IOT 환경의 임베디드 개발자 양성과정

3일차 과제 문한나

문제 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)

<코드>

mhn@mhn-900X3L: ~/my_proj/c

```c
#include <stdio.h>

int main(){

        int rental = 37500; //변수 선언 후 값 저장
        int day=7;

        if(day >= 3) //3일 이상 이용할 경우
                rental = rental - (rental*0.2); //할인

        printf("result = %d",rental); //출력

        return 0;
}
~
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c$ ./a.out
result = 30000mhn@mhn-900X3L:~/my_proj/c$
```

* 조건문 if 사용
  조건식이 0이 아니면(참) 다음 문장을 실행하고 0이면(거짓) 실행하지 않는다.

문제 3. 1 ~ 1000사이에 3의 배수의 합을 구하시오.

<코드>

mhn@mhn-900X3L: ~/my_proj/c

```c
#include <stdio.h>

void func(int start,int end){

        int result;
        while(start <= end){ //1~1000까지 반복
                if(start % 3 == 0){ //3의 배수
                        result += start; //값 저장
                }       start++;
        }       printf("1~1000사이 3의 배수의 합:%d",result); //출력
}

int main(){

        func(1,1000);//함수호출

        return 0;
}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c$ ./a.out
1~1000사이 3의 배수의 합:166833mhn@mhn-900X3L:~/my_proj/c$
```

* 반복문 while 사용
  반복 조건을 판별하여 0이면(거짓) while문을 종료하고, 0이 아니면(참) 실행한다.

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

<코드>

mhn@mhn-900X3L: ~/my_proj/c

```c
#include <stdio.h>

void func(int num1,int num2){ //리턴값이 없으므로 반환형을 void로 기술,함수 선언문

        int result; //결과 값을 저장할 변수 선언

        while(num1 <= num2){ //변수 num1(1)이 num2(1000)까지 반복
                if(num1 % 4 == 1 || num1 % 6 == 1){ /* 논리연산자를 사용하여 조건 중
                                                       하나만 성립이 되어도 조건문 수행 */
                        result += num1; //값 저장(result = result+num1 ->
                                                result += num1 축약대입연산자사용)
                }num1++; //증감 연산자를 사용하여 조건문이 끝난 후 num1 숫자증가
        }printf("1~1000사이 4나 6으로 나누어도 나머지가 1인 수의 합:%d",result);//결과출력
}

int main(){

        func(1,1000); //함수호출

        return 0;

}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c$ ./a.out
1~1000사이 4나 6으로 나누어도 나머지가 1인 수의 합:166167
```

* 논리연산자 || 사용
  두 피연산자 중에서 하나만 0이 아니면(참) 1이고, 모두 거짓이면 0이다.
  (논리연산자 &&과 ||은 두 개 중에서 왼쪽 피연산자만으로 논리연산 결과가 결정된다면
  오른쪽 피연산자는 평가하지 않는다->단축평가)

문제 5. 7의 배수로 이루어진 값들이 나열되어 있다고 가정한다.

함수의 인자(input)로 항의 갯수를 받아서 마지막 항의 값을 구하는 프로그램을 작성하라.

<코드>

mhn@mhn-900X3L: ~/my_proj/c

```c
#include <stdio.h>

void func(int input){ //함수 선언

        printf("%d",input * 7); //결과 출력

}


int main(){

        int input; //입력받은 값을 저장 할 변수 선언

        printf("7의 배수 중 입력한 숫자번째 값을 알려드립니다");
        scanf("%d",&input); //입력값 저장(&->변수의 주소를 의미)
        func(input);//함수 호출

        return 0;

}
~
```

<결과>

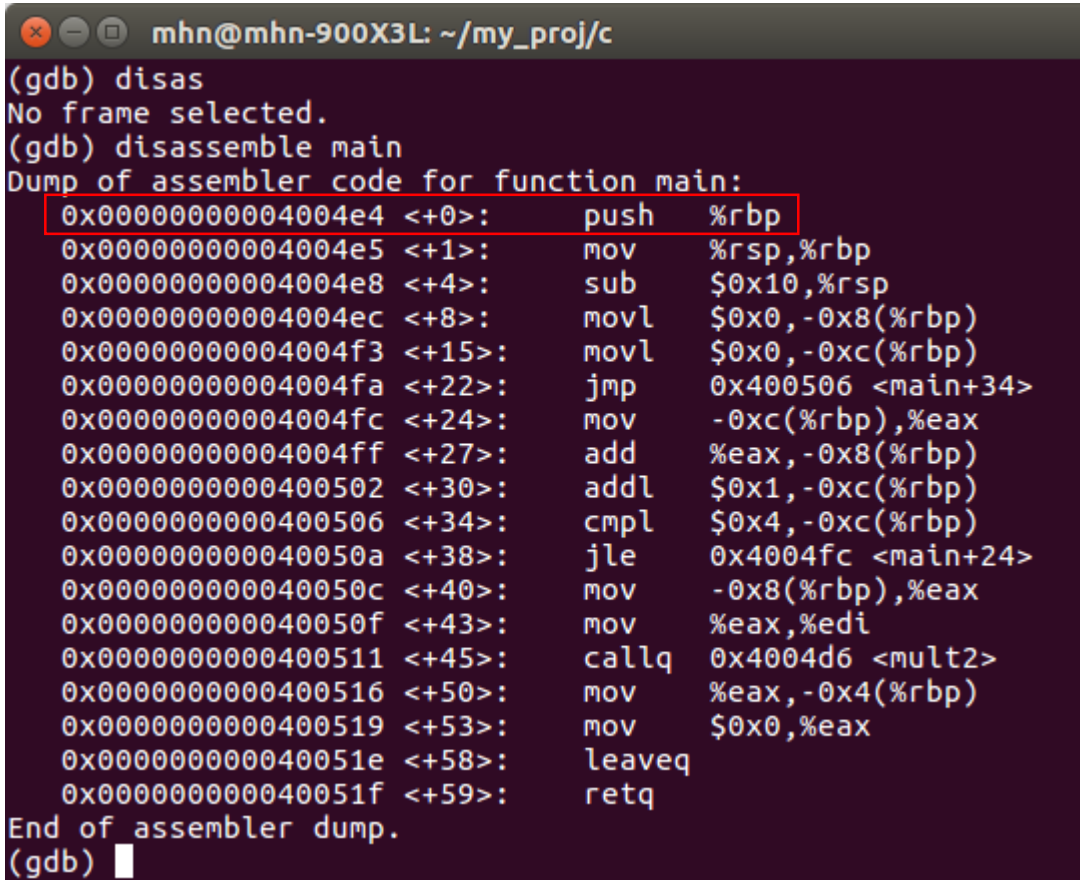
```
mhn@mhn-900X3L:~/my_proj/c$ ./a.out
7의 배수 중 입력한 숫자번째 값을 알려드립니다4
28mhn@mhn-900X3L:~/my_proj/c$
```

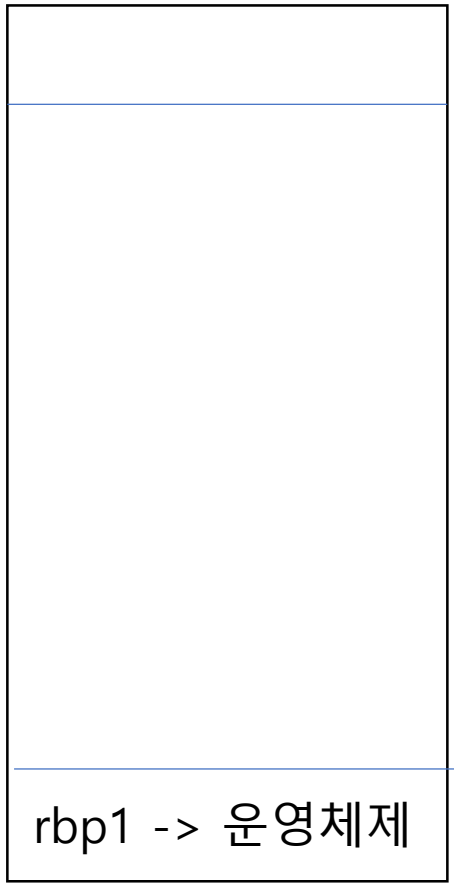
* 입력함수 scanf() 사용
  자료값을 입력 받는 함수이다. 첫 번째 인자는 제어문자열(%d,%f 등)이 오며, 두번째 인자는 입력값이 저장되는 변수가 온다. 변수명 앞에는 변수의 주소를 의미하는 &를 꼭 써야한다.

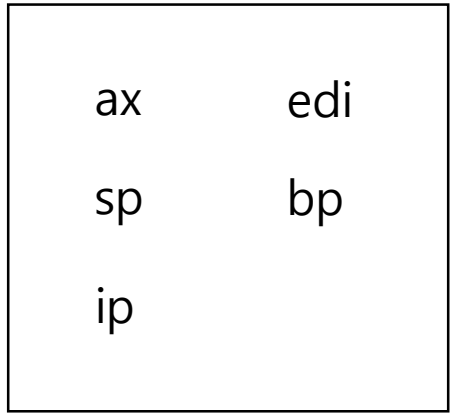
문제 7. C로 함수를 만들 때, Stack이란 구조가 생성된다.
이 구조가 어떻게 동작하는지 Assembly Language를 해석하며 기술해보시오.
esp, ebp, eip등의 Register에 어떤 값이 어떻게 들어가는지 등등
메모리에 어떤 값들이 들어가는지 등을 자세히 기술하시오.

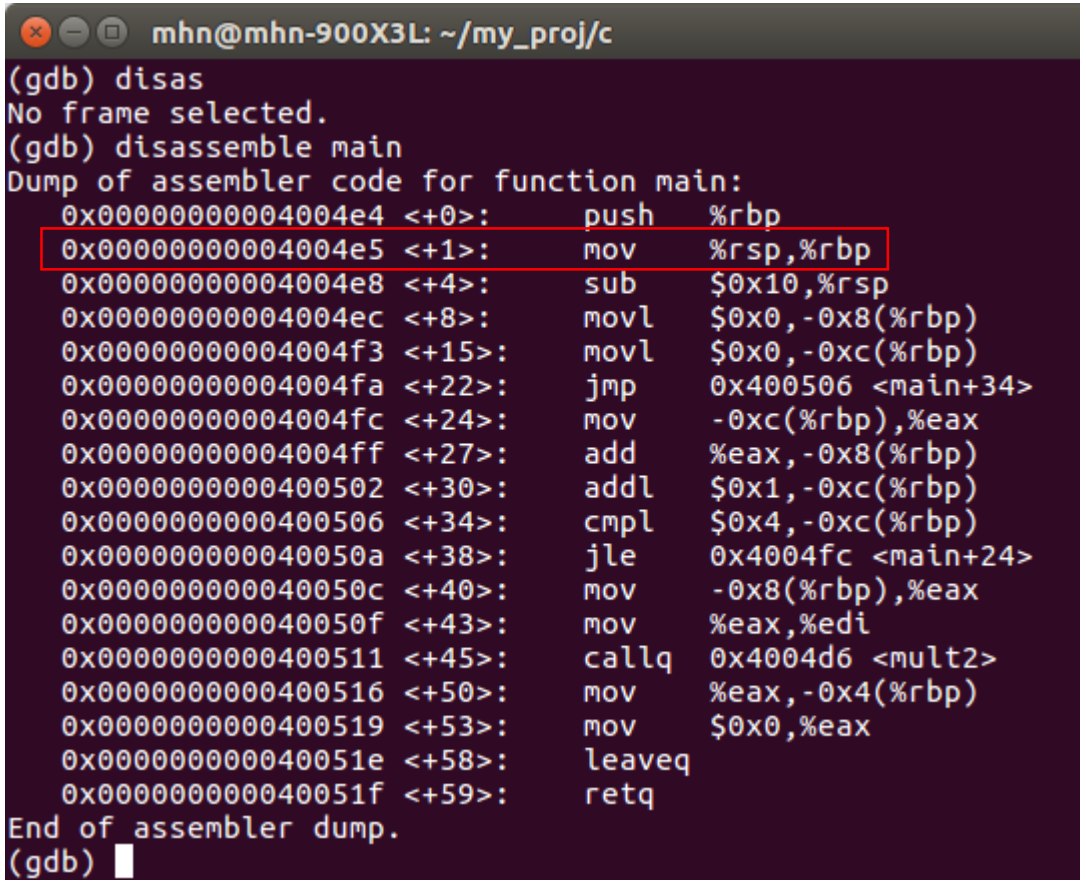
```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```

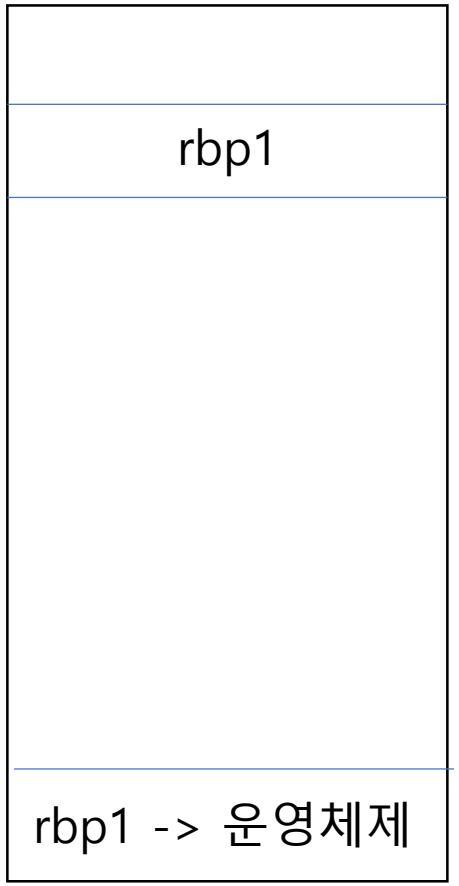
```
(gdb) disassemble mult2
Dump of assembler code for function mult2:
   0x00000000004004d6 <+0>:     push   %rbp
   0x00000000004004d7 <+1>:     mov    %rsp,%rbp
   0x00000000004004da <+4>:     mov    %edi,-0x4(%rbp)
   0x00000000004004dd <+7>:     mov    -0x4(%rbp),%eax
   0x00000000004004e0 <+10>:    add    %eax,%eax
   0x00000000004004e2 <+12>:    pop    %rbp
   0x00000000004004e3 <+13>:    retq
End of assembler dump.
```

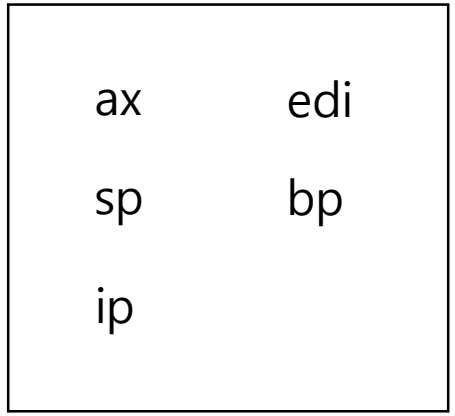

mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
Dump of assembler code for function main:
0x00000000004004e4 <+0>: push %rbp
0x00000000004004e5 <+1>: mov %rsp,%rbp
0x00000000004004e8 <+4>: sub $0x10,%rsp
0x00000000004004ec <+8>: movl $0x0,-0x8(%rbp)
0x00000000004004f3 <+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 <mult2>
0x0000000000400516 <+50>: mov %eax,-0x4(%rbp)
0x0000000000400519 <+53>: mov $0x0,%eax
0x000000000040051e <+58>: leaveq
0x000000000040051f <+59>: retq
End of assembler dump.
(gdb)


rbp1 -> 운영체제

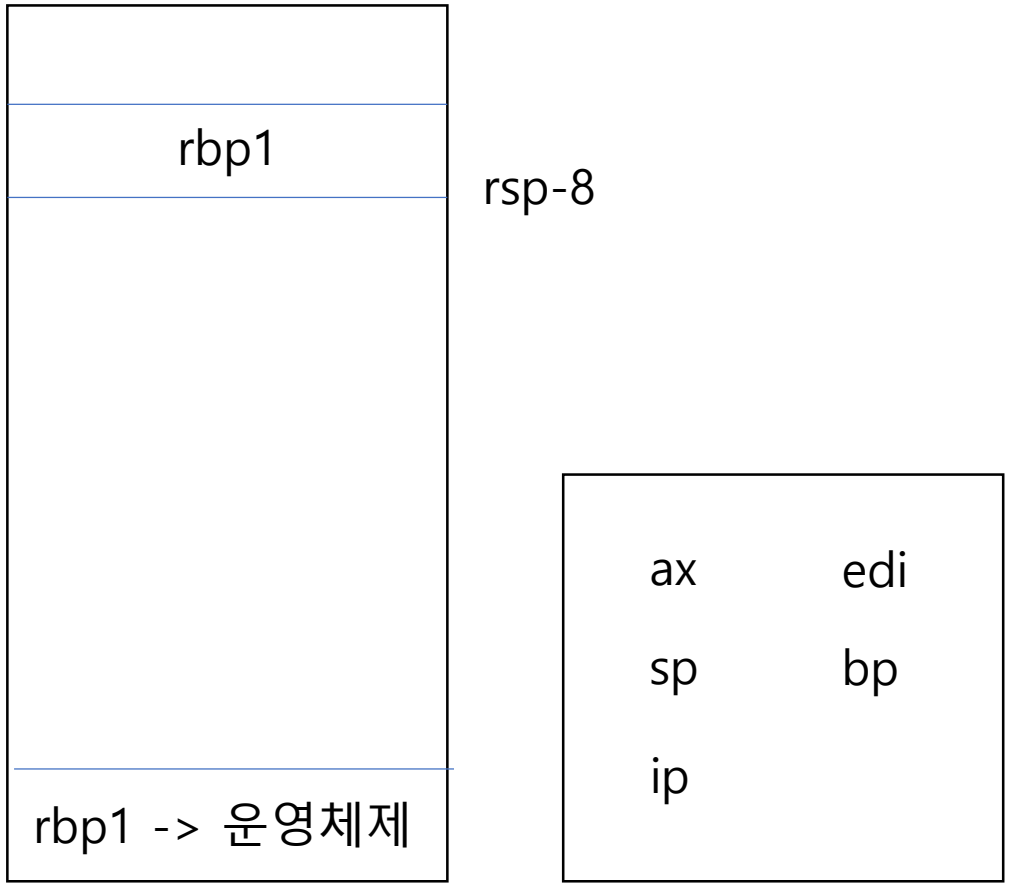

ax
edi
sp
bp
ip


mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
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 <mult2>
0x0000000000400516 <+50>: mov %eax,-0x4(%rbp)
0x0000000000400519 <+53>: mov $0x0,%eax
0x000000000040051e <+58>: leaveq
0x000000000040051f <+59>: retq
End of assembler dump.
(gdb)

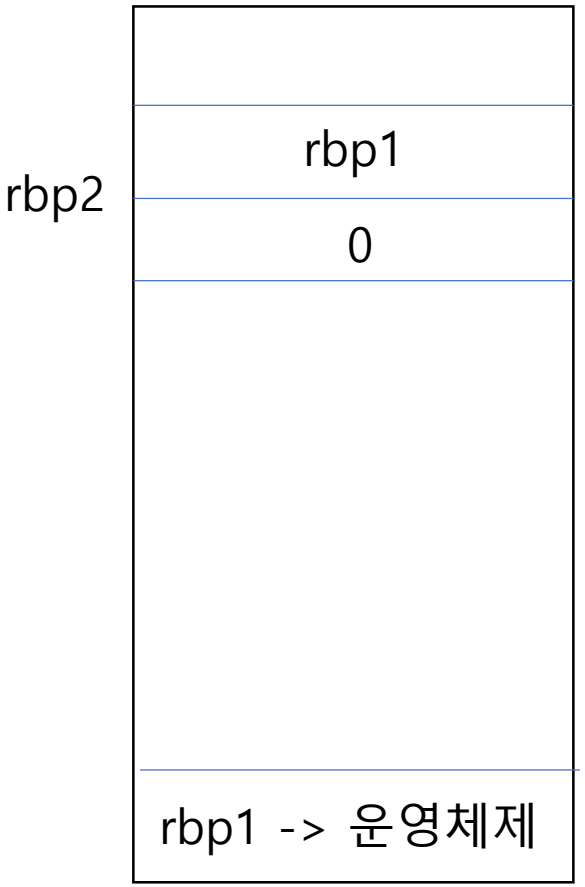

rbp1
rbp1 -> 운영체제

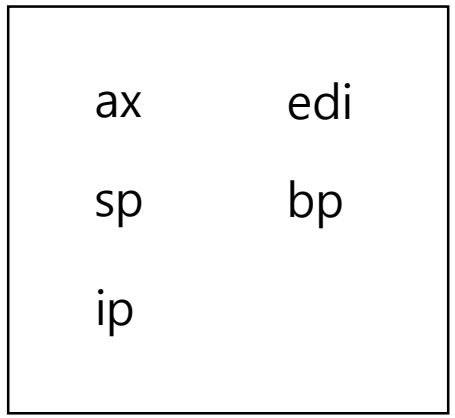

ax
edi
sp
bp
ip

```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```

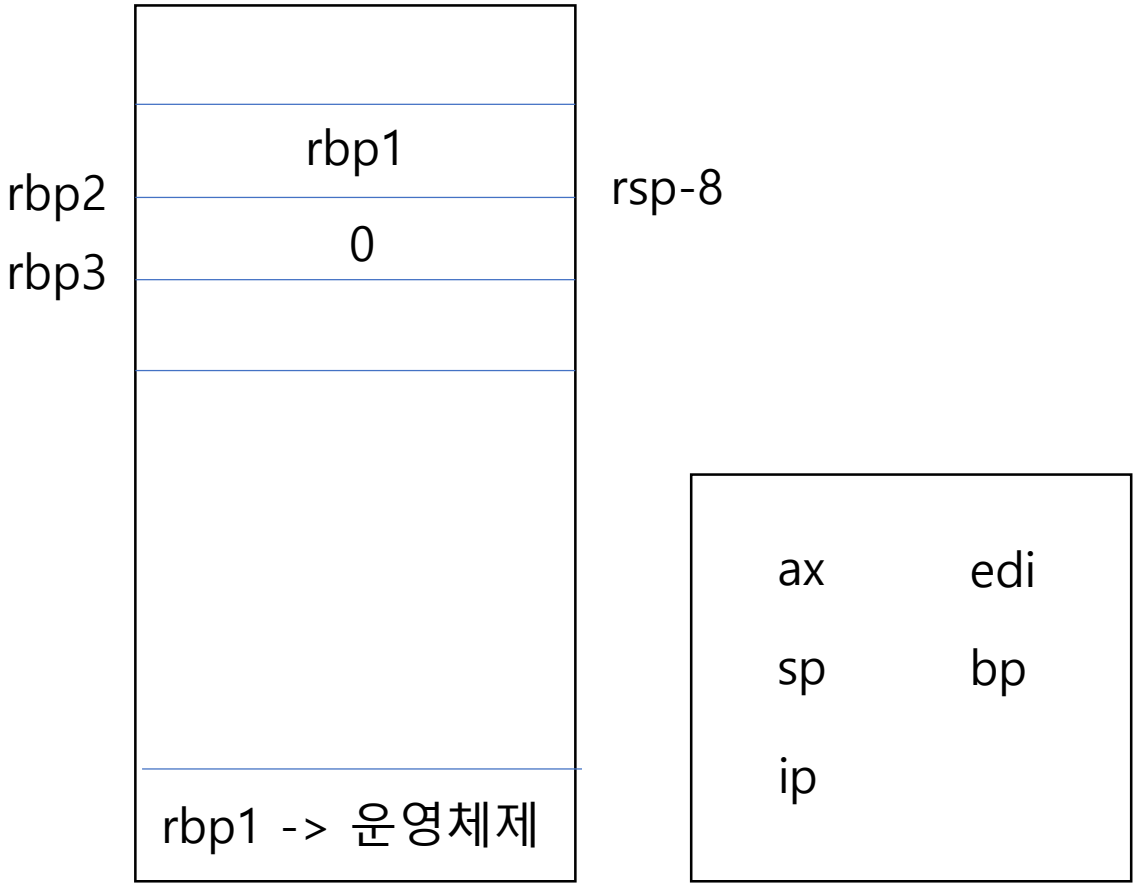


```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```

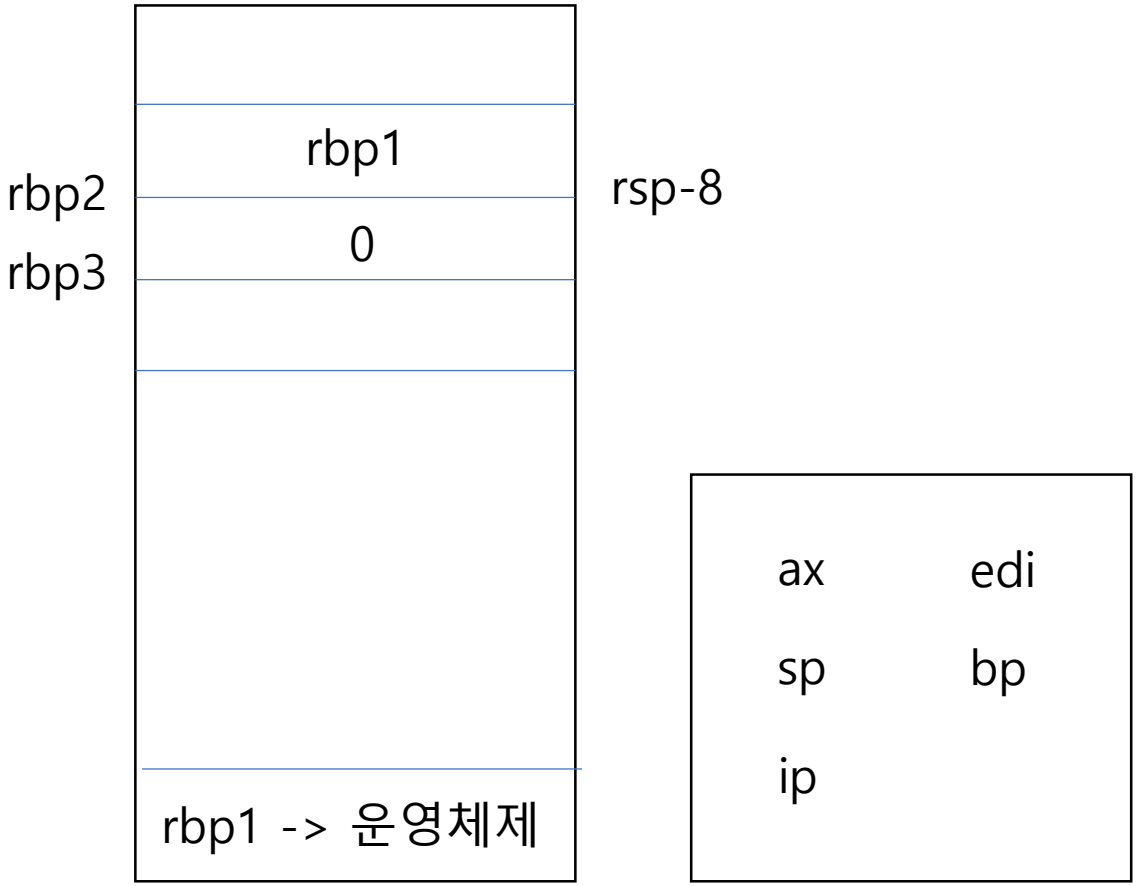




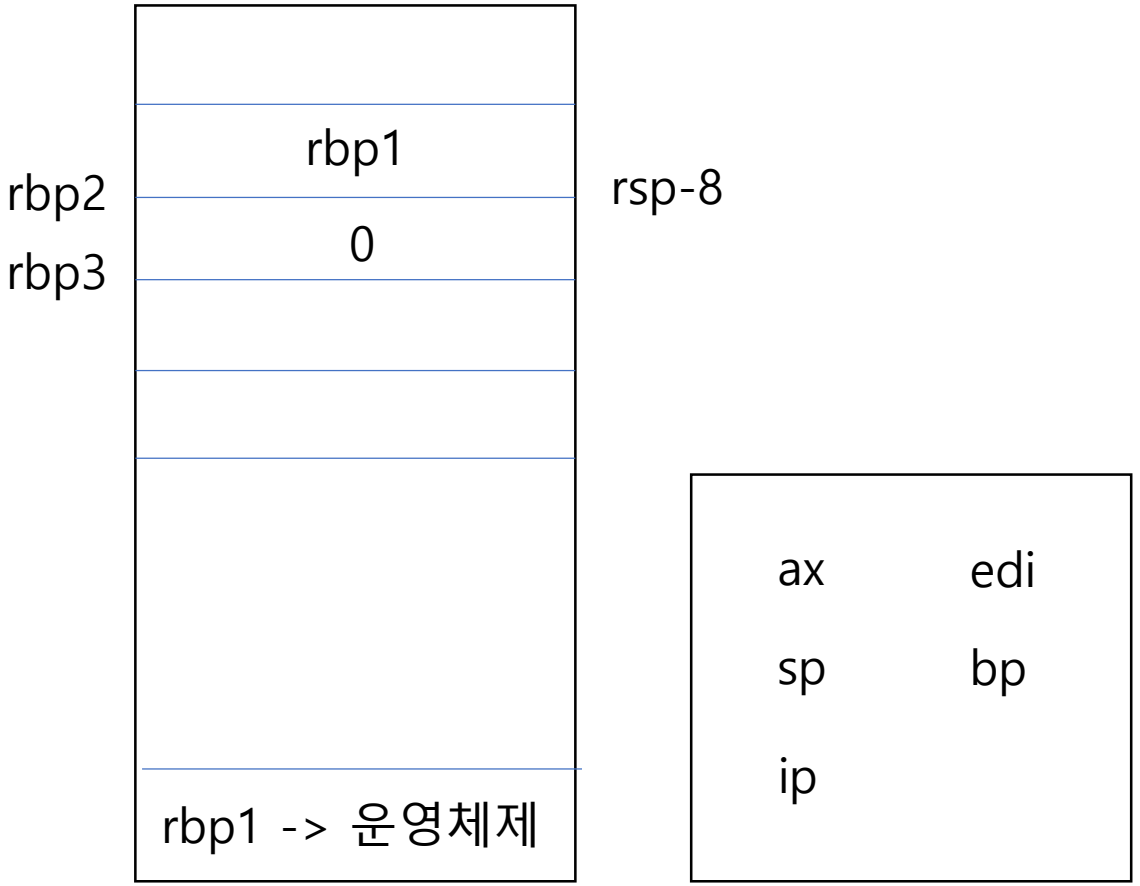
```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
Dump of assembler code for function main:
   0x00000000004004e4 <+0>:     push   %rbp
   0x00000000004004e5 <+1>:     mov    %rsp,%rbp
   0x00000000004004e8 <+4>:     sub    $0x10,%rsp
   0x00000000004004ec <+8>:     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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```



```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```



```
mhn@mhn-900X3L: ~/my_proj/c
(gdb) disas
No frame selected.
(gdb) disassemble main
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 <mult2>
   0x0000000000400516 <+50>:    mov    %eax,-0x4(%rbp)
   0x0000000000400519 <+53>:    mov    $0x0,%eax
   0x000000000040051e <+58>:    leaveq
   0x000000000040051f <+59>:    retq
End of assembler dump.
(gdb)
```



더 공부하겠습니다...

문제 10. 구구단을 만들어보시오.

<코드>

mhn@mhn-900X3L: ~/my_proj/c

```c
#include <stdio.h>

int main(){

        int num1=1,num2;

        while(num1 <= 9){ // 1~9단까지 반복
                num2=1; //9까지 곱해진 후 다시 1을 곱하기 위해
                while(num2 <= 9){ //1~9까지 곱하기 반복
                        printf("%d * %d = %d\n",num1,num2,num1*num2); //출력
                        num2++; //곱할 값 증가
                }printf("\n");//단이 끝난 후 띄어쓰기
                num1++; //9까지 곱해진 후 다음 단으로 넘어가기 위해
        }
        return 0;
}
```

<결과>

```
mhn@mhn-900X3
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

3 * 1 = 3
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 * 1 = 4
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 * 1 = 5
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 * 1 = 6
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 * 1 = 7
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 * 1 = 8
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 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

- 중첩 While문 사용
  안쪽 while문이 끝나면 바깥쪽 While문을 실행한다.

문제 12. 리눅스에서 Debugging하는 방법에 대해 기술해보시오.
Break Point는 어떻게 잡으며, 조사식, 메모리, 레지스터등의 디버그 창은
각각 어떤 역할을 하고 무엇을 알고자 할 때 유용한지 기술하시오.
gdb 상에서 아직 소개하지 않은 명령들 bt, c 이 2 개에 대해 조사해보고 활용해보자 ~

- 디버거의 목적은 다른 프로그램 수행 중에 그 프로그램 '내부에서' 무슨 일이 일어나고 있는지 보여주거나 프로그램이 잘못 실행되었을 때 무슨 일이 일어나고 있는지 보여주는 것

- Debugging상태 들어가기
  gcc –g –o debug 파일명.c : 디버깅 + 컴파일
  gcc –g –o0 –o debug 파일명.c : 분석을 쉽게 하기 위해 최적화를 방지한다.
  gdb 파일명 : gdb 실행

- 명령어
  b *주소 : Break Point를 주소에 건다.
  disassemble 주소/함수명 : 해당 함수를 디스어셈블하여 실행한다.
  list (l) : 소스를 출력한다.
  r : 디버깅 실행
  si : 명령어 한 줄을 실행한다.
  bt : 오류가 발생한 함수를 역으로 찾아간다.
  c : 브레이크 포인트를 만날 때 까지 계속 진행한다.
  quit (q) : GDB 를 종료한다.

bt : 오류가 발생한 함수를 역으로 찾아간다.

```
(gdb) bt
#0  main () at func1.c:12
(gdb) l
7               // return num << 1;
8       }
9
10      int main(void)
11      {
12              int num = 3, res;
13              res = myfunc(num);
14              printf("res = %d\n", res);
15
16              return 0;
(gdb)
```

c : 브레이크 포인트를 만날 때 까지 계속 진행한다.

```
mhn@mhn-900X3L: ~/Homework/sanghoonlee
(gdb) disas
No frame selected.
(gdb) disas main
Dump of assembler code for function main:
   0x0000000000400535 <+0>:     push   %rbp
   0x0000000000400536 <+1>:     mov    %rsp,%rbp
   0x0000000000400539 <+4>:     sub    $0x10,%rsp
   0x000000000040053d <+8>:     movl   $0x3,-0x8(%rbp)
   0x0000000000400544 <+15>:    mov    -0x8(%rbp),%eax
   0x0000000000400547 <+18>:    mov    %eax,%edi
   0x0000000000400549 <+20>:    callq  0x400526 <myfunc>
   0x000000000040054e <+25>:    mov    %eax,-0x4(%rbp)
   0x0000000000400551 <+28>:    mov    -0x4(%rbp),%eax
   0x0000000000400554 <+31>:    mov    %eax,%esi
   0x0000000000400556 <+33>:    mov    $0x4005f4,%edi
   0x000000000040055b <+38>:    mov    $0x0,%eax
   0x0000000000400560 <+43>:    callq  0x400400 <printf@plt>
   0x0000000000400565 <+48>:    mov    $0x0,%eax
   0x000000000040056a <+53>:    leaveq
   0x000000000040056b <+54>:    retq
End of assembler dump.
```

```
(gdb) b *0x0000000000400549
```

```
(gdb) c
Continuing.

Breakpoint 2, 0x0000000000400549 in main () at func1.c:13
13              res = myfunc(num);
(gdb)
```

- [메모리 계층 구조]

<속도>	<용량>
1. 레지스터	1. 디스크
2. 캐시	2. 메모리
3. 메모리	3. 캐시
4. 디스크	4. 레지스터

- [x86 범용 레지스터들]
  ax: 함수의 return 값을 저장함
  cx: 무언가를 반복하고자 할 때 사용
  bp: 스택의 기준점
  sp: 스택의 최상위점
  ip: 다음에 실행할 명령어의 주소