IOT 환경의 임베디드 개발자 양성과정

4일차 과제 문한나

4일차 내용 정리

- 변수 초기화
  연산을 하는 작업에서는 값을 초기화 시켜주는 작업이 필요하다
  변수 선언 시 주소에 알 수 없는 값이 저장되어 있기 때문

- 전역변수
  함수 밖에 있으며 프로그램 전체에서 공유되는 변수
  어떤 변수 영역 내에서도 접근할 수 있다

- 지역변수
  특정 함수 안에 존재하는 변수
  변수 영역이 정해져 있다

- 정적변수
  정적으로 할당되는 변수이며, 프로그램 실행 전반에 걸쳐 변수의 수명이 유지된다.
  static 사용

- goto

  프로그램의 어느 부분에서 다른 부분으로 건너뛸 때 사용하는 명령어

  Goto는 jmp하나로 이루어져 있는데 이것은 큰 이점

  If문은 mov, cmp, jmp로 이루어져 있어 jmp명령어를 쓸 때 cpu의 파이프라인에서

  치명적인 손실을 가져다 줌

- 파이프라인

  명령어를 읽어 순차적으로 실행하는 프로세서에 적용되는 기술로,

  한 번에 하나의 명령어만 실행하는 것이 아니라 하나의 명령어가 실행되는 도중에

  다른 명령어 실행을 시작하는 식으로 동시에 여러 개의 명령어를 실행하는 기법이다.

  CPU 의 파이프라인을 설명하자면 아래와 같은 3 단계로 구성된다(가장 단순,수십단계까지 존재)

  1. Fetch – 실행해야 할 명령어를 물어옴
  2. Decode - 어떤 명령어인지 해석함
  3. Execute - 실제 명령어를 실행시킴

While로 짠 코드를 for문으로 바꾸기

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/4_h

```c
#include <stdio.h>

// input: first - day, second - 37500
int borrow_equip(int day, double money)
{
        int i = 0, res = 0;
        double rate = 1.0;
        double tmp = 0;

        if(day >= 3)
        {
                rate = 0.8;
                tmp = money * rate;
        }

        for(i=0;i<7;i++){
                res += tmp;
                printf("res = %d\n",res);
        }

        return res;
}

int main(void)
{
        printf("res = %d\n", borrow_equip(7, 37500));
        return 0;
}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c/4_h$ ./a.out
res = 30000
res = 60000
res = 90000
res = 120000
res = 150000
res = 180000
res = 210000
res = 210000
```

문제 3. 1 ~ 1000사이에 3의 배수의 합을 구하시오.

<코드>

mhn@mhn-900X3L: ~/my_proj/c/4_h

```c
#include <stdio.h>

// synthesis: first - start, second - end, three - times
int syn(int start, int end, int times)
{
        int res = 0, i = start;

        for(i=start;i<end+1;i++){
                if(!(i%3)){
                        res += i;
                }
        }
        return res;
}

int main(void)
{
        printf("tot series sum = %d\n", syn(1, 1000, 3));
        return 0;
}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c/4_h$ ./a.out
tot series sum = 166833
```

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

<코드>

mhn@mhn-900X3L: ~/my_proj/c/4_h

```c
#include <stdio.h>

int syn(int start, int end, int t1, int t2)
{
        int res = 0, i = start;

        for(i=start;i<end+1;i++){
                if(((i % 4) == 1) || ((i % 6) == 1)){
                        res += i;
                }
        }
        return res;
}

int main(void)
{
        printf("tot series sum = %d\n", syn(1, 1000, 4, 6));
        return 0;
}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c/4_h$ ./a.out
tot series sum = 166167
```

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/4_h

```c
#include <stdio.h>

void print_seven_series(int num)
{
        int i = 1;

        for(i=i;i<num+1;i++)
        {
                if(i < num)
                {
                        printf("%d\t", i * 7);
                }
                else
                {
                        printf("%d\n", i * 7);
                }
        }
}

int main(void)
{
        int num;
        scanf("%d", &num);
        print_seven_series(num);
        return 0;
}
```

<결과>

```
mhn@mhn-900X3L:~/my_proj/c/4_h$ ./a.out
5
7       14      21      28      35
```

문제 10. 구구단을 만들어보시오.

<코드>

mhn@mhn-900X3L: ~/my_proj/c/4_h

```c
#include <stdio.h>

void print_rom(void)
{
        int i = 2, j = 1;

        for(i=2;i<10;i++)
        {
                for(j=1;j<10;j++)
                {
                        printf("%d x %d = %d\n", i, j, i * j);

                }
                j = 1;
        }
}

int main(void)
{
        print_rom();
        return 0;
}
```

<결과>

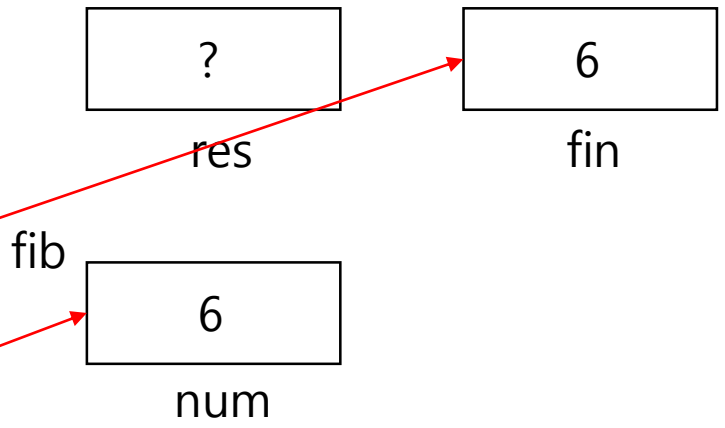
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

```
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
```

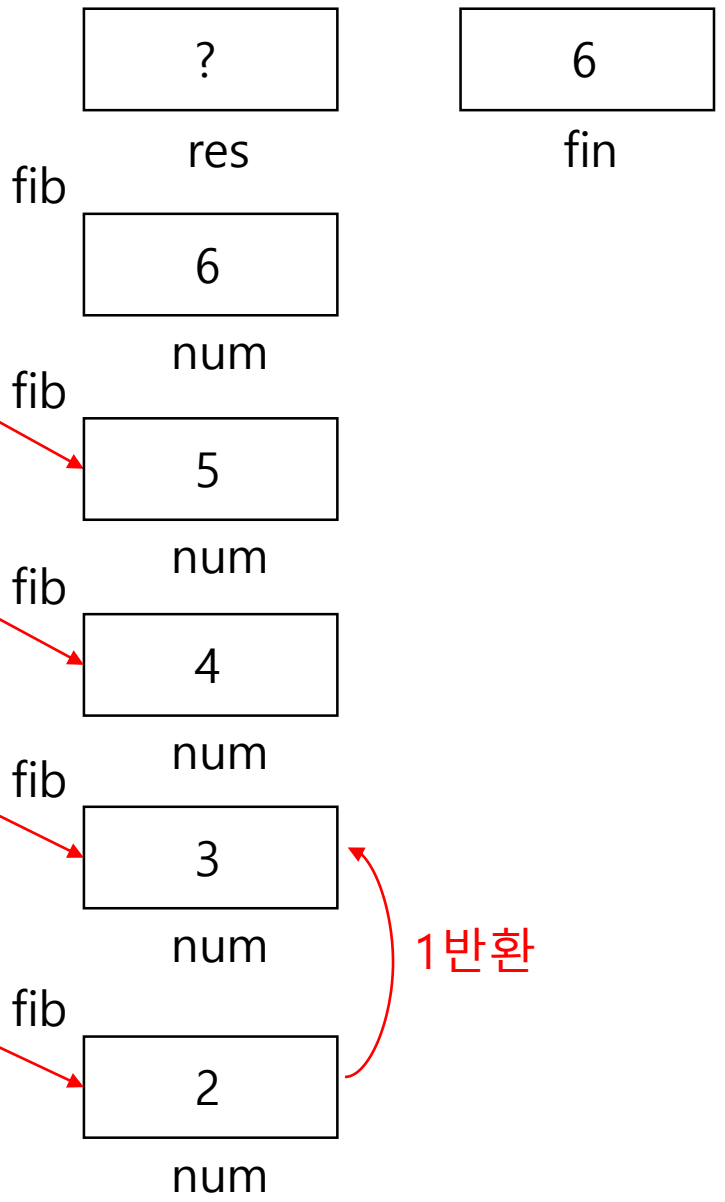
```
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

fib 함수 동작 분석(디버깅 및 그림 그리기)

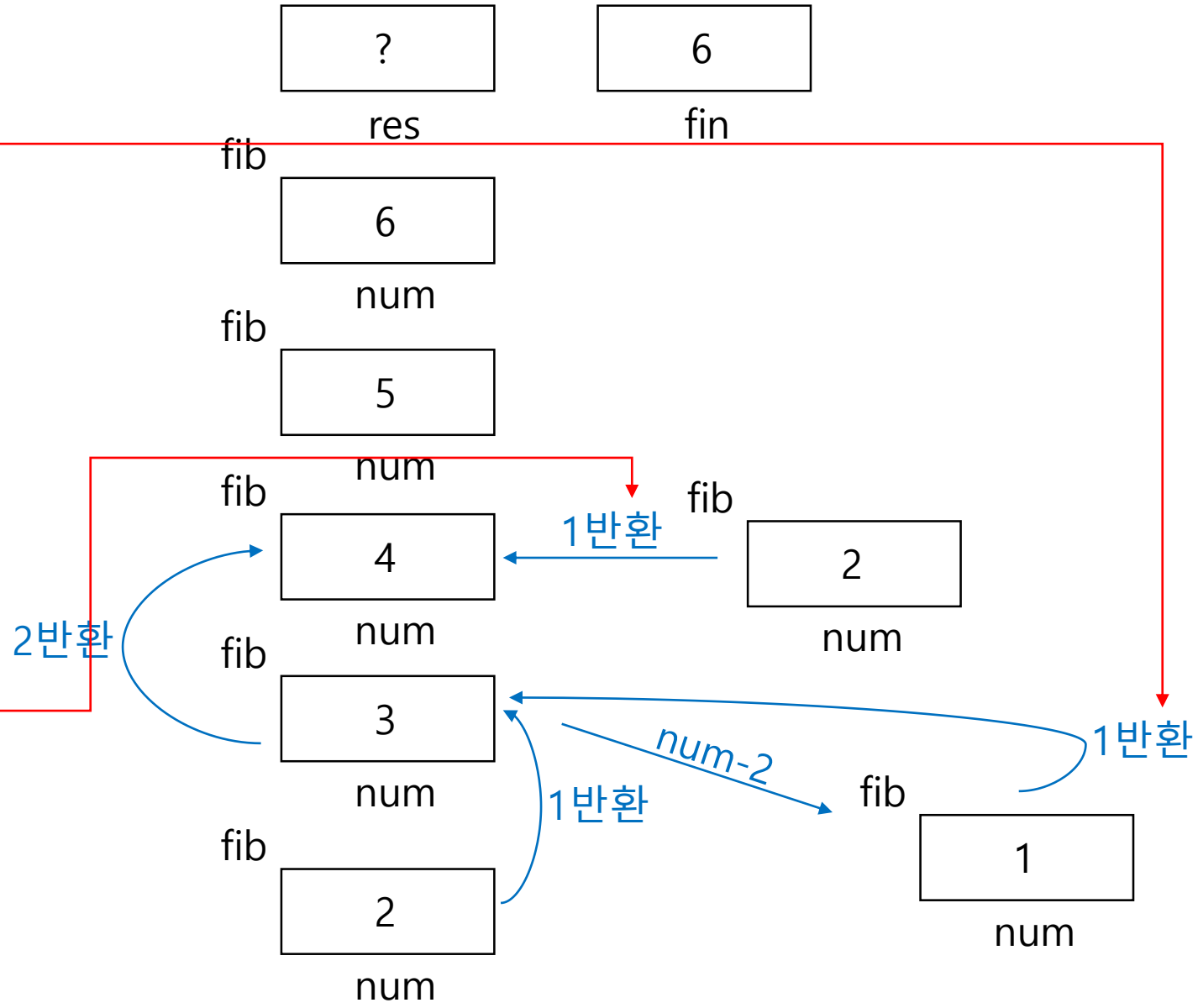
```
(gdb) l
7               else
8                       return fib(num-1)+fib(num-2);
9
10      }
11
12      int main(){
13
14              int result,final_val;
15              printf("피보나치 수열의 항의 개수를 입력하세요: ");
16              scanf("%d",&final_val);
(gdb) s
15              printf("피보나치 수열의 항의 개수를 입력하세요: ");
(gdb) n
16              scanf("%d",&final_val);
(gdb) n
피보나치 수열의 항의 개수를 입력하세요: 6
17              result = fib(final_val);
(gdb) s
fib (num=6) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) s
8                       return fib(num-1)+fib(num-2);
(gdb) bt
#0  fib (num=6) at test36.c:8
#1  0x0000000000400680 in main () at test36.c:17
(gdb) s
fib (num=5) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) s
```



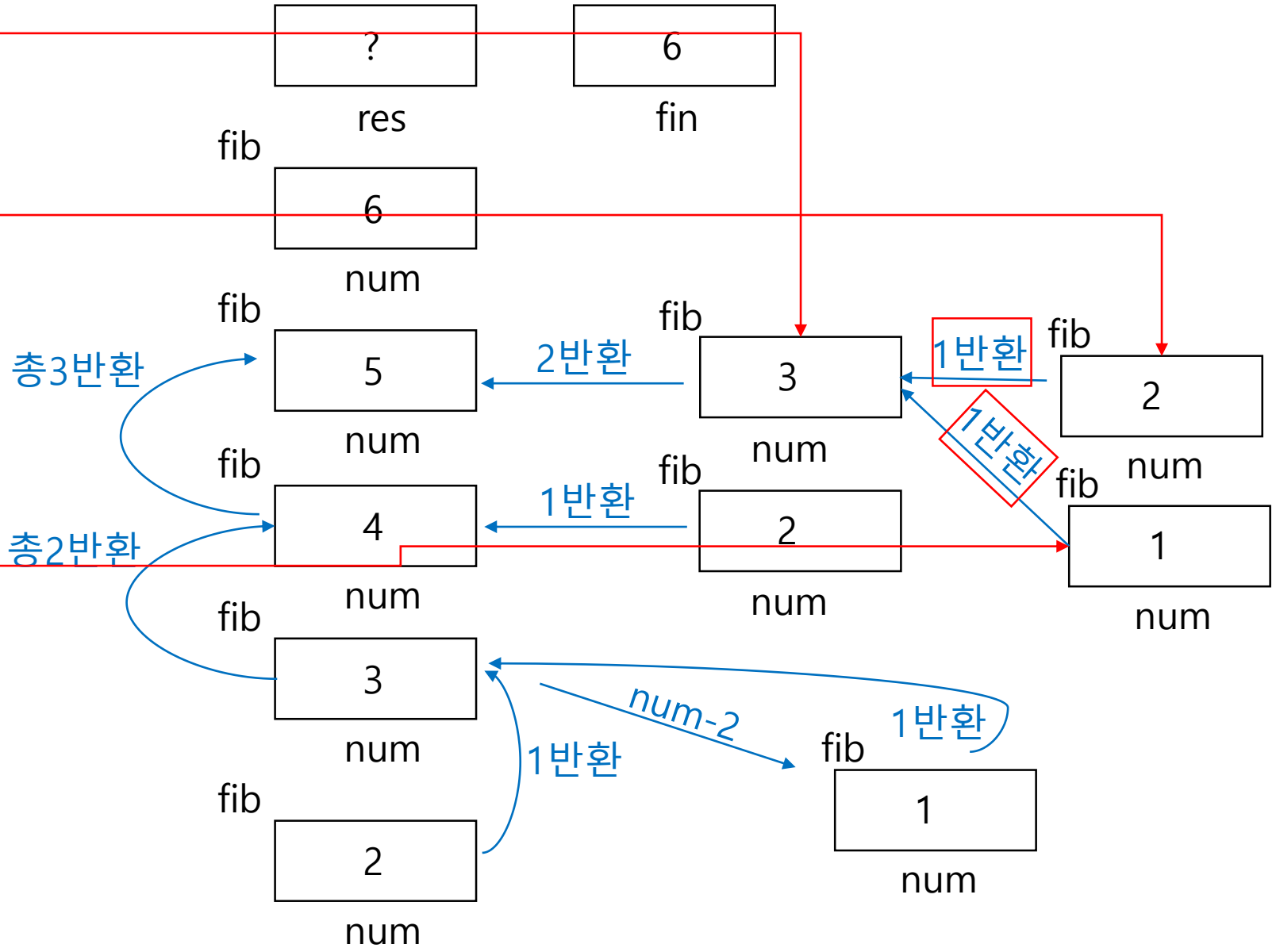
```
(gdb) bt
#0  fib (num=5) at test36.c:8
#1  0x0000000000400622 in fib (num=6) at test36.c:8
#2  0x0000000000400680 in main () at test36.c:17
(gdb) s
fib (num=4) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) s
8                       return fib(num-1)+fib(num-2);
(gdb) bt
#0  fib (num=4) at test36.c:8
#1  0x0000000000400622 in fib (num=5) at test36.c:8
#2  0x0000000000400622 in fib (num=6) at test36.c:8
#3  0x0000000000400680 in main () at test36.c:17
(gdb) s
fib (num=3) at test36.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 test36.c:8
#1  0x0000000000400622 in fib (num=4) at test36.c:8
#2  0x0000000000400622 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
fib (num=2) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) s
6                       return 1;
(gdb) bt
#0  fib (num=2) at test36.c:6
#1  0x0000000000400622 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) bt
#0  fib (num=2) at test36.c:10
#1  0x0000000000400622 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
fib (num=1) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) bt
#0  fib (num=1) at test36.c:5
#1  0x0000000000400631 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
6                       return 1;
(gdb) bt
```

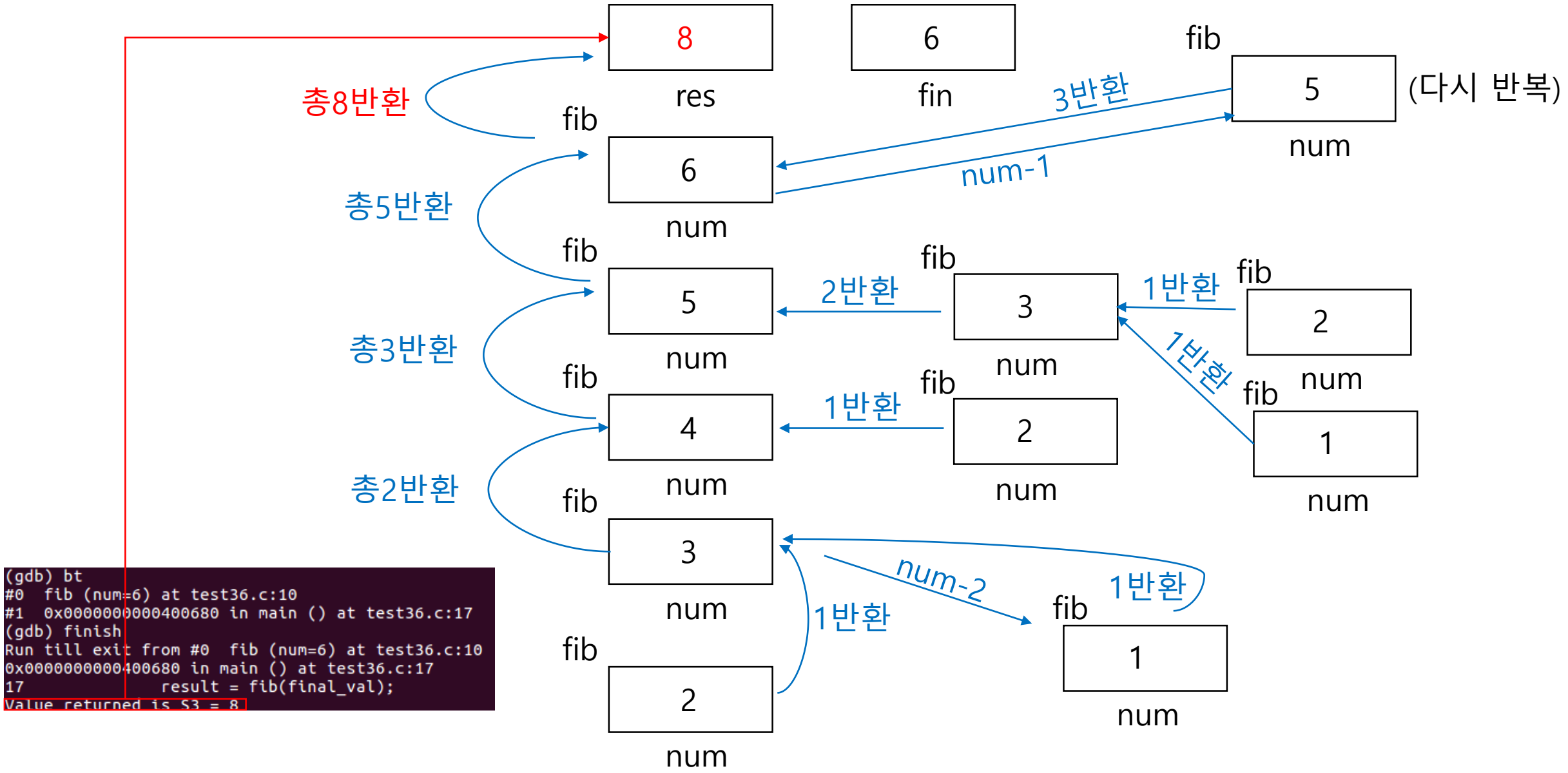


```
(gdb) bt
#0  fib (num=1) at test36.c:5
#1  0x0000000000400631 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
6                   return 1;
(gdb) bt
#0  fib (num=1) at test36.c:6
#1  0x0000000000400631 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) bt
#0  fib (num=1) at test36.c:10
#1  0x0000000000400631 in fib (num=3) at test36.c:8
#2  0x0000000000400622 in fib (num=4) at test36.c:8
#3  0x0000000000400622 in fib (num=5) at test36.c:8
#4  0x0000000000400622 in fib (num=6) at test36.c:8
#5  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) s
fib (num=2) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) bt
#0  fib (num=2) at test36.c:5
#1  0x0000000000400631 in fib (num=4) at test36.c:8
#2  0x0000000000400622 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
6                   return 1;
(gdb) bt
#0  fib (num=2) at test36.c:6
#1  0x0000000000400631 in fib (num=4) at test36.c:8
#2  0x0000000000400622 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) bt
#0  fib (num=2) at test36.c:10
#1  0x0000000000400631 in fib (num=4) at test36.c:8
#2  0x0000000000400622 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) s
fib (num=3) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) bt
```



```
(gdb) bt
#0  fib (num=3) at test36.c:5
#1  0x0000000000400631 in fib (num=5) at test36.c:8
#2  0x0000000000400622 in fib (num=6) at test36.c:8
#3  0x0000000000400680 in main () at test36.c:17
(gdb) s
8                       return fib(num-1)+fib(num-2);
(gdb) s
fib (num=2) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) bt
#0  fib (num=2) at test36.c:5
#1  0x0000000000400622 in fib (num=3) at test36.c:8
#2  0x0000000000400631 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
6                       return 1;
(gdb) bt
#0  fib (num=2) at test36.c:6
#1  0x0000000000400622 in fib (num=3) at test36.c:8
#2  0x0000000000400631 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) s
fib (num=1) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) bt
#0  fib (num=1) at test36.c:5
#1  0x0000000000400631 in fib (num=3) at test36.c:8
#2  0x0000000000400631 in fib (num=5) at test36.c:8
#3  0x0000000000400622 in fib (num=6) at test36.c:8
#4  0x0000000000400680 in main () at test36.c:17
(gdb) s
6                       return 1;
(gdb) s
10      }
(gdb) s
10      }
(gdb) bt
#0  fib (num=3) at test36.c:10
#1  0x0000000000400631 in fib (num=5) at test36.c:8
#2  0x0000000000400622 in fib (num=6) at test36.c:8
#3  0x0000000000400680 in main () at test36.c:17
(gdb) s
10      }
(gdb) s
fib (num=4) at test36.c:5
5               if(num == 1 || num == 2)
(gdb) br
Breakpoint 2 at 0x400602: file test36.c, line 5.
(gdb) bt
#0  fib (num=4) at test36.c:5
```





다음에는 더 깔끔하게 정리하겠습니다.