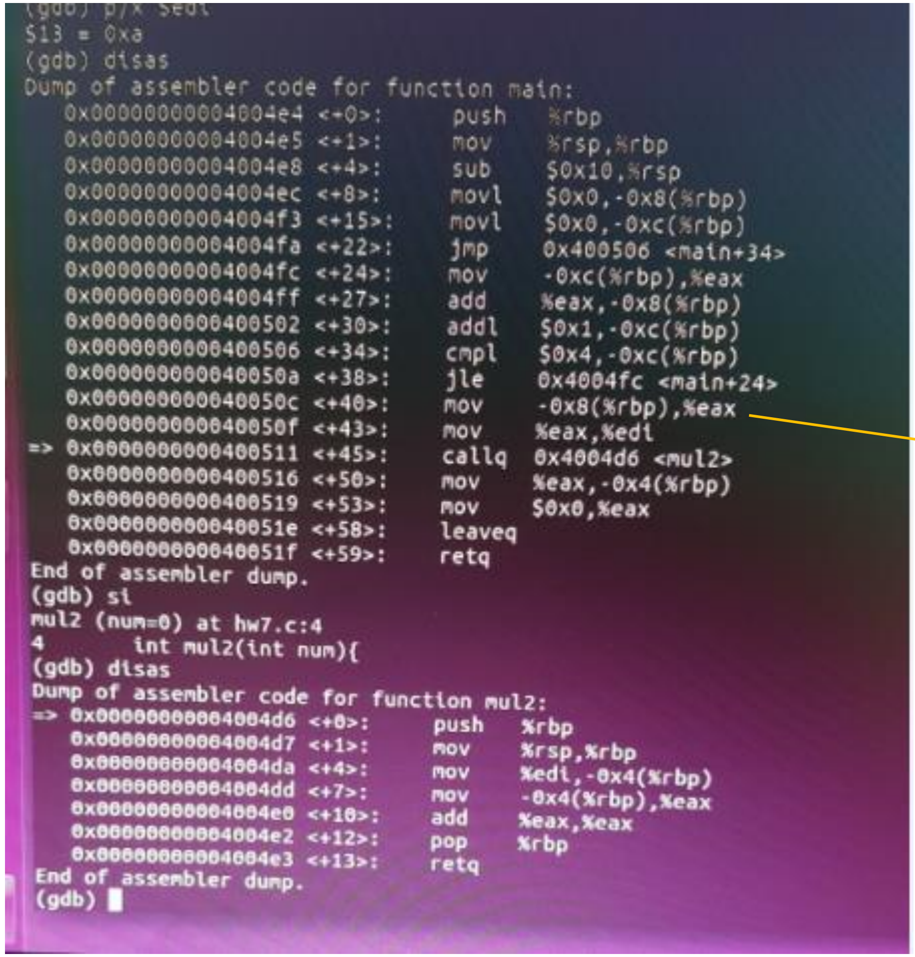


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

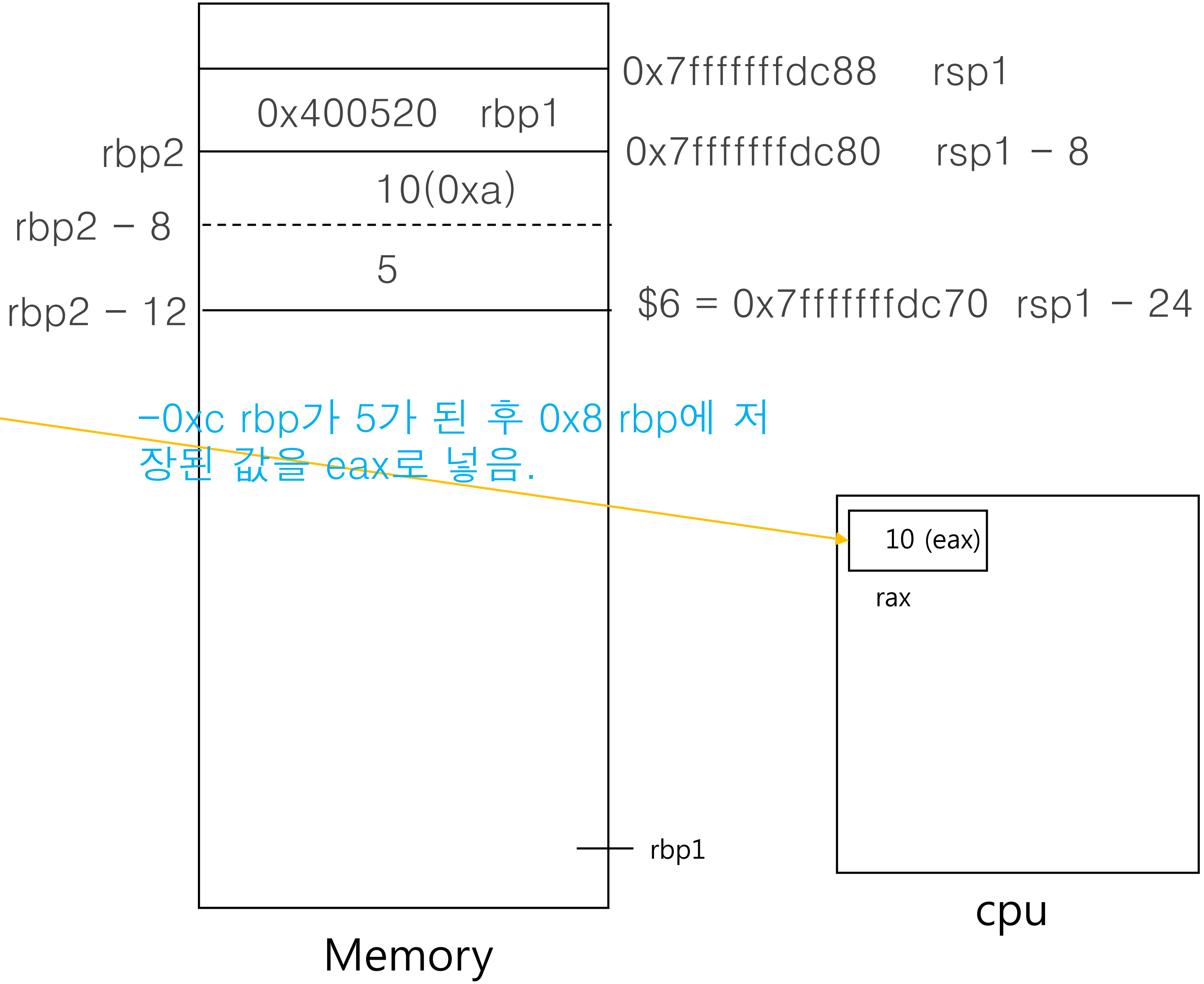


과제 7 - 19 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



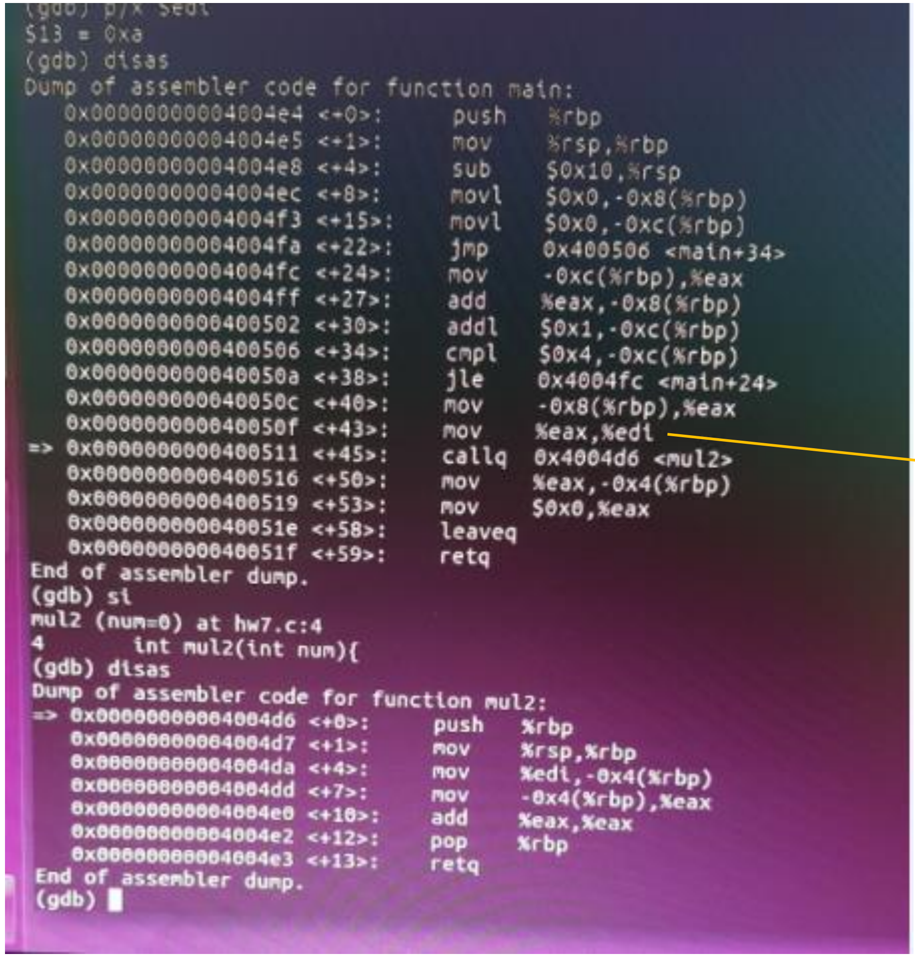
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



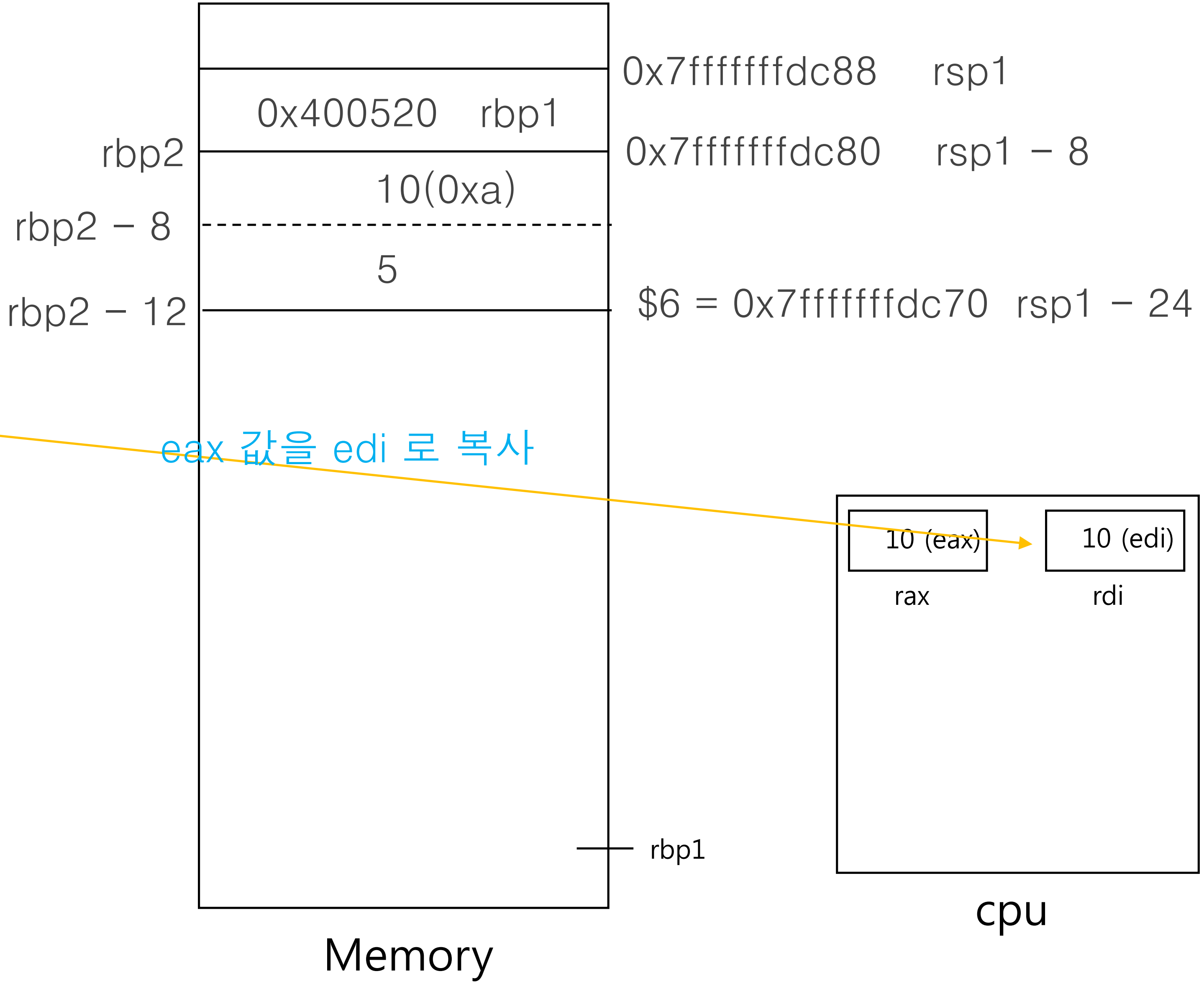
–0xc rbp가 5가 된 후 0x8 rbp에 저장된 값을 eax로 넣음.

과제 7 - 20 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) █
```



0x7fffffffdc88 rsp1

0x400520 rbp1

rbp2

0x7fffffffdc80 rsp1 – 8

10(0xa)

rbp2 – 8

5

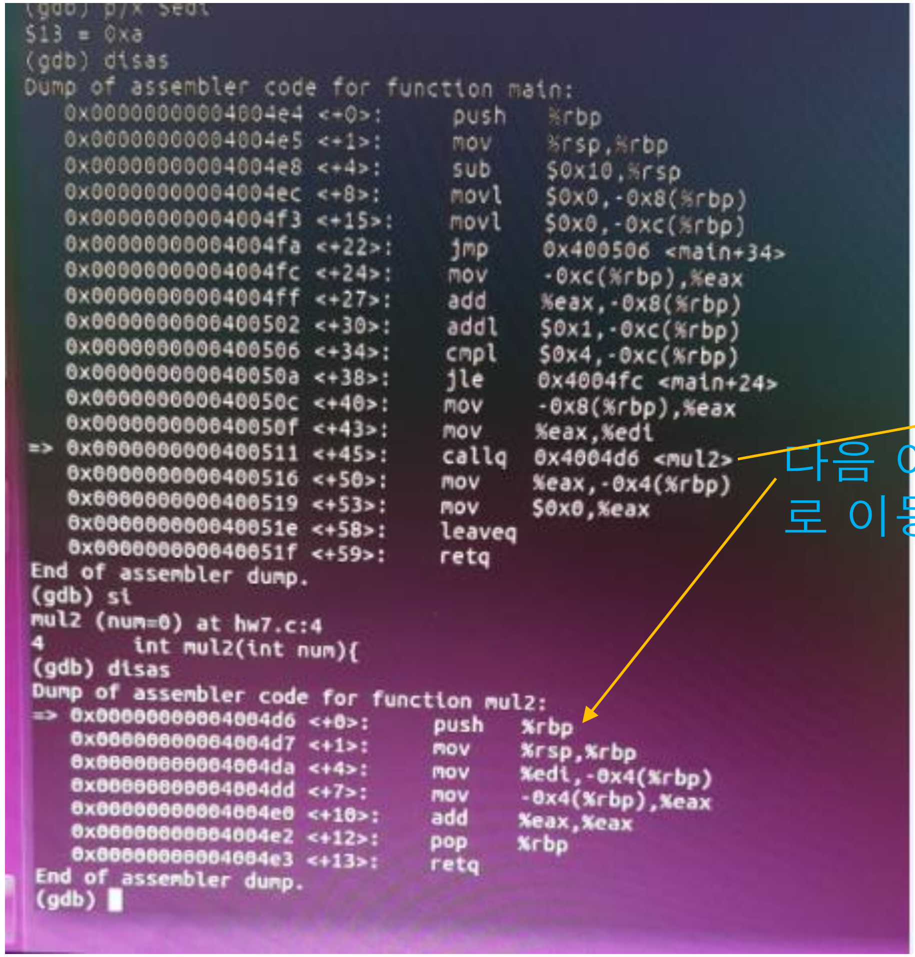
rbp2 – 12

$6 = 0x7fffffffdc70 rsp1 – 24

eax 값을 edi 로 복사

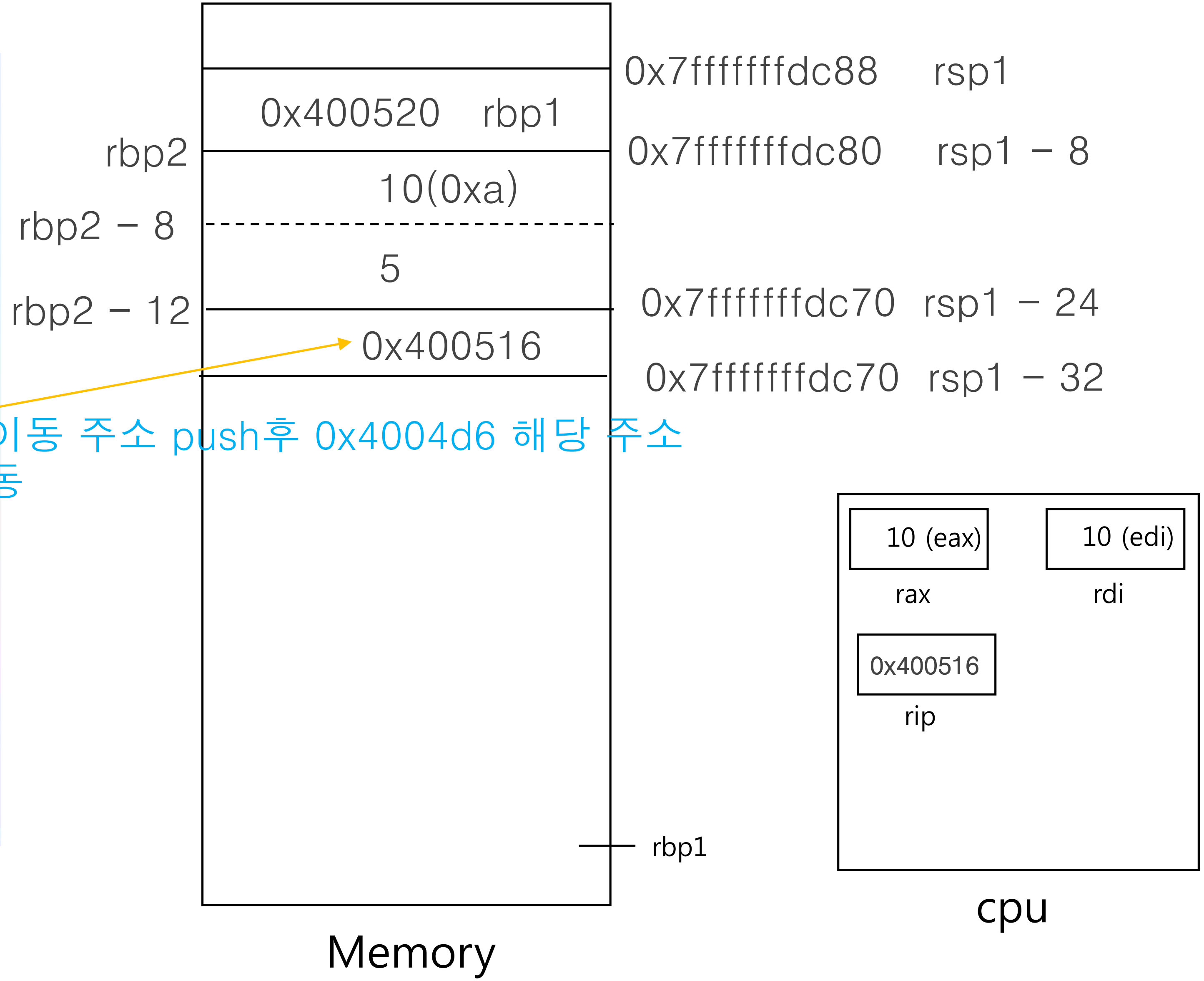
과제 7 - 21 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



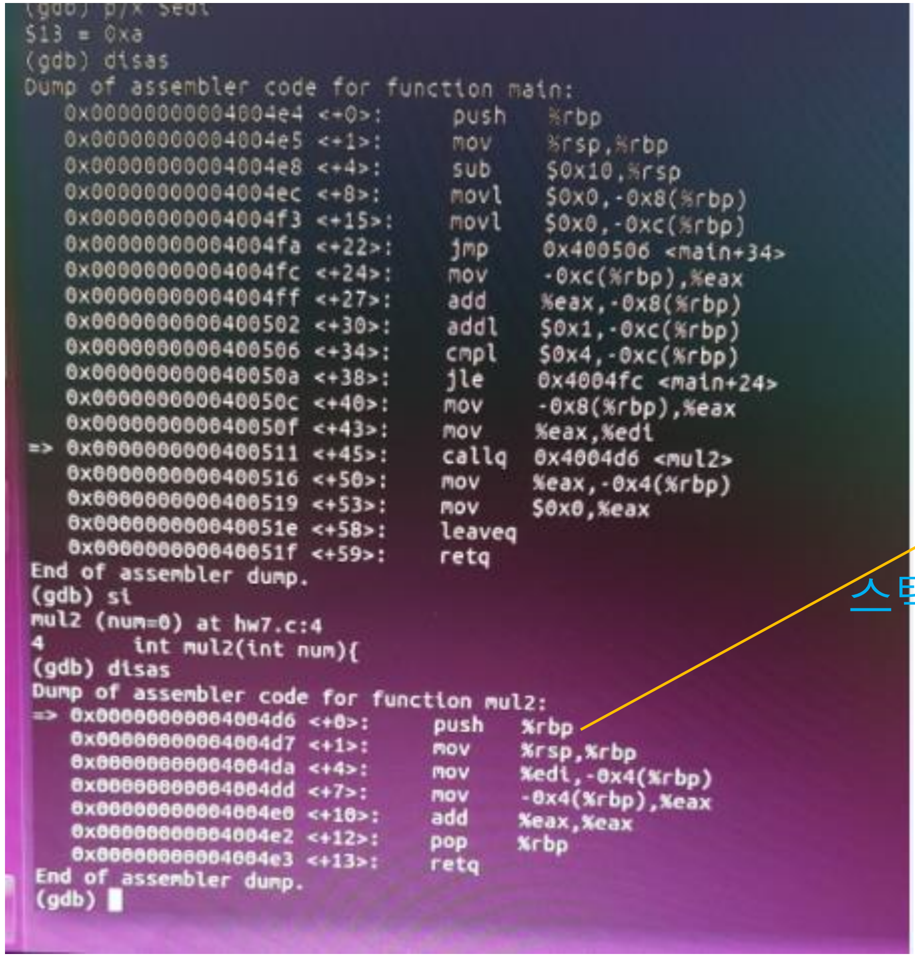
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:	push   %rbp
   0x00000000004004e5 <+1>:	mov    %rsp,%rbp
   0x00000000004004e8 <+4>:	sub    $0x10,%rsp
   0x00000000004004ec <+8>:	movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:	movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:	jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:	mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:	add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:	addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:	cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:	jle    0x4004fc <main+24>
   0x000000000040050c <+40>:	mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:	mov    %eax,%edi
=> 0x0000000000400511 <+45>:	callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:	mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:	mov    $0x0,%eax
   0x000000000040051e <+58>:	leaveq
   0x000000000040051f <+59>:	retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4	    int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:	push   %rbp
   0x00000000004004d7 <+1>:	mov    %rsp,%rbp
   0x00000000004004da <+4>:	mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:	mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:	add    %eax,%eax
   0x00000000004004e2 <+12>:	pop    %rbp
   0x00000000004004e3 <+13>:	retq
End of assembler dump.
(gdb) 
```

다음 이동 주소 push후 0x4004d6 해당 주소로 이동



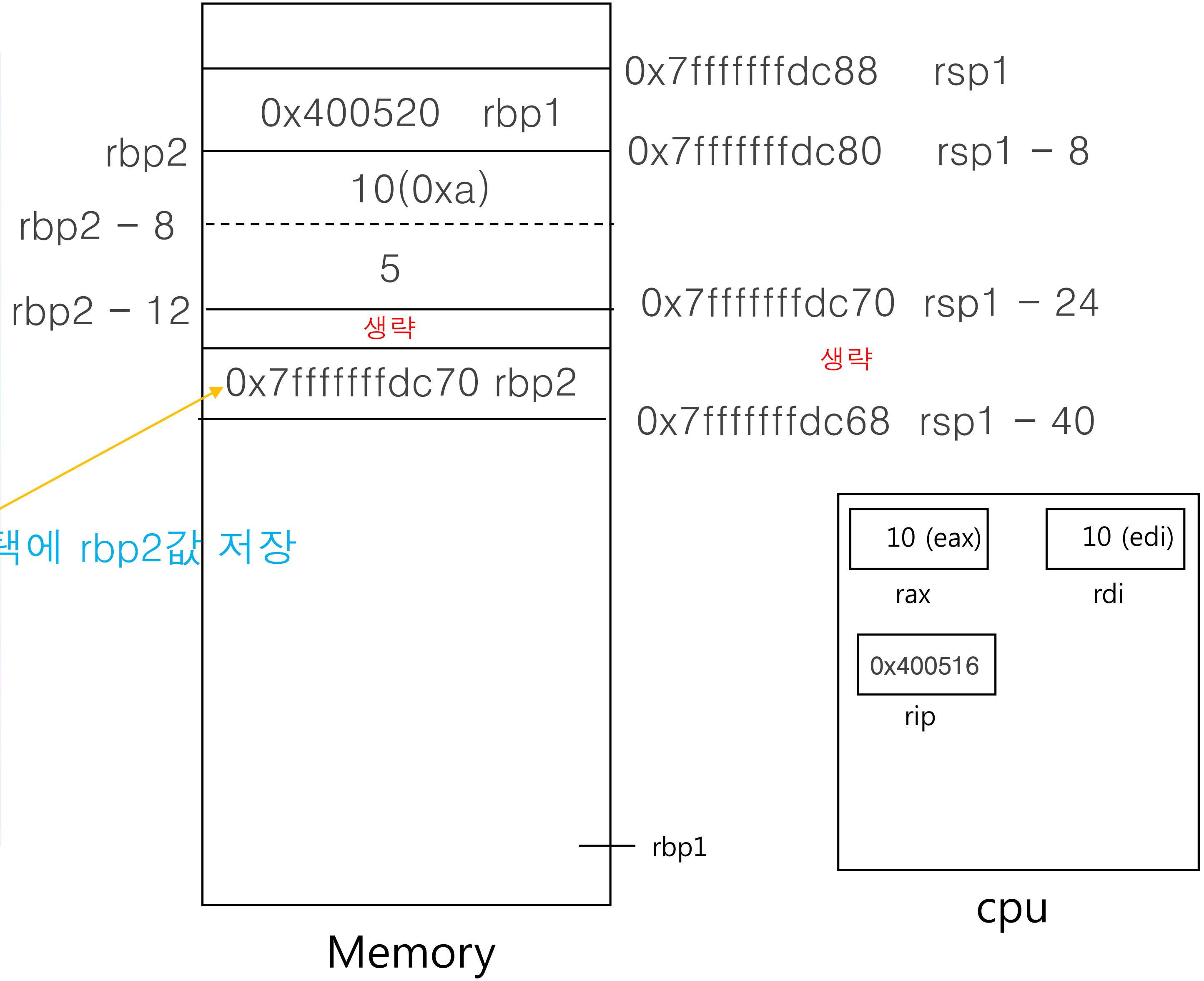
과제 7 - 22 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



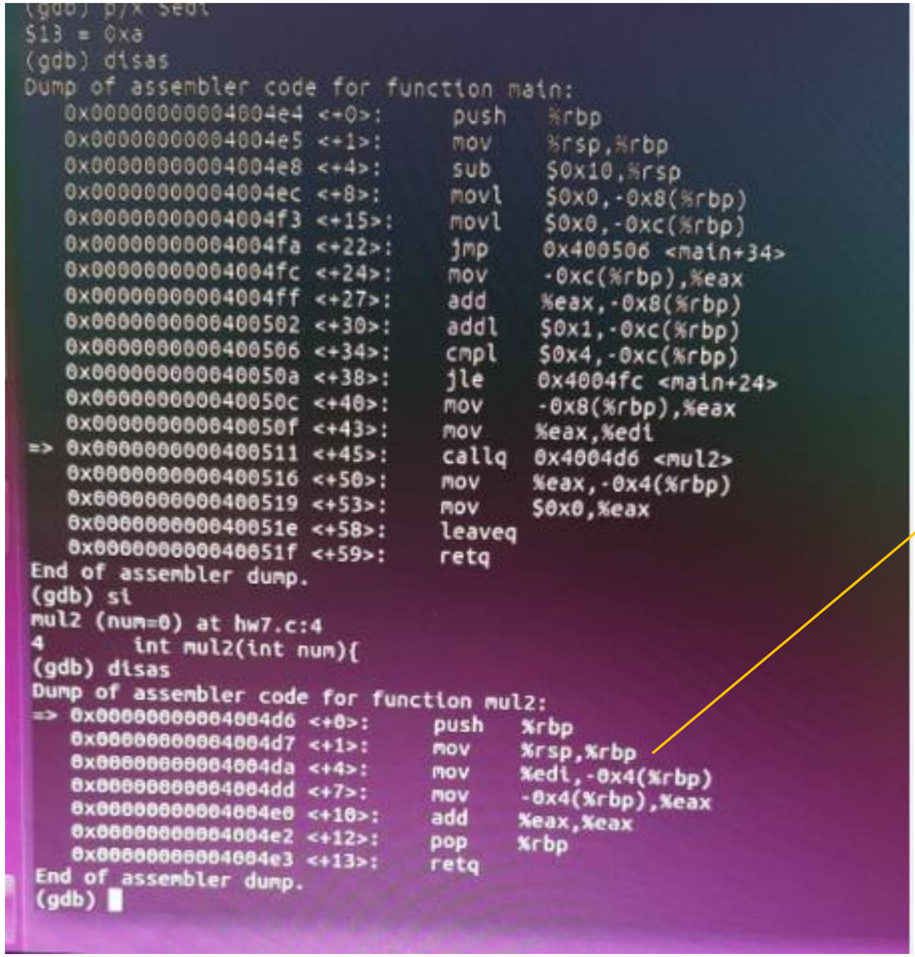
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) █
```

스택에 rbp2값 저장

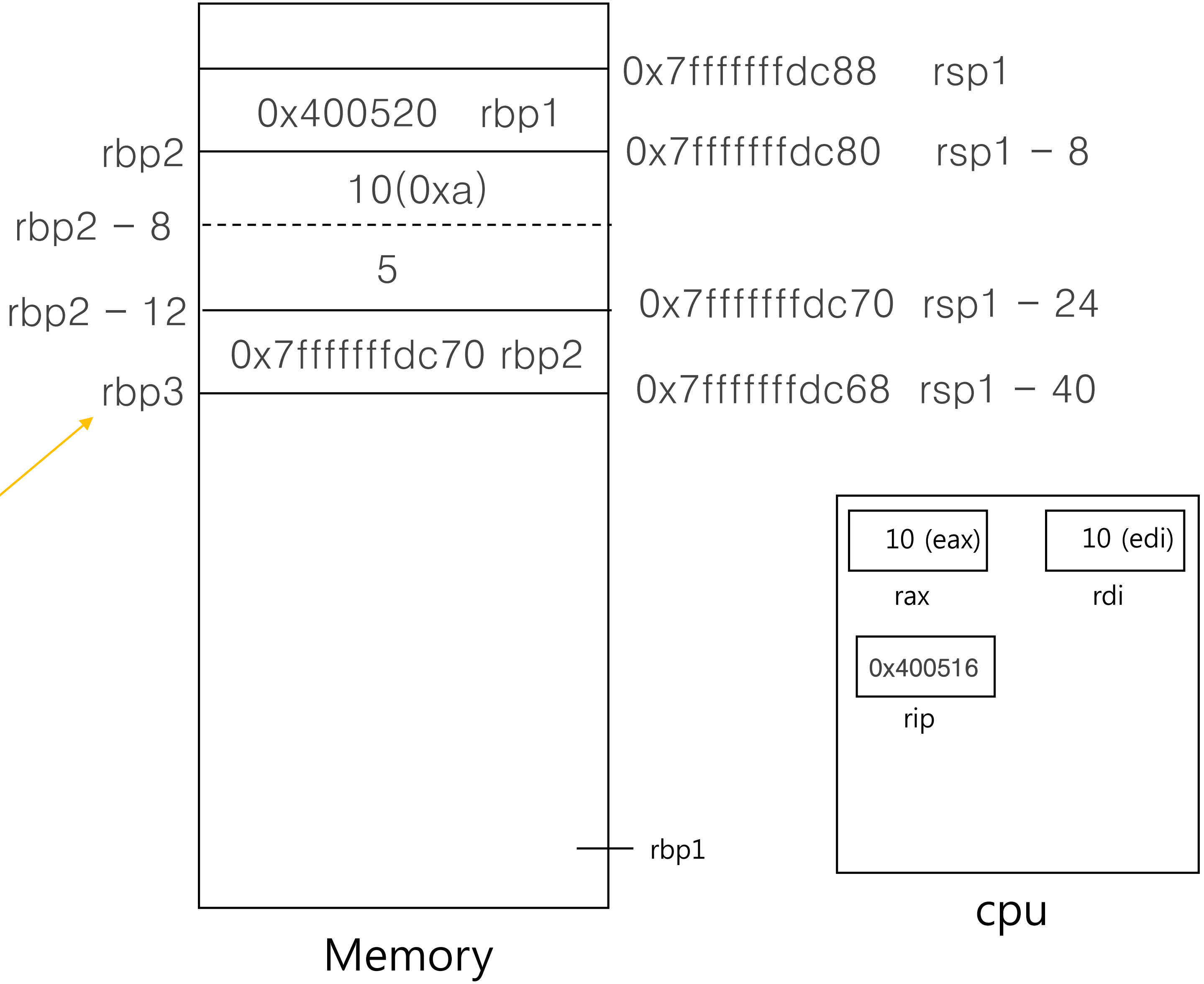


과제 7 - 23 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

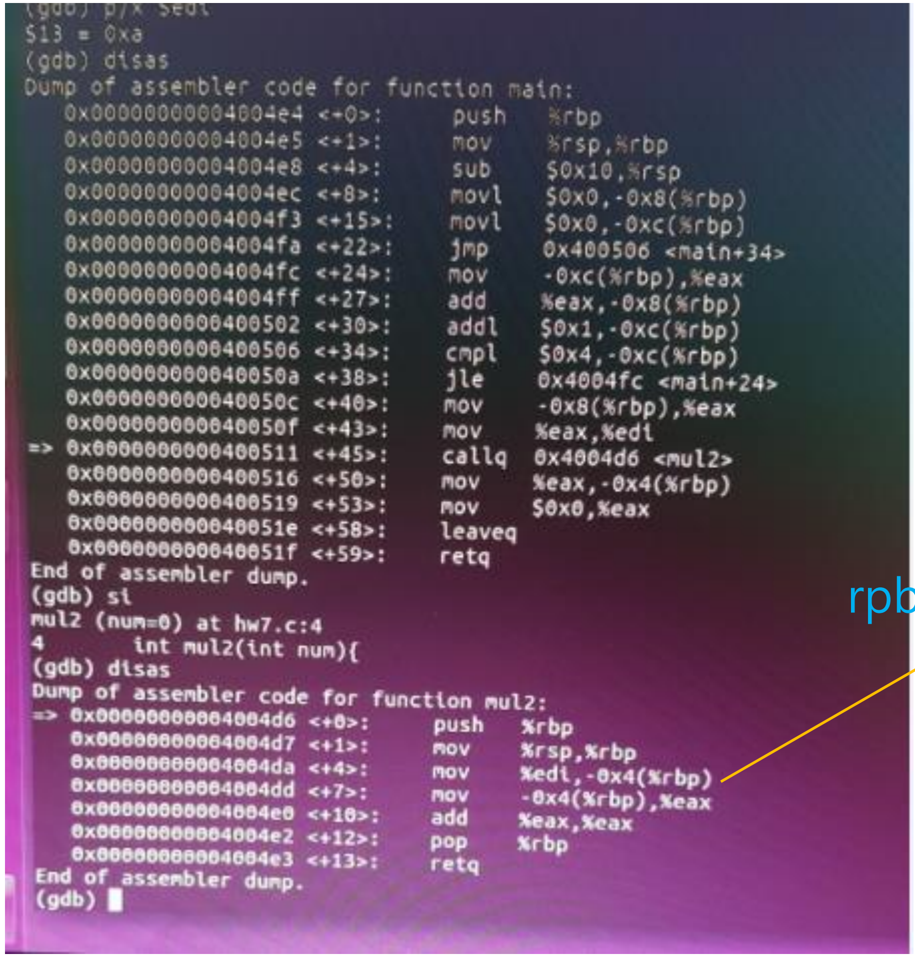


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



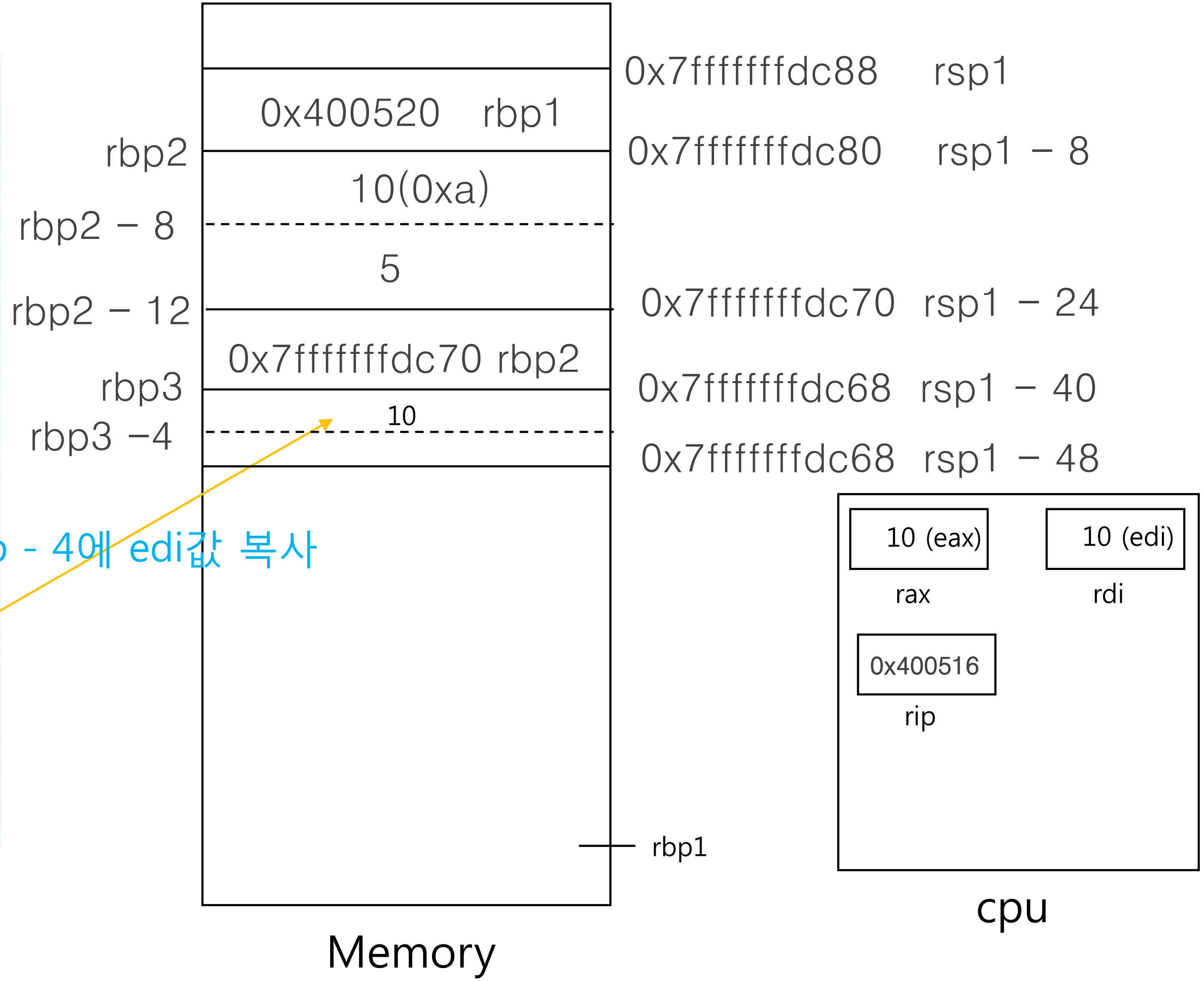
과제 7 - 24 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



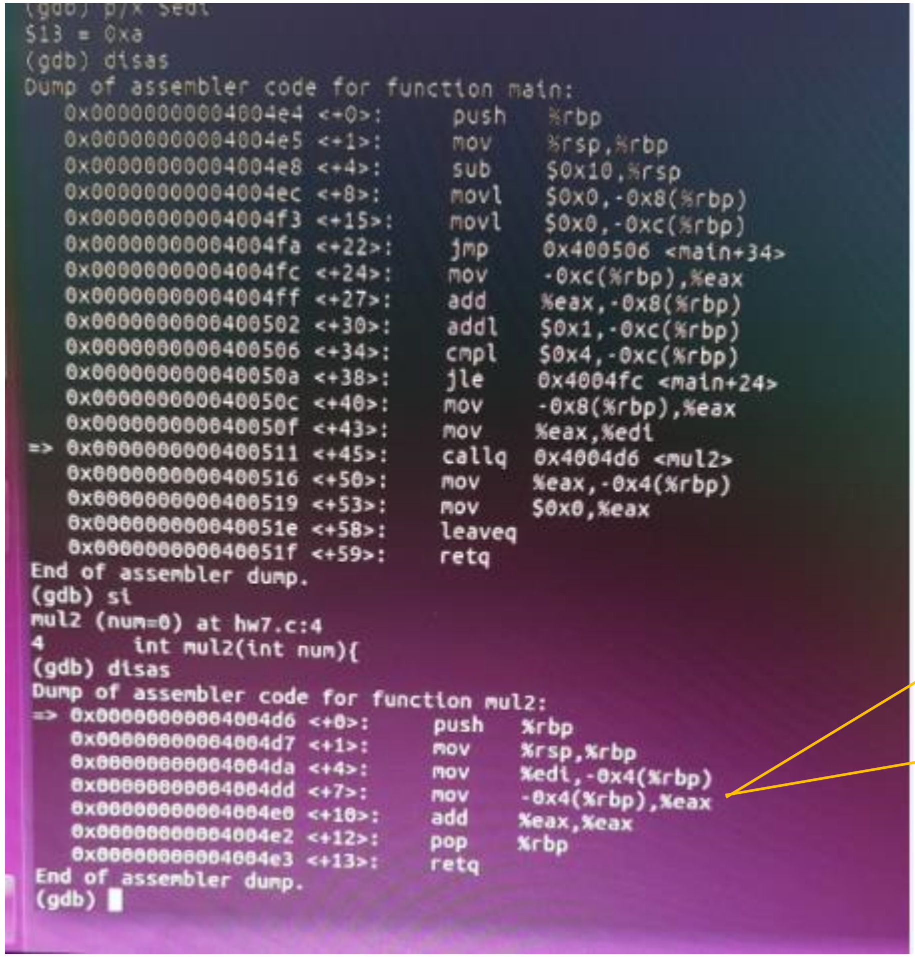
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

rpb - 40에 edi값 복사

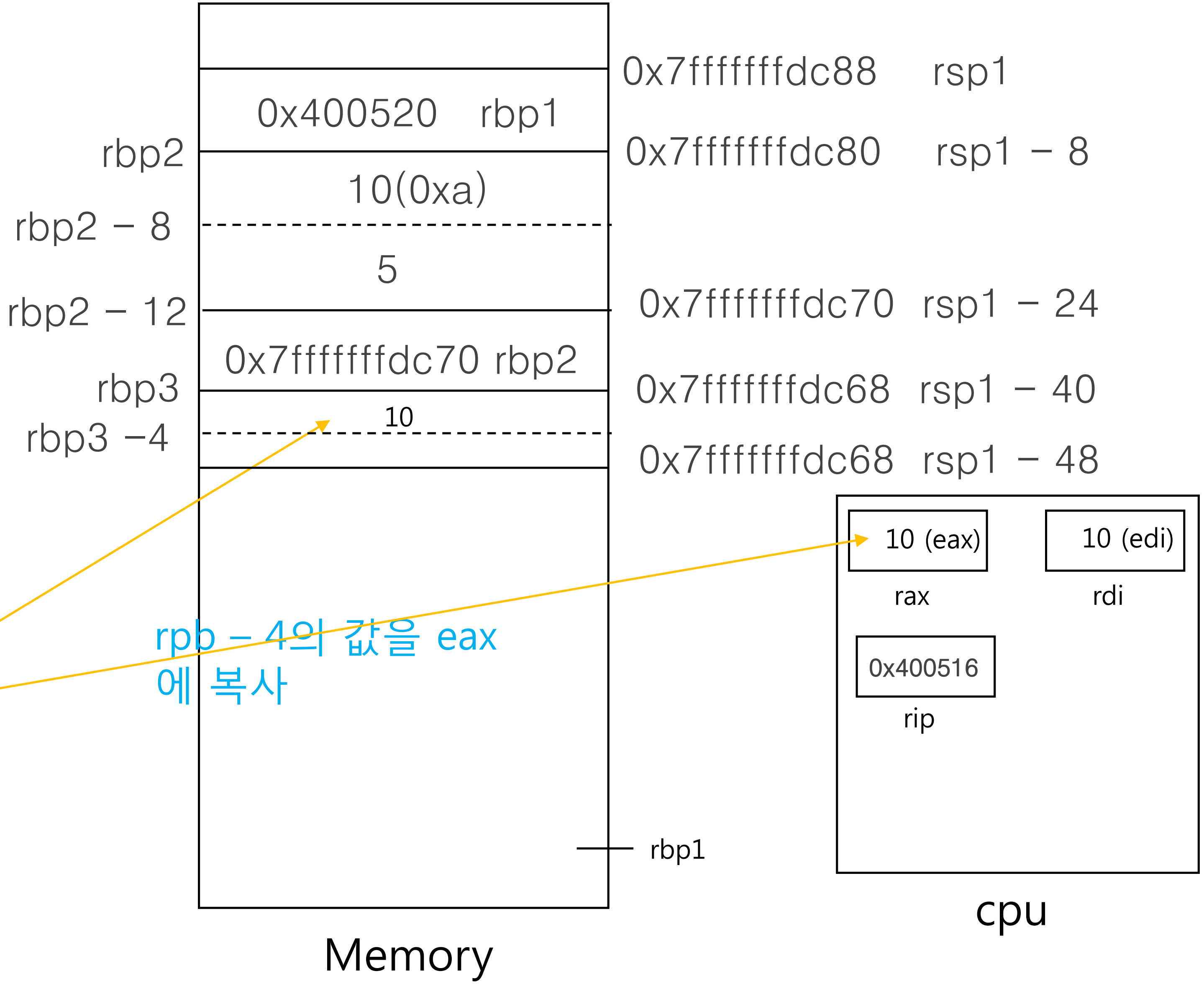


과제 7 - 25 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



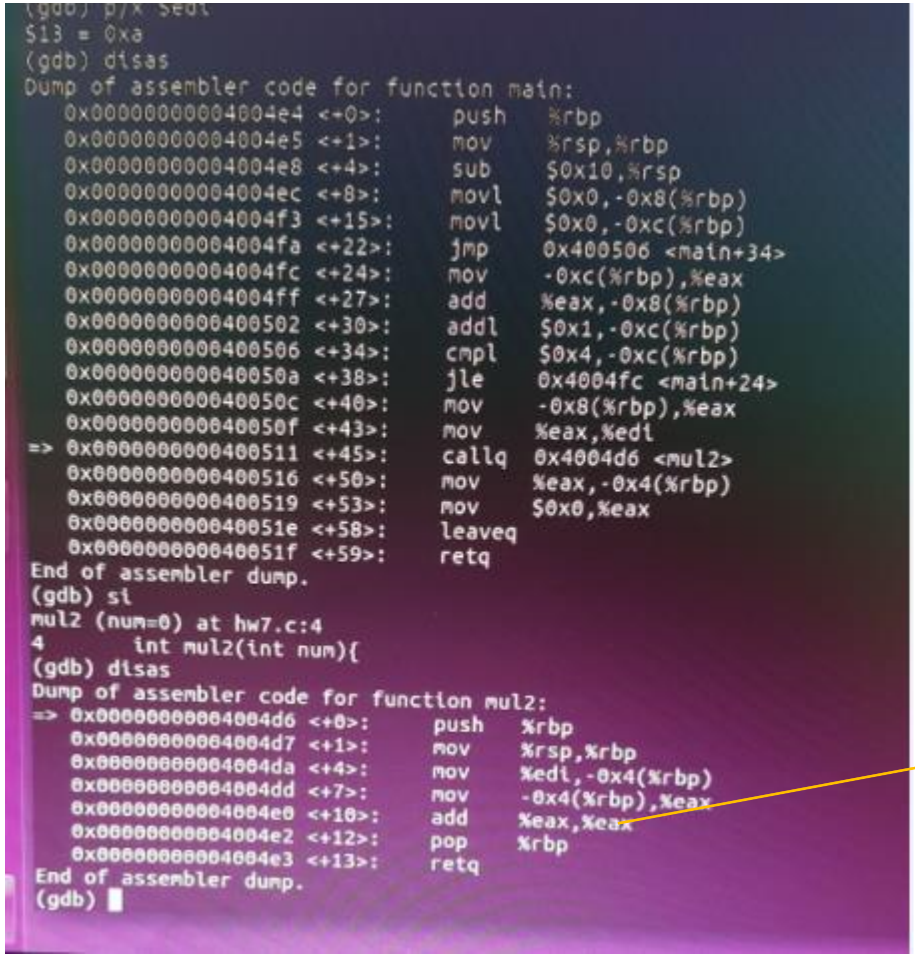
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:	push   %rbp
   0x00000000004004e5 <+1>:	mov    %rsp,%rbp
   0x00000000004004e8 <+4>:	sub    $0x10,%rsp
   0x00000000004004ec <+8>:	movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:	movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:	jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:	mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:	add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:	addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:	cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:	jle    0x4004fc <main+24>
   0x000000000040050c <+40>:	mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:	mov    %eax,%edi
=> 0x0000000000400511 <+45>:	callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:	mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:	mov    $0x0,%eax
   0x000000000040051e <+58>:	leaveq
   0x000000000040051f <+59>:	retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4	int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:	push   %rbp
   0x00000000004004d7 <+1>:	mov    %rsp,%rbp
   0x00000000004004da <+4>:	mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:	mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:	add    %eax,%eax
   0x00000000004004e2 <+12>:	pop    %rbp
   0x00000000004004e3 <+13>:	retq
End of assembler dump.
(gdb) 
```



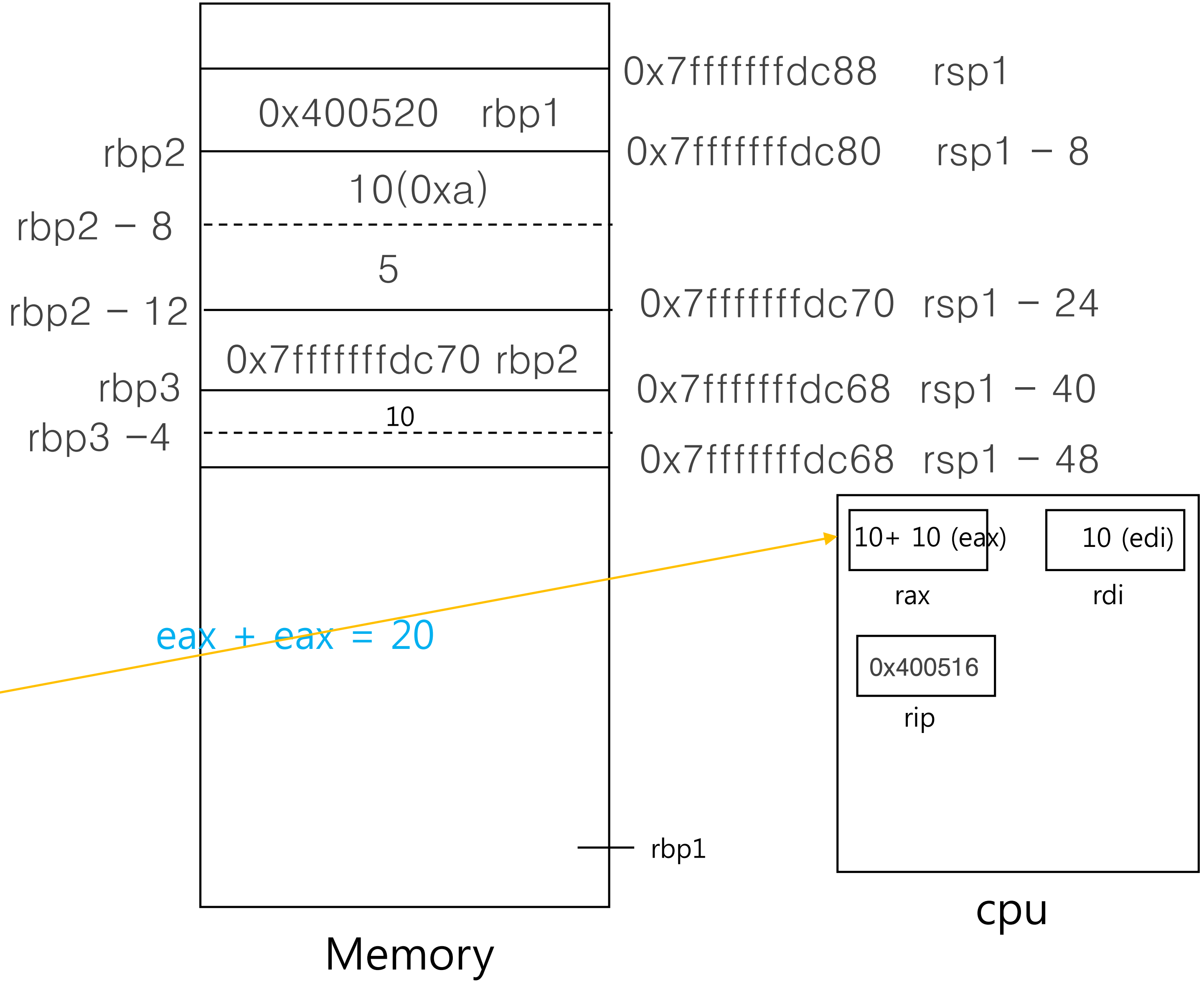
rpb – 4의 값을 eax 에 복사

과제 7 - 26 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

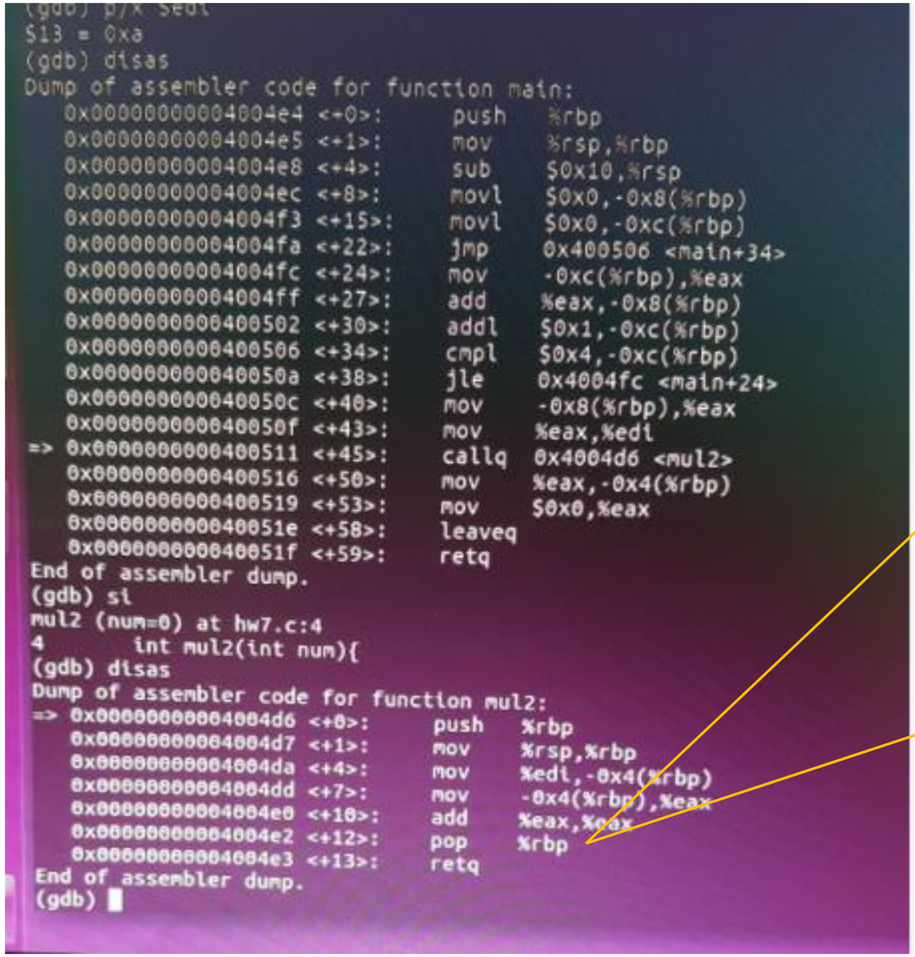


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

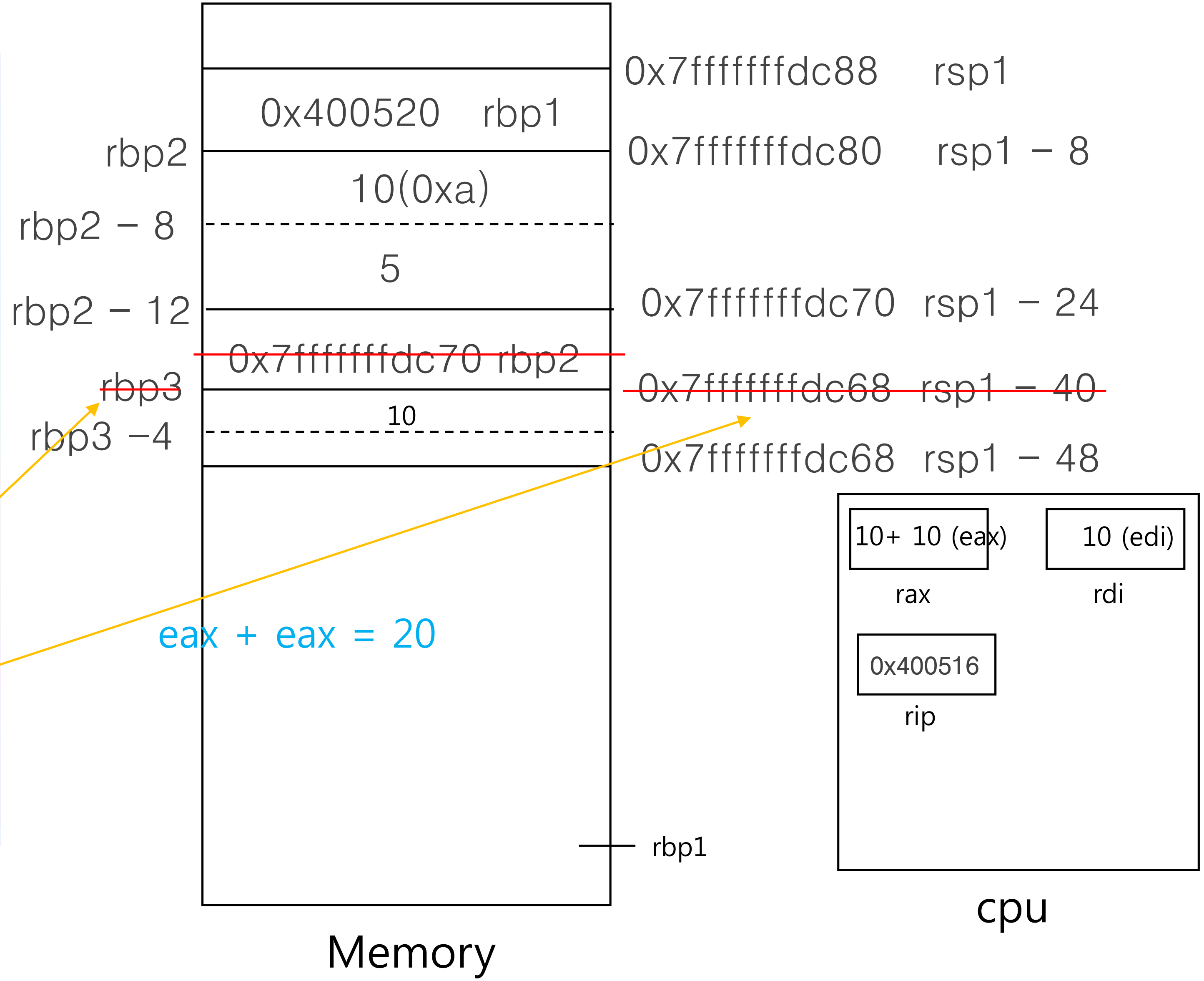


과제 7 - 27 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

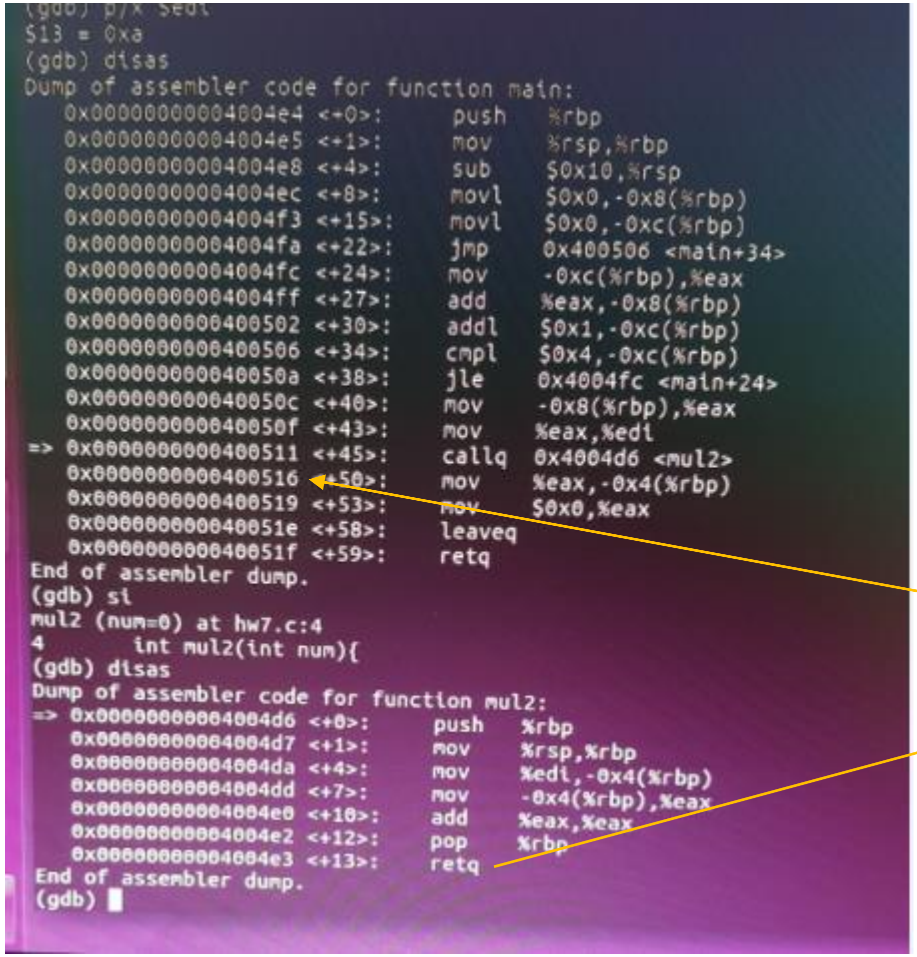


```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

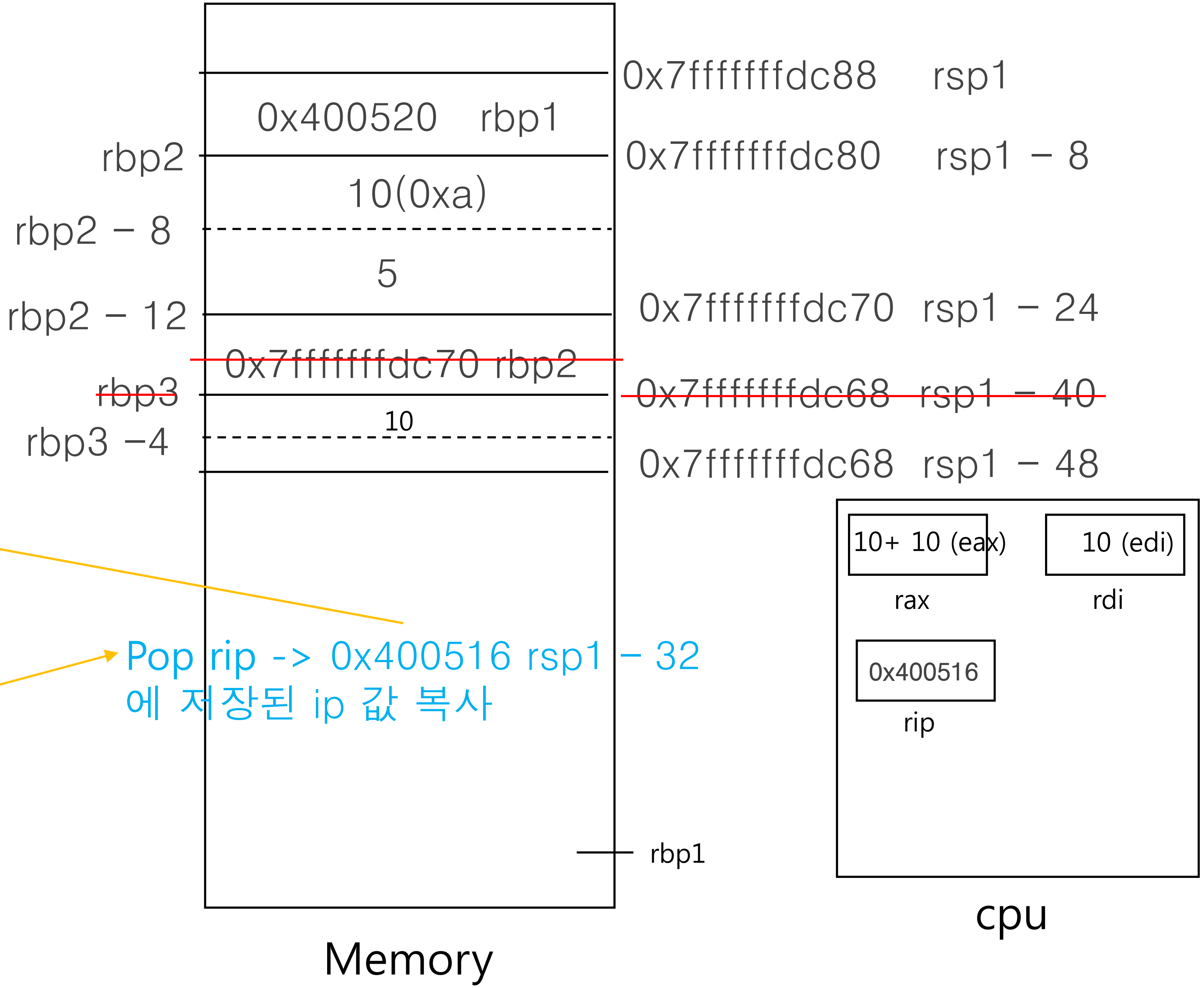


과제 7 - 28 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
 이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
 esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
 메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



0x7fffffffdc88 rsp1

0x400520 rbp1

rbp2

0x7fffffffdc80 rsp1 – 8

10(0xa)

rbp2 – 8

5

rbp2 – 12

0x7fffffffdc70 rsp1 – 24

~~0x7fffffffdc70 rbp2~~

~~rbp3~~

~~0x7fffffffdc68 rsp1 – 40~~

10

rbp3 –4

0x7fffffffdc68 rsp1 – 48

Pop rip -> 0x400516 rsp1 – 32
에 저장된 ip 값 복사

rbp1

Memory

10+ 10 (eax)

rax

10 (edi)

rdi

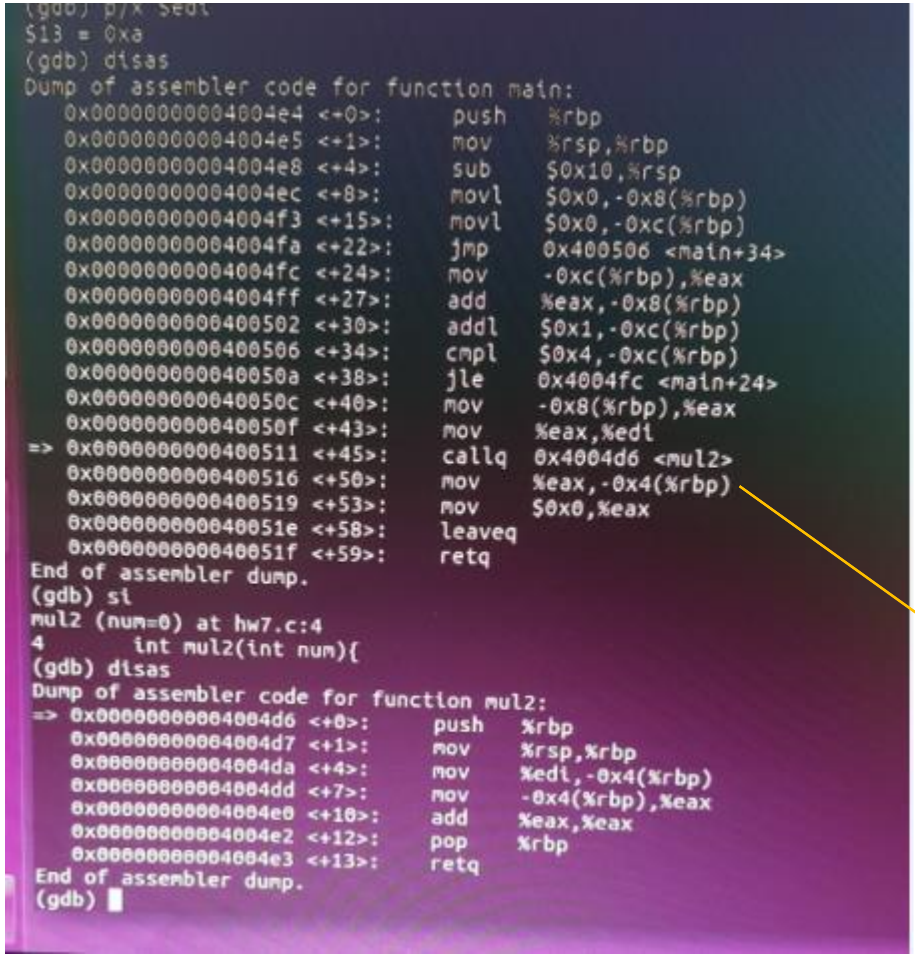
0x400516

rip

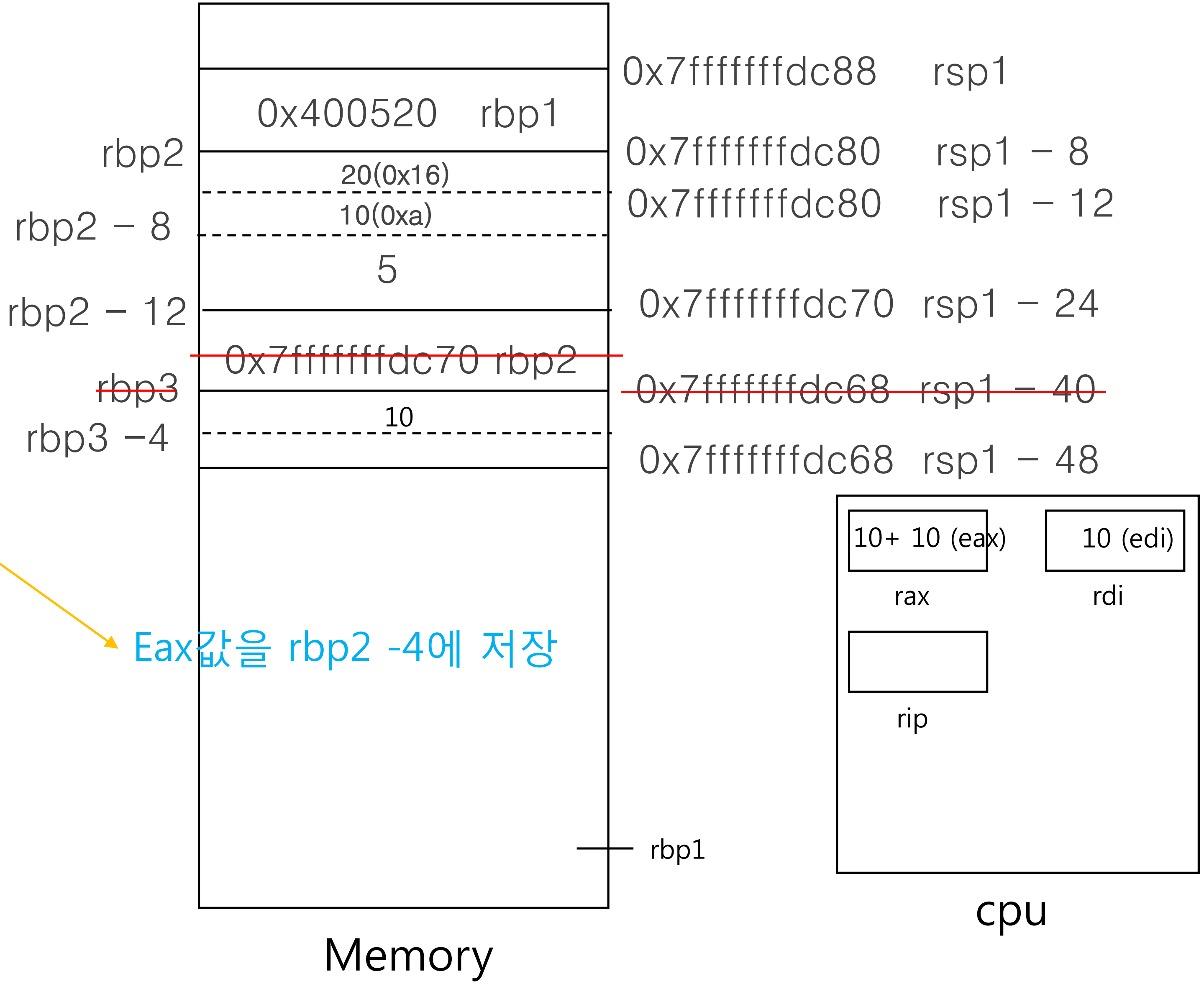
cpu

과제 7 - 29 -

7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



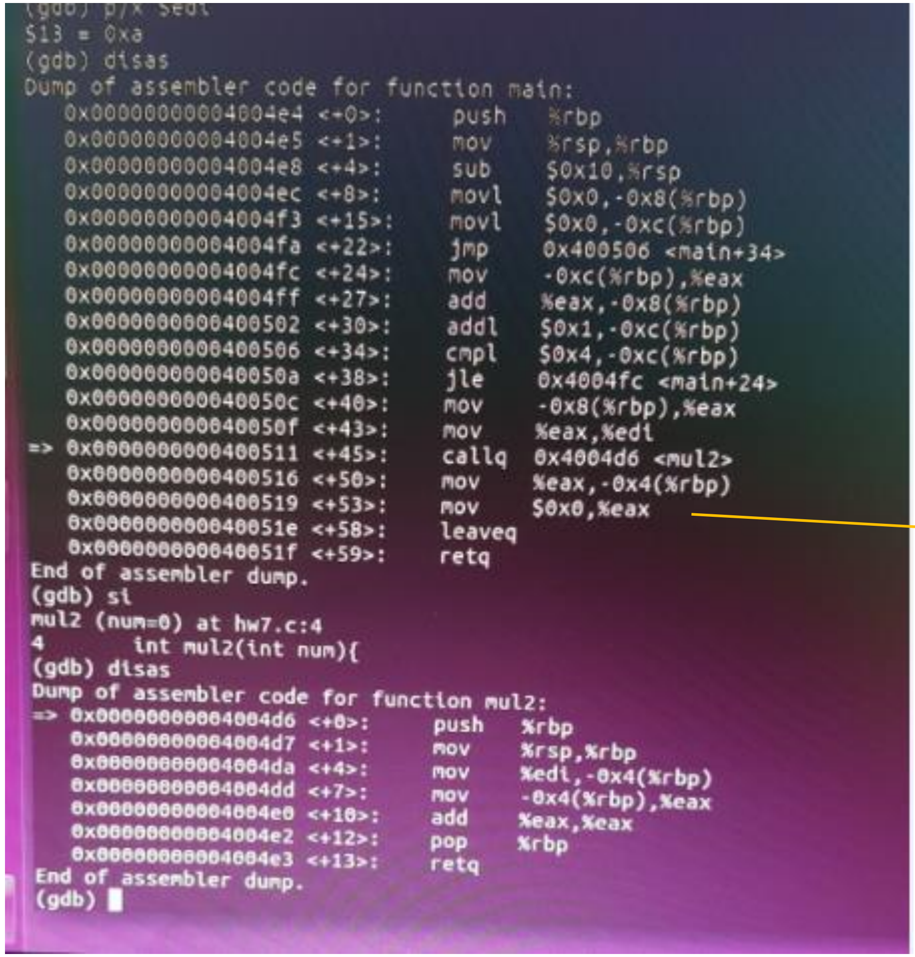
```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```



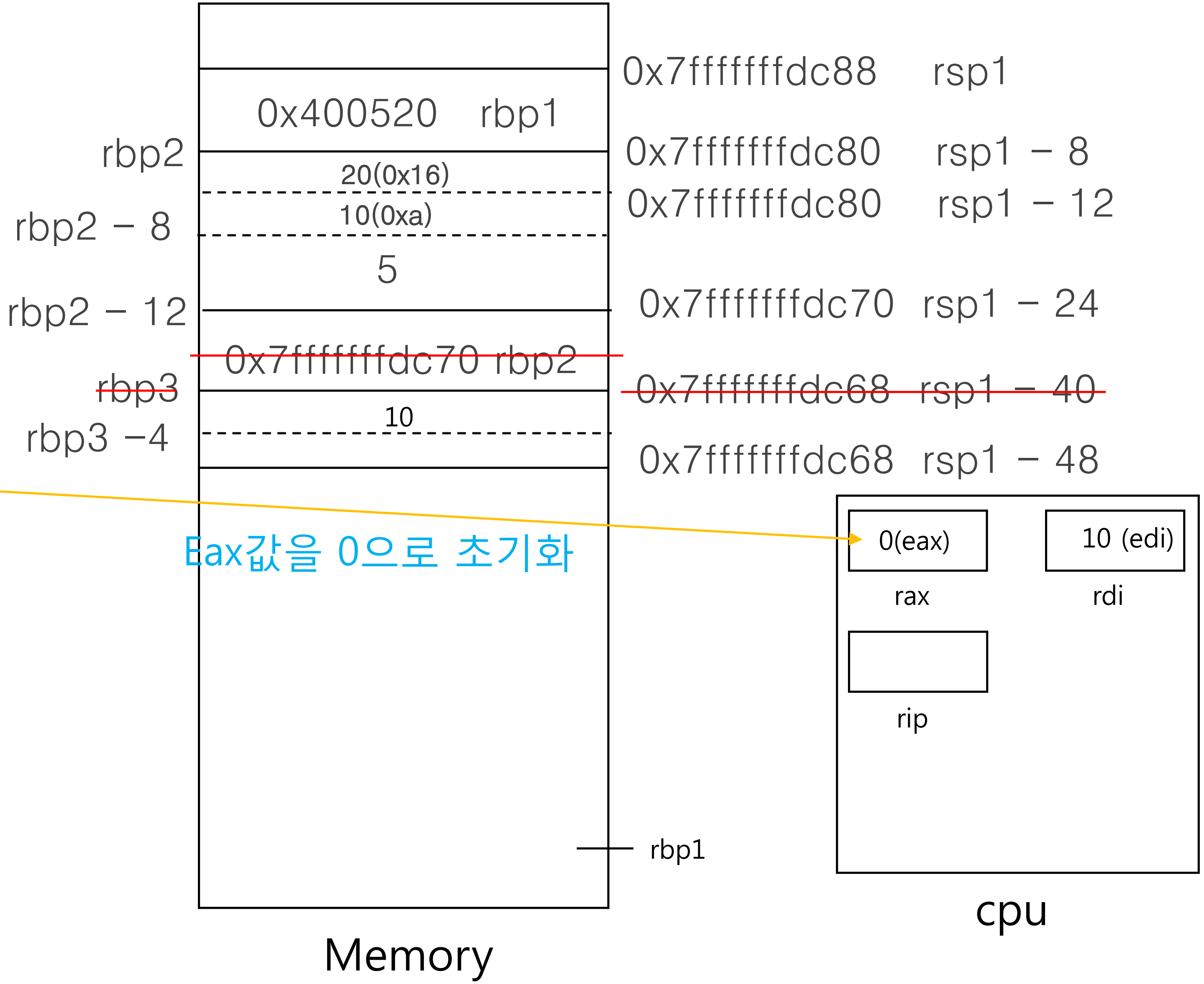
Eax값을 rbp2 -4에 저장

과제 7 - 30 -

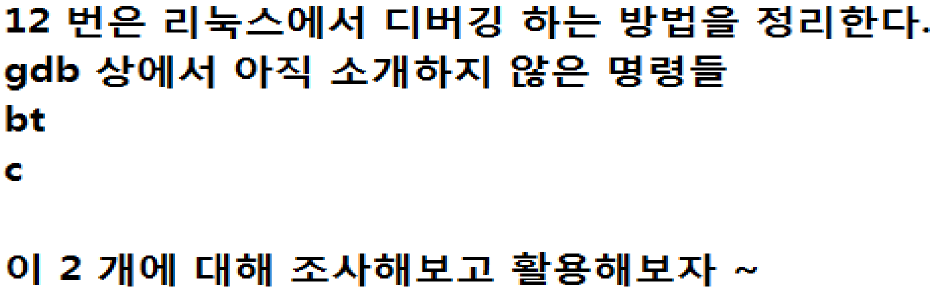
7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
   이 구조가 어떻게 동작하는지 Assem bly Language를 해석하며 기술해보시오.
   esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
   메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.



```
(gdb) p/x $edi
$13 = 0xa
(gdb) disas
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     movl   $0x0,-0x8(%rbp)
   0x00000000004004f3 <+15>:    movl   $0x0,-0xc(%rbp)
   0x00000000004004fa <+22>:    jmp    0x400506 <main+34>
   0x00000000004004fc <+24>:    mov    -0xc(%rbp),%eax
   0x00000000004004ff <+27>:    add    %eax,-0x8(%rbp)
   0x0000000000400502 <+30>:    addl   $0x1,-0xc(%rbp)
   0x0000000000400506 <+34>:    cmpl   $0x4,-0xc(%rbp)
   0x000000000040050a <+38>:    jle    0x4004fc <main+24>
   0x000000000040050c <+40>:    mov    -0x8(%rbp),%eax
   0x000000000040050f <+43>:    mov    %eax,%edi
=> 0x0000000000400511 <+45>:    callq  0x4004d6 <mul2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb) si
mul2 (num=0) at hw7.c:4
4         int mul2(int num){
(gdb) disas
Dump of assembler code for function mul2:
=> 0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
(gdb) 
```

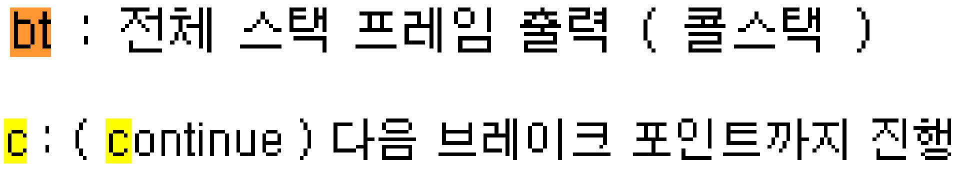


과제 12



12 번은 리눅스에서 디버깅 하는 방법을 정리한다.
gdb 상에서 아직 소개하지 않은 명령들
bt
c

이 2 개에 대해 조사해보고 활용해보자 ~



bt : 전체 스택 프레임 출력 (콜스택)

c : (continue) 다음 브레이크 포인트까지 진행