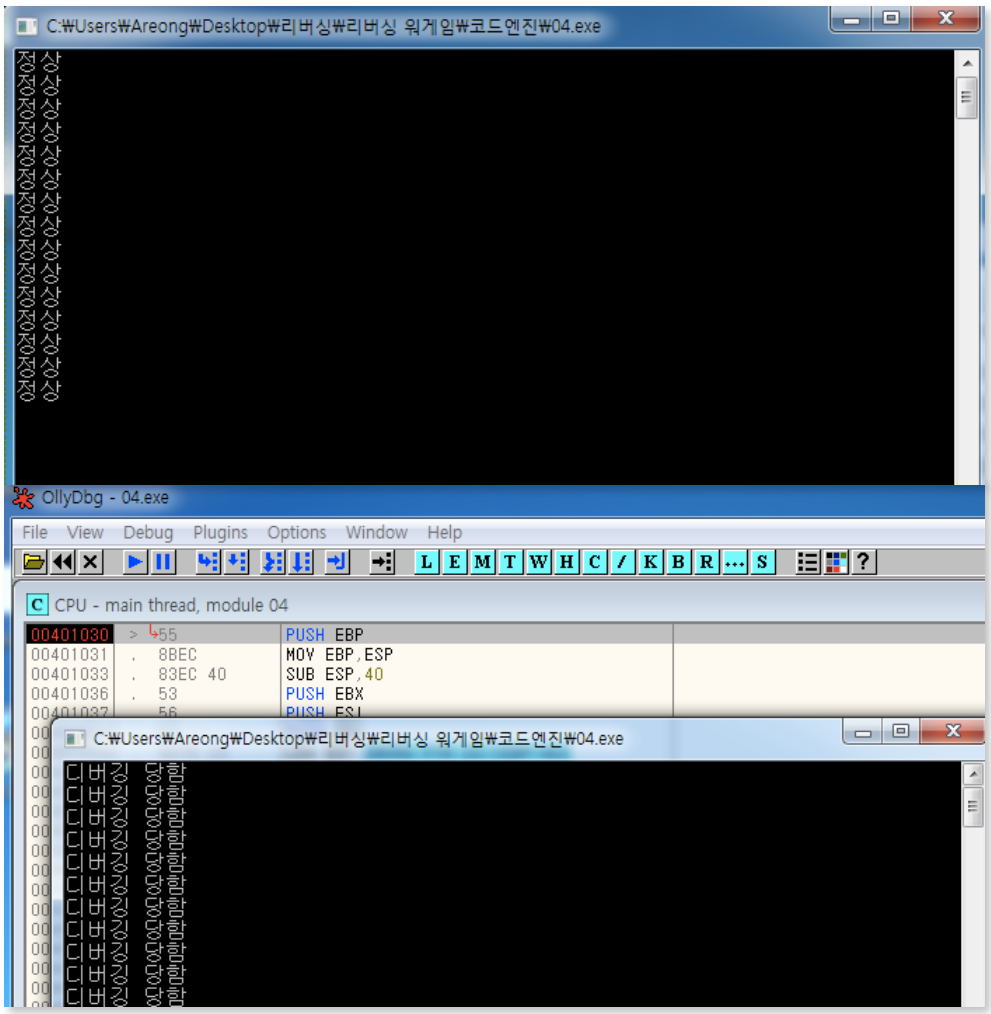
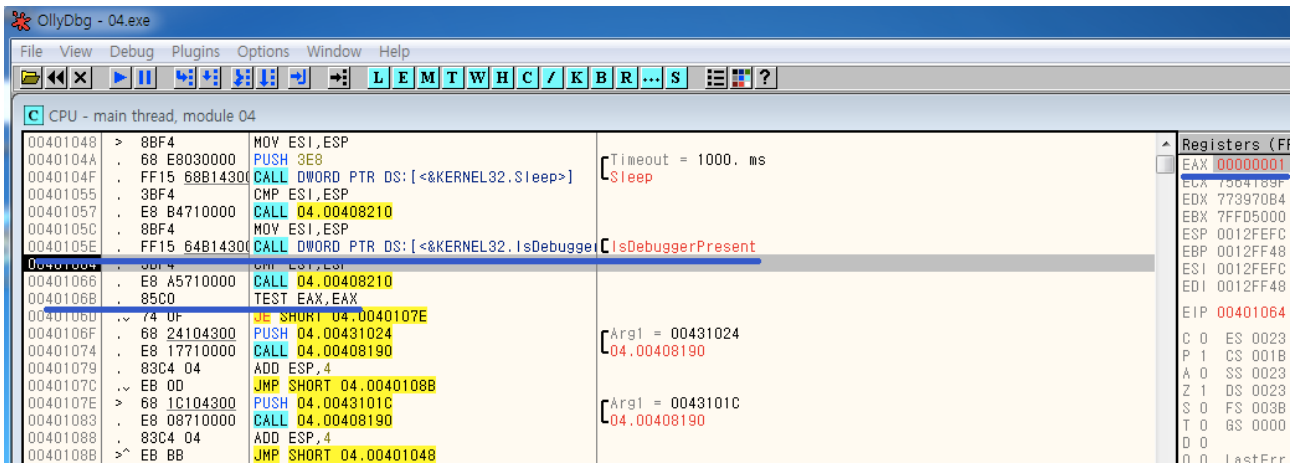


위는 그냥 실행한 화면이고 아래는 올리디버거를 통해 실행한 화면입니다.

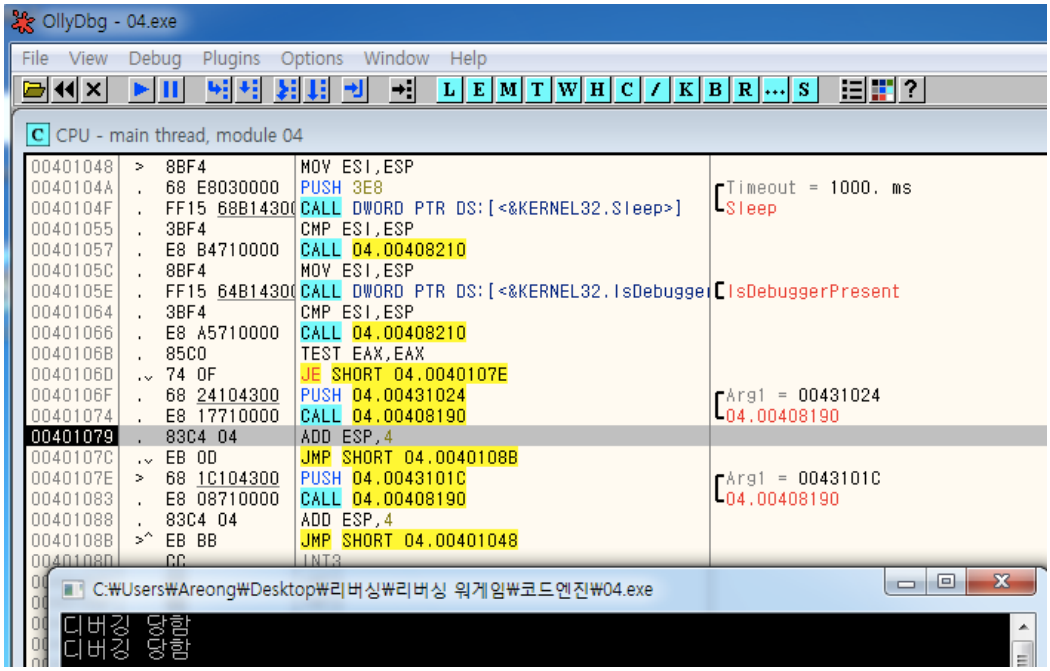
그냥 실행했을 땐 디버거를 감지하지 못했기 때문에 '정상' 이라는 단어가 계속 출력됩니다.

올리디버거를 통해 실행하면 프로그램이 이를 감지하고 '디버깅 당함' 이라는 문자열을 출력합니다.

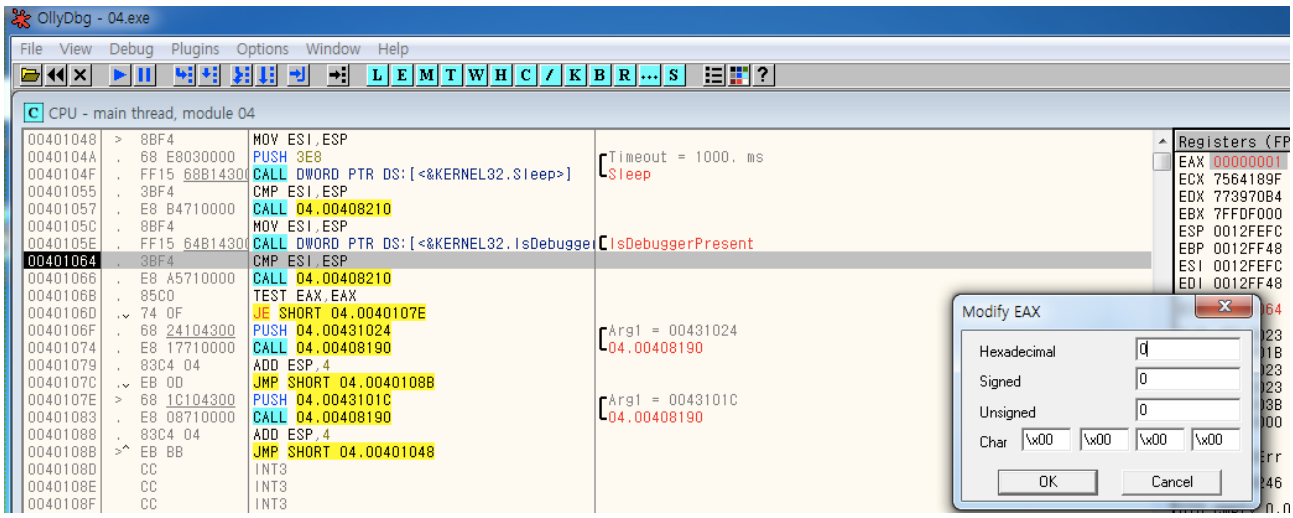


프로그램 내부를 살펴보면 위의 그림과 같이 함수를 통해 디버깅을 감지하고 TEST 구문으로 분기 합니다.

함수의 리턴값인 EAX의 값이 1임을 알 수 있는데 이는 디버깅을 감지했다는 리턴값입니다.

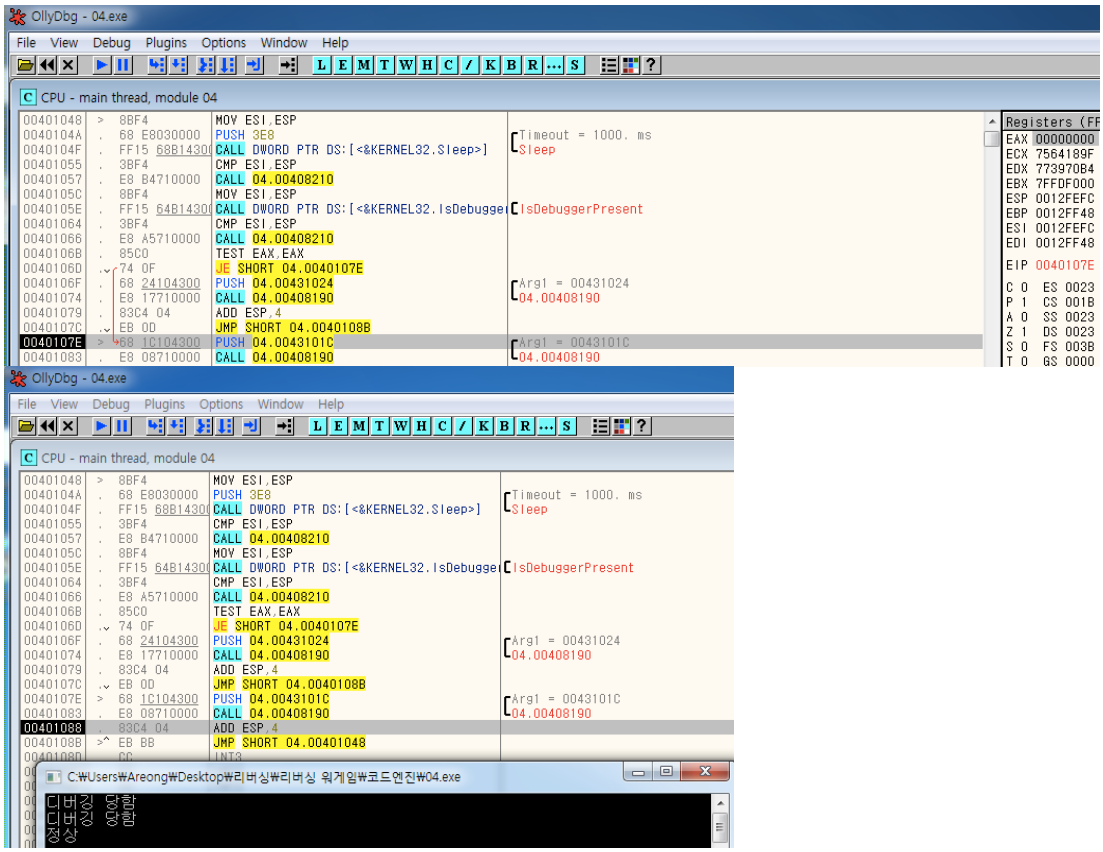


계속 실행하면 디버깅 당함이라는 결과가 출력되게 됩니다.



이를 우회하기 위한 방법으로 리턴값을 수정하는 방법이 있습니다.

함수의 결과가 1이지만 임의적으로 0(디버깅 체크를 못한 리턴값)으로 바꾸게 되면



위 그림과 같이 분기를 다른 곳으로 하고 결과는 정상이라고 출력되게 됩니다.

분기의 TEST EAX, EAX는 EAX값이 0인가 아닌가를 판별하는 어셈블리어 입니다.

AND연산을 하는데 같은 값을 AND 하기 때문에 TEST의 결과로는 EAX가 되게 됩니다.

이때 EAX가 0이면 ZF(Zero Flag)값이 1이 됩니다.

이건 CMP로 비교했을 시의 출력 0, 즉 두 값이 같다는 의미로 JE(Jump Equal)에서 점프를 하게 됩니다.