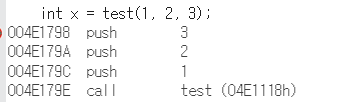
인자와 반환값은 어떻게 처리하는가?

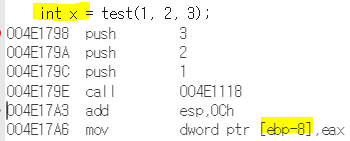
인자는 오른쪽부터 들어간다.



그리고 test 안으로 들어간다.

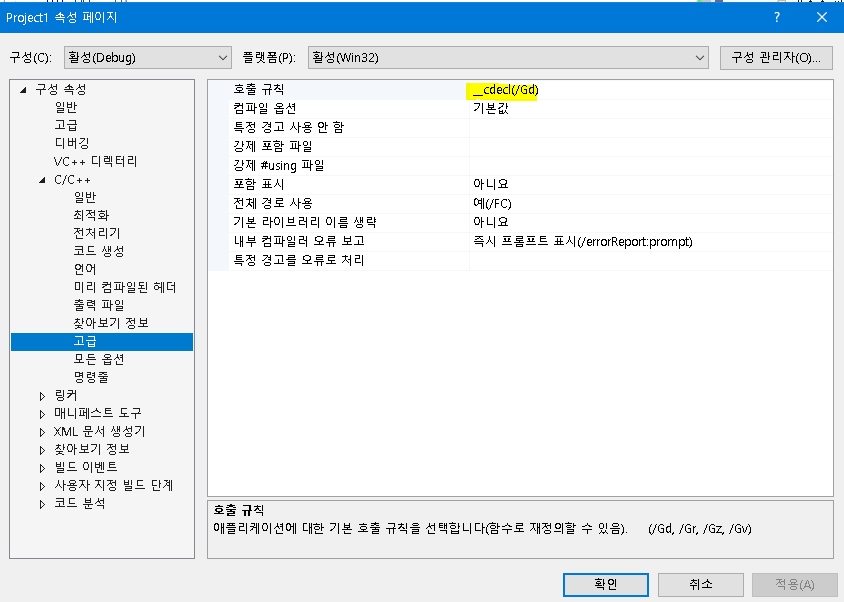


리턴값을 eax에 넣어두고



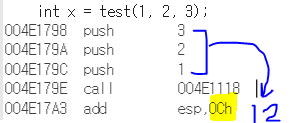
변수x에 eax의 값을 바로 받아온다.

함수호출규약

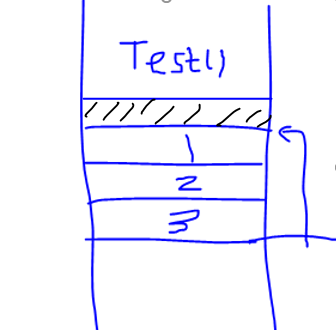


컴파일러에 지정된 옵션. 인자를 던지는 방법 그리고 인자로 던졌던 내용을 정리하는 주체가 가장 큰 차이점. 그냥 신경쓰지 않아도 됐던 이유는 우리가 짠 코드를 우리만 호출했기 때문. 근데 따로 정의해야하는 경우가 있다. 대부분 정해져있다. 윈도우api 같은 경우 std call을 쓰는데 내가 만든 함수를 윈도우에서 등록한다. 그럼 거기서 내 함수를 호출한다. 스레드도 우리가 함수를만들지만, 함수를 호출하는건 os안에 만들어진 함수에서.

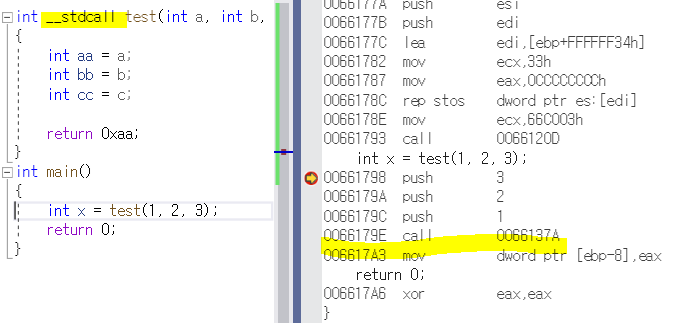
Cdecl : RTL 오른쪽에서 왼쪽으로 넘어간다. 이건 STDCALL도 마찬가지./ 근데 차이는 스택정리해주는 애가 다르다. 얘는 호출자가 스택을 정리하는데, stdcall은 피호출자가 스택을 정리한다.



Push 번으로 스택이 올라가고,

 (검정 빗금은 함수 return 후 돌아갈 주소가 들어가있음)

Call 이후에 호출자가 스택 push 3번으로 올라갔던 esp를 add해서 밑으로 내려준다.



그런데 \_\_stdcall 호출규약으로 하겠다고 하니까 아까 봤던 call 밑에 스택을 정리하는 부분이 사라져있다. 정리는 그럼 함수를 빠져나오기 전에 하겠다는 것



함수 안에서 이렇게 되었는데 ret를 하면서 12를 내려달라는 것. 규약을 양쪽에서 맞추지 않으면 스택이 밀리거나 낭비됨. 두번 스택이 정리되거나, 아예 정리가 안돼서 12바이트가 낭비되던지.

Static 으로 선언을 하고 매번 지역변수 선언을 안하고 재사용을 한다면? 그러면 이동을 데이터영역까지 더 멀리해야하는데 이동이 문제가 아니라 캐시 히트 문제 때문에 속도차이가 난다. 전역을 쓰면 캐시미스가 날 확률이 크다.

그리고 객체를 지역에서 static으로 만들 경우 불필요한 if문으로의 비교가 들어간다.

또 전역으로 static으로 선언하면 소스파일마다 생기고 서로 다른 전역변수가 된다. 전역이지만 내 파일 소스코드 안에서만 사용하게 해달라고 한다. 다른 파일에서 같은 이름으로 선언하면 다른 메모리로 잡힌다.

선행처리기

#달린거

\_\_DATA\_\_ 컴파일 시점의 날짜가 하드코딩으로 기입이 됨 %s로 출력가능. 빌드할때의 날짜.  
\_\_FILE\_\_ 현재 처리중인 소스파일의 이름 출력  
\_\_LINE\_\_ 사용중인 라인번호  
\_\_TIME\_\_ 컴파일 시간

Typedef WINAPI에서 자기네가 선언한 변수타입을 다 typedef CHAR, DWORD 이런걸 선언하는데 왜일까? WINAPI는 C만을 위해 나온게 아니다. API문서 기준으로 봤을 때 하나의 통일된 규격이 필요하다.

재귀함수 왜 쓸까? 그 앞에 보관됐던 값이 보존되니까 그게 필요해서.