

Embedded Class

Homework#3

-목차

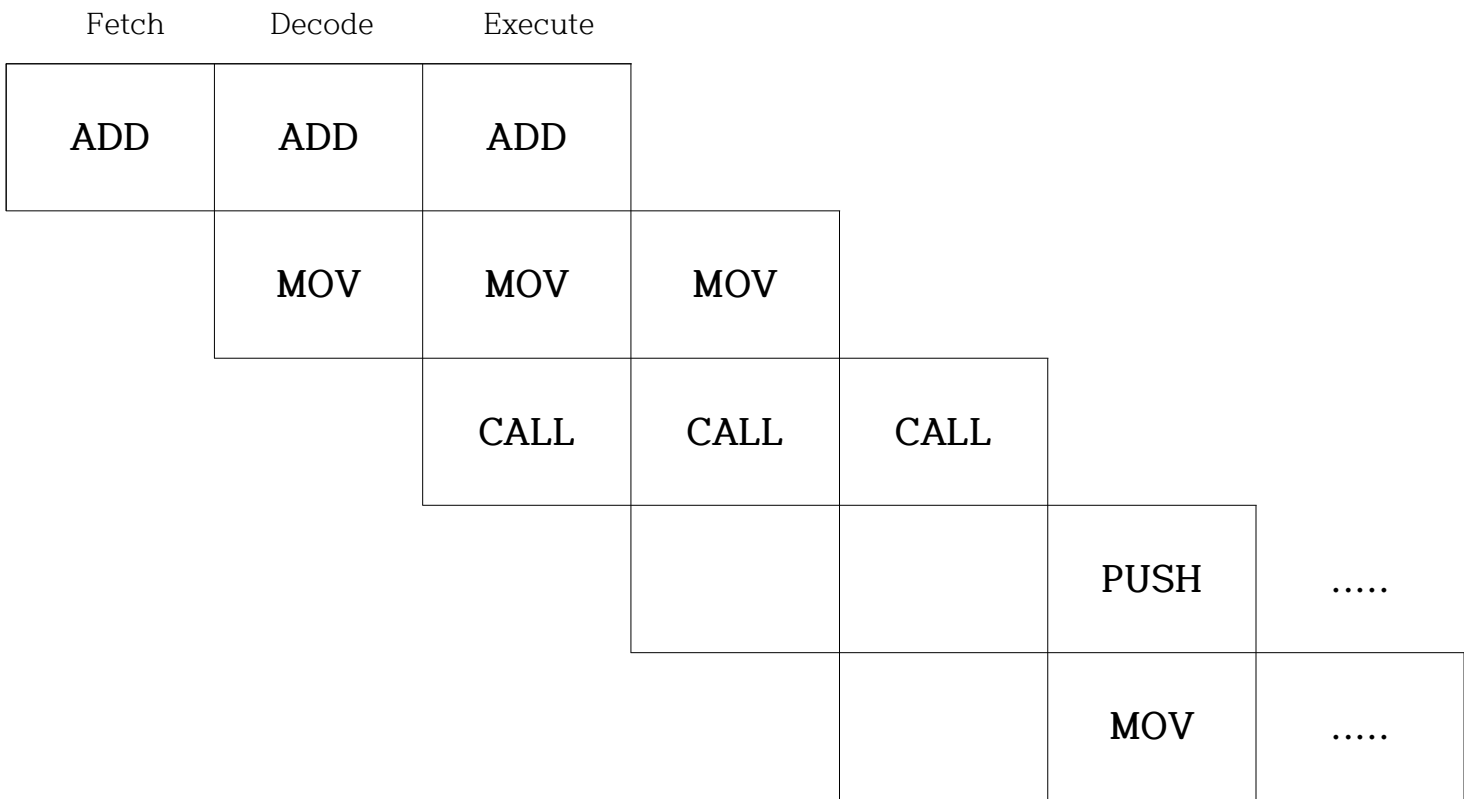
1. 배운내용 복습 (goto,파이프라인,for)
- 2.문제은행 어제 내용 for문으로
- 3.fib 함수 동작분석

김 시 윤

1. 배운 내용 복습.

1) goto 문.

goto 는 if와 break를 조합한 코드와 비교했을 때 goto 로 처리하는 소스코드가 더 성능이 좋다.
그 이유는 if 문은 기본적으로 mov,cmp,jmp를 한다. 문제는 jmp 명령이다.
call 이나 jmp를 cpu instruction 레벨에서 분기 명령이라고 하고 이들은 cpu 파이프 라인에 치명적이다.



한클럭에 세 개의 동작을 동시에 하면 3파이프라인이라 한다.
위에 jmp를 예를들면
CALL을 하는순간 새로운 함수로 점프하게되고 필요없는 스택공간을 만들어낸다.
또 한 클럭에 여러 동작을 하게되면 CPU 파이프라인이 깨지게 되 성능이 저하된다.

하지만 goto 문은 jmp 하나로 끝이기 때문에 여러번 jmp 하는 if문보다 유용하다.

```

siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ vi goto2.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ cat goto2.c
#include <stdio.h>

int main(void)
{
    int i,j,k;
    for(i=0; i<5; i++)
    {
        for(j =0; j<5; j++)
        {
            for(k=0;k<5;k++)
            {
                if((i ==2) && (j ==2) && (k ==2))
                {
                    printf("error!!!!\n");
                    goto err_handler;
                }
                else
                {
                    printf("data\n");
                }
            }
        }
    }
    return 0;
err_handler:
    printf("Goto Zzang!\n");
    return -1;
}
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ gcc goto2.c

```

```

data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
data
error!!!!
Goto Zzang!
siyun@siyun-Z20NH-A

```

[그림1. goto 실습]

2.문제은행 문제 for문으로 실습.

1)

```
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ vi h1.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ vi h1.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ gcc h1.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ ./a.out
res = 30000
res = 60000
res = 90000
res = 120000
res = 150000
res = 180000
res = 210000
res = 210000
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ cat h1.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<=6;i++)
    {
        res += money * 0.8;
        printf("res = %d\n",res);
    }
    return res;
}

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

    return 0;
}
```

2)



Terminal window showing the execution of a C program. The sidebar on the left contains icons for a gear, a circular arrow, a document with a pencil, a camera, a hard drive, a CD/DVD, another camera, and a glass of water.

```
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ vc h2.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ gcc h2.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ ./a.out
sum of tot series = 166833
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ cat h2.c
#include <stdio.h>

int syn3(int start, int end,int times)
{
    int i = start,res=0;
    for(i=start;i<end+1;i++)
    {
        if(!(i%3))
        {
            res +=i;
        }
    }
    return res;
}

int main(void)
{
    printf("sum of tot series = %d\n",syn3(1,1000,3));
    return 0;
}
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$
```

3)

```
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ ./a.out
tot series sum = 166167
siyun@siyun-Z20NH-AS51B5U:~/my_proj/homework1$ cat h3.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;
}
```

4)

```
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ ./a.out
4
7      14      21      28
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ cat h5.c
#include <stdio.h>

void print_seven_series(int num)
{
    int i;
    for(i=0;i<num;i++)
    {
        if(i<num -1)
        {
            printf("%d\t",(i+1)*7);
        }
        else
        {
            printf("%d\n",(i+1)*7);
        }
    }
}

int main(void)
{
    int num;
    scanf("%d",&num);
    print_seven_series(num);

    return 0;
}
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$
```

5)

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 1 = 6
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 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
siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$ cat h6.c
#include <stdio.h>

void print_rom(void)
{
    int i , j ;
    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;
}

siyun@siyun-Z20NH-AS51B5U:~/my_proj/Homework/siyunkim$
```


3. fib 동작분석

```
피보나치 수열의 항의 개수를 입력하시오 : 6
15         result = fib(final_val);
(gdb) s
fib (num=6) at fib.c:4
4         if(num ==1 || num ==2)
(gdb) s
7         return fib(num-1) + fib(num -2);
(gdb) bt
#0  fib (num=6) at fib.c:7
#1  0x0000000000400680 in main () at fib.c:15
(gdb) s
fib (num=5) at fib.c:4
4         if(num ==1 || num ==2)
(gdb) s
7         return fib(num-1) + fib(num -2);
(gdb) bt
#0  fib (num=5) at fib.c:7
#1  0x0000000000400622 in fib (num=6) at fib.c:7
#2  0x0000000000400680 in main () at fib.c:15
(gdb) s
fib (num=4) at fib.c:4
4         if(num ==1 || num ==2)
(gdb) s
7         return fib(num-1) + fib(num -2);
(gdb) bt
#0  fib (num=4) at fib.c:7
#1  0x0000000000400622 in fib (num=5) at fib.c:7
#2  0x0000000000400622 in fib (num=6) at fib.c:7
#3  0x0000000000400680 in main () at fib.c:15
(gdb) s
fib (num=3) at fib.c:4
4         if(num ==1 || num ==2)
(gdb) s
7         return fib(num-1) + fib(num -2);
(gdb) bt
#0  fib (num=3) at fib.c:7
#1  0x0000000000400622 in fib (num=4) at fib.c:7
#2  0x0000000000400622 in fib (num=5) at fib.c:7
#3  0x0000000000400622 in fib (num=6) at fib.c:7
#4  0x0000000000400680 in main () at fib.c:15
(gdb) s
fib (num=2) at fib.c:4
4         if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=2) at fib.c:4
#1  0x0000000000400622 in fib (num=3) at fib.c:7
#2  0x0000000000400622 in fib (num=4) at fib.c:7
#3  0x0000000000400622 in fib (num=5) at fib.c:7
#4  0x0000000000400622 in fib (num=6) at fib.c:7
#5  0x0000000000400680 in main () at fib.c:15
(gdb) finish
Run till exit from #0  fib (num=2) at fib.c:4
0x0000000000400622 in fib (num=3) at fib.c:7
7         return fib(num-1) + fib(num -2);
Value returned is $19 = 1
```

피보나치수열 6을 입력하였다. 계산되는 과정을 Assembly Language를 통해 관찰한다.

우선 입력된 fib(num 6)이 호출된다. 컴파일러가 똑똑해 자동으로 if문을 거르고 바로 else문으로 들어간다. else 문에서 num -1 과 num -2가 있는데, 여기 이 컴파일러는 num-1부터 시행하였다. num6 -1 해서 num5가 호출되고 if 문에 비교한다.

하지만 if 문은 1 또는 2만 통과되어 리턴값 1을 받는데 1보다 크기 때문에 num5는 통과하지 못한다.
따라서 else 문에 있는 num -1 이 반복된다.
그것이 반복되어 num 2 가 되었을 때, if문을 통과하여 return 값이 1이된 모습을 스크린샷에 담았다.

```
(gdb) s
fib (num=1) at fib.c:4
4             if(num ==1 || num ==2)
(gdb) finish
Run till exit from #0  fib (num=1) at fib.c:4
0x0000000000400631 in fib (num=3) at fib.c:7
7             return fib(num-1) + fib(num -2);
Value returned is $20 = 1
(gdb)
```

num3 - 1 이 num2가 되어 if문을 통과하여 return값 1을 받았다.
그리고 시행하지 않았던 num -2를 num3에 먼저 시행해준다.
num3 -2를 하면 num1이 되어 if 문을 통과해 리턴값 1을 받는다.
따라서 num3의 리턴 총합은 2가 된다. ----- num3 -ret 2

```
(gdb) s
8         }
(gdb) s
fib (num=2) at fib.c:4
4             if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=2) at fib.c:4
#1  0x0000000000400631 in fib (num=4) at fib.c:7
#2  0x0000000000400622 in fib (num=5) at fib.c:7
#3  0x0000000000400622 in fib (num=6) at fib.c:7
#4  0x0000000000400680 in main () at fib.c:15
(gdb) finish
Run till exit from #0  fib (num=2) at fib.c:4
0x0000000000400631 in fib (num=4) at fib.c:7
7             return fib(num-1) + fib(num -2);
Value returned is $21 = 1
```

num2 갑자기 나와서 당황해 bt를 해보았더니, num4 가 num-2를 시행한 후였다.
num2는 if문을 통과해 ret1을 받기 때문에 바로 finish를 해버렸다. num2 - ret1
그러면 여기서 num4 -1 인 num3 의 ret = 2
num4 -2 인 num2 의 ret= 1
따라서 num4 의 ret은 3이된다.

```

(gdb) s
8      }
(gdb) s
fib (num=3) at fib.c:4
4          if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=3) at fib.c:4
#1  0x0000000000400631 in fib (num=5) at fib.c:7
#2  0x0000000000400622 in fib (num=6) at fib.c:7
#3  0x0000000000400680 in main () at fib.c:15
(gdb) s
7          return fib(num-1) + fib(num -2);
(gdb) s
fib (num=2) at fib.c:4
4          if(num ==1 || num ==2)
(gdb) finish
Run till exit from #0  fib (num=2) at fib.c:4
0x0000000000400622 in fib (num=3) at fib.c:7
7          return fib(num-1) + fib(num -2);
Value returned is $22 = 1

```

이제 여기서 num 5 -2를 시행한다.

num3은 위에서 구했던 것처럼 ret 값이 2이 지만 한번 더 확인해 보았다.

num3 에 -1 을 먼저 시행하여 num2가 되고 num2는 if문을 통과하여 ret 1을 받는다.

```

(gdb) s
fib (num=1) at fib.c:4
4          if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=1) at fib.c:4
#1  0x0000000000400631 in fib (num=3) at fib.c:7
#2  0x0000000000400631 in fib (num=5) at fib.c:7
#3  0x0000000000400622 in fib (num=6) at fib.c:7
#4  0x0000000000400680 in main () at fib.c:15
(gdb) finish
Run till exit from #0  fib (num=1) at fib.c:4
0x0000000000400631 in fib (num=3) at fib.c:7
7          return fib(num-1) + fib(num -2);
Value returned is $23 = 1

```

num3 -2 는 num1 이 되어 if문을 통과하여 ret 1의 값을 받는다.

따라서 num3 의 총 return 값은 2가 된다.

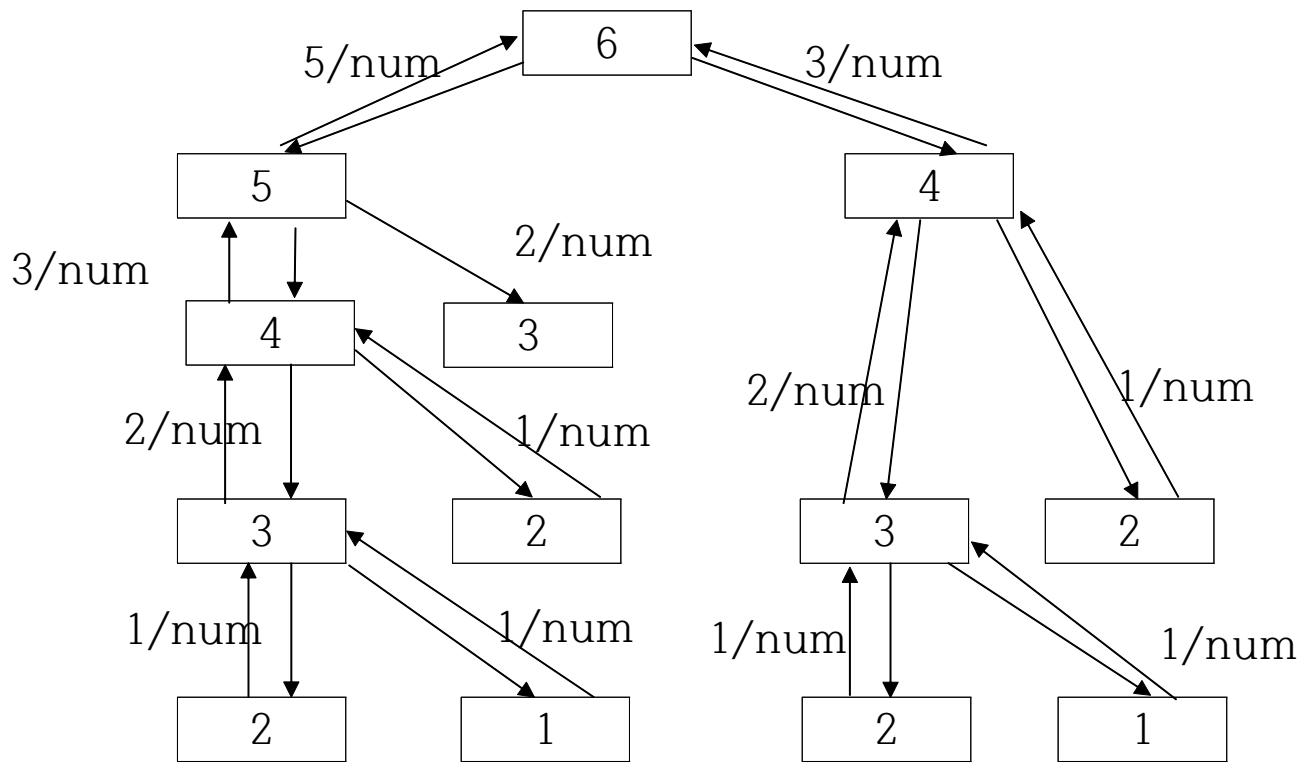

```

Value returned is $23 = 1
(gdb) s
8      }
(gdb) s
8      }
(gdb) s
fib (num=4) at fib.c:4
4          if(num ==1 || num ==2)
(gdb) bt
#0  fib (num=4) at fib.c:4
#1  0x0000000000400631 in fib (num=6) at fib.c:7
#2  0x0000000000400680 in main () at fib.c:15
(gdb) s
7          return fib(num-1) + fib(num -2);
(gdb) bt
#0  fib (num=4) at fib.c:4
#1  0x0000000000400631 in fib (num=6) at fib.c:7
#2  0x0000000000400680 in main () at fib.c:15
(gdb) s
fib (num=3) at fib.c:4
4          if(num ==1 || num ==2)
(gdb) finish
Run till exit from #0  fib (num=3) at fib.c:4
0x0000000000400622 in fib (num=4) at fib.c:7
7          return fib(num-1) + fib(num -2);
Value returned is $24 = 2
(gdb) █

```

앞에서 구한값이 총 5 이고 위의 확인하면 num4가 return3이 나오므로
두 개 총 합한 값으 8이 된다.

위의 과정을 그림으로 설명해보면 아래와 같다.



여기서 위로 올라가는 화살표는 return의 이동경로이다.
 밑으로 내려가는 화살표는 $num - 1$ 또는 $num - 2$ 의 행위이다.