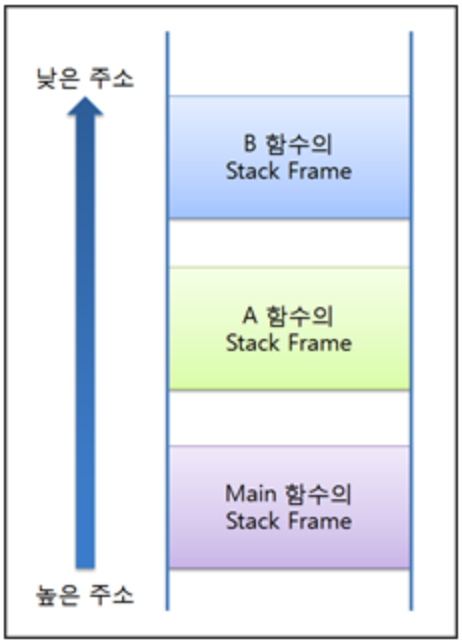
10 함수 호출 규약

1. Calling Convention

> *'함수를 호출할 때 파라미터를 어떤 식으로 전달하는가?'에 대한 일종의 약속*

- 함수 호출 전 파라미터를 스택을 통해 전달
- 스택이란 프로세스에 정의된 메모리 공간이다.
  - 큰 메모리 주소 → 작은 메모리 주소쪽으로 자란다.
  - PE 헤더에 그 크기가 명시되어 있다.
    - 즉 프로세스 실행 시 스택 메모리의 크기가 결정된다.
- 함수 실행 완료 후 ESP 값은 함수 호출 전으로 복원되어야 한다.
  - 참조 가능한 스택의 크기가 줄지 않는다.
- 스택 메모리는 고정되어 있고 ESP로 스택의 현재 위치를 가리키는데, 만약 ESP가 스택의 끝을 가리킨다면 더이상 스택을 사용할 수 없다.
  - 함수 호출 후 ESP를 어떻게 정리하는지에 대한 약속이 바로 함수 호출 규약이다.
    - cdecl
    - stdcall
    - fastcall



▲함수의 Stack Frame

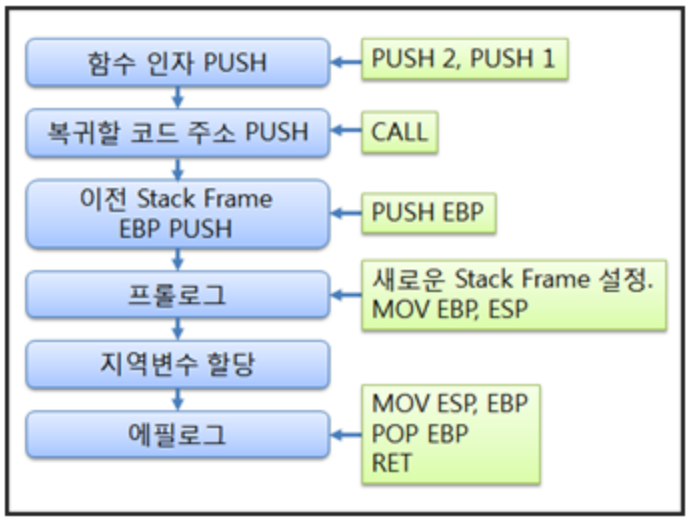
▲Stack Frame의 구조

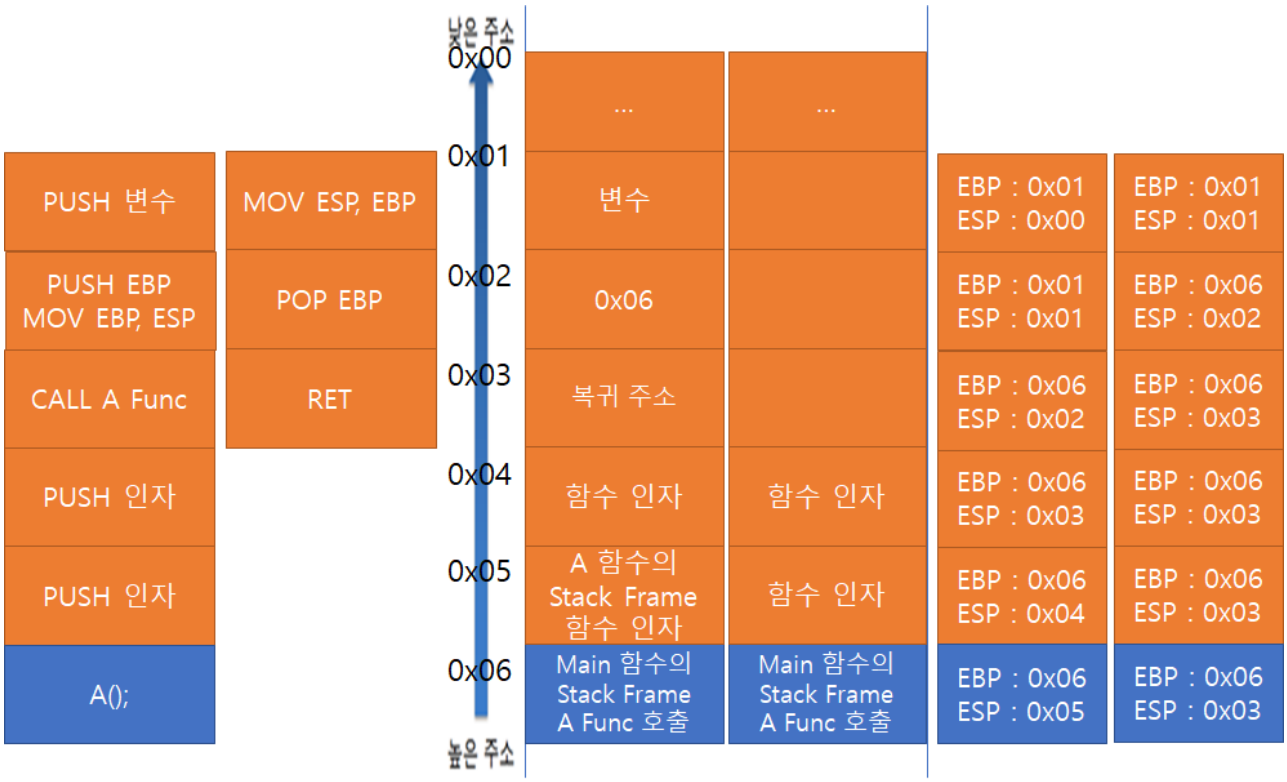
Stack Frame은 함수의 호출 과정에서 호출되는 함수가 사용하기 위해 할당되는 Stack의 공간을 의미한다.

프로그램이 실행되면 가장 먼저 Main 함수의 Stack Frame을 할당한다. Main 함수가 A 함수를 호출하면서 A의 Stack Frame을 할당하고, 다시 A 함수 내에서 B함수를 호출하면 B의 Stack Frame을 할당한다.

Stack Frame에는 함수를 호출할 때 입력한 전달 인자, 함수가 종료될 때 복귀할 명령어의 주소(return address), 이전 Stack Frame의 EBP값을 저장하고, 지역변수를 저장하기 위한 공간을 할당한다.

1) 함수 호출 시 **Stack**의 변화 순서





2. 주요 함수 호출 규약

*) 함수 호출 규약의 구분 방법

> *Caller : 호출자, 함수를 호출한 쪽*
> *Callee : 피 호출자, 호출을 당한 함수*

인자 전달	인자 전달의 순서 (왼쪽 인자부터 / 오른쪽 인자부터)
	인자 전달에 사용하는 매체 (Stack / 레지스터)
Stack Frame 정리 방법	Caller를 이용한 정리
	Callee를 이용한 정리

규약	인자 전달 순서	인자 전달 매체	Stack를 정리하는 함수
cdecl	←	Stack	Caller
stdcall	←	Stack	Callee
fastcall	←	레지스터 + Stack	Callee

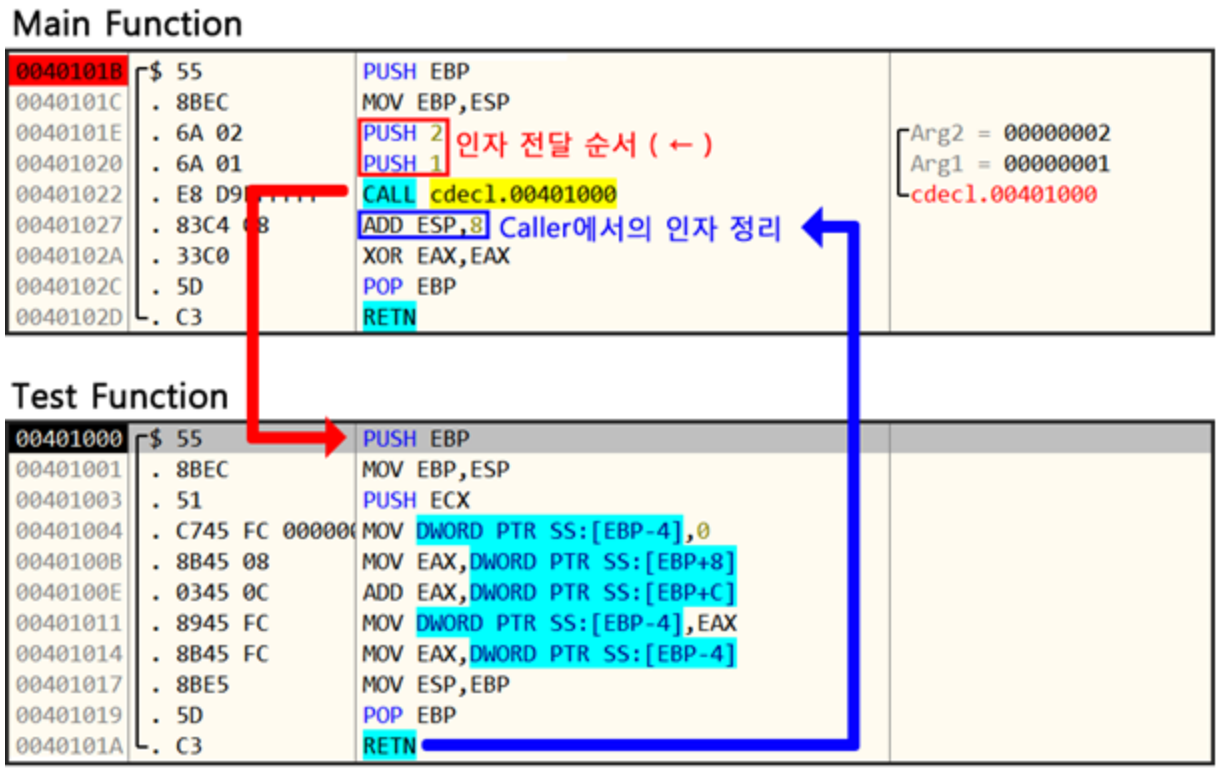
1) cdecl

> *C 언어의 printf() 함수와 같이 가변 길이 파라미터를 전달 할 수 있다는 장점이 있다.*

인자 전달 순서	가장 오른쪽 인자부터 전달한다. (←)
인자 전달 매체	Stack 을 사용한다.
Stack Frame 정리 방법	함수를 호출한 Caller가 인자를 정리한다.
C언어와 C++에서의 표준 함수 호출 규약이다. Caller가 인자를 정리하는 규약이므로, 가변인자를 사용할 수 있다.	

```
#include <stdio.h>
int __cdecl cdecl_Test(int a, int b)
{
  int nSum = 0;
  nSum = a + b;
  return nSum;
}

int main()
{
  cdecl_Test(1, 2);
  return 0;
}
```



2) stdcall

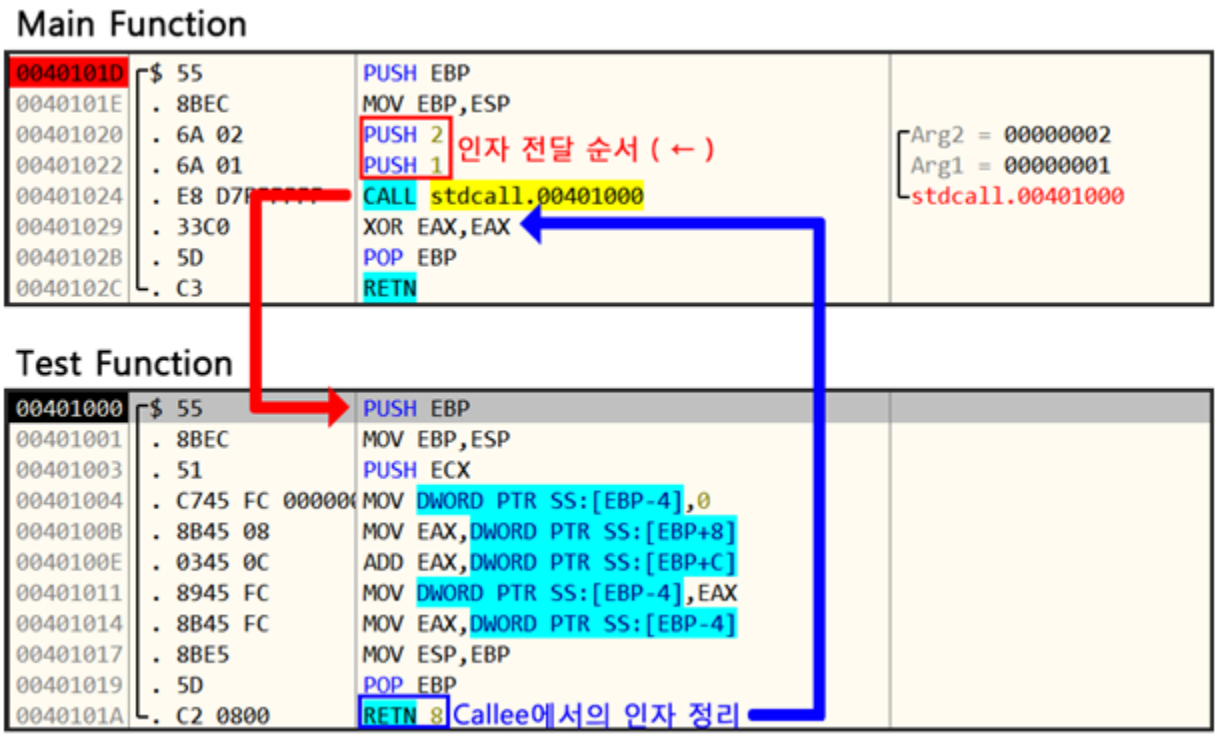
> *Win32 API에서 사용된다.*
> *호출되는 함수 내부에서 스택 정리 코드가 존재하므로 함수를 호출할 때마다 ADD ESP, XXX 명령을 써줘야 하는 cdecl 방식에 비해서 코드 크기가 작아진다.*

인자 전달 순서	가장 오른쪽 인자부터 전달한다. (←)
인자 전달 매체	Stack 을 사용한다.
Stack Frame 정리 방법	호출을 당한 Callee가 함수를 종료하면서 인자를 정리한다.
Window API, Visual Basic에서 사용하는 표준 규약이다. 코드가 간결하지만 가변 인자를 사용할 수 없다.	

```
#include <stdio.h>
int __stdcall stdcall_Test(int a, int b)
{
  int nSum = 0;
  nSum = a + b;
```

```
  return nSum;
}

int main()
{
  stdCall_Test(1, 2);
  return 0;
}
```



3) fastcall

- 기본적으로 stdcall 방식과 같다.
  - 다만, 함수에 전달하는 파리미터 일부(2개 까지)를 스택 메모리가 아닌 레지스터를 이용하여 전달
    - 파리미터가 4개라면 앞의 두개의 파라미터는 각각 ECX, EDX 레지스터를 이용하여 전달