

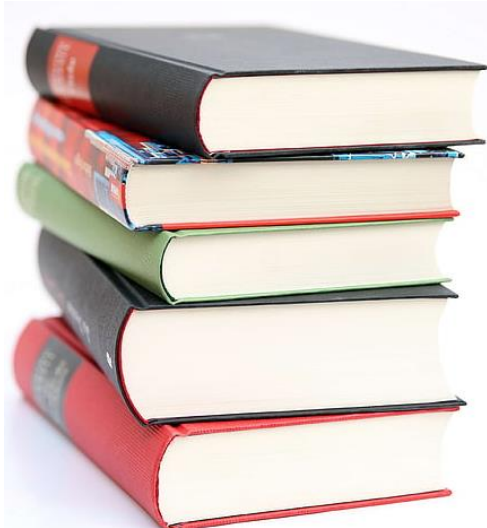
자료 구조 (STACK) 발표

홍정완

- 목 차 -

1. 스택(STACK)이란?
2. 스택 (시스템)
3. 스택 추상 자료형 (ADT)
4. 스택 (배열)
5. 스택 (연결 리스트)
6. 구현

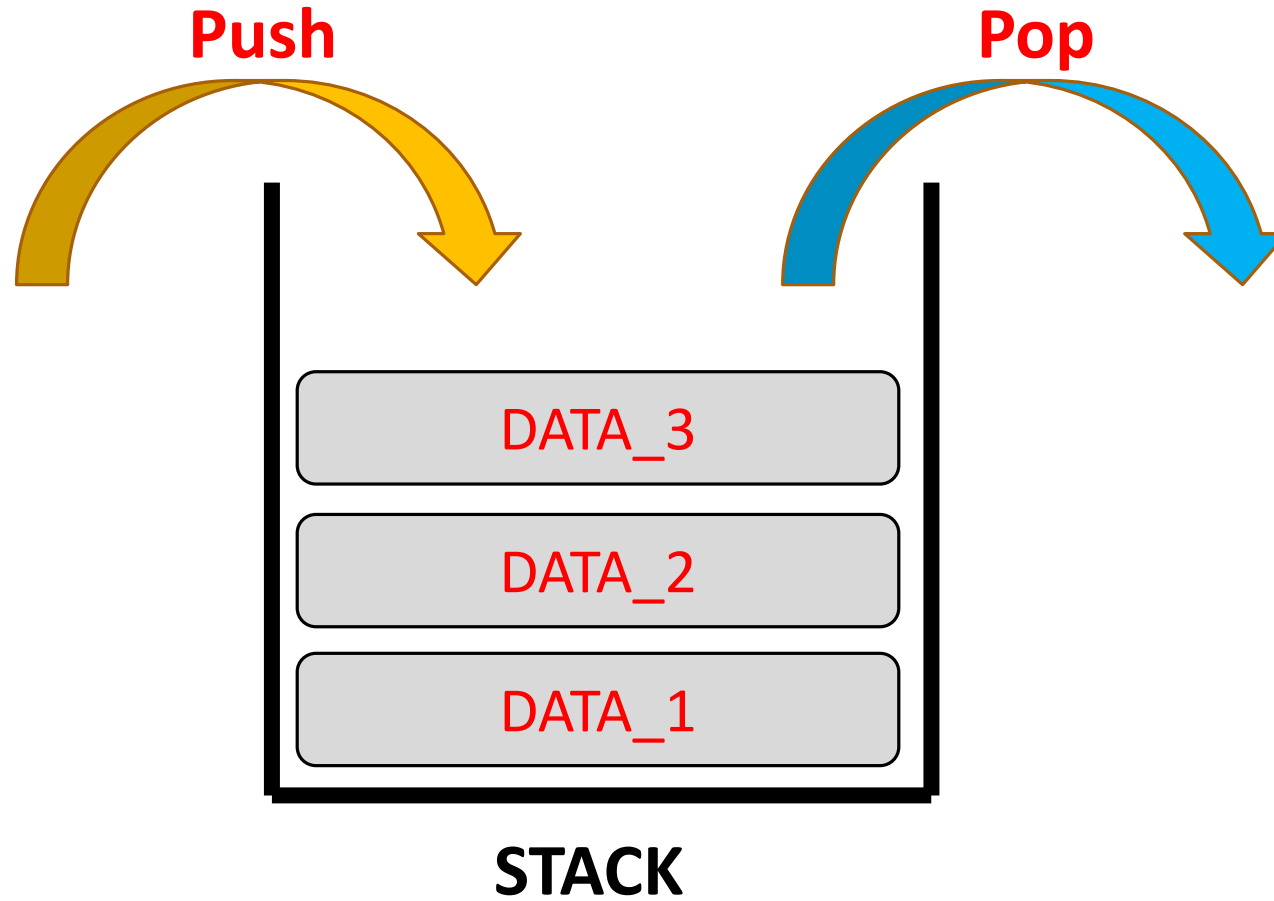
1. 스택(STACK)이란?



· 스택(STACK)
⇒ 쌓여 있는 것.

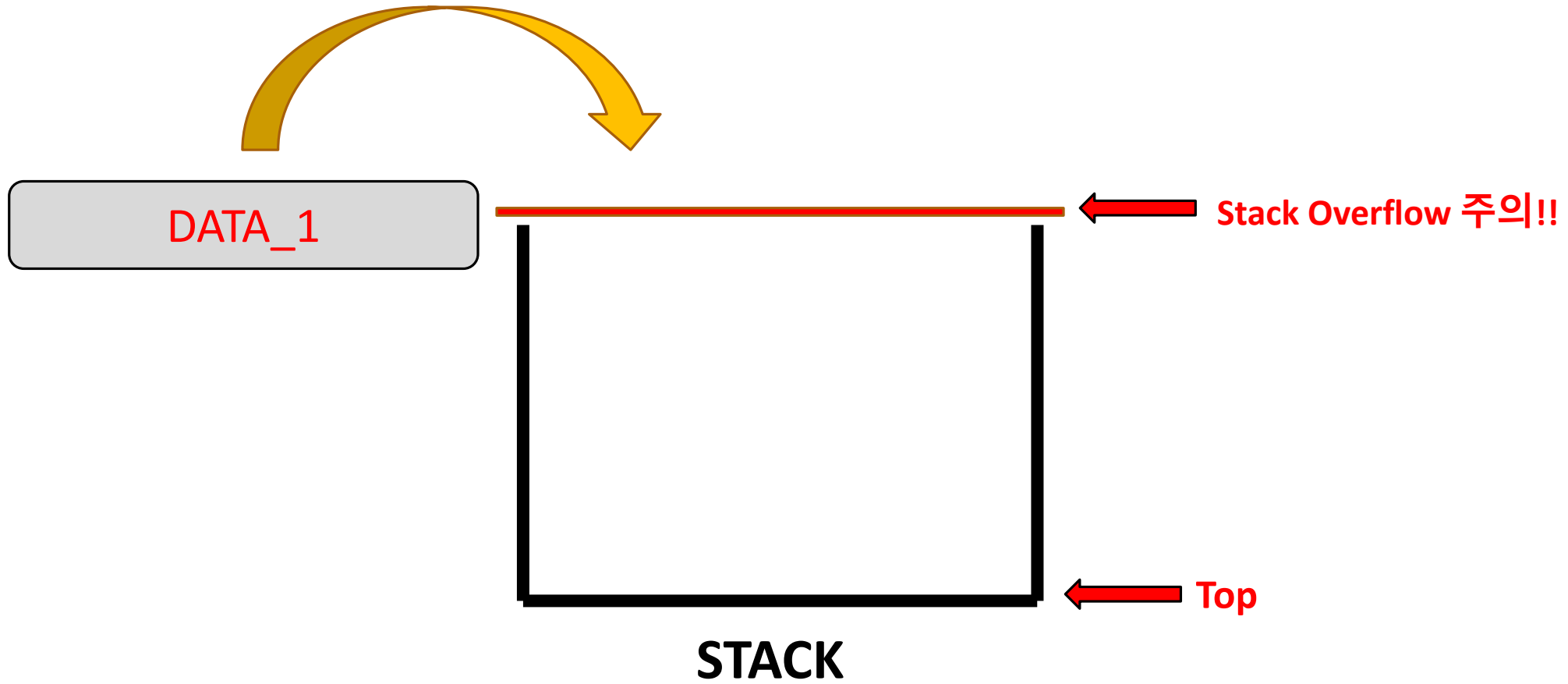
· LIFO(후입 선출)방식의
자료구조

1. 스택(STACK)이란?



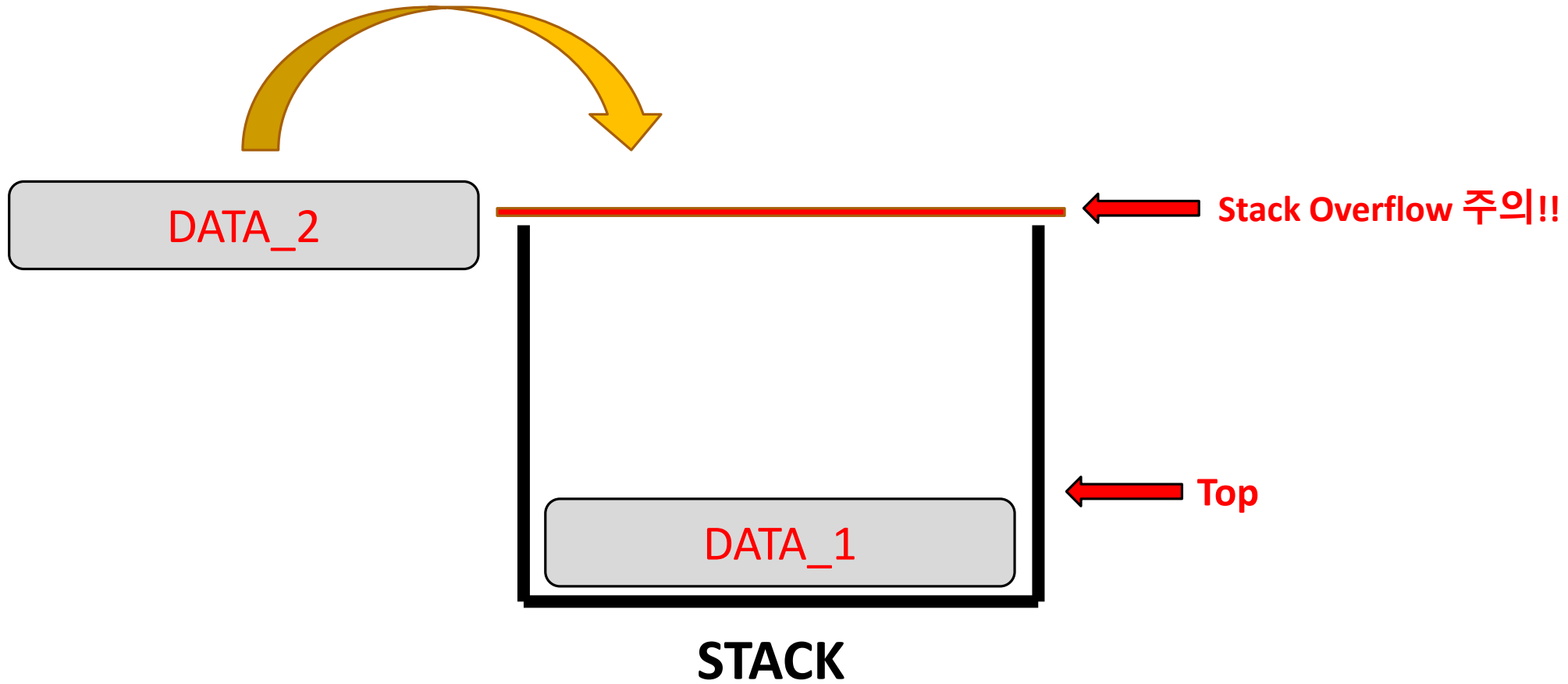
1. Push

Push



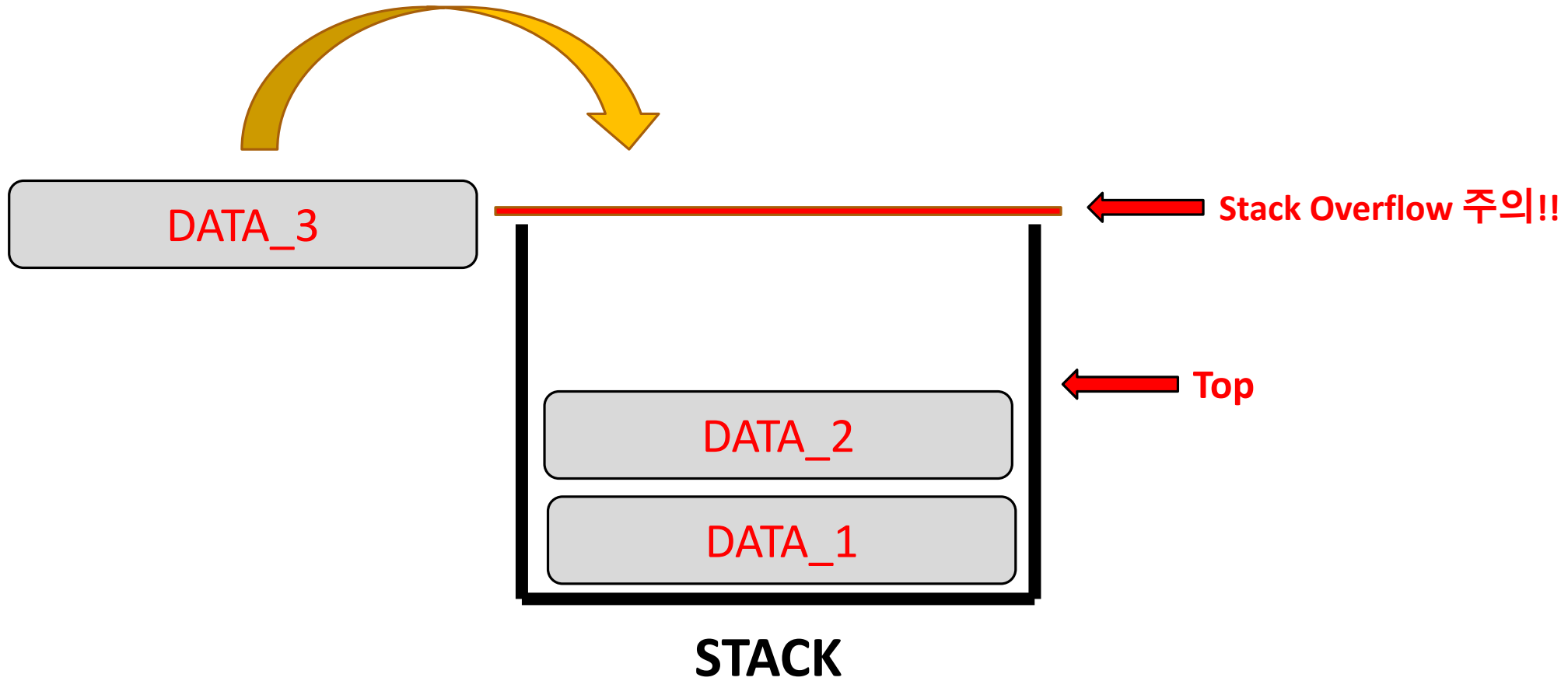
1. Push

Push



1. Push

Push

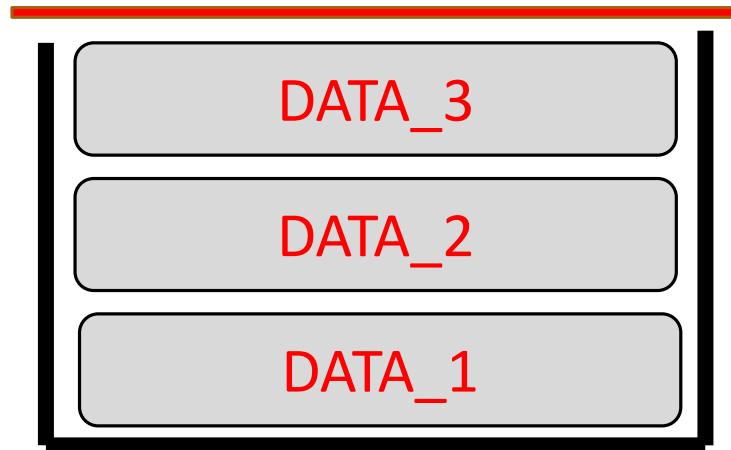


1. Push

Push



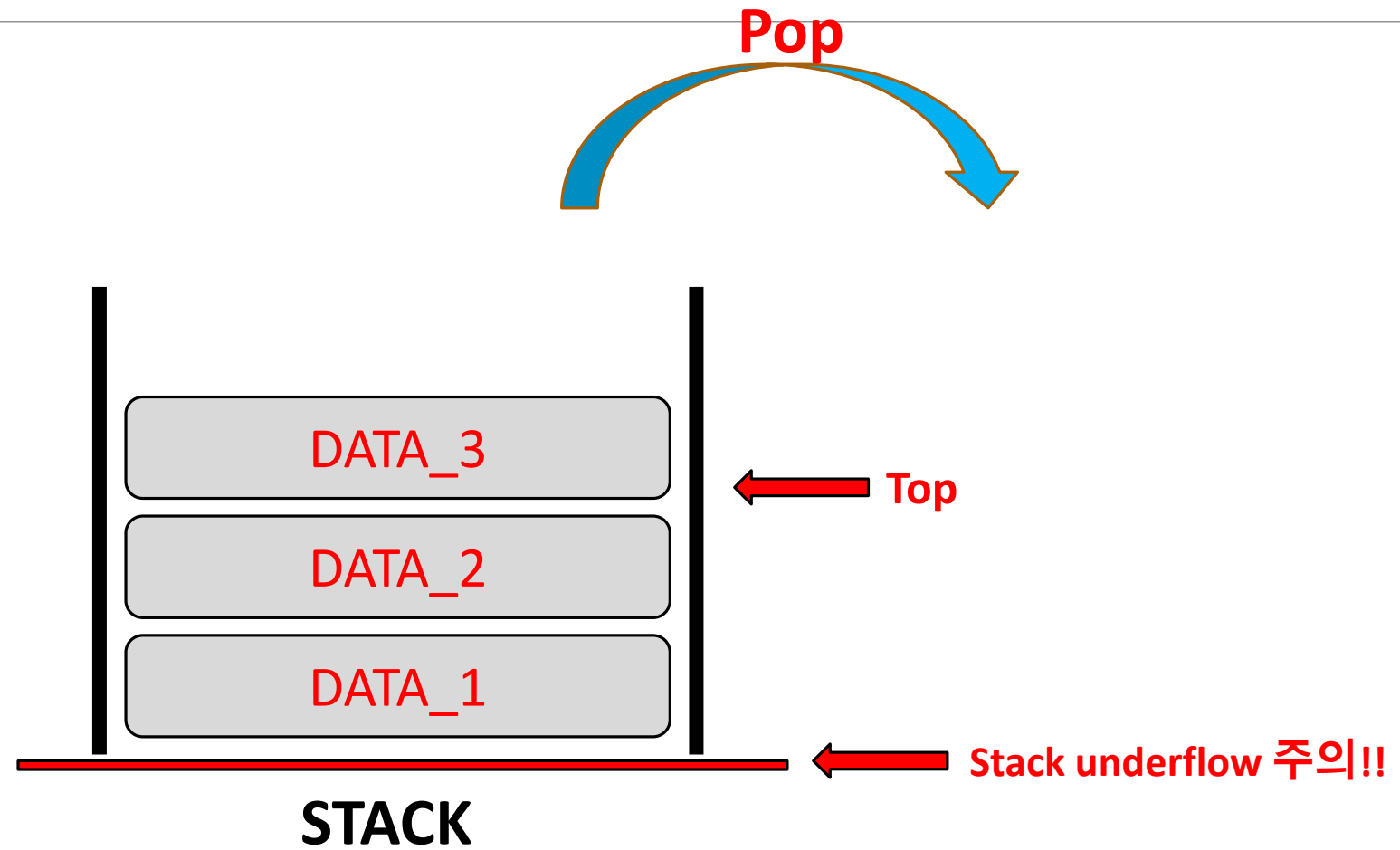
DATA_4



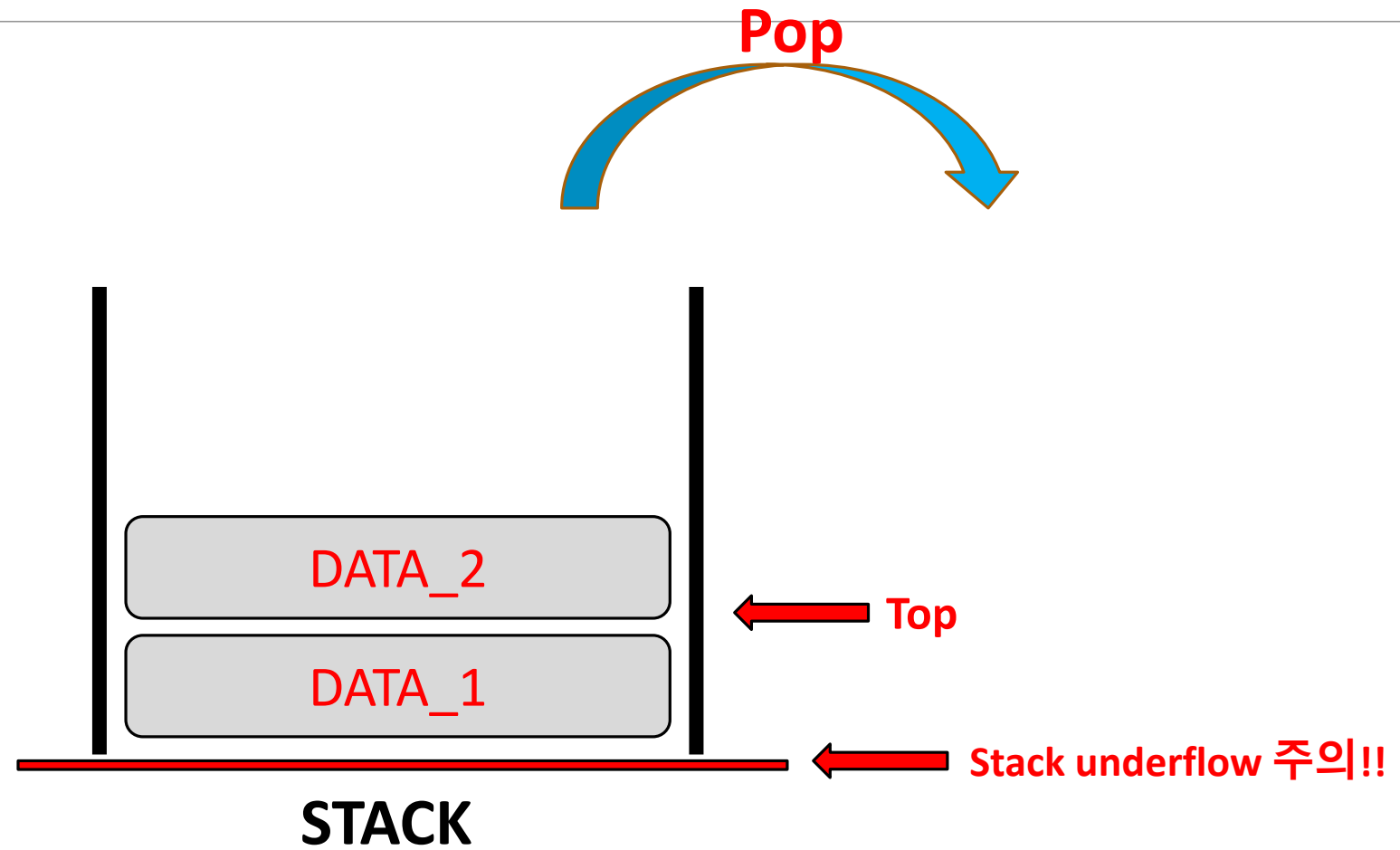
← Top

STACK

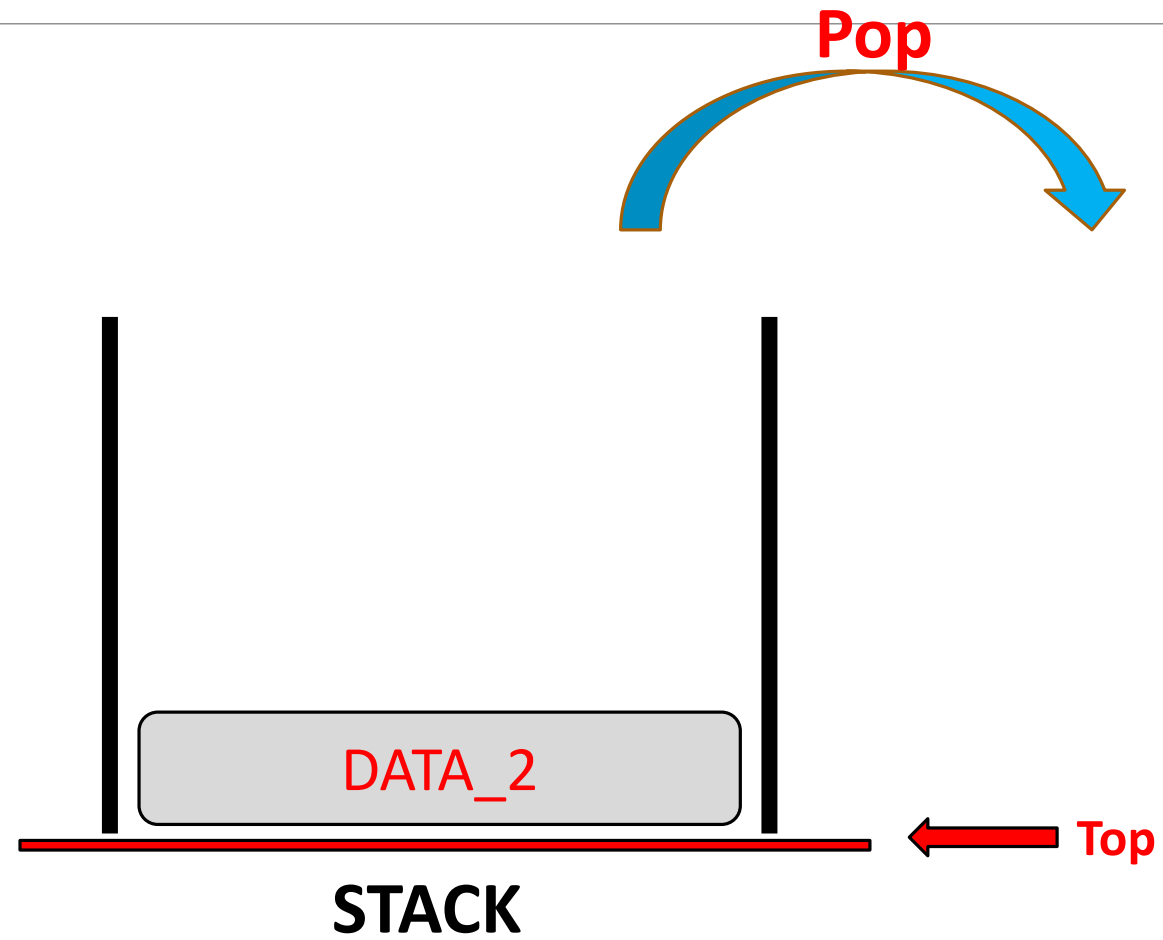
1. Pop



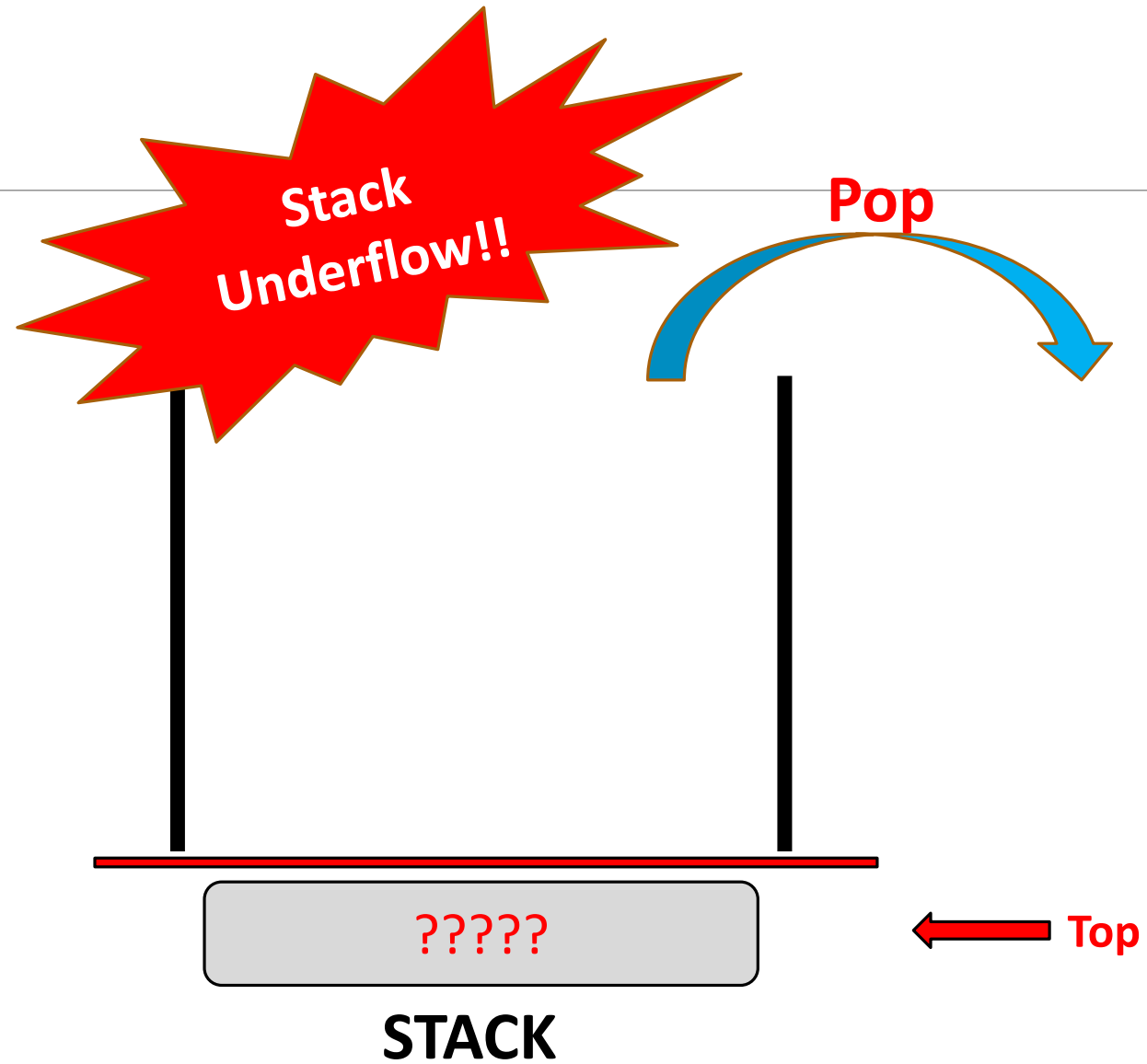
1. Pop



1. Pop

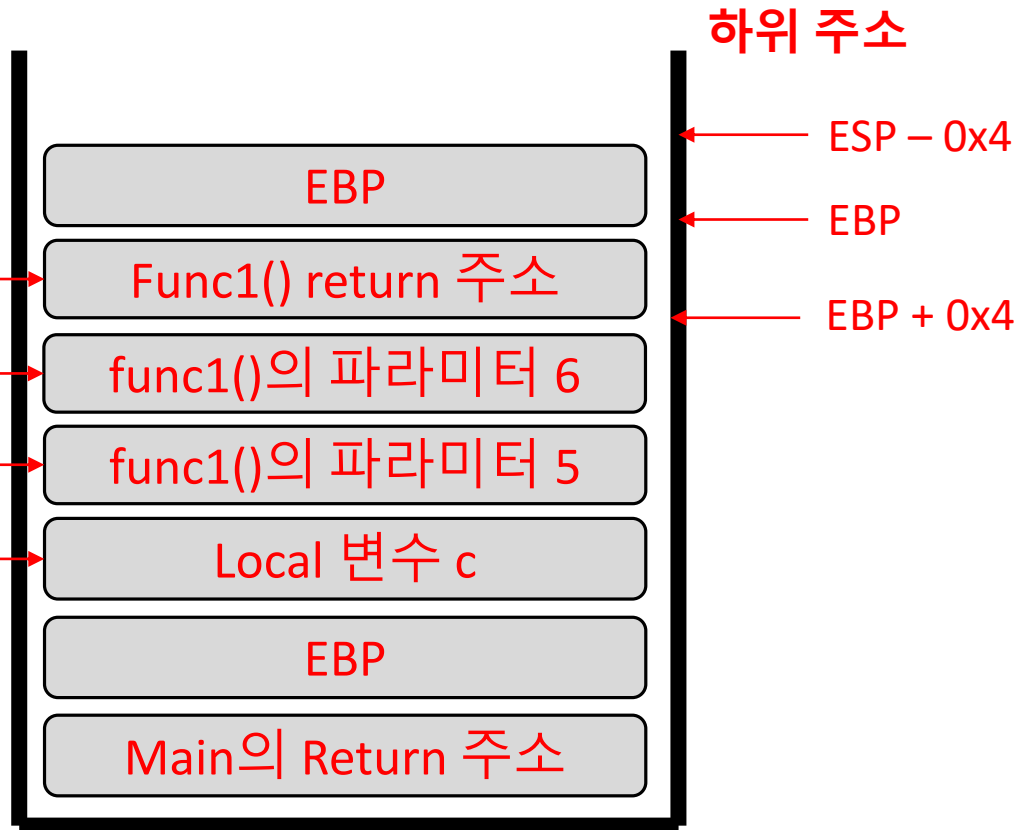


1. Pop



2. 시스템 스택(STACK)이란?

```
1  #include <stdio.h>
2
3  int func1(int, int);
4
5  int main()
6  {
7      int c;
8      c = func1(6, 5);
9      printf("%d", c);
10
11     return 0;
12 }
13
14 int func1(int a, int b)
15 {
16     return a+b;
17 }
```



STACK

3. 스택 추상 자료형(ADT)

- $\text{create()} ::=$ 스택을 생성
- $\text{is_empty}(s) ::=$ 스택이 비어 있는지를 검사
- $\text{is_full}(s) ::=$ 스택이 가득 찼는가를 검사
- $\text{push}(s, e) ::=$ 스택의 맨 위에 요소 e 를 추가
- $\text{pop}(s) ::=$ 스택의 맨 위에 있는 요소를 삭제
- $\text{peek}(s) ::=$ 스택의 맨 위에 있는 요소를 삭제하지 않고 반환

4. 스택 – 배열 (초기화 / 검사)

```
1  #include <stdio.h>
2
3  #define MAX_STACK_SIZE 5
4  typedef int element;
5  element stack[MAX_STACK_SIZE];
6
7  int top = -1;
8
9  int is_empty()
10 {
11     return (top == -1);
12 }
13
14 int is_full()
15 {
16     return (top == (MAX_STACK_SIZE - 1));
17 }
```

① stack 크기

② stack 배열

③ top 위치 : -1 부터 시작

④ Underflow 검사

⑤ Overflow 검사

메모리 2

주소: 0x0077A5A0 열: 자동

0x0077A5A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0077A5B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

조사식 1

0077a000 검색 심도: 3

이름	값	형식
stack	0x0077a5a0 {0, 0, 0, 0, 0}	int[5]
top	-1	int

감시할 항목 추가

Stack 메모리

4. 스택 – 배열(push)

```
19 void push(element item) ①
20 { 경과 시간 1ms 이하
21   if (is_full()) ②
22   {
23     fprintf(stderr, "스택 포화 에러\n");
24     return;
25   }
26   else stack[++top] = item; ③ ④
27 }
49 void main()
50 {
51   push(1);
52   push(2);
53   push(3);
```

① push

② Overflow 검사

③ top 위치 1 증가

④ 매개변수 (item) **'1'** 입력

조사식 1

이름	값
stack	0x0077a5a0 {1, 0, 0, 0, 0}
top	0
item	1

조사식 1

이름	값
stack	0x0077a5a0 {1, 0, 0, 0, 0}
[0]	1
[1]	0
[2]	0
[3]	0
[4]	0
top	0
item	1

메모리 2

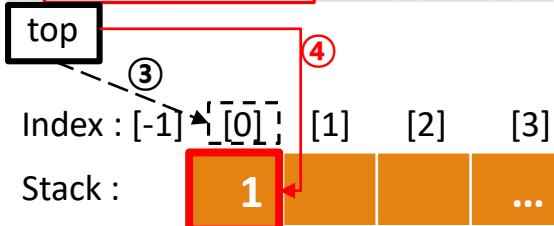
주소: 0x0077A5A0

0x0077A5A0

01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x0077A5AF

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



4. 스택 – 배열(push)

```
19 void push(element item) ①
20 { 경과 시간 1ms 이하
21   if (is_full()) ②
22   {
23     fprintf(stderr, "스택 포화 에러\n");
24     return;
25   }
26   else stack[++top] = item; ③ ④
27 }
49 void main()
50 {
51   push(1);
52   push(2);
53   push(3);
```

① push

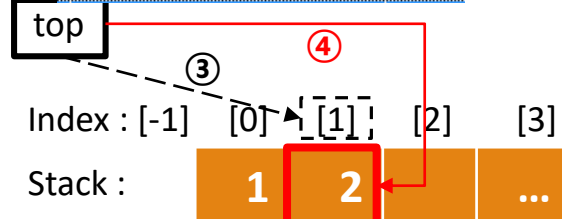
② Overflow 검사

③ top 위치 1 증가

④ 매개변수 (item) 입력
[1 -> **'2'**]

조사식 2	
검색(Ctrl+E)	
이름	값
top	1
item	2

조사식 1	
0077a5a0	
이름	값
stack	0x0077a5a0 (1, 2, 0, 0, 0)
[0]	1
[1]	2
[2]	0
[3]	0
[4]	0
top	1
item	2



메모리 2	
주소: 0x0077A5A0	
0x0077A5A0	01 00 00 00 02 00 00 00 00
0x0077A5AF	00 00 00 00 00 00 00 00 00

4. 스택 – 배열(push)

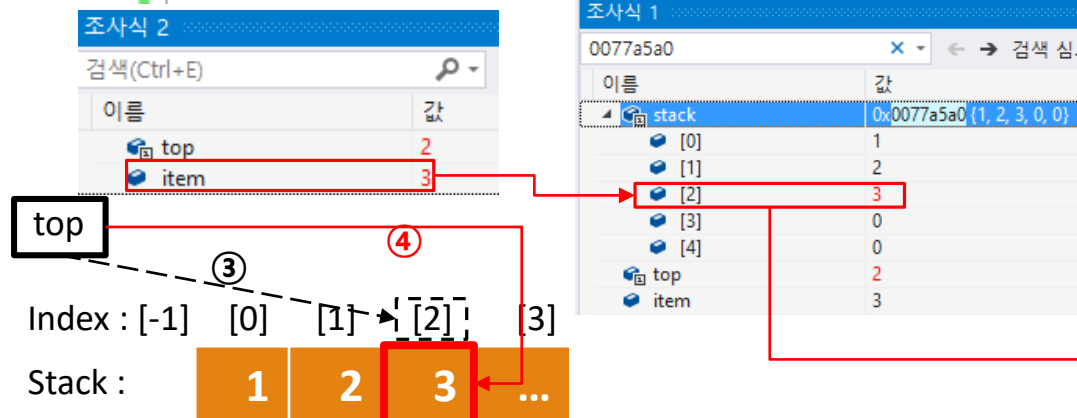
```
19 void push(element item) ①
20 { 경과 시간 1ms 이하
21   if (is_full()) ②
22   {
23     fprintf(stderr, "스택 포화 에러\n");
24     return;
25   }
26   else stack[++top] = item; ③ ④
27 }
49 void main()
50 {
51   push(1);
52   push(2);
53   push(3);
```

① push

② Overflow 검사

③ top 위치 1 증가

④ 매개변수 (item) 입력
[1 -> 2 -> **3**]



메모리 2

주소	0x0077A5A0
0x0077A5A0	01 00 00 00 02 00 00 00 03 00 00 00
0x0077A5AF	00 00 00 00 00 00 00 00 00 00 00 00

4. 스택 – 배열(pop)

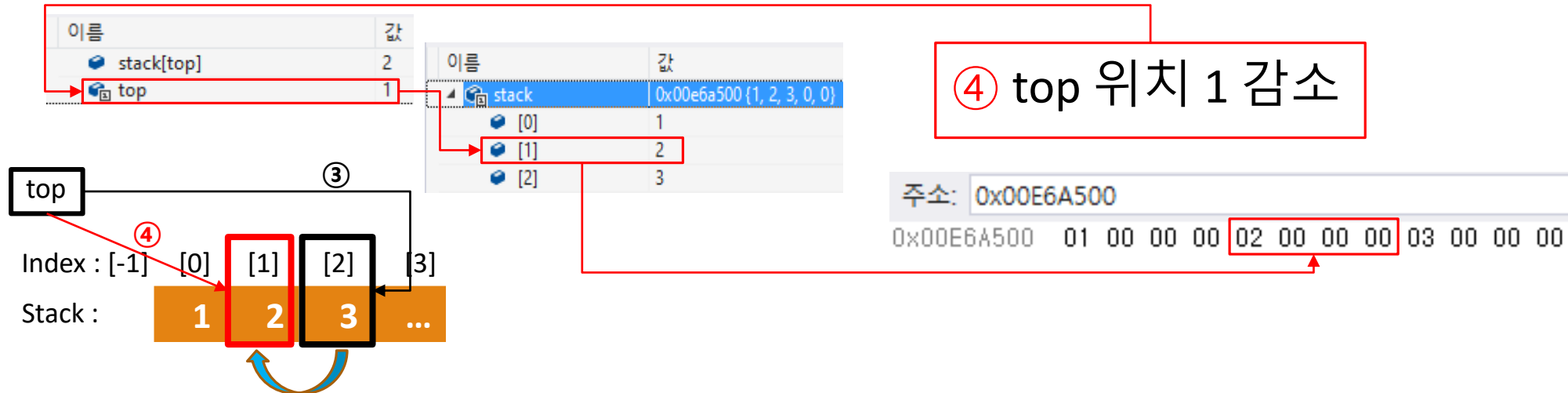
```
29 element pop() ①
30 {
31     if (is_empty()) ②
32     {
33         fprintf(stderr, "스택 공백 에러\n");
34         exit(1);
35     }
36     else return stack[top--]; ③
37 }
```

① pop

② Underflow 검사

③ stack 데이터 '3' 출력

④ top 위치 1 감소



4. 스택 – 배열(pop)

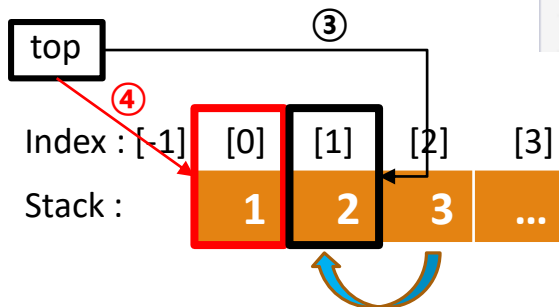
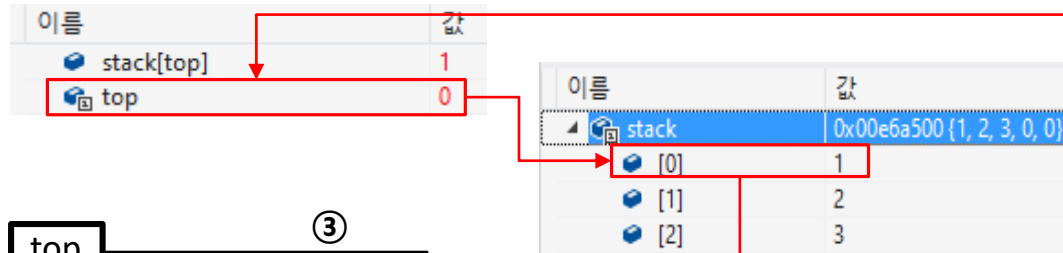
```
29 element pop() ①
30 {
31     if (is_empty()) ②
32     {
33         fprintf(stderr, "스택 공백 에러\n");
34         exit(1);
35     }
36     else return stack[top--]; ③
37 }
```

① pop

② Underflow 검사

③ stack 데이터 '2' 출력

④ top 위치 1 감소



주소: 0x00E6A500

0x00E6A500 01 00 00 00 02 00 00 00 03 00 00 00

4. 스택 – 배열(pop)

```
29 element pop() ①
30 {
31     if (is_empty()) ②
32     {
33         fprintf(stderr, "스택 공백 에러\n");
34         exit(1);
35     }
36     else return stack[top--]; ③
37 }
```

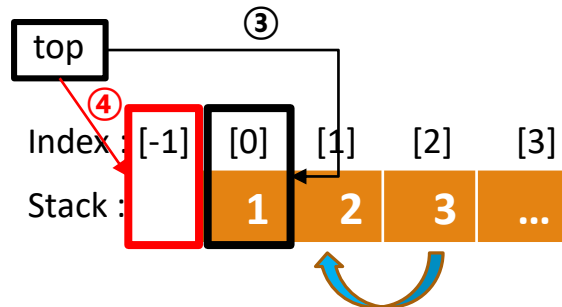
① pop 총 3회

② Underflow 검사

③ stack 데이터 '1' 출력

④ top 위치 1 감소

이름	값
stack[top]	0
top	-1



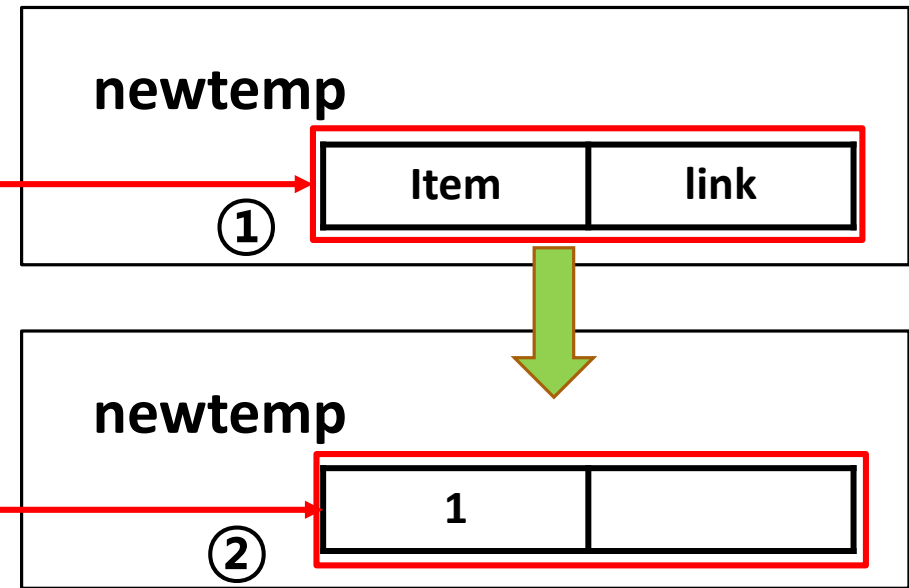
주소: 0x00E6A000
0x00E6A000 ff ff ff ff

4. 스택 – 연결 리스트(push)

```
// 삽입 함수
void push(LinkedStackType* s, element item)
{
    StackNode* newtemp = (StackNode*)malloc(sizeof(StackNode));

    if (newtemp == NULL)
    {
        fprintf(stderr, "메모리 할당에러\n");
        return;
    }
    else
    {
        newtemp->item = item;
        newtemp->link = s->top;
        s->top = newtemp;
    }
}
```

이름	값
s	0x00aff898 {t
item	1
newtemp	0x00df67e0 {
item	1
link	0xcdcdcdcd



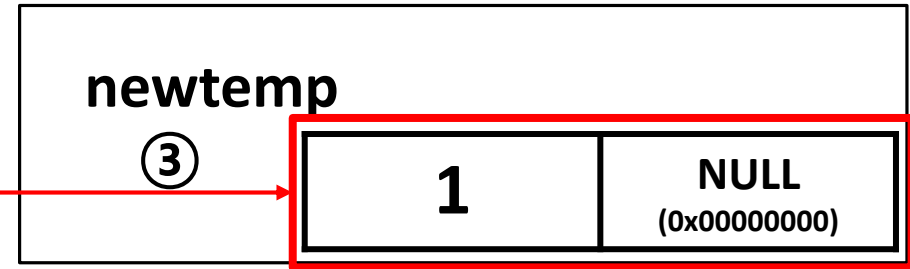
주소: 0x00DF67E0

0x00DF67E0 → 01 00 00 00

4. 스택 – 연결 리스트(push)

```
// 삽입 함수
void push(LinkedListType* s, element item)
{
    StackNode* newtemp = (StackNode*)malloc(sizeof(StackNode));

    if (newtemp == NULL)
    {
        fprintf(stderr, "메모리 할당에러\n");
        return;
    }
    else
    {
        newtemp->item = item;
        newtemp->link = s->top;
        s->top = newtemp;
    }
}
```



이름	값
s	0x00aff898 {top=0x00000...
item	1
newtemp	0x00df67e0 {item=1 link=..
item	1
link	0x00000000 <NULL>

주소: 0x00DF67E0

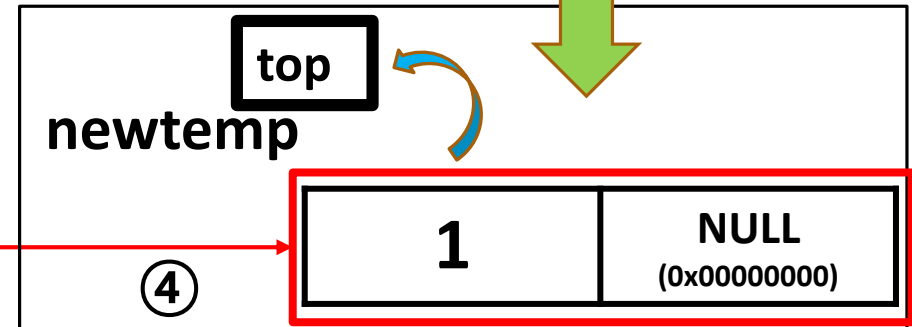
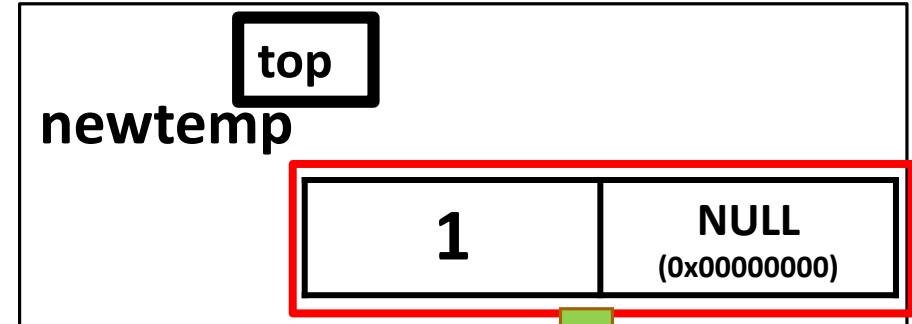
0x00DF67E0	01 00 00 00	00 00 00 00
------------	-------------	-------------

4. 스택 – 연결 리스트(push)

```
// 삽입 함수
void push(LinkedListType* s, element item)
{
    StackNode* newtemp = (StackNode*)malloc(sizeof(StackNode));

    if (newtemp == NULL)
    {
        fprintf(stderr, "메모리 할당에러\n");
        return;
    }
    else
    {
        newtemp->item = item;
        newtemp->link = s->top;
        s->top = newtemp;
    }
}
```

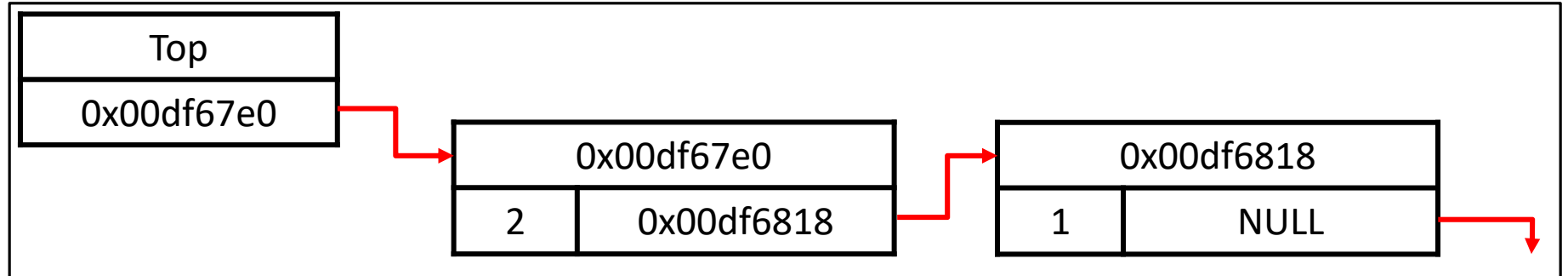
이름	값
s	0x00aff898 {top=0x00d
s->top	0x00df67e0 {item=1 lin
item	1
link	0x00000000 <NULL>



주소: 0x00AFF898
0x00AFF898 e0 67 df 00 cc cc cc cc

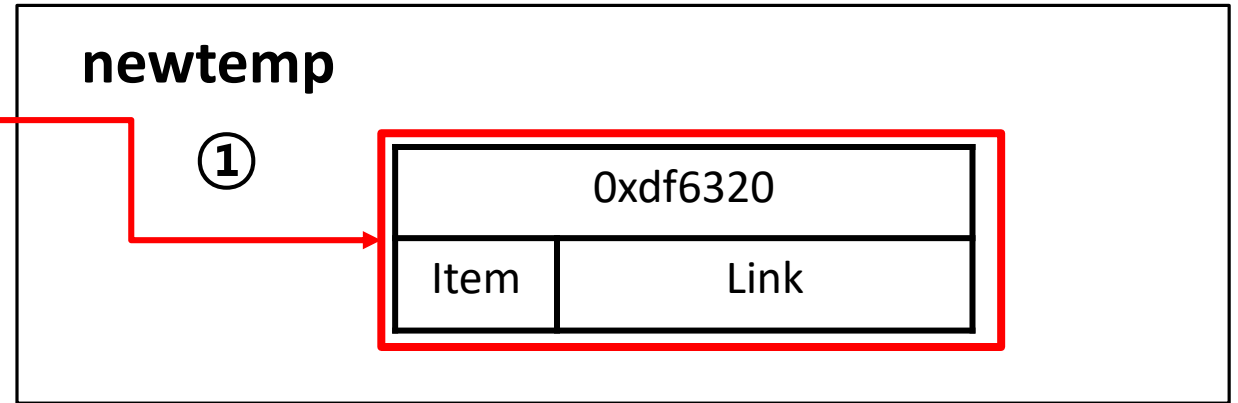
4. 스택 – 연결 리스트(push)

```
push(&s, 1);  
push(&s, 2);  
push(&s, 3);
```



```
// 삽입 함수  
void push(LinkedListType* s, element item)  
{  
    StackNode* newtemp = (StackNode*)malloc(sizeof(StackNode));  
  
    if (newtemp == NULL)  
    {  
        fprintf(stderr, "메모리 할당에러\n");  
        return;  
    }  
}
```

이름	값
newtemp	0x00df6320
item	-842150451
link	0xcdcdcdcd



메모리 2
주소: 0x00DF6320
0x00DF6320 cd cd cd cd cd cd cd cd

4. 스택 – 연결 리스트(push)

```
newtemp->item = item;  
newtemp->link = s->top;  
s->top = newtemp;
```

이름	값
s->top	0x00df6818 {
newtemp	0x00df6320 {
item	3
link	0xcdcdcdcd

newtemp

0xdf6320

②

3

주소: 0x00DF6320

0x00DF6320 03 00 00 00 cd cd cd cd

```
newtemp->item = item;  
newtemp->link = s->top;  
s->top = newtemp;
```

이름	값
s->top	0x00df6818 {
newtemp	0x00df6320 {
item	3
link	0x00df6818

newtemp

0xdf6320

③

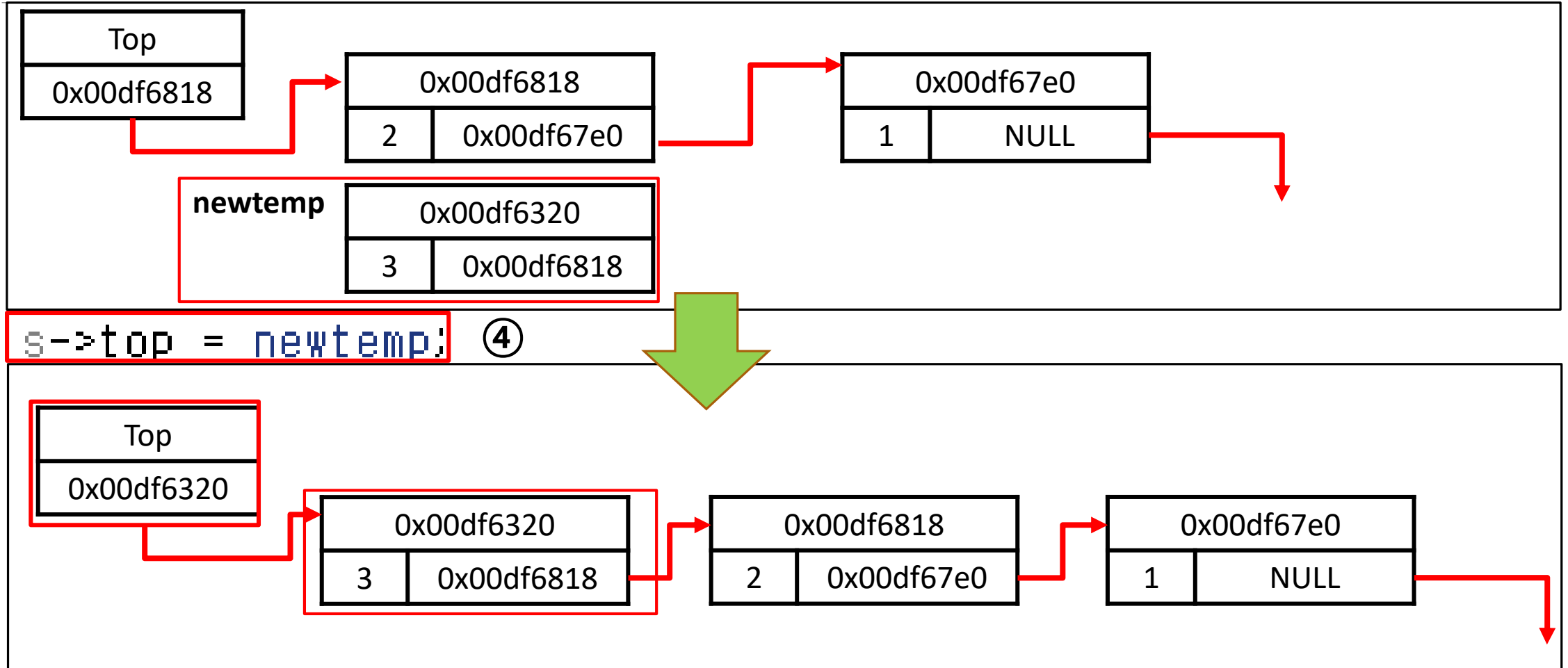
3

0x00df6818

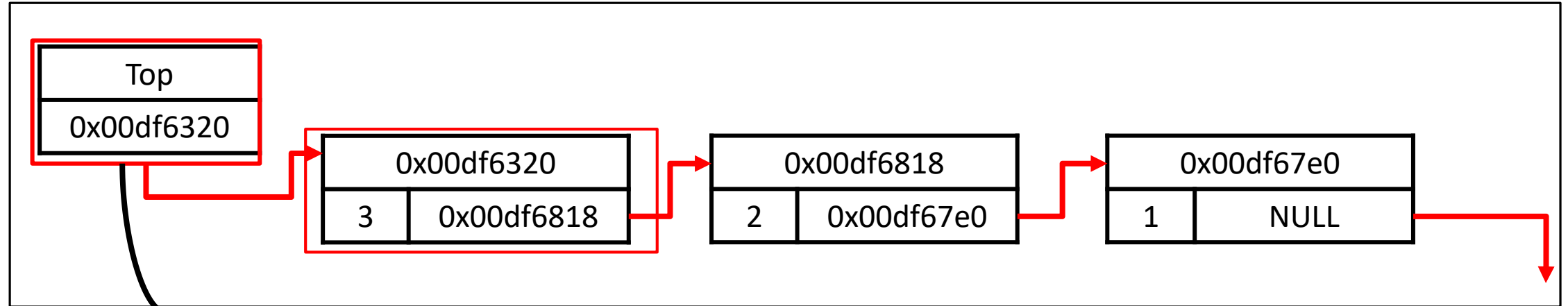
주소: 0x00DF6320

0x00DF6320 03 00 00 00 18 68 df 00

4. 스택 – 연결 리스트(push)



4. 스택 – 연결 리스트(push)



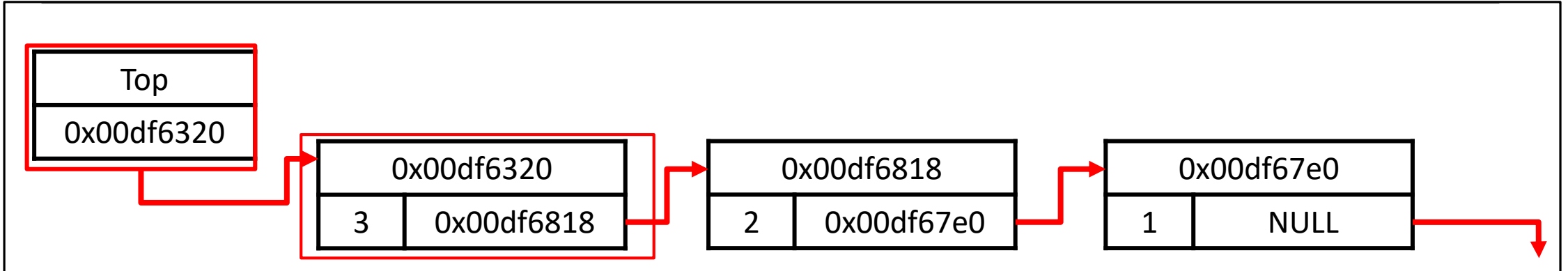
이름	값
s->top	0x00df6320
item	3
link	0x00df6818
newtemp	0x00df6320
s->top	0x00df6818

이름	값
s->top	0x00df6320
item	3
link	0x00df6818
newtemp	0x00df6320
item	3
link	0x00df6818
item	2
link	0x00df67e0
item	1
link	0x00000000

주소: 0x00AFF898
0x00AFF898 20 63 df 00

Push 연산을 하는 동안 top은
1 -> 2 -> '3'.
즉, 최종적으로 newtemp를
가리킨다.

4. 스택 – 연결 리스트(pop)



```
// 삭제 함수
element pop(LinkedStackType* s) ①
{
    경과시간 1ms 이하
    if (is_empty(s)) ②
    {
        fprintf(stderr, "스택이 비어있음\n");
        exit(1);
    }
    else
    {
        StackNode* deltemp = s->top; ③
        int item = deltemp->item;

        s->top = s->top->link; ④
        free(deltemp); ⑤

        return item;
    }
}
```

① pop 총 3회

② Underflow 검사

③ stack 데이터 '1' 출력

④ top 위치 1 감소

⑤ 기존 top 동적 메모리 해제

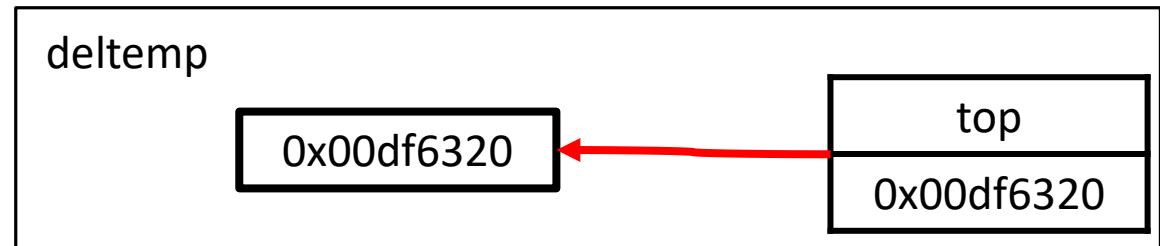
4. 스택 – 연결 리스트(pop)

```
StackNode* deltemp = s->top;  
... ..
```

③ top의 메모리 주소 백업

이름	값
▶ s->top	0x00df6320
▶ deltemp	0xffffffff

이름	값
▶ s->top	0x00df6320
▶ item	3
▶ link	0x00df6818
▶ deltemp	0x00df6320
▶ &deltemp	0x00aff7b4

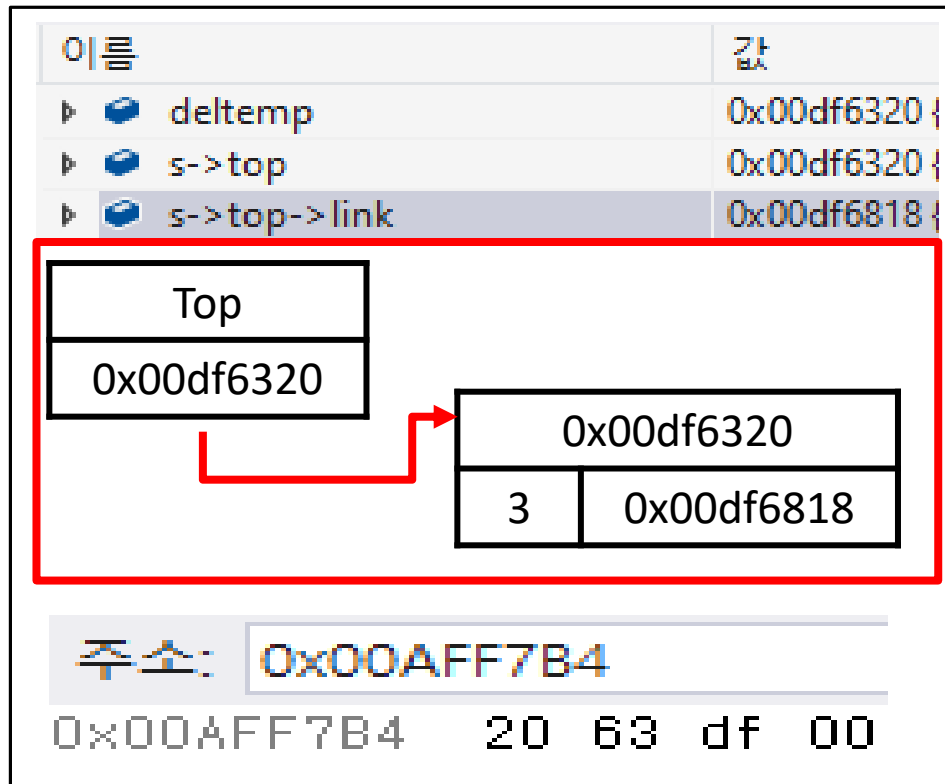


이름	값
▶ deltemp	0x00df6320
▶ s->top	0x00df6320

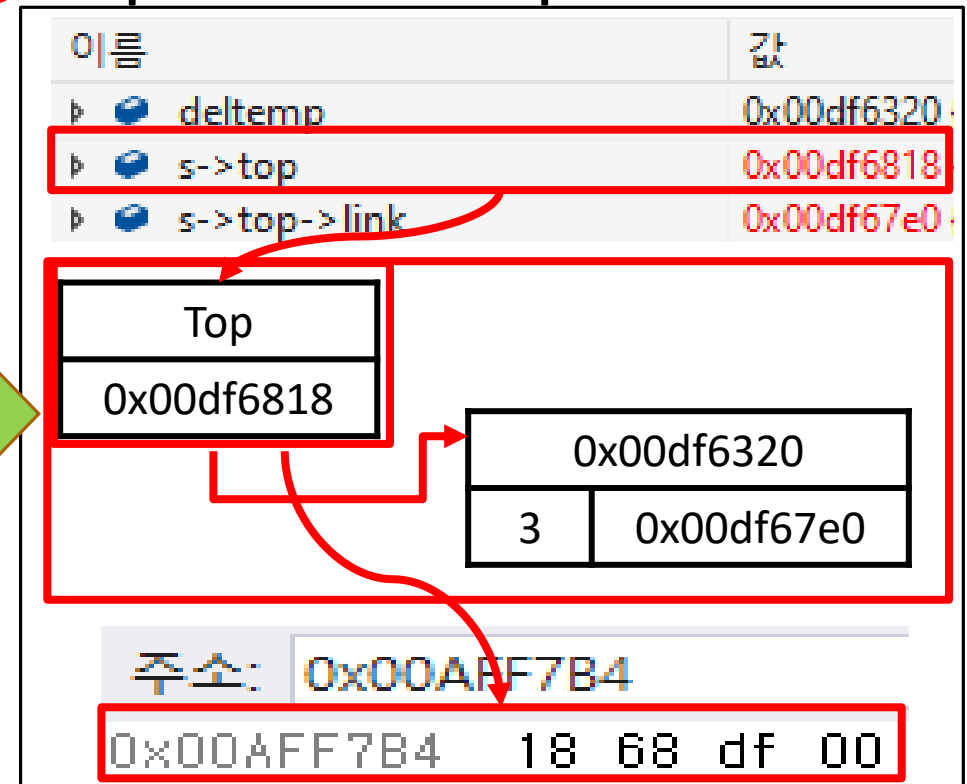
주소:	0x00AFF7B4
0x00AFF7B4	20 63 df 00

4. 스택 – 연결 리스트(pop)

`s->top = s->top->link;` ④



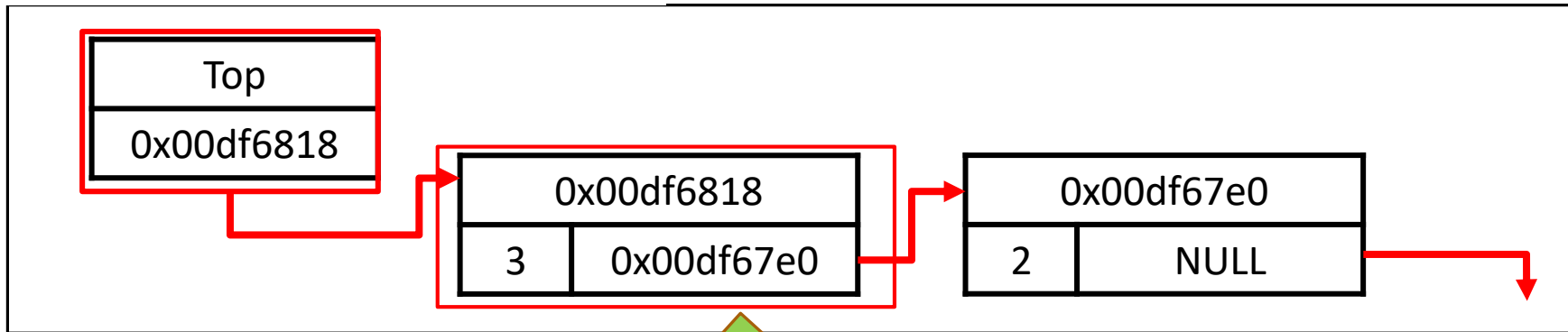
④ top의 값을 top->link로 변경



4. 스택 – 연결 리스트(pop)

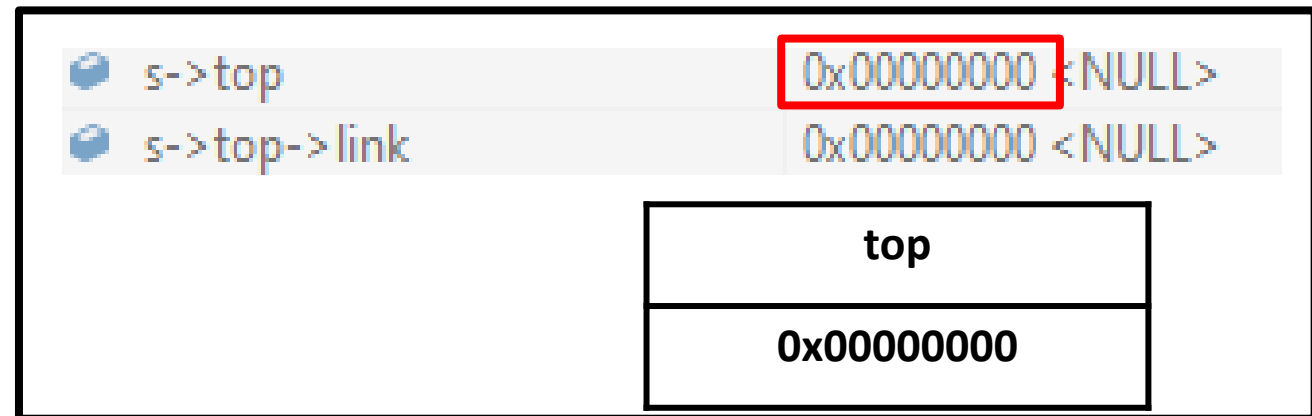
`free(deltemp);` ⑤

⑤ 기존 top 동적 메모리 해제



```
printf("%d\n", pop(&s));
printf("%d\n", pop(&s));
printf("%d\n", pop(&s));
```

• 총 `pop()` 연산 3회 수행하여
결과는 3 -> 2 -> 1 순으로 데이터를
`pop` 한다.



5. 구현

구현 환경

OS : Windows 10 / 64 bit

IDE : Visual Studio 2019