Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – 이상훈
gcccompil3r@gmail.com

수강생 – 서재언
20640@naver.com

1. day_7/01.c

- 함수 포인터의 리턴

```
1 #include <stdio.h>
2
3 void aaa(void)
4 {
5    printf("aaa called\n");
6
7 }
8
9 int number(void)
10 {
11    printf("number called\n");
12
13
14    return 7;
15 }
16 void (*bbb(void))(void)
17 {
18   printf("bbb called\n");
19   return aaa;
20 }
21
22
23 void ccc(void(*p)(void))
24 {
25   printf("ccc: I can call aaa!\n");
26   p();
26 }
27
```

1. day_7/01.c

```
28 int (*ddd(void))(void)
29 {
30   printf("ddd: I can call number\n");
31   return number;
32 }
33
34 int main(void)
35 {
36    int res;
37
38    bbb()();
39    ccc(aaa);
40    ddd()();
41
42   res = ddd()();
43   printf("res = %d\n", res);
44
45  return 0;
46 }
```

함수	리턴	이름	인자
void(*bbb(void)(void)	void(*)(void)	bbb	void
void ccc(void(*p)(void))	void	ccc	void (*p)(void)
int (*ddd(void))(void)	int (*)(void)	ddd	void
void (*bbb(void (*p)(void)))(void)	void (*)(void)	bbb	void (*p)(void)

2. day_7/02.c

– 함수포인터 인자

```
1 #include <stdio.h>
2
3 void aaa(void)
4 {
5   printf("aaa called\n");
6 }
7
8 void (*bbb(void(*p)(void)))(void)
9 {
10   p();
11   printf("bbb called\n");
12   return aaa;
13 }
14
15
 16 int main(void)
17 {
18   bbb(aaa)();
19   return 0;
20 }
```

함수	리턴	이름	인자
void (*bbb(void(*p)(void)))(void)	void(*)(void)	bbb	void (*p)(void)

3. day_7/03.c

– 함수포인터 함수, 인자, 반환형 예제

```
1 #include <stdio.h>
2

3 // int (*)[2] aaa(void)
4 // return: int(*)[2]
5 // name: aaa
6 // parameter: void
7 int (* aaa(void))[2]
8 {
9    static int a[2][2] = {{10, 20}, {30, 40}};
10  printf("aaa called\n");
11  return a;
12 }
13

14 // int (*(* bbb(void))(void))[2]
15
16 // case 1: criteria: function name
17 // int (*(*)(void))[2] bbb(void)
18 // int (*)[2] (*)(void) bbb(void)
19
20 // case 2: first '(*' last ')~~'
21 // int (*)[2](* bbb(void))(void)
22 // int (*)[2](*)(void) bbb(void)
23

24 // return: int(*)[2](*)(void)
25 // name: bbb
26 // parameter: void
27
```

3. day_7/03.c

```
28 // int (*)[2] aaa(void)
29 // int (*)[2] (*)(void) bbb(void)
30 int (*(* bbb(void))(void))[2]
31 {
32   printf("bbb called\n");
33   return aaa;
34 }
35
 36 int (*(* ccc(void))(void))[2]
37 {
38   printf("ccc called\n");
39   return aaa;
40 }
41
 42 int (*(* ddd(void))(void))[2]
43 {
44   printf("ddd called\n");
45   return aaa;
46 }
47
 48 int (*(* eee(void))(void))[2]
49 {
50   printf("eee called\n");
51   return aaa;
52 }
53
```

3. day_7/03.c

```
54 int main(void)
55 {
56   // int (*)[2] ret;
57   int (*ret)[2];

58   // int (*)[2] (*)(void) bbb(void)
59   // int (*)[2] (*)(void) (*)(void) p[][2]
60   int (*(*(*p[][2])(void))(void))[2] = {{bbb, ccc}, {ddd, eee}};

61   // int (*)[2] (*)(void) (*)(void) [ ][2] p
62   // int (*)[2] (*)(void) (*)(void) (*)[2] p1
63   int (*(*(*(*p1)[2])(void))(void))[2] = p;
64
      // int (*)[2] (*)(void) (*)(void) (*)[2] p1
65   // ((*(*(*(*p1)[3])))()())
66   // (*(*(*(*p1)[3])))()()
67   // (*)(*)(*)(*) p1[3] ()() = eee()()
68   // eee()() = aaa()
69   // ret = address of array a
70   ret = ((*(*(*(*p1)[3])))()());

71   printf("*ret[0]= %d\n",*ret[0]); // 10
72   printf("ret[0][0] = %d\n",  ret[0][0]); // 10
73   printf("*ret[1] = %d\n",*ret[1]); // 30
74   printf("ret[1][1] = %d\n", ret[1][1]);// 40
75   return 0;
76 }
```

3. day_7/03.c

함수	리턴	이름	인자
int (*) (aaa(void))[2]	int *()[2]	aaa	void
int (*(* bbb(void))(void))[2]	int *()[2](void)	bbb	void
int (*(* ccc(void))(void))[2]	int *()[2](void)	ccc	void
int (*(* ddd(void))(void))[2]	int *()[2](void)	ddd	void
int (*(* eee(void))(void))[2]	int *()[2](void)	eee	void
int (*(*(*p[][2])(void))(void))[2]	int *()[2](void)	*()(void)	*p[][2](void)

4. day_7/04.c

(문제1)
함수포인터 함수, 인자, 반환형 1. (int를 반환하고 float, double, int를 인자로 취하는 함수 포인터)를 반환하고 (float을 반환하고 int 2개를 인자로 취하는 함수포인터)를 인자로 취하는 함수를 작성하여프로그램이 정상적으로 동작하도록 프로그래밍하시오.
(힌트 : 함수는 총 main, pof_test_main, pof_test1, pof_test2으로 4개를 만드세요

```
#include <stdio.h>

float pof_test1(int n1, int n2)
{
		return (n1+n2)*0.23573;
}

int pof_test2(float n1, double n2, int n3)
{
		return (n1+n2+n3)/3.0;
}
//int (*)(float, double, int) pof_test_main(float (*)(int, int))
//int (*(pof_test_main(float (*)(int, int)))(float, double, int)
int (*pof_test_main(float (*p)(int, int)))(float, double, int)
{
		float res;
		res = p(4, 3);
		printf("res = %f₩n", res);
		return pof_test2;
}
```

4. day_7/04.c

(문제1)
함수포인터 함수, 인자, 반환형 1. (int를 반환하고 float, double, int를 인자로 취하는 함수 포인터)를 반환하고 (float을 반환하고 int 2개를 인자로 취하는 함수포인터)를 인자로 취하는 함수를 작성하여프로그램이 정상적으로 동작하도록 프로그래밍하시오.
(힌트 : 함수는 총 main, pof_test_main, pof_test1, pof_test2으로 4개를 만드세요

```
int main(void)
{
		int res;
		res = pof_test_main(pof_test1)(3.7, 2.4, 7);
		printf("pof_test_main res = %d₩n", res);
		return 0;
}
```

5. day_7/05.c

(문제2)
(int 2개를 인자로 취하고 int를 반환하는 함수포인터를 반환하며 인자로 int를 취하는 함수포인터)를 반환하는 함수는(float을 반환하고 인자로 int, double을 취하는 함수 포인터)를 인자로 취한다.이를 프로그래밍하고 역시 정상적으로 동작하도록 프로그래밍하시오.
(힌트 : 함수는 총 main, pof_test_main, pof1, subpof1, pof2로 5개를 취한다.)

```
#include <stdio.h>

//int(*)(int, int)
int pof1(int n1, int n2)
{
		return n1+n2;
}

int (*subpof1(int n))(int, int)
{
		printf("n = %d₩n", n);
		return pof1;
}

//float (*)(int, double)
float pof2(int n1, double d1)
{
		return n1*d1;
}
```

5. day_7/05.c

```
//int (*)(int, int) (*)(int) pof_test_main(float (*)(int, double))
//int (*(* por_test_main(float(*)(int, double)))(int))(int, int);
//int(*)(int, int)(*pof_test_main(float (*)(int, double)))(int)
//int(*)(int, int) (*)(int) pof_test_main(float (*)(int, double))

int (*(*pof_test_main(float (*p)(int, double)))(int))(int, int)
{
		float res;
		res = p(3, 7.7);			//pof2 호출
		printf("res = %f₩n", res);
		return subpof1;
}


int main(void)
{

		int res;
		res = pof_test_main(pof2)(3)(7, 3);  //subpof1함수 포인터 호출
		printf("res = %d₩n", res);

		return 0;
}
```

5. day_7/05.c

(case1)
포인터 함수의 주소를 다른 함수로 단순 변환하는 경우 => How?

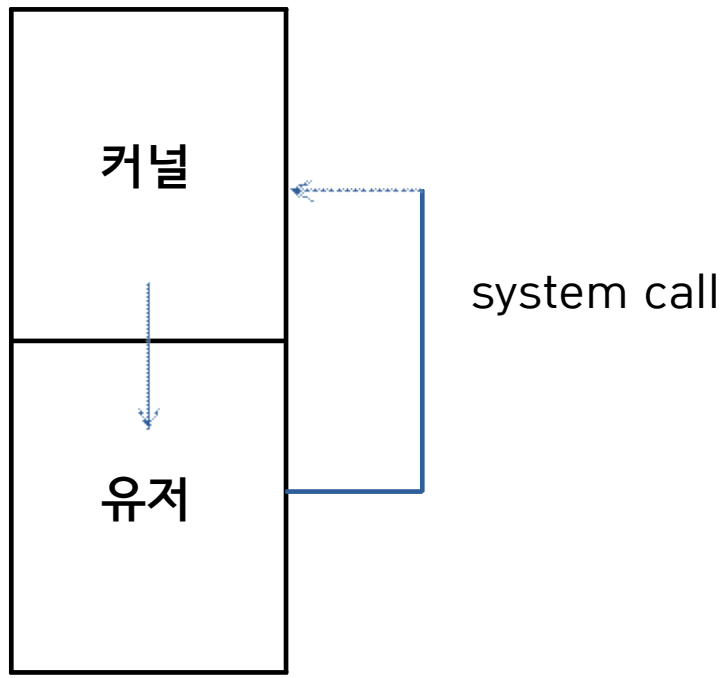
(case2)
포인터 함수를 함수의 인자로 넣는 경우 => How?

(case3)
포인터 함수를 함수 내에서 리턴하는 경우 => How?

∴ 3가지 생각을 통해 포인터 함수를 보다 잘 이해할 수 있음

6. 시스템 콜(System Call)

시스템은 함수포인터에 대한 주소를 테이블로 관리



7. 함수포인터

시스템에서 비동기 동작에 사용되는 방법 - 인터럽트 핸들러

8. memmove(day7/06.c)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
		int i=0;
		int src[5] = {1, 2, 3, 4, 5};
		int dst[5];

		memmove(dst, src, sizeof(src));

		for(i=0; i<5; i++)
		{
			printf("dst[%d] = %d₩n", i, dst[i]);

		}

		return 0;
}
```

9. memcpy(day_7/07.c)

- memcpy가 memmove에 비해 속도 빠름

```
#include <stdio.h>
#include <string.h>

int main(void)
{
		char src[30] = "this is amazing";
		char *dst = src + 3;

		printf("before memmove = %s₩n", src);
		memcpy(dst, src, 3);
		printf("after memmove = %s₩n", dst);

		return 0;
}
```

10. argc, argv, env명령어와 env환경 설정값 가지고 오는 방법(day_7/08.c)

- 프로그램 명령어 옵션을 실행하는 프로그램으로 활용

```
#include <stdio.h>
#include <string.h>

int main(void)
{
		char src[30] = "this is amazing";
		char *dst = src + 3;

		printf("before memmove = %s₩n", src);
		memcpy(dst, src, 3);
		printf("after memmove = %s₩n", dst);

		return 0;
}
```

11. gets(), getchar(), strlen()(day_7/09.c)

- strlen은 '\0'제외한 길이를 구함

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
        char *str = "This is the string";
        int len = strlen(str);
        printf("len = %d₩n", len);

        return 0;
}
```

12. strncpy()(day_7/10.c)

- strcpy는 해킹당할 위험이 있으므로 strncpy() 사용권함

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
		char src[20] = "abcdef";
		char dst[20];
		strncpy(dst, src, 3);
		printf("dst = %s₩n", dst);

		return 0;
}
```

13. strncmp()(day_7/11.c)

- 비교하는 값이 같으면 반환값 '0'

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
		char src[20] = "made in korea";
		char dst[20] = "made in china";

		if(!strncmp(src, dst, 8))
			printf("src, dst는 서로 같음₩n");
		else
			printf("src, dst는 서로 다름₩n");

		return 0;
}
```

13. stack (day_7/12.c)

- 스택구현
  top 이 위로 올라가면서 아래로 단일링크 연결해 쌓아나가는 방식

```
#include <stdio.h>
#include <stdlib.h>

#define EMPTY 0

struct node{
        int data;
        struct node *link;
};

typedef struct node Stack;

Stack *get_node()
{
        Stack *tmp;
        tmp = (Stack *)malloc(sizeof(Stack));
        tmp->link = EMPTY;

        return tmp;
}
```

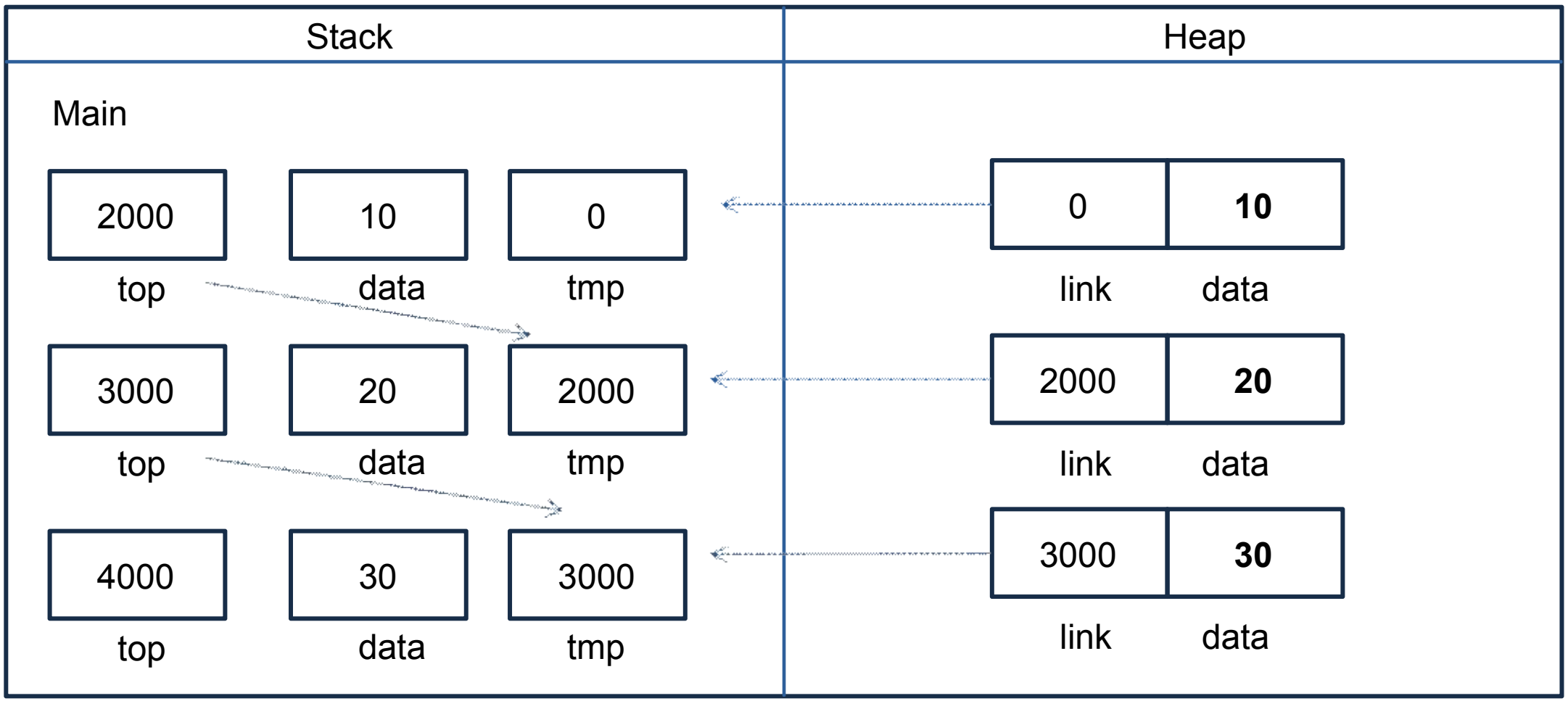
13. stack (day_7/12.c)

```
void push(Stack **top, int data)
{
		Stack *tmp;
		tmp = *top;
		*top = get_node();
		(*top)->data = data;
		(*top)->link = tmp;
}
```

13. stack (day_7/12.c)

- push() 구현

(1) tmp를 통해 heap에 메모리 생성
(2) 자기참조 구조체 통해 링크 생성 및 연결
(3) top이 roof로써 설정된 후 단일 링크드 리스트로 연결



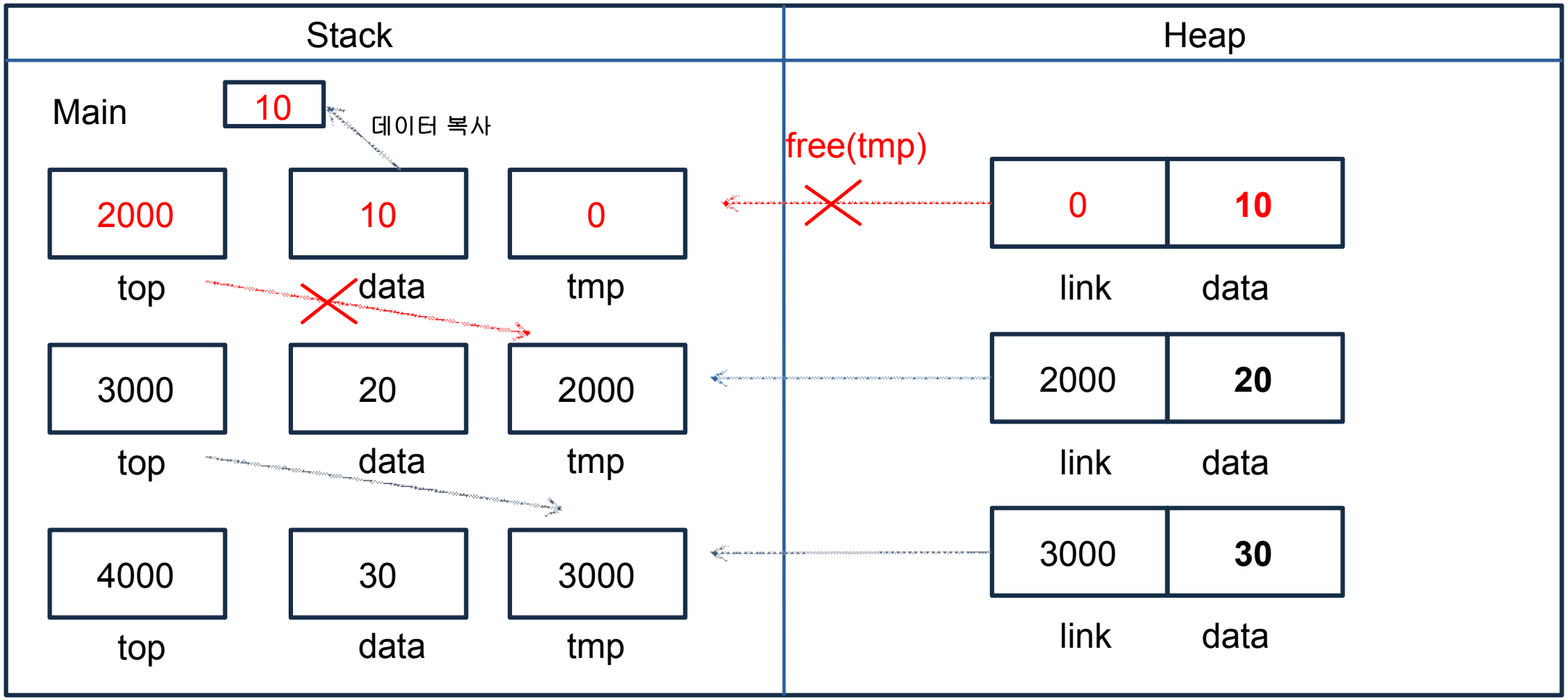
13. stack (day_7/12.c)

```
int pop(Stack **top)
{
		Stack *tmp;
		int num;
		tmp = *top;

		if(*top == EMPTY)
		{
			printf("Stack is empty!!₩n");
			return 0;
		}

		num = tmp->data;
		*top = (*top)->link;
		free(tmp);

		return num;
}
```

13. stack (day_7/12.c)

- pop()구현

(1) 링크드리스트로 연결된 top의 주소를 tmp에 대입
(2) tmp통해 top이 가지고 있던 data를 꺼냄(또는 복사)
(3) top의 기존 주소를 top이 다음으로 가리키고 있던 주소로 치환
(4) data 리턴



13. stack (day_7/12.c)

```
int main(void)
{
		Stack *top = EMPTY;
		push(&top, 10);
		push(&top, 20);
		push(&top, 30);
		printf("%d₩n", pop(&top));   //와 pop()비교
		printf("%d₩n", pop(&top));
		printf("%d₩n", pop(&top));
		printf("%d₩n", pop(&top));

		return 0;
}
```

숙제

위와 같은 프로토타입의 함수가 구동되도록 프로그래밍 float (* (* test(void (*p)(void)))(float (*)(int, int)))(int, int)하시오.

float *() (int, int) , *test( )(float (*)(int, int)), void (*p)(void)

```
#include <stdio.h>


void a_func()
{
	printf("a_func\n");
}

float b_func(int n1, int n2)
{
	printf("b_func\n");

	return (float)(n1+n2);

}

float (*c_func(float (*p1)(int, int)))(int, int)
{
	float res;


	res = p1(10, 10);

	printf("*c_func():%f\n", res);

	return b_func;

}
```

숙제

위와 같은 프로토타입의 함수가 구동되도록 프로그래밍 float (* (* test(void (*p)(void)))(float (*)(int, int)))(int, int)하시오.

float *() (int, int) , *test( )(float (*)(int, int)), void (*p)(void)

```
float (* (* test(void (*p)(void)))(float (*)(int, int)))(int, int)
{

	printf("test func\n");

	p();

	return c_func;


}


int main(void)
{
	float res;

	res = (*test(a_func))(b_func)(19,20);

	printf("main: %f\n", res);

	return 0;
}
```