과제 1 – 1번 문제 (for 문으로 변경)

hyunwoopark@hyunwoopark-P65-P67SG: ~/hw

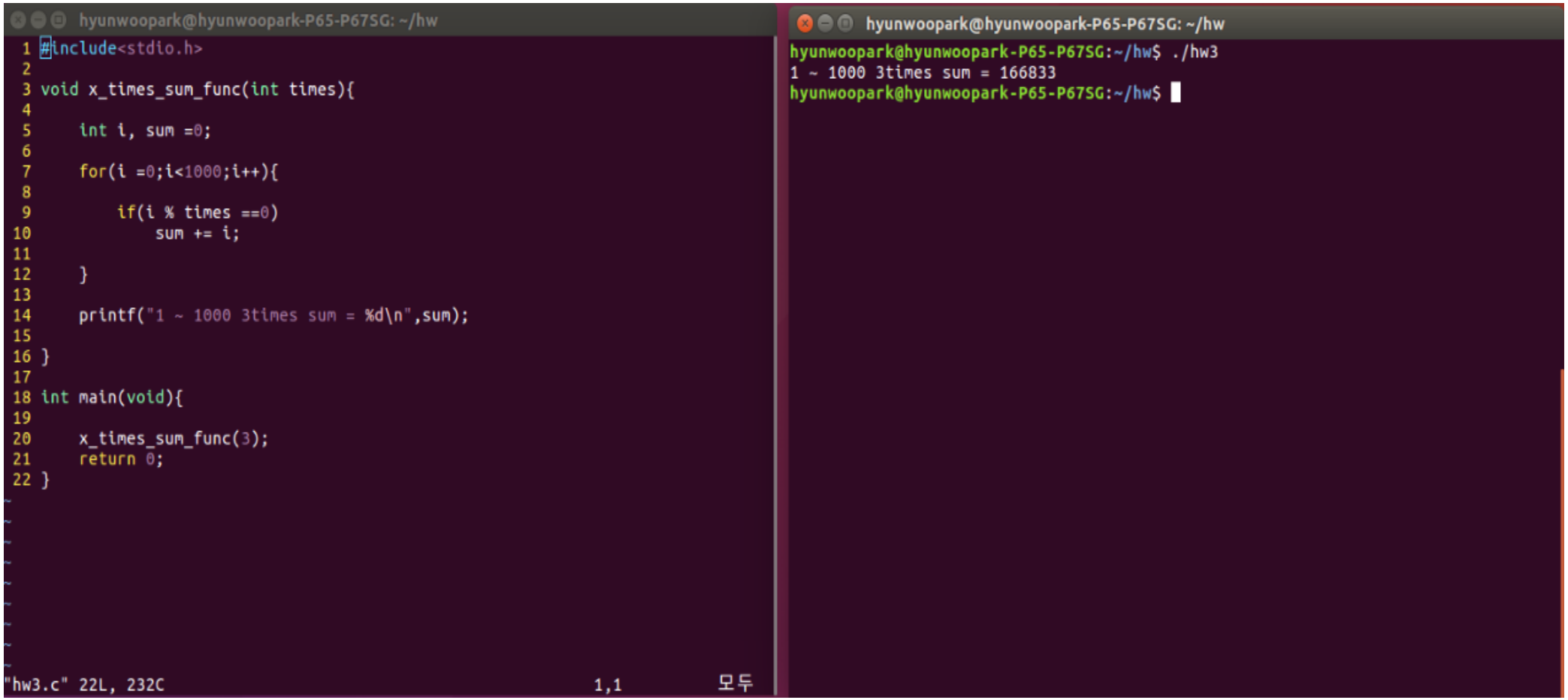
```
1  #include<stdio.h>
2
3  void discount_func(int date, int price){
4
5      int i;
6      int dis_price = price;
7
8      int tot_price;
9
10     if(date>=3)
11         dis_price *= 0.8;
12
13     for(i =0; i<date;i++){
14
15        tot_price += dis_price;
16     }
17
18     printf("total_fee = %d\n", tot_price);
19 }
20
21 int main(void){
22
23   discount_func(7,37500);
24
25   return 0;
26 }
```

1,1 모두

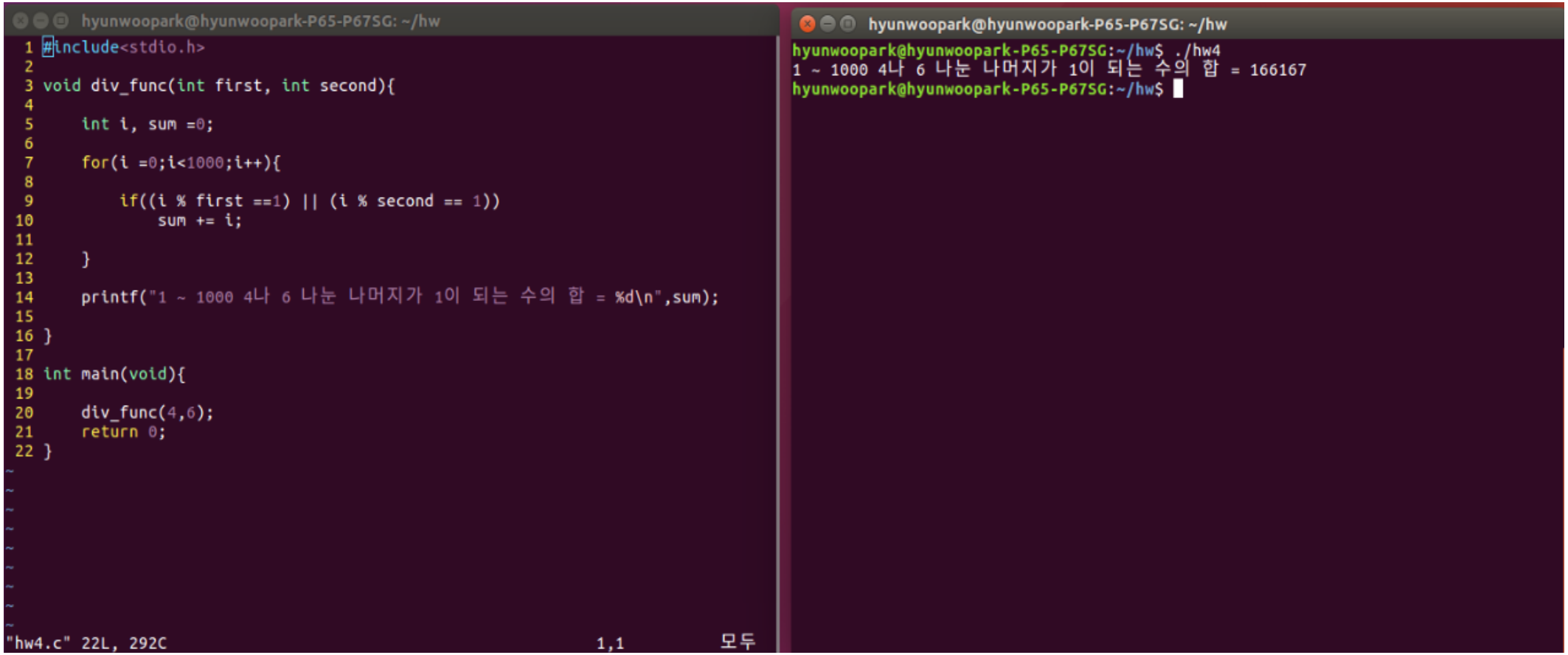
hyunwoopark@hyunwoopark-P65-P67SG: ~/hw

```
hyunwoopark@hyunwoopark-P65-P67SG:~$ cd hw
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$ clear
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$ ./hw1
total_fee = 210000
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$
```

과제 1 – 3번 문제 (for 문으로 변경)



과제 1 – 4번 문제 (for 문으로 변경)



과제 1 – 10번 문제 (for 문으로 변경)

hyunwoopark@hyunwoopark-P65-P67SG: ~/hw

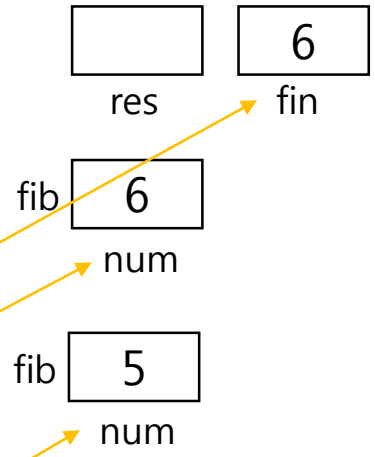
```
 1 #include<stdio.h>
 2
 3 void gugu_func(int num){
 4
 5     int i,j;
 6
 7     for(i=0; i<num-1;i++){
 8
 9         for(j=0; j<num;j++){
10
11             printf("%d * %d = %d\n", i+2, j+1, (i+2) * (j+1));
12
13         }
14         j=0;
15     }
16
17 }
18
19
20 int main(void){
21
22     gugu_func(9);
23     return 0;
24
25 }
~
~
~
~
~
~
"hw10.c" 25L, 225C                                1,1           모두
```

hyunwoopark@hyunwoopark-P65-P67SG: ~/hw

```
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
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$
```

과제 2 – fib 함수 동작 분석 (디버깅 및 그림 그리기)

```
Breakpoint 1, main () at fib.c:12
12      int main(void){
(gdb) l
7               else
8                       return fib(num-1) + fib(num-2);
9
10      }
11
12      int main(void){
13
14              int result, final_val;
15              printf("피보나치 수열의 항의 개수를 입력하시오: ");
16              scanf("%d", &final_val);
(gdb) n
15              printf("피보나치 수열의 항의 개수를 입력하시오: ");
(gdb) n
16              scanf("%d", &final_val);
(gdb)
피보나치 수열의 항의 개수를 입력하시오: 6
17              result = fib(final_val);
(gdb) s
fib (num=6) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=6) at fib.c:5
#1  0x0000000000400680 in main () at fib.c:17
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=5) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=5) at fib.c:5
#1  0x0000000000400622 in fib (num=6) at fib.c:8
#2  0x0000000000400680 in main () at fib.c:17
(gdb)
```



과제 2 – fib 함수 동작 분석 (디버깅 및 그림 그리기)

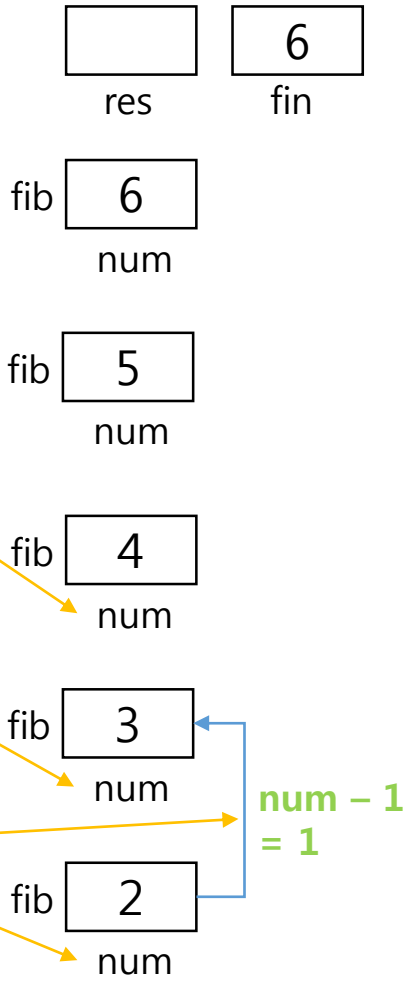
```
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=4) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=4) at fib.c:5
#1  0x0000000000400622 in fib (num=5) at fib.c:8
#2  0x0000000000400622 in fib (num=6) at fib.c:8
#3  0x0000000000400680 in main () at fib.c:17
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=3) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=3) at fib.c:5
#1  0x0000000000400622 in fib (num=4) at fib.c:8
#2  0x0000000000400622 in fib (num=5) at fib.c:8
#3  0x0000000000400622 in fib (num=6) at fib.c:8
#4  0x0000000000400680 in main () at fib.c:17
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=2) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=2) at fib.c:5
#1  0x0000000000400622 in fib (num=3) at fib.c:8
#2  0x0000000000400622 in fib (num=4) at fib.c:8
#3  0x0000000000400622 in fib (num=5) at fib.c:8
#4  0x0000000000400622 in fib (num=6) at fib.c:8
#5  0x0000000000400680 in main () at fib.c:17
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) bt
#0  fib (num=2) at fib.c:10
#1  0x0000000000400622 in fib (num=3) at fib.c:8
#2  0x0000000000400622 in fib (num=4) at fib.c:8
#3  0x0000000000400622 in fib (num=5) at fib.c:8
#4  0x0000000000400622 in fib (num=6) at fib.c:8
#5  0x0000000000400680 in main () at fib.c:17
(gdb) s
fib (num=1) at fib.c:5
5               if(num ==1 || num ==2)
(gdb)
```

4번 째

5번 째

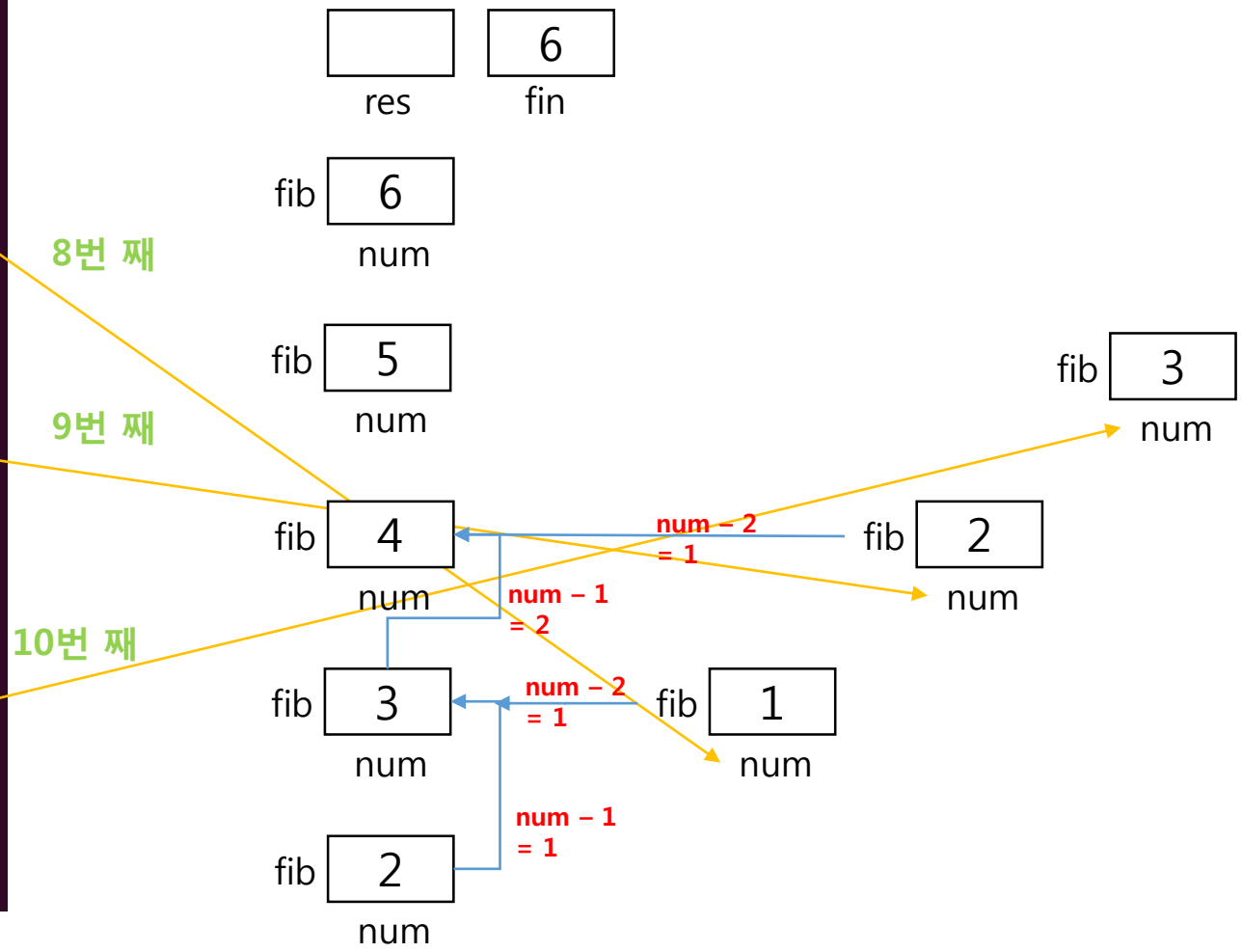
6번 째

7번 째



과제 2 – fib 함수 동작 분석 (디버깅 및 그림 그리기)

```
(gdb) bt
#0  fib (num=1) at fib.c:5
#1  0x0000000000400631 in fib (num=3) at fib.c:8
#2  0x0000000000400622 in fib (num=4) at fib.c:8
#3  0x0000000000400622 in fib (num=5) at fib.c:8
#4  0x0000000000400622 in fib (num=6) at fib.c:8
#5  0x0000000000400680 in main () at fib.c:17
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) s
10      }
(gdb) s
fib (num=2) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=2) at fib.c:5
#1  0x0000000000400631 in fib (num=4) at fib.c:8
#2  0x0000000000400622 in fib (num=5) at fib.c:8
#3  0x0000000000400622 in fib (num=6) at fib.c:8
#4  0x0000000000400680 in main () at fib.c:17
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) s
10      }
(gdb) s
fib (num=3) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=3) at fib.c:5
#1  0x0000000000400631 in fib (num=5) at fib.c:8
#2  0x0000000000400622 in fib (num=6) at fib.c:8
#3  0x0000000000400680 in main () at fib.c:17
(gdb)
```



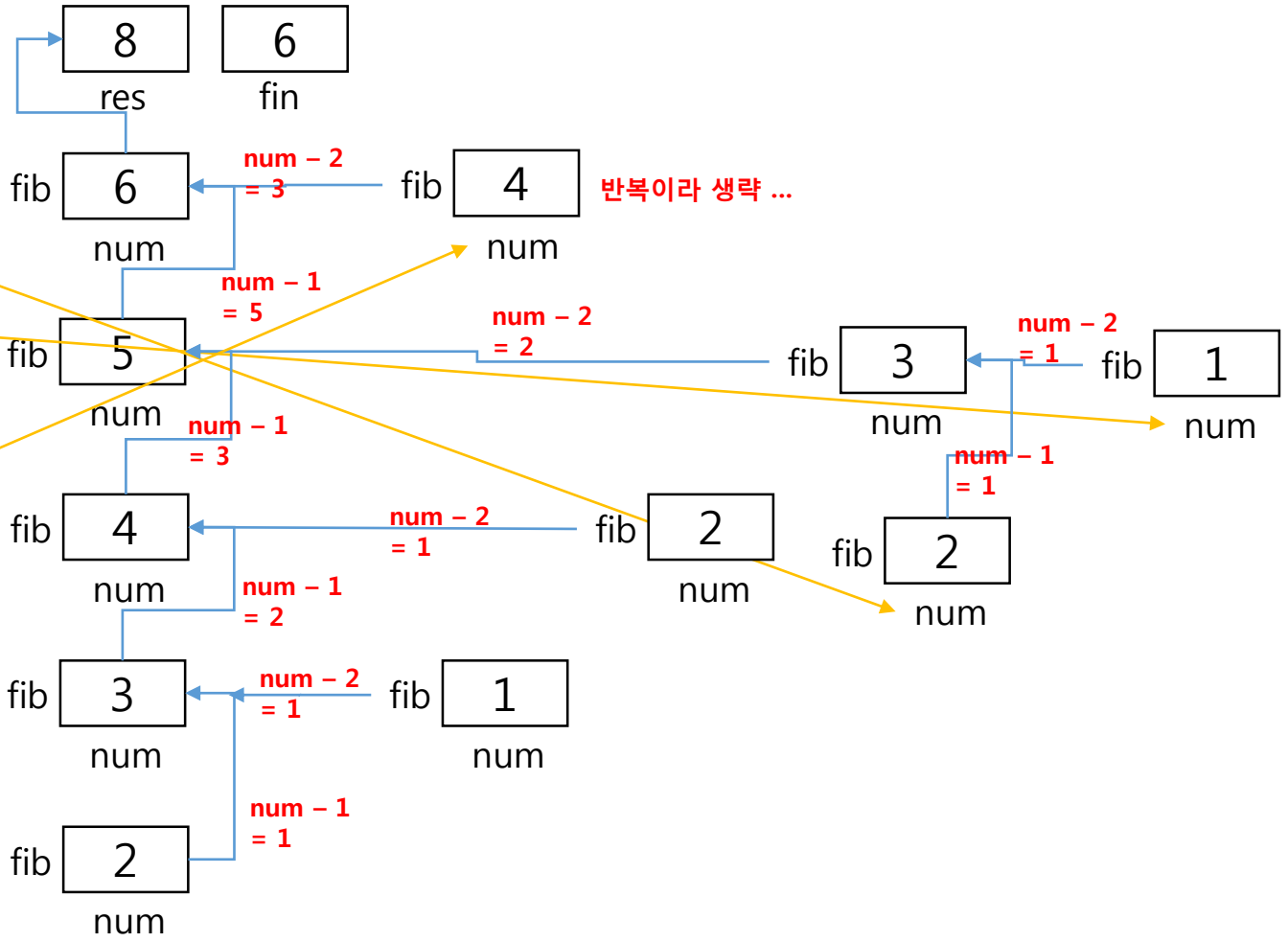
과제 2 – fib 함수 동작 분석 (디버깅 및 그림 그리기)

```
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=2) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) s
fib (num=1) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=1) at fib.c:5
#1  0x0000000000400631 in fib (num=3) at fib.c:8
#2  0x0000000000400631 in fib (num=5) at fib.c:8
#3  0x0000000000400622 in fib (num=6) at fib.c:8
#4  0x0000000000400680 in main () at fib.c:17
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) s
10      }
(gdb) s
10      }
(gdb) s
fib (num=4) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=4) at fib.c:5
#1  0x0000000000400631 in fib (num=6) at fib.c:8
#2  0x0000000000400680 in main () at fib.c:17
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) s
fib (num=3) at fib.c:5
5               if(num ==1 || num ==2)
(gdb) s
8                       return fib(num-1) + fib(num-2);
(gdb) bt
#0  fib (num=3) at fib.c:8
#1  0x0000000000400622 in fib (num=4) at fib.c:8
#2  0x0000000000400631 in fib (num=6) at fib.c:8
#3  0x0000000000400680 in main () at fib.c:17
(gdb)
```

11번 째

12번 째

13번 째



과제 3 – 배운 내용 복습(goto, 파이프라인, for)

- goto 문법의 이점

예를 들어, for문을 중복해서 사용하고 if와 break로 for문에서 빠져 나올 때, 상당히 많은 불필요한 명령어들이 사용된다.

이러한 단점을 보완하기 위해 goto 문법을 사용하면 불필요한 코드를 사용하지 않고 한 번에 for문을 빠져나올 수 있다.

If와 break를 사용하면 기본적으로 mov, cmp, jmp를 해야한다. 하지만, goto는 jmp 하나로 끝난다.

또한, for문과 if,break를 여러 개 조합을 할수록 mov, cmp, jmp가 늘어난다. 여기서 문제는 jmp임.

Call 이나 jmp를 cpu instruction 레벨에서 분기 명령어라고 하고 이들은 cpu 파이프라인에 치명적인 손실을 가져다 준다.

과제 3 – 배운 내용 복습(goto, 파이프라인, for)

- call, jmp 등 분기 명령어의 문제점.

기본적으로 분기 명령어는 파이프라인을 부순다.
이 뜻은 위의 가장 단순한 cpu가 실행까지 3 clock을 소요하는데
기존의 파이프라인이 깨지니
쓸데없이 또 다시 3clock을 버려야 함을 의미한다.

만약 이러한 파이프라인의 단계가 수십 단계라면

여기서 낭비되는 cpu clock이 수없이 많음.

즉, 성능면으로만 보아도 goto가 월등히 좋다는 것을 알 수 있다.

과제 3 – 배운 내용 복습(goto, 파이프라인, for)

- for 문

While문을 사용시 번거로움을 간소화 시킨 것이 for문이다.

for (초기화; 조건; 증감) 으로 문법을 사용한다.

예를 들어,

Int I =0;

While(i<5){

    i++;
}

를 for문으로 바꾸면

For(i =0; i<5; i++) 과 같이 간략하게 표현할 수 있다.