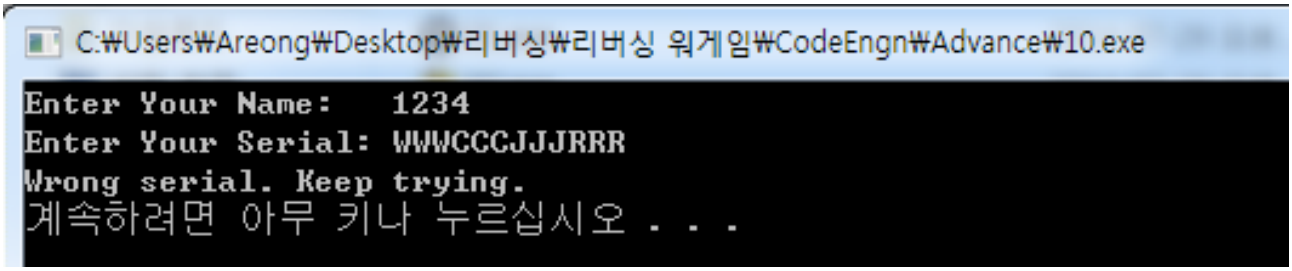
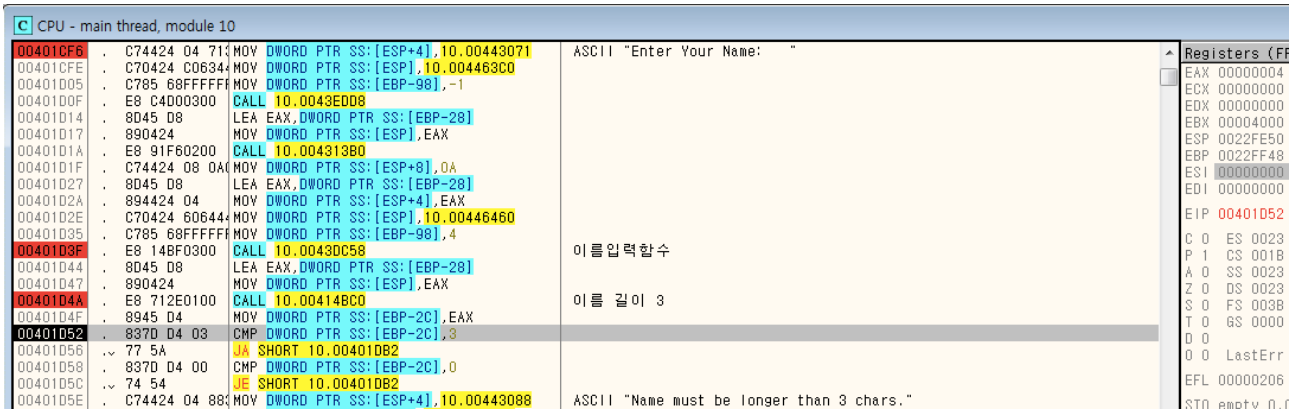


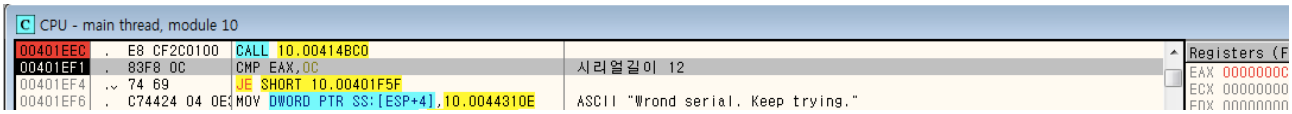
이름과 시리얼을 입력받고 인증하는 프로그램입니다.

패킹되어있지 않고, Dev C++로 개발되었음을 확인했고 분석을 진행합니다.



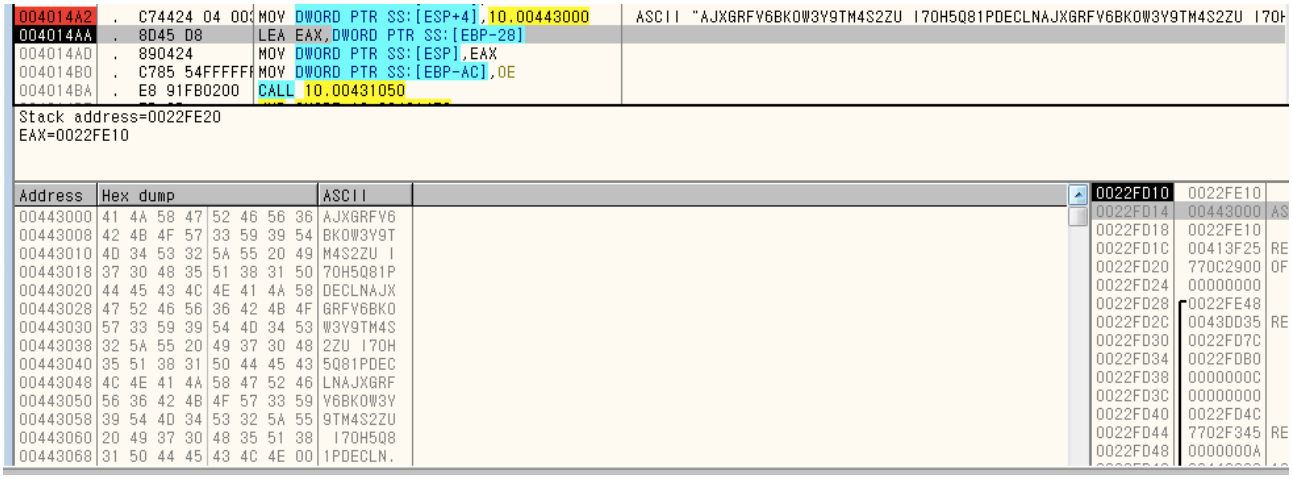
이름을 입력받는데 그 길이는 3글자 이상이어야 합니다.

3글자보다 적을 경우 에러메세지가 출력됩니다.



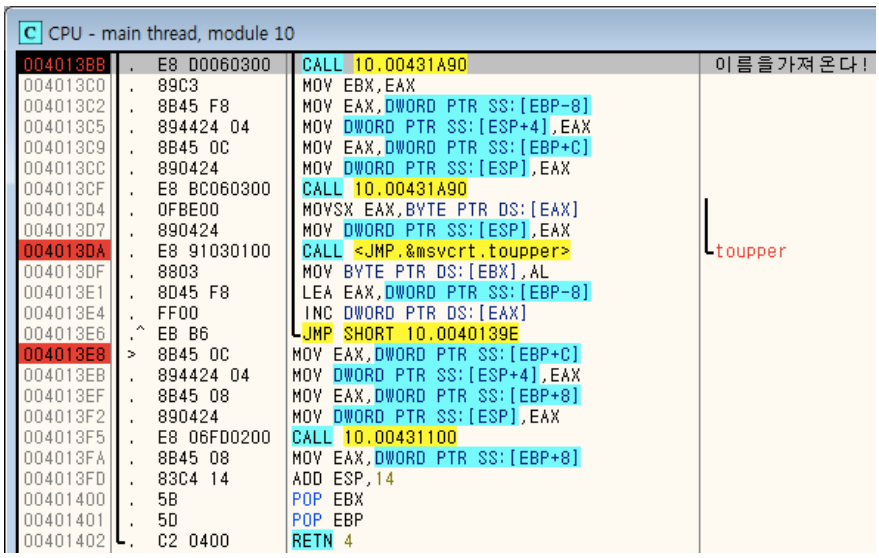
패스워드도 길이의 제한이 있습니다. 길이가 12(0x0C)여야 합니다.

만일 길이가 12가 아니면 에러메세지가 출력됩니다.



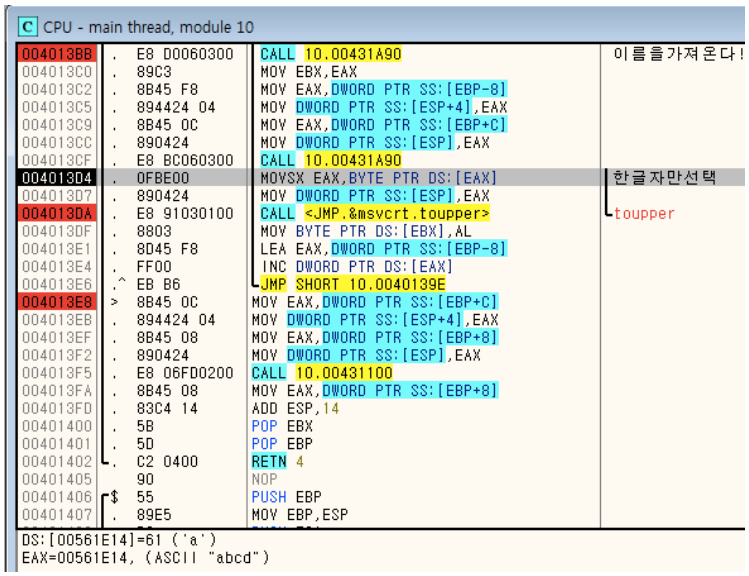
진행하다 보면 엄청 긴 문자열을 저장합니다.

이 문자열은 특정 문자열이 반복되는 형태입니다.

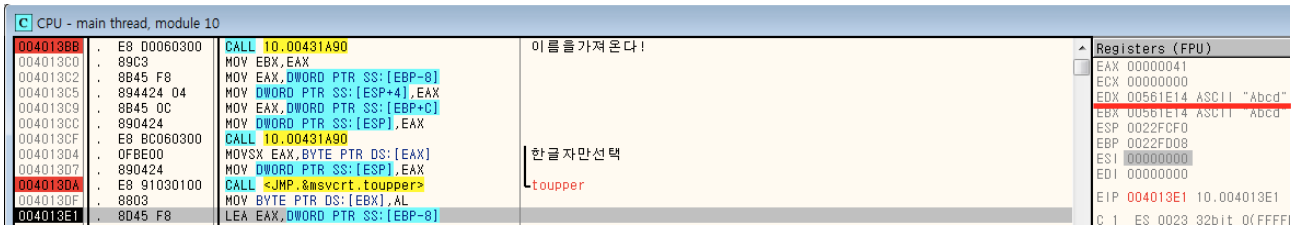


진행하다보면 다음과 같은 루틴이 나옵니다.

BreakPoint가 설정되어 있는 중간 부분을 보시면 'toupper'라는 함수 콜이 있습니다.



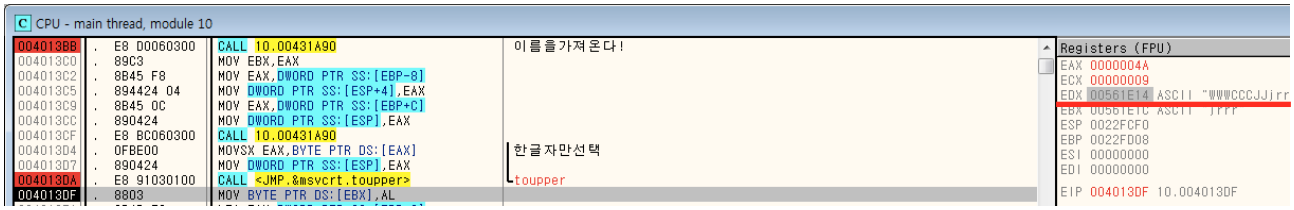
이름을 받아오고 한 글자만 EAX에 저장합니다.



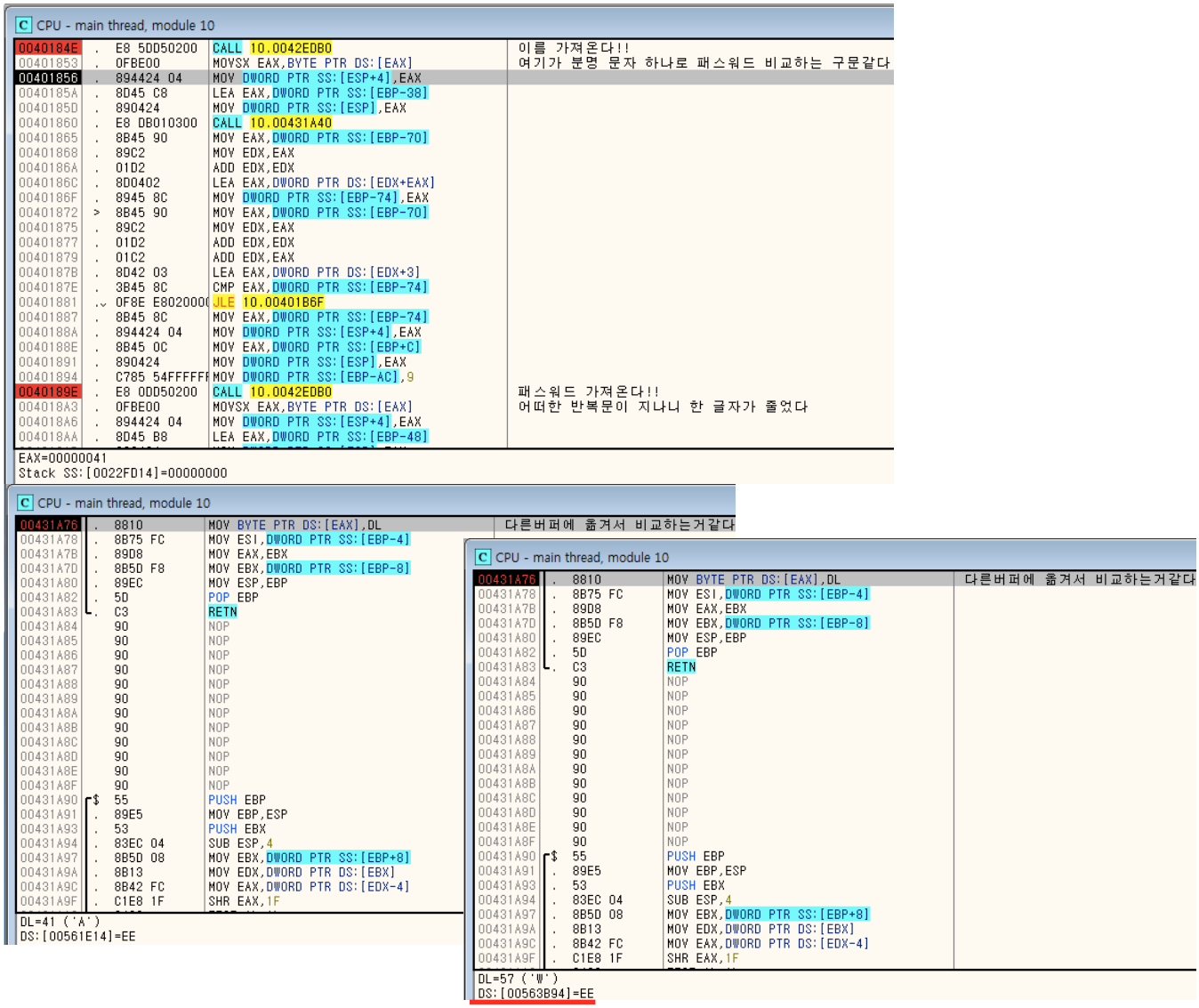
toupper는 소문자를 대문자로 바꿔주는 함수로 'abcd'가 'Abcd'로 바뀐 것을 확인할 수 있습니다.

이름에 존재하는 소문자를 전부 대문자로 바꾸고 다음 연산을 진행하는 것으로 보아

대, 소문자의 구분은 없음을 알 수 있습니다.

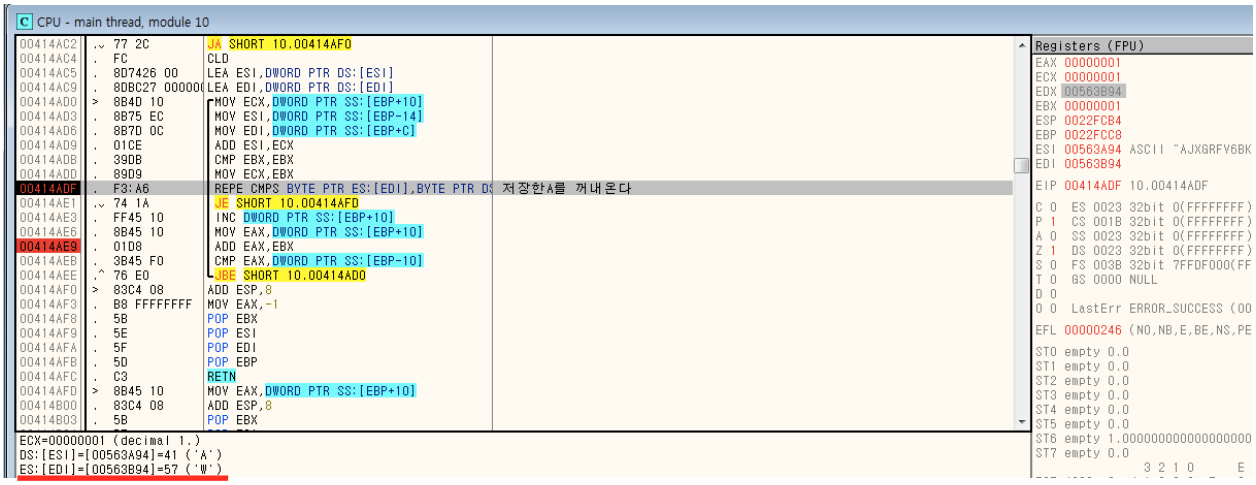


패스워드로 입력한 값도 대문자로 바뀌는 것을 확인할 수 있습니다.



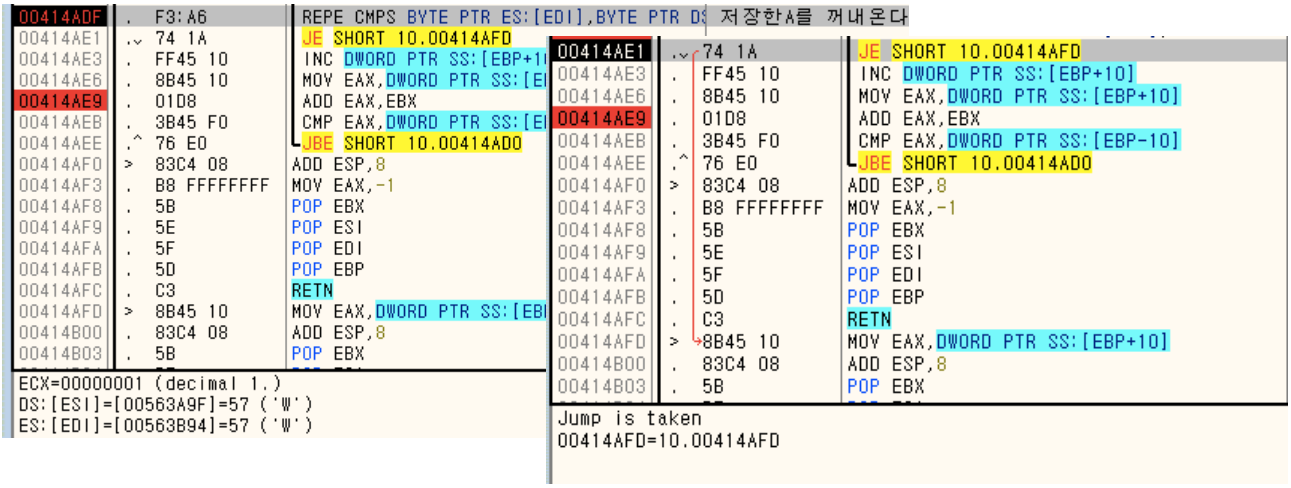
위 그림을 보면 이름으로 입력한 값의 한 글자를 가져와서 다른 버퍼에 옮깁니다.

패스워드 역시 같은 동작을 합니다.



그렇게 따로 버퍼에 저장한 값들을 위에서 저장한 특정 문자열의 반복과 비교하게 됩니다.

이전 그림의 패스워드가 저장된 버퍼와 밑줄 친 버퍼의 주소가 같은 것을 확인할 수 있습니다.



문자열에서 입력한 값과 같은 값을 찾을 때까지 반복하다가 찾으면 루틴을 빠져나옵니다.

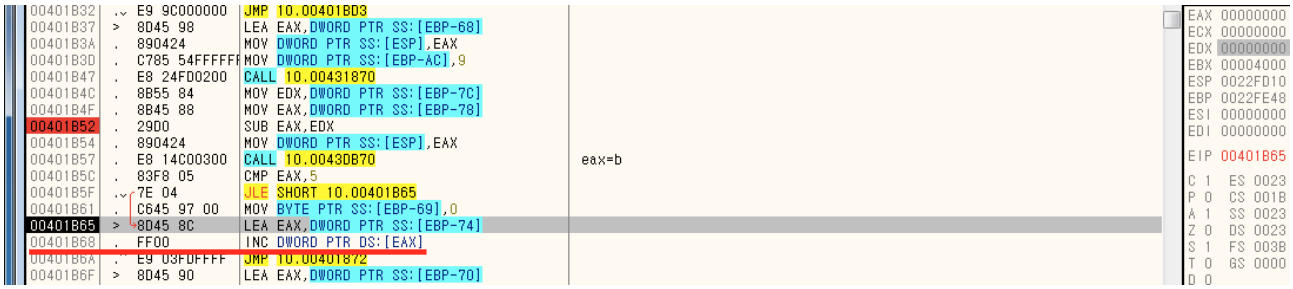
루틴 도중에는 카운트를 증가시켜 입력한 값과 같은 값이 문자열 내에서 어디에 있는지 체크합니다.

그리고 루틴을 빠져나올 때 카운트 값을 버퍼에 저장합니다.

```
00401B4C  .  8B55 84      MOV EDX,DWORD PTR SS:[EBP-7C]
00401B4F  .  8B45 88      MOV EAX,DWORD PTR SS:[EBP-78]
00401B52  .  29D0         SUB EAX,EDX
00401B54  .  890424       MOV DWORD PTR SS:[ESP],EAX
00401B57  .  E8 14C00300  CALL 10.0043DB70
00401B5C  .  83F8 05      CMP EAX,5
00401B5F  .v 7E 04        JLE SHORT 10.00401B65
00401B61  .  C645 97 00   MOV BYTE PTR SS:[EBP-69],0
00401B65  >  8D45 8C      LEA EAX,DWORD PTR SS:[EBP-74]
00401B68  .  FF00         INC DWORD PTR DS:[EAX]
00401B6A  .^ E9 03FDFFFF  JMP 10.00401872
00401B6F  >  8D45 90      LEA EAX,DWORD PTR SS:[EBP-70]
00401B72  .  FF00         INC DWORD PTR DS:[EAX]
00401B74  .^ E9 3BFCFFFF  JMP 10.004017B4
00401B79  >  0FB645 97    MOVZX EAX,BYTE PTR SS:[EBP-69]
00401B7D  .  8985 F4FEFFF MOV DWORD PTR SS:[EBP-10C],EAX
00401B83  .  8D45 B8      LEA EAX,DWORD PTR SS:[EBP-48]
00401B86  .  890424       MOV DWORD PTR SS:[ESP],EAX
00401B89  .  C785 54FFFFF MOV DWORD PTR SS:[EBP-AC],0B
00401B93  .  E8 D8FC0200  CALL 10.00431870
00401B98  .  8D45 C8      LEA EAX,DWORD PTR SS:[EBP-38]
EDX=00000026
EAX=00000026
```

EDX와 EAX는 이름과 패스워드로 입력한 값의 한 글자가 특정 문자열에서 어디에 있는지 index값을 가집니다.

EAX와 EDX를 빼고 그 값을 5와 비교합니다.

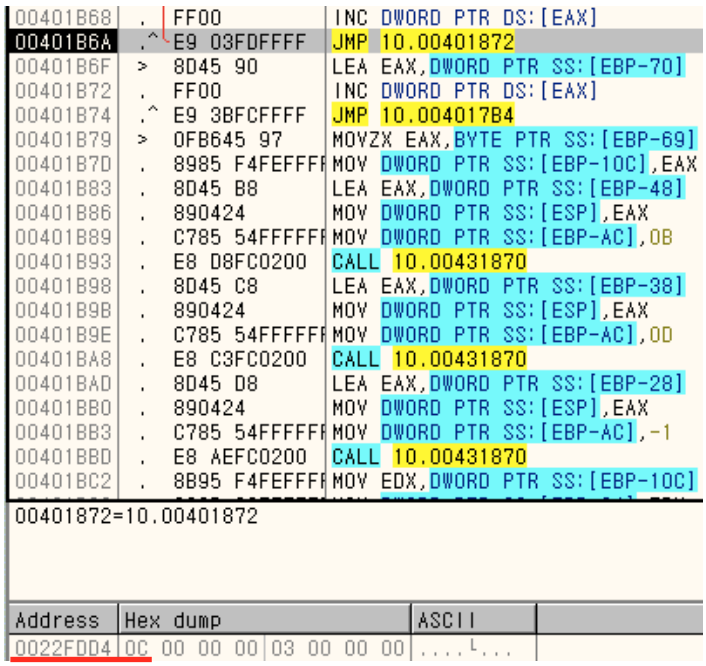


현재 이름으로 WCJR을 입력했고 시리얼은 WWWCCCJJJRRR로 입력했습니다.

문자열에서 W의 위치는 변하지 않기 때문에 EAX는 0의 값을 가지게 되고 분기문에서 점프하게 됩니다.

```
00401B68  .  FF00         INC DWORD PTR DS:[EAX]
00401B6A  .^ E9 03FDFFFF  JMP 10.00401872
00401B6F  >  8D45 90      LEA EAX,DWORD PTR SS:[EBP-70]
00401B72  .  FF00         INC DWORD PTR DS:[EAX]
00401B74  .^ E9 3BFCFFFF  JMP 10.004017B4
00401B79  >  0FB645 97    MOVZX EAX,BYTE PTR SS:[EBP-69]
00401B7D  .  8985 F4FEFFF MOV DWORD PTR SS:[EBP-10C],EAX
00401B83  .  8D45 B8      LEA EAX,DWORD PTR SS:[EBP-48]
00401B86  .  890424       MOV DWORD PTR SS:[ESP],EAX
00401B89  .  C785 54FFFFF MOV DWORD PTR SS:[EBP-AC],0B
00401B93  .  E8 D8FC0200  CALL 10.00431870
00401B98  .  8D45 C8      LEA EAX,DWORD PTR SS:[EBP-38]
Stack DS:[0022FDD4]=00000006
```

이름과 패스워드의 문자열의 위치가 같거나 +-5의 위치를 가지는 값의 경우 밑줄 친 부분의 값이 증가합니다.



이름과 시리얼이 문자열 내에서 모두 같은 위치에 존재하기 때문에

버퍼의 값은 시리얼의 길이인 0x0C(12)가 되고 점프 구문으로 점프하게 됩니다.

알고리즘을 정리하도록 하겠습니다.

1. 이름과 시리얼을 입력받는다.
2. 특정 문자열 내에서 이름의 각 글자들이 어디에 위치하는지 찾아낸다.
3. 시리얼 역시 2와 같은 동작을 한다.
4. 이름 한글자당 시리얼 3글자의 위치를 비교한다.
5. 그 위치가 문자열에서 +-5 이내의 위치면 카운트를 1 증가시킨다.
6. 카운트가 시리얼의 길이와 같으면 인증을 성공시킨다.

와 같은 루틴이 존재하는 프로그램입니다.

이를 프로그램으로 만들어서 실행한 뒤 인증하면 성공입니다!