사물인터넷(IoT/ICT)환경에서의 임베디드 응용SW 개발자 양성과정

정한별

문제 은행 (어제 내용을 for 문으로 작성하시오.)

<1번>

#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("\n일주일간 사용 요금은 res = %d 이다.\n", borrow_equip(7, 37500));
	return 0;
}
```

<3번>

```
#include <stdio.h>

// synthesis: first - start, second - end, three - times
int syn(int start, int end, int times)
{

	int res = 0, i = start;
	for(res=0;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;
}
```

<4번>

```
#include <stdio.h>

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


	for(res=0;i++;i<end+1)
	{
		if(((i % t1) == 1) || ((i % t2) == 1))
		{
			res += i;
		}
	}

	return res;
}

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

<5번>

#include <stdio.h>

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

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

int main(void)
{
	int num;
	printf("보고싶은 7의 배수의 n번째 항은?");
	scanf("%d", &num);
	print_seven_series(num);
	return 0;
}
```

<10번>

```
#include <stdio.h>

void print_rom(void)
{
	int i = 2;
	int 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;
}
```

<학습 내용>

scope란 무엇일까?

'{'로 시작해서 '}'으로 끝나는 영역, 누군가만의 공간.

static 키워드의 역할은?

static 변수는 전역 변수와 마찬가지로 data 역역에 지역 변수를 static을 선언하면 이변수는 변수를 정의한 함수 내에서만 접근 가능.(함수 밖으로 나가면 사용 못함)

static을 함수에 쓰면 다른 파일에서 접근이 불가능 하다.(private 같은 역할)

continue

NaN = x/0, Inf =∞ 일 때, contionue로 제껴라. (실행하다가 걸치면 에러나거나 빠질 부분에 써주면 된다)

do while 은 왜 사용할까?

무조건 한번 사용해야 할 변수나 함수나 등등의 상황이 있을 시에 쓴다.

#define

#define A B : A가 나올 때 B로 대체해서 사용한다.
사용하는 이유: 예를 들어 회사 코드에 100번 루프(반복)을 돌아가게 하는 코드가 777개 있을 때 while 문의 조건을 i<100 -> i<500으로 바꿀 시, 변수로 넣어두면 한번에 바꿀 수 있다.

for 문

리눅스에서 for(int i=0; ;) 의 int는 사용하면 안 된다. 선언은 위에서 한번 해주는 것이 좋다.
for(;;) ->이렇게 쓰면 무한 루프가 된다.

goto 문(goto의 이점과 파이프 라인)

goto는 말이 많은 녀석이다. 잘못 쓰면 코드가 이상해진다. 그래서 예로부터 쓰지말라는 이야기들을 많이 듣는데 실상은 goto를 잘써야 클럭상으로도 많은 효율적 손해를 줄일 수 있다.
(예로 if,break와 goto문을 비교해 보면 알 수 있다.)

if는 ->어셈블리에서 mov, cmp, jmp 등의 명려어를 전달 받는다.
goto는 -> jmp 명령어 하나로 끝난다.

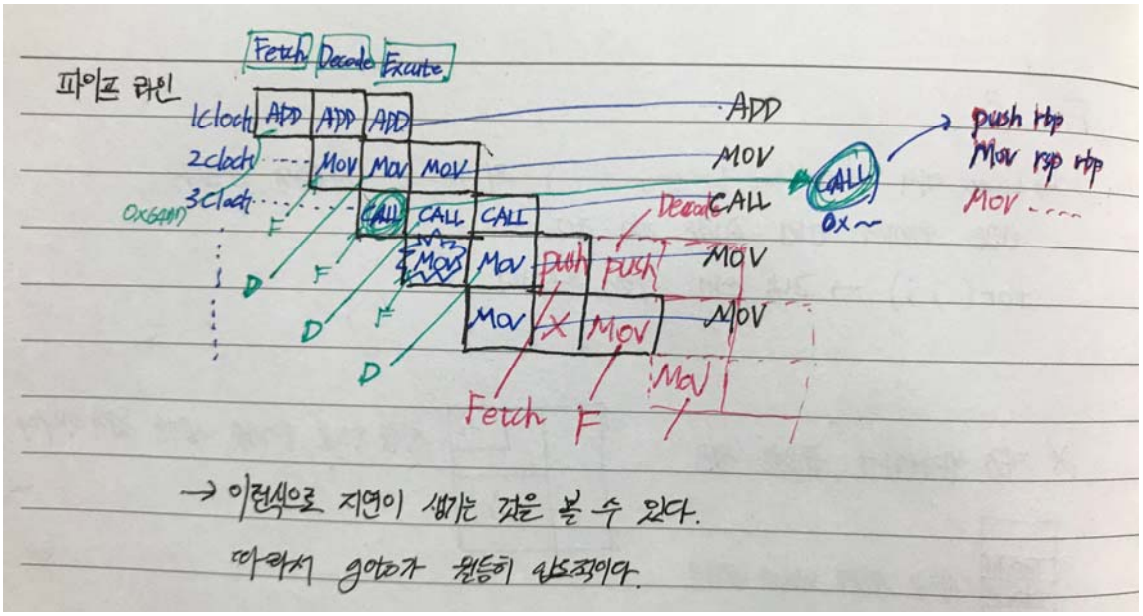
위와 같은 call, jmp 는 CPU instruction(명령어) 레벨에서 분기 명령어라고 하며 기본적으로 3단계로 구성된다.

1. Fetch - 실행한 명령어를 물어옴.
2. Decode - 어떤 명령어 인지 해석함.
3. Excute - 실제 명령어를 실행함.

왜 분기명령어가 문제가 된다고 할까?

분기 명령어는 파이프라인을 부순다.
실행까지 3clock을 낭비한다. 그래서 파이프라인을 깨뜨린다.
수십개의 파이프라인이면 분기 발생마다 cpu clock을 낭비하니 문제가 된다.

<파이프 라인>



****SW와 HW 동작을 생각할 때 주의점****
SW: 멀티코어 상황이 아니면 한번에 한가지 동작만 수행
HW: cpu는 오로지 한 순간에 1가지 동작만 한다.

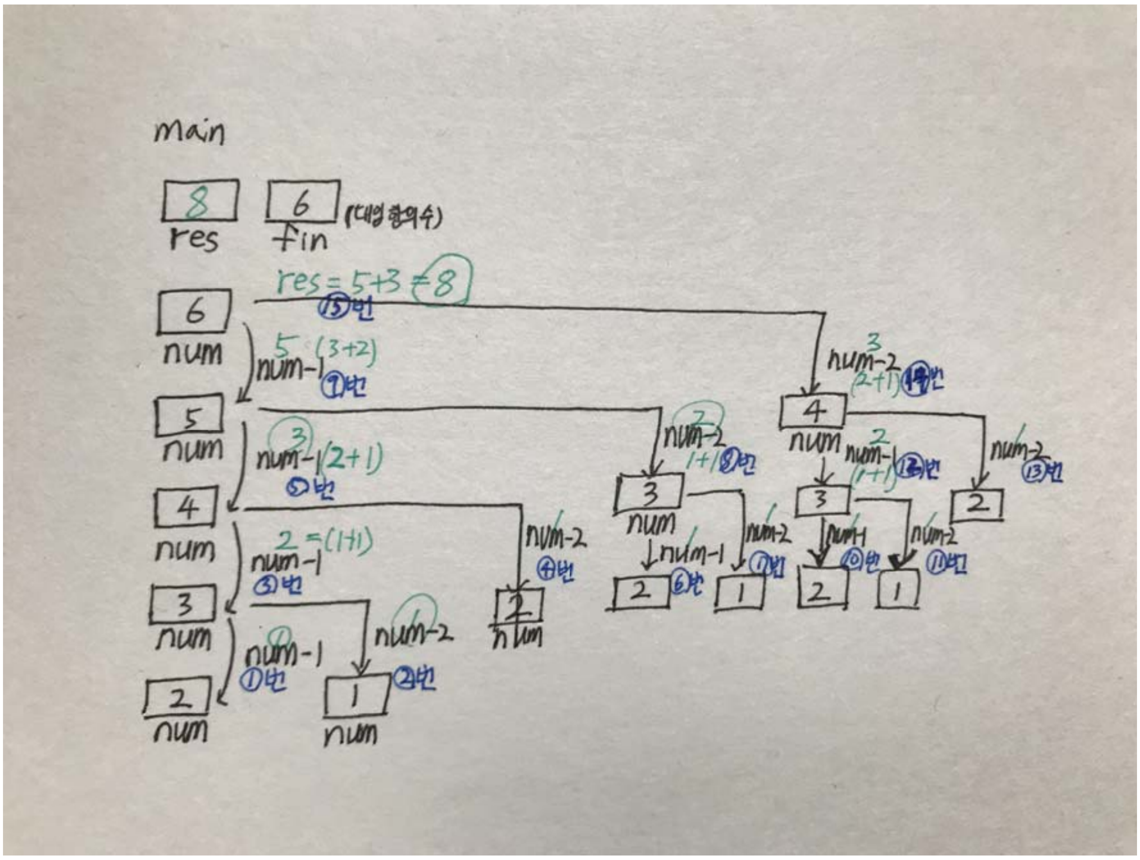
반면 HW회로는 병렬회로가 존재하듯이 모든 회로가 동시에 동작 가능.
파이프라인은 cpu로 구성된 회로이기 때문에 모든 모듈에서 동시에 동작이 가능하다.

재귀 함수 호출

-사용한 함수를 다시 호출하는 방식, program 구현상 반드시 필요한 경우가 있다.
-무조건 좋지만은 않음(stack이 새로 잡히니까)
-편의가 따른다.

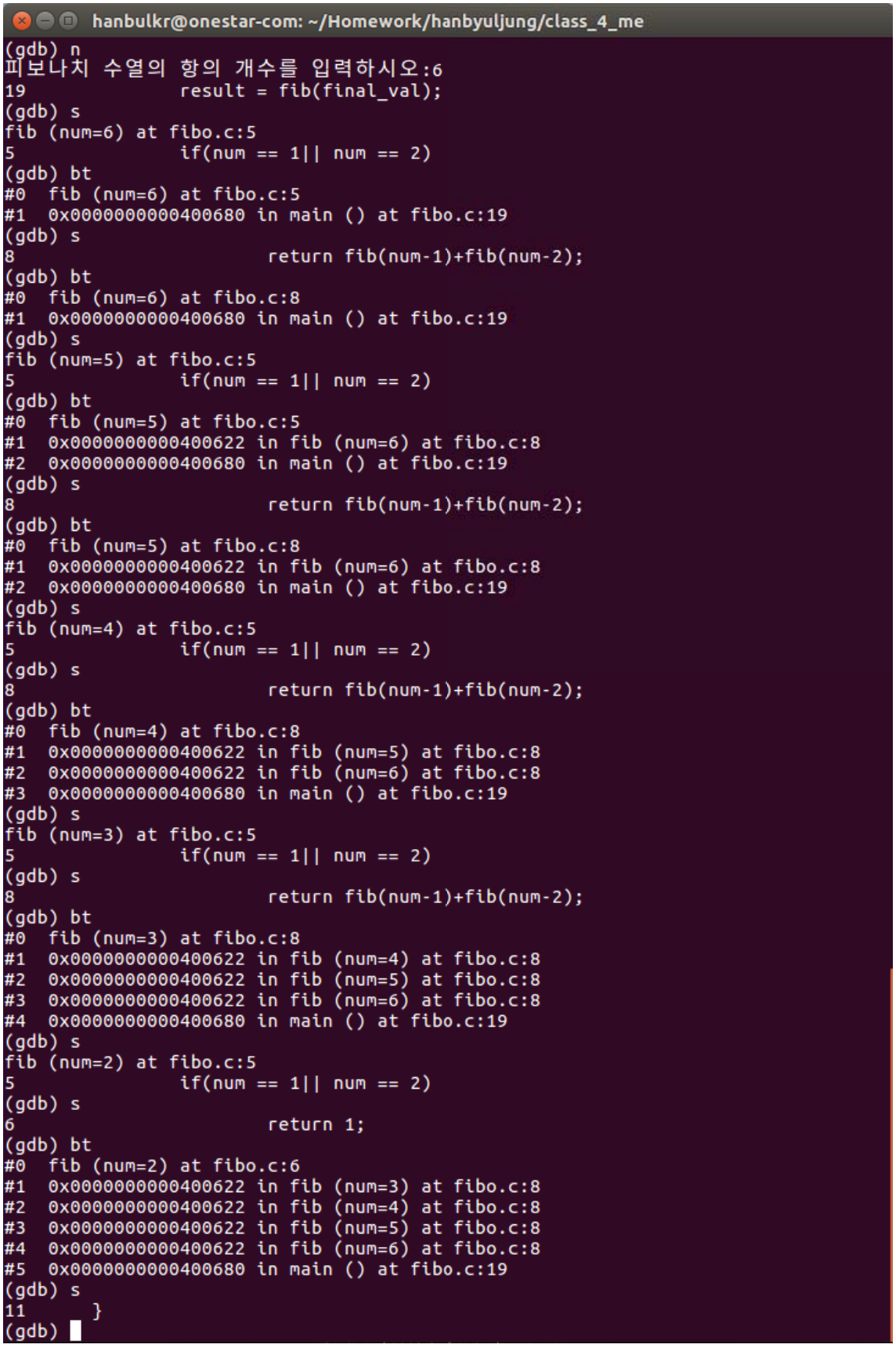
**재귀함수에서 중요한 점은 재귀함수를 부르게 되면서 스택 속에 또 다른 공간을 잘라 잡는 것이 아니라 새로운 스택이 생성되어 공간을 만드는 식이다.

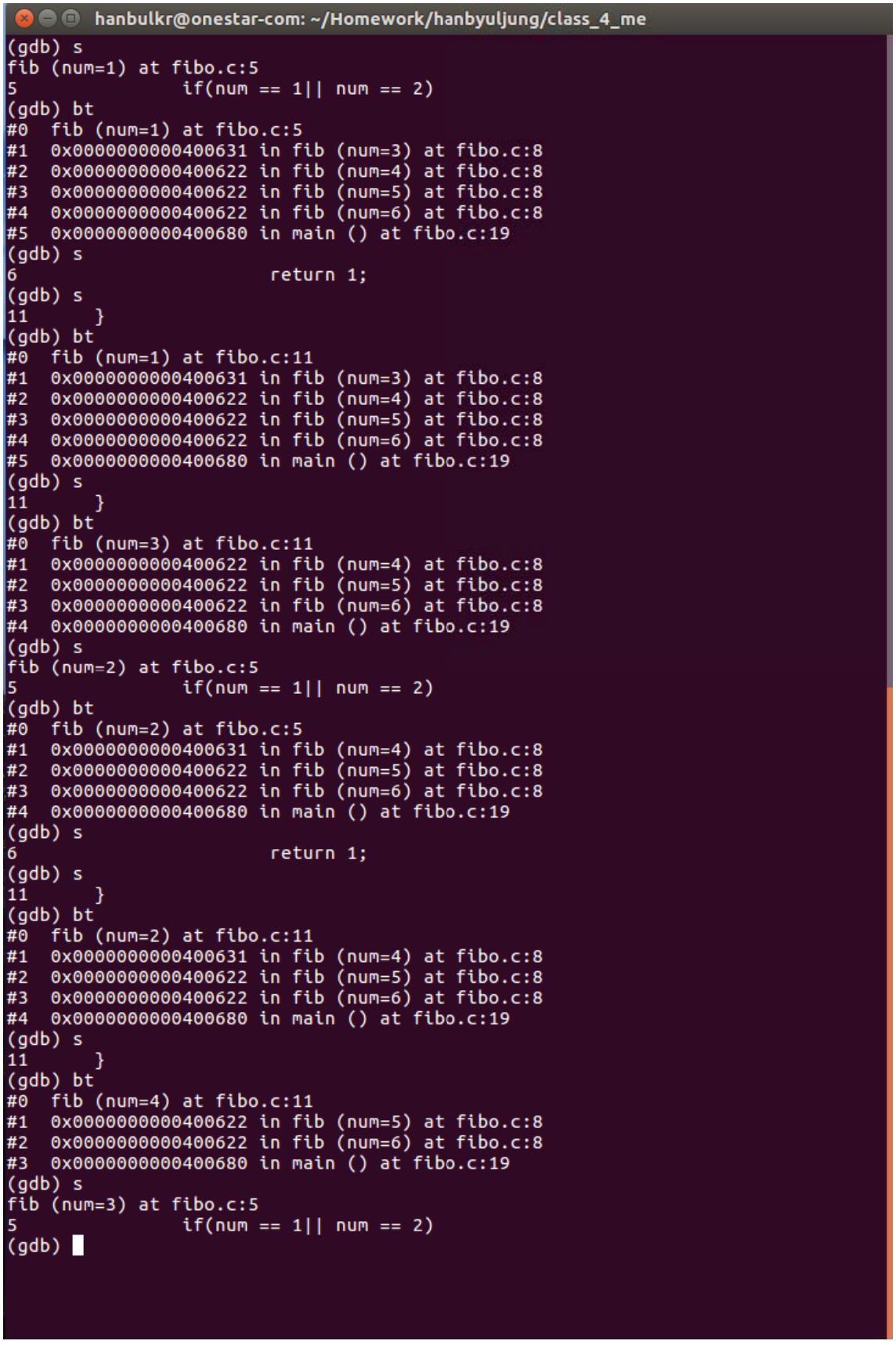
fib 함수 동작 분석 (디버깅 및 그림 그리기)

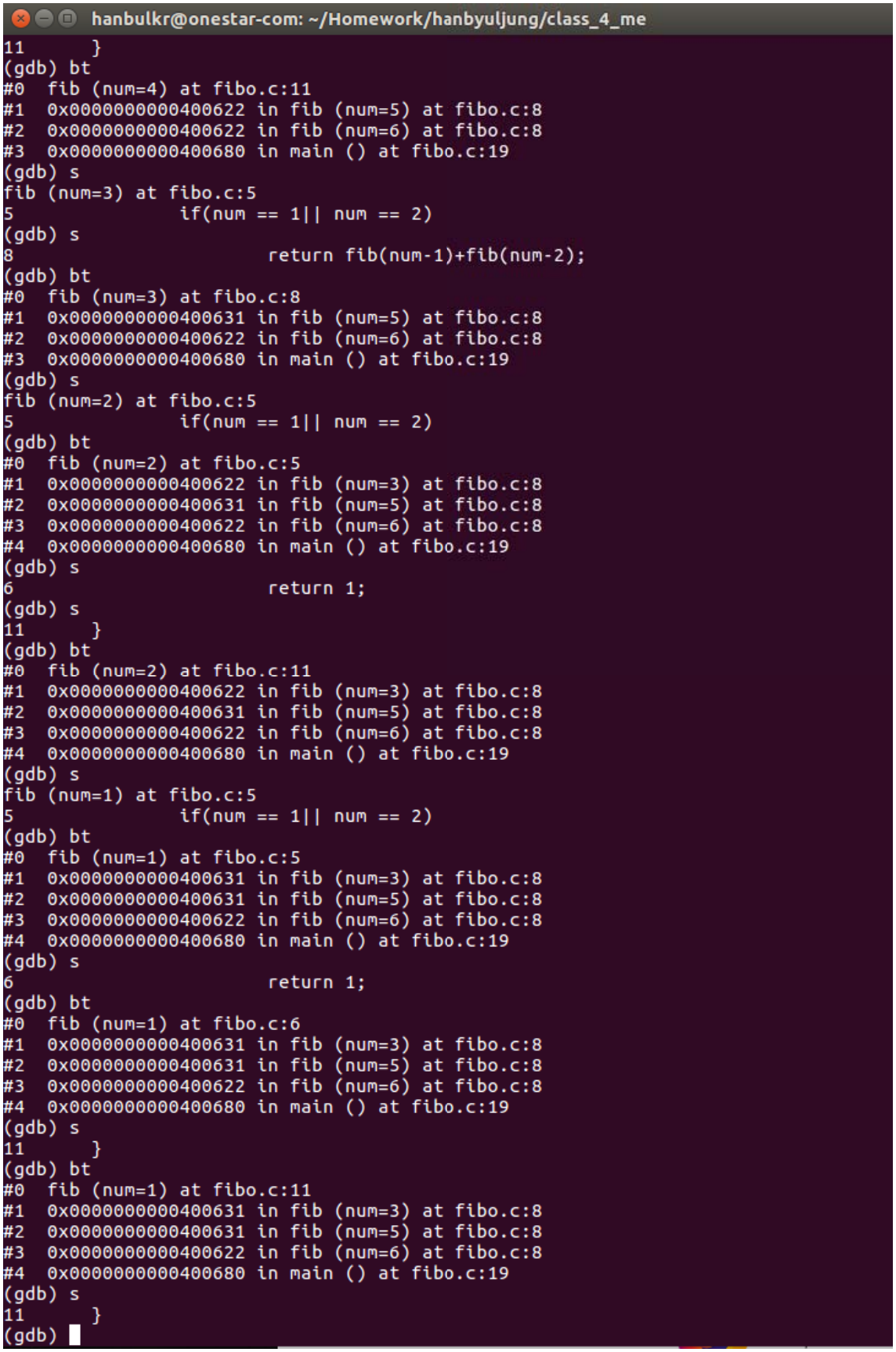


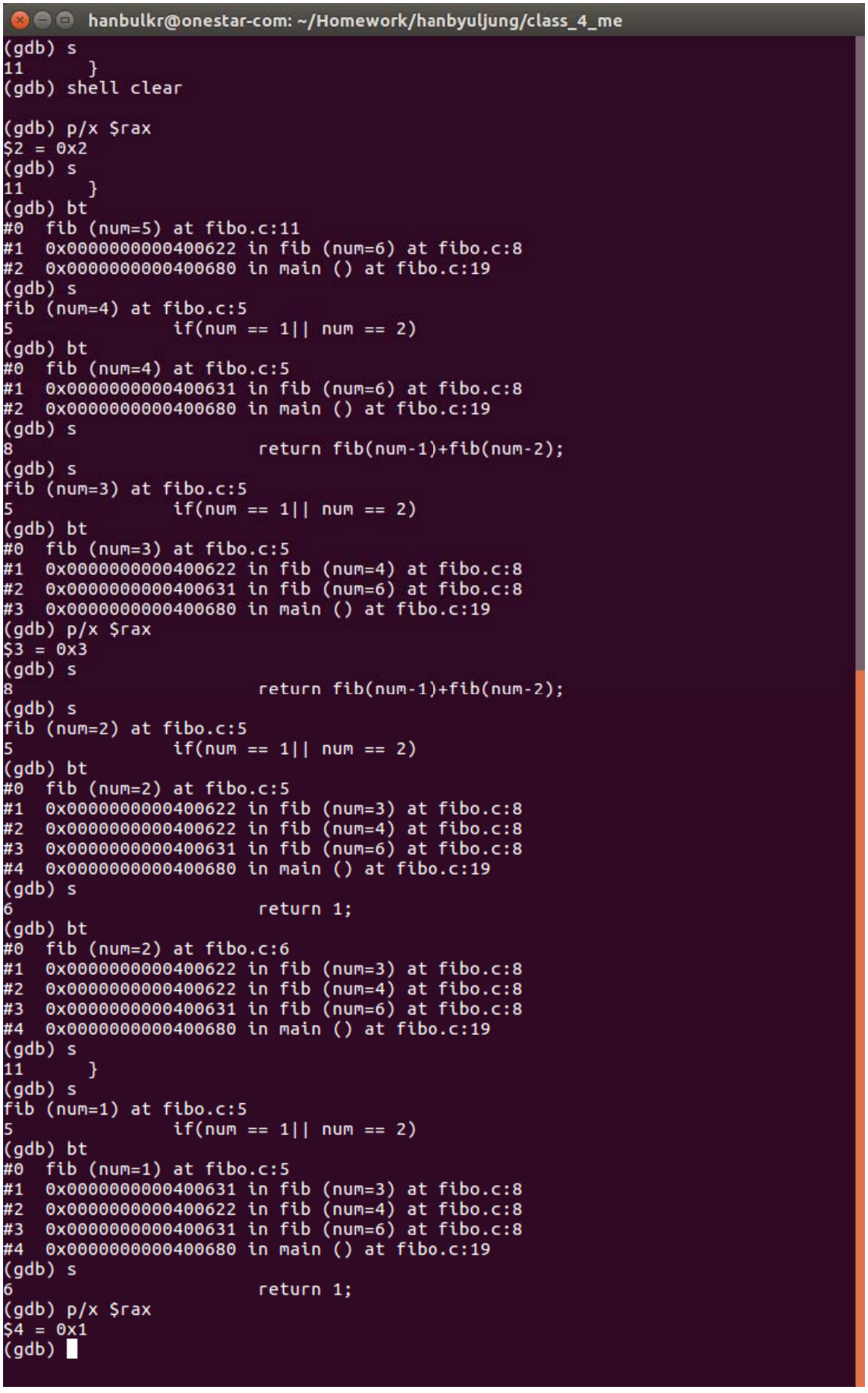
1. 처음 메인문을 들어가 ‘6’라는 숫자를 입력 받고 재귀함수 호출시 num=6부터 아래로 계속적으로 재귀함수(num-1)이 호출 된다.
2. 그렇게 맨 밑인 num=2 부터 재귀함수 제일 말단 부 ①부터 ①>②>③>④>⑤>⑥>⑦>⑧>⑨>⑩>⑪>⑫>⑬>⑭>⑮ 순으로 rax에 리턴 값을 저장하면서 연산과정을 거친다.

*밑에 동작과정을 스크린 샷으로 남긴 디버깅 자료를 참고로 이해한다.


hanbulkr@onestar-com: ~/Homework/hanbyuljung/class_4_me
(gdb) n
피보나치 수열의 항의 개수를 입력하시오:6
19   result = fib(final_val);
(gdb) s
fib (num=6) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=6) at fibo.c:5
#1 0x0000000000400680 in main () at fibo.c:19
(gdb) s
8   return fib(num-1)+fib(num-2);
(gdb) bt
#0 fib (num=6) at fibo.c:8
#1 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=5) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=5) at fibo.c:5
#1 0x0000000000400622 in fib (num=6) at fibo.c:8
#2 0x0000000000400680 in main () at fibo.c:19
(gdb) s
8   return fib(num-1)+fib(num-2);
(gdb) bt
#0 fib (num=5) at fibo.c:8
#1 0x0000000000400622 in fib (num=6) at fibo.c:8
#2 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=4) at fibo.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 fibo.c:8
#1 0x0000000000400622 in fib (num=5) at fibo.c:8
#2 0x0000000000400622 in fib (num=6) at fibo.c:8
#3 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=3) at fibo.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 fibo.c:8
#1 0x0000000000400622 in fib (num=4) at fibo.c:8
#2 0x0000000000400622 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=2) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) s
6   return 1;
(gdb) bt
#0 fib (num=2) at fibo.c:6
#1 0x0000000000400622 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400622 in fib (num=5) at fibo.c:8
#4 0x0000000000400622 in fib (num=6) at fibo.c:8
#5 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb)


hanbulkr@onestar-com: ~/Homework/hanbyuljung/class_4_me
(gdb) s
fib (num=1) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=1) at fibo.c:5
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400622 in fib (num=5) at fibo.c:8
#4 0x0000000000400622 in fib (num=6) at fibo.c:8
#5 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) s
11   }
(gdb) bt
#0 fib (num=1) at fibo.c:11
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400622 in fib (num=5) at fibo.c:8
#4 0x0000000000400622 in fib (num=6) at fibo.c:8
#5 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb) bt
#0 fib (num=3) at fibo.c:11
#1 0x0000000000400622 in fib (num=4) at fibo.c:8
#2 0x0000000000400622 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=2) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=2) at fibo.c:5
#1 0x0000000000400631 in fib (num=4) at fibo.c:8
#2 0x0000000000400622 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) s
11   }
(gdb) bt
#0 fib (num=2) at fibo.c:11
#1 0x0000000000400631 in fib (num=4) at fibo.c:8
#2 0x0000000000400622 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb) bt
#0 fib (num=4) at fibo.c:11
#1 0x0000000000400622 in fib (num=5) at fibo.c:8
#2 0x0000000000400622 in fib (num=6) at fibo.c:8
#3 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=3) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb)


hanbulkr@onestar-com: ~/Homework/hanbyuljung/class_4_me
11   }
(gdb) bt
#0 fib (num=4) at fibo.c:11
#1 0x0000000000400622 in fib (num=5) at fibo.c:8
#2 0x0000000000400622 in fib (num=6) at fibo.c:8
#3 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=3) at fibo.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 fibo.c:8
#1 0x0000000000400631 in fib (num=5) at fibo.c:8
#2 0x0000000000400622 in fib (num=6) at fibo.c:8
#3 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=2) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=2) at fibo.c:5
#1 0x0000000000400622 in fib (num=3) at fibo.c:8
#2 0x0000000000400631 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) s
11   }
(gdb) bt
#0 fib (num=2) at fibo.c:11
#1 0x0000000000400622 in fib (num=3) at fibo.c:8
#2 0x0000000000400631 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=1) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=1) at fibo.c:5
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400631 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) bt
#0 fib (num=1) at fibo.c:6
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400631 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb) bt
#0 fib (num=1) at fibo.c:11
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400631 in fib (num=5) at fibo.c:8
#3 0x0000000000400622 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb)


hanbulkr@onestar-com: ~/Homework/hanbyuljung/class_4_me
(gdb) s
11   }
(gdb) shell clear
(gdb) p/x $rax
$2 = 0x2
(gdb) s
11   }
(gdb) bt
#0 fib (num=5) at fibo.c:11
#1 0x0000000000400622 in fib (num=6) at fibo.c:8
#2 0x0000000000400680 in main () at fibo.c:19
(gdb) s
fib (num=4) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=4) at fibo.c:5
#1 0x0000000000400631 in fib (num=6) at fibo.c:8
#2 0x0000000000400680 in main () at fibo.c:19
(gdb) s
8   return fib(num-1)+fib(num-2);
(gdb) s
fib (num=3) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=3) at fibo.c:5
#1 0x0000000000400622 in fib (num=4) at fibo.c:8
#2 0x0000000000400631 in fib (num=6) at fibo.c:8
#3 0x0000000000400680 in main () at fibo.c:19
(gdb) p/x $rax
$3 = 0x3
(gdb) s
8   return fib(num-1)+fib(num-2);
(gdb) s
fib (num=2) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=2) at fibo.c:5
#1 0x0000000000400622 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400631 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) bt
#0 fib (num=2) at fibo.c:6
#1 0x0000000000400622 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400631 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
11   }
(gdb) s
fib (num=1) at fibo.c:5
5   if(num == 1|| num == 2)
(gdb) bt
#0 fib (num=1) at fibo.c:5
#1 0x0000000000400631 in fib (num=3) at fibo.c:8
#2 0x0000000000400622 in fib (num=4) at fibo.c:8
#3 0x0000000000400631 in fib (num=6) at fibo.c:8
#4 0x0000000000400680 in main () at fibo.c:19
(gdb) s
6   return 1;
(gdb) p/x $rax
$4 = 0x1
(gdb)

```
hanbulkr@onestar-com: ~/Homework/hanbyuljung/class_4_me
(gdb) s
fib (num=2) at fibo.c:5
5               if(num == 1|| num == 2)
(gdb) bt
#0  fib (num=2) at fibo.c:5
#1  0x0000000000400631 in fib (num=4) at fibo.c:8
#2  0x0000000000400631 in fib (num=6) at fibo.c:8
#3  0x0000000000400680 in main () at fibo.c:19
(gdb) s
6                       return 1;
(gdb) s
11      }
(gdb) bt
#0  fib (num=2) at fibo.c:11
#1  0x0000000000400631 in fib (num=4) at fibo.c:8
#2  0x0000000000400631 in fib (num=6) at fibo.c:8
#3  0x0000000000400680 in main () at fibo.c:19
(gdb) s
11      }
(gdb) bt
#0  fib (num=4) at fibo.c:11
#1  0x0000000000400631 in fib (num=6) at fibo.c:8
#2  0x0000000000400680 in main () at fibo.c:19
(gdb) s
11      }
(gdb) s
main () at fibo.c:20
20              printf("%d번째 항의 수는 = %d\n", final_val, result);
(gdb) bt
#0  main () at fibo.c:20
(gdb) disas
Dump of assembler code for function main:
   0x000000000040063a <+0>:     push   %rbp
   0x000000000040063b <+1>:     mov    %rsp,%rbp
   0x000000000040063e <+4>:     sub    $0x10,%rsp
   0x0000000000400642 <+8>:     mov    %fs:0x28,%rax
   0x000000000040064b <+17>:    mov    %rax,-0x8(%rbp)
   0x000000000040064f <+21>:    xor    %eax,%eax
   0x0000000000400651 <+23>:    mov    $0x400748,%edi
   0x0000000000400656 <+28>:    mov    $0x0,%eax
   0x000000000040065b <+33>:    callq  0x4004c0 <printf@plt>
   0x0000000000400660 <+38>:    lea    -0x10(%rbp),%rax
   0x0000000000400664 <+42>:    mov    %rax,%rsi
   0x0000000000400667 <+45>:    mov    $0x400781,%edi
   0x000000000040066c <+50>:    mov    $0x0,%eax
   0x0000000000400671 <+55>:    callq  0x4004e0 <__isoc99_scanf@plt>
   0x0000000000400676 <+60>:    mov    -0x10(%rbp),%eax
   0x0000000000400679 <+63>:    mov    %eax,%edi
   0x000000000040067b <+65>:    callq  0x4005f6 <fib>
   0x0000000000400680 <+70>:    mov    %eax,-0xc(%rbp)
=> 0x0000000000400683 <+73>:    mov    -0x10(%rbp),%eax
   0x0000000000400686 <+76>:    mov    -0xc(%rbp),%edx
   0x0000000000400689 <+79>:    mov    %eax,%esi
   0x000000000040068b <+81>:    mov    $0x400784,%edi
   0x0000000000400690 <+86>:    mov    $0x0,%eax
   0x0000000000400695 <+91>:    callq  0x4004c0 <printf@plt>
   0x000000000040069a <+96>:    mov    $0x0,%eax
   0x000000000040069f <+101>:   mov    -0x8(%rbp),%rcx
   0x00000000004006a3 <+105>:   xor    %fs:0x28,%rcx
   0x00000000004006ac <+114>:   je     0x4006b3 <main+121>
   0x00000000004006ae <+116>:   callq  0x4004b0 <__stack_chk_fail@plt>
   0x00000000004006b3 <+121>:   leaveq
   0x00000000004006b4 <+122>:   retq
End of assembler dump.
(gdb) p/x $rax
$5 = 0x8
(gdb)
```

**윗 사진을 보면 알 듯이 처음엔 재귀함수가 재귀함수를 불러 새로운 스택을 만들어 그 메모리 공간으로 들어가고 if문에 걸리는 제일 말단부터 차례로 num-1 , num-2의 연산을 하고 들어오기 이전 재귀함수로 빠져 나가면서 다시 새로이 num-1, num-2의 가지들이 연산과정을 거듭반복하게 된다.