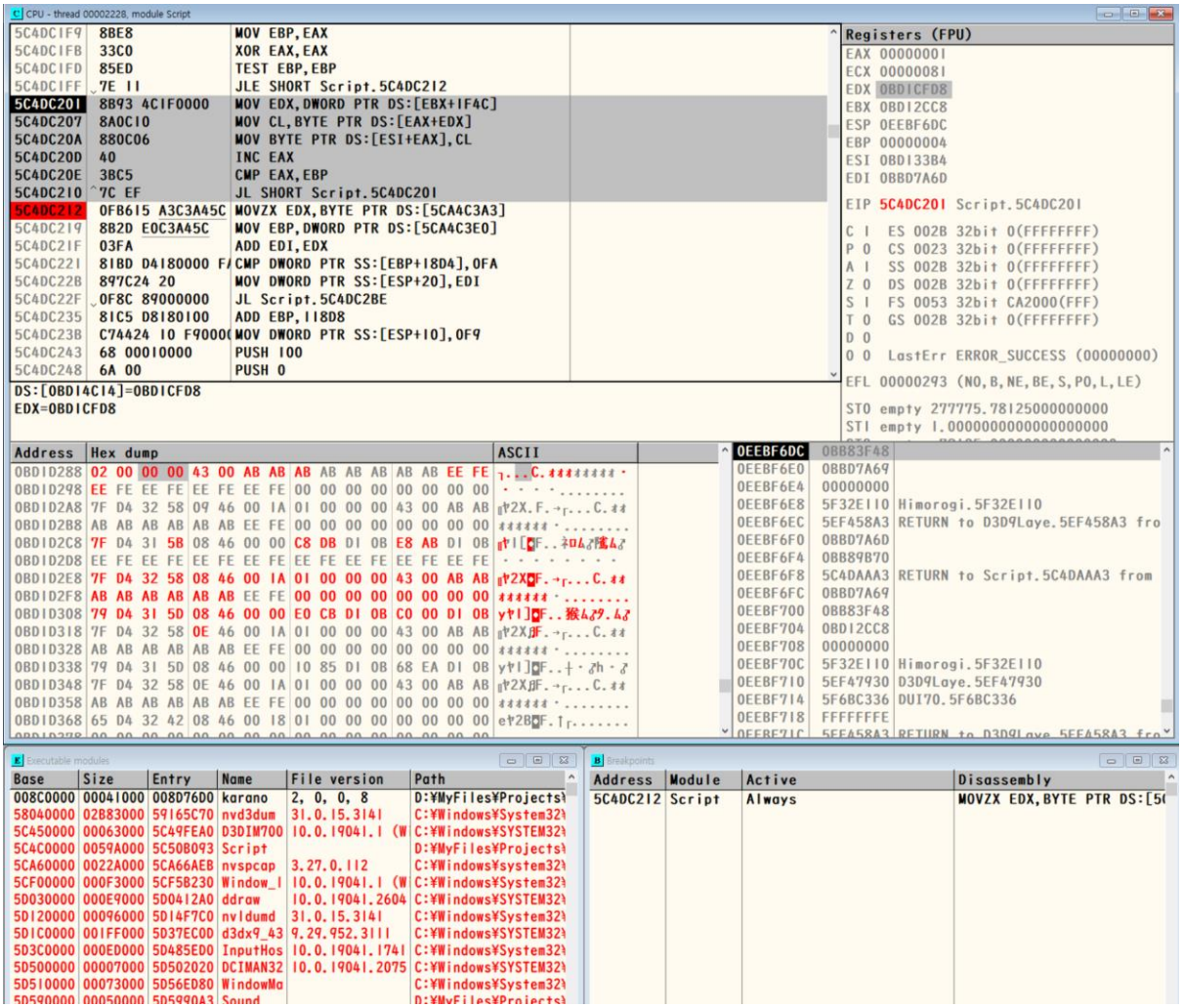


여기가 이름 받아오는파트는 맞음

5C4DC207 MOVE CL BYTE PTR DS:[EAX+EDX]

여기서 CL에 EDX로부터 EAX만큼의 오프셋을 이동시킨 WORD하나를 추출해온다.

5C4DC20A MOV BYTE PTR DS:[ESI+EAX],CL

이부분이 (CL에 추출된 글자)를 메모리에 하나씩 넣는 부분.

그렇다면 원본은 어디서부터 시작됐느냐?

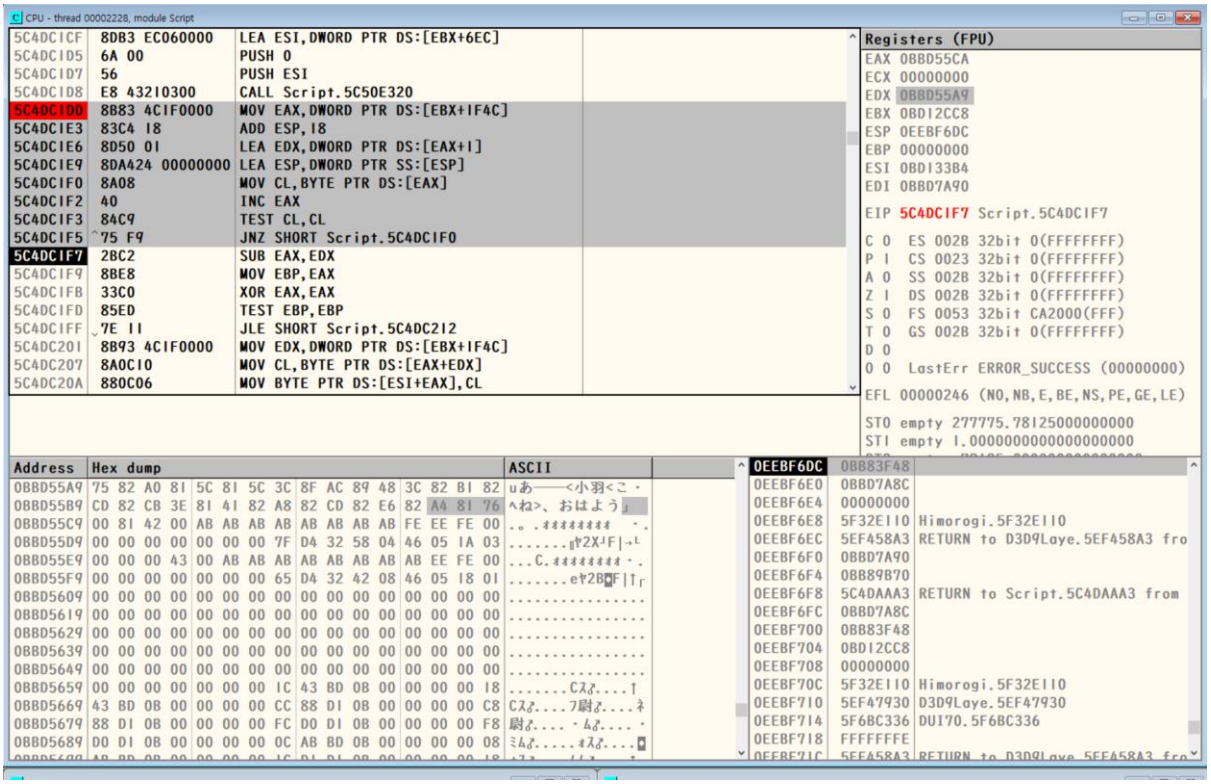
반복구문이 시작되는 곳에서 EDX에 받아왔음.

5C4DC201 MOV EDX,DWORD PTR DS:[EBX+1F4C]

EDX에 [EBX+1F4C]라는 주소에 있는 값을 넣는데 이걸 자세히 생각하기에는 뇌에 과부하가 왔으니 생략

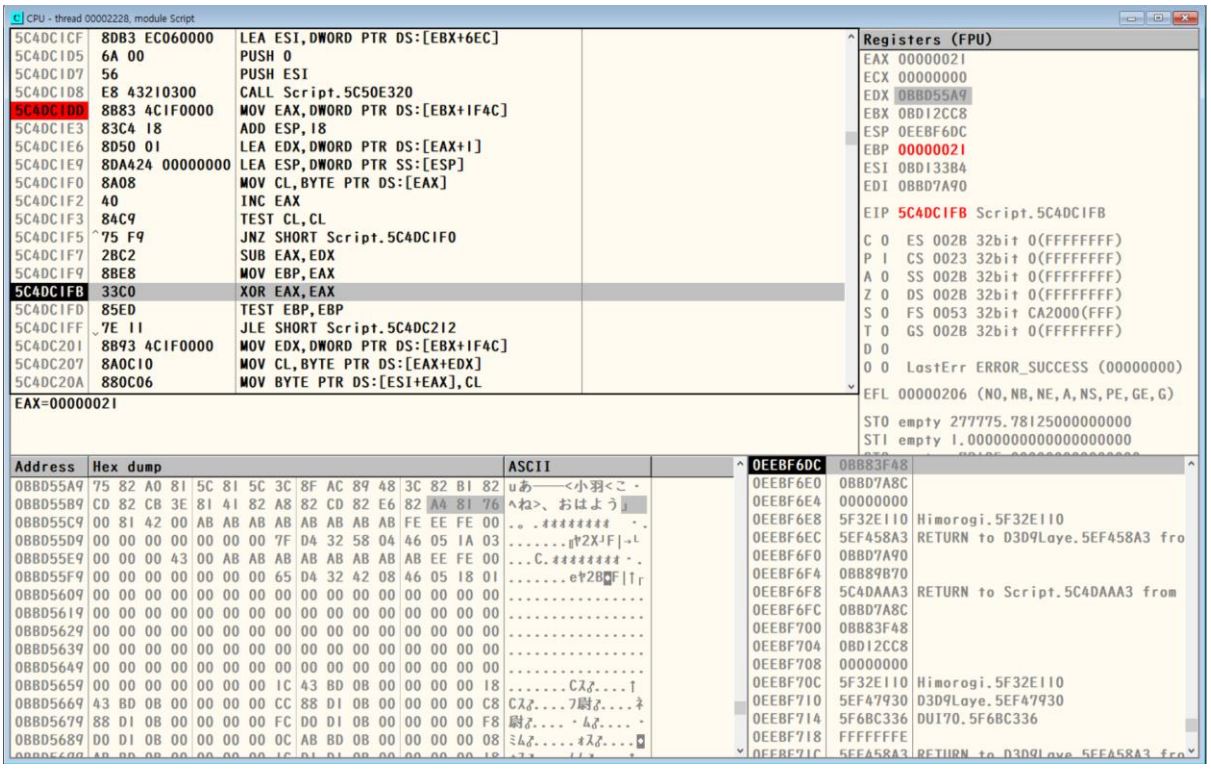
어차피 EDX에 다 받아왔고 EDX가 가리키는 부분의 값을 바꾸면 게임화면 출력값도 바뀜

결론1 : 이름추출은 5C4DC201, EDX 인줄 알았는데 아닌가봄 ㅠㅠㅠㅠㅠ

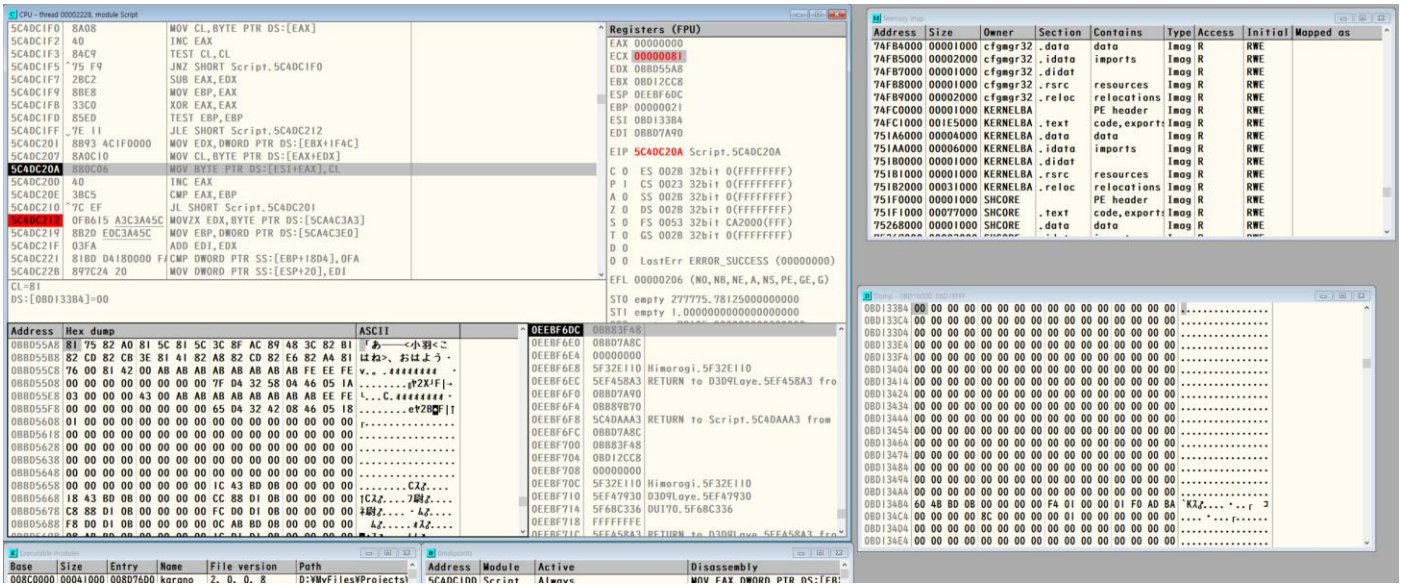


아 여기에서 속아서 한참 헤맨듯

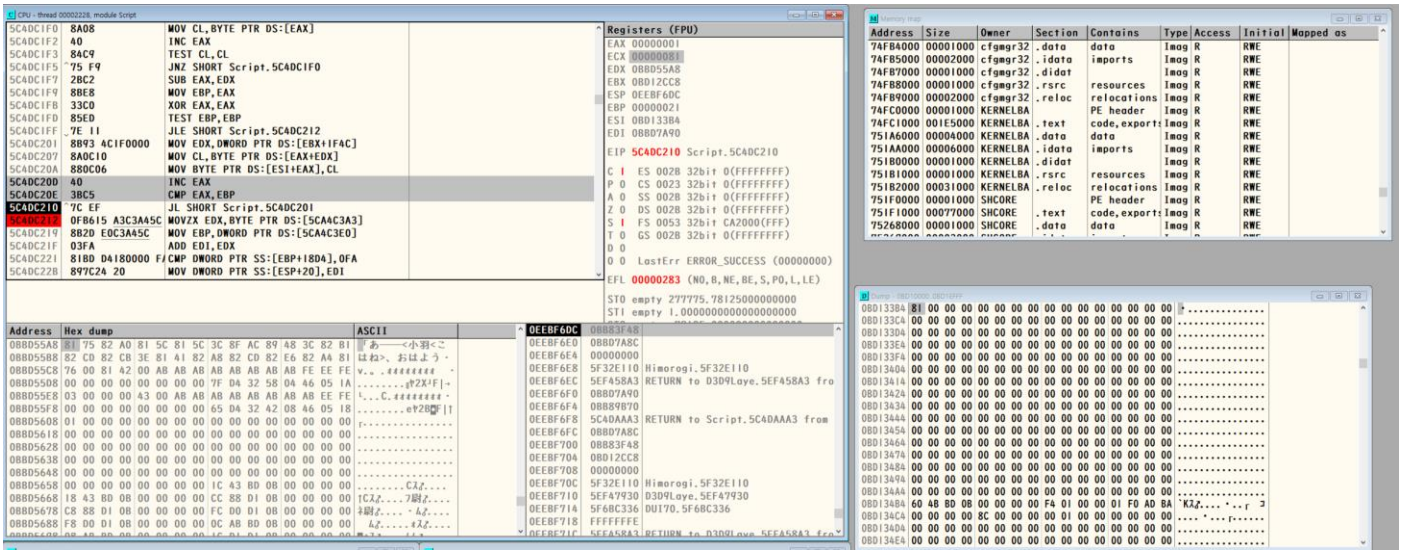
회색으로 강조된곳이 TEST CL,CL을 반복하며 CL(한글자씩 받아오는거)가 0이되면 문장이 끝났음으로 보고 루프를 종료하는데, 이후 SUB EAX,EDX로 문장길이를 알아내는듯



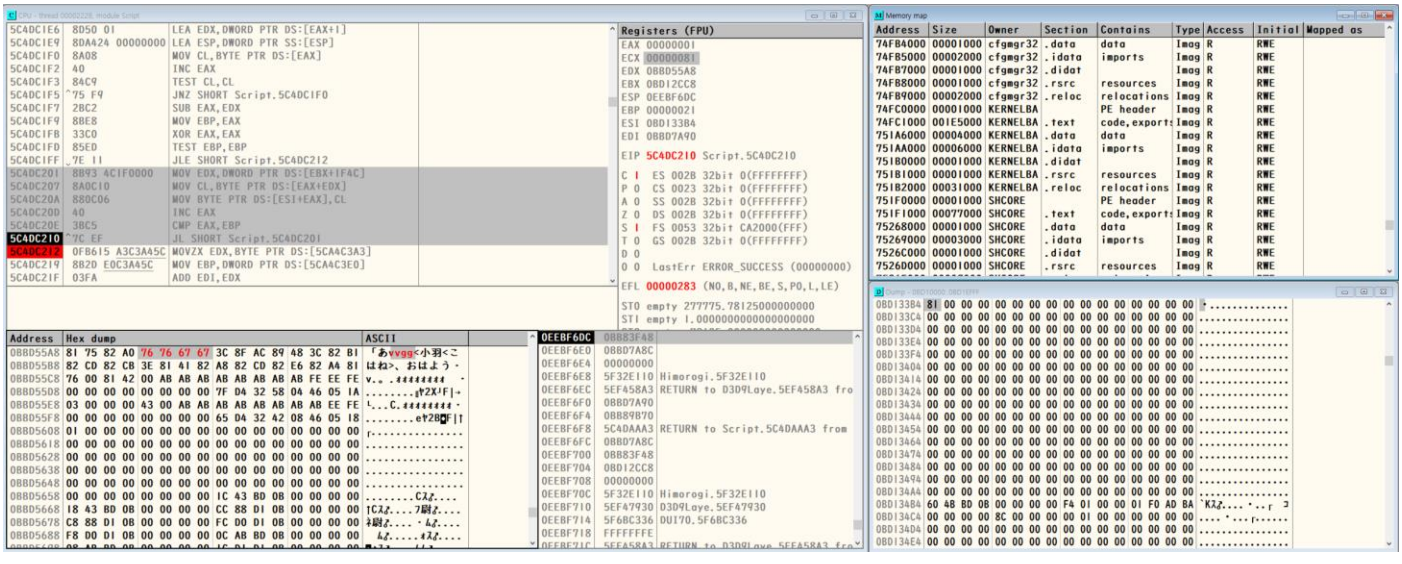
이렇게 EAX에 21이 있다가 EBP에 저장후 XOR로 EAX를 0으로 초기화함. 그다음 TEST EBP,EBP를 하는 이유는 바로다음 JLE이 있음을 보니까 EBP가 0인경우를 걸러내려고 하는 것임을 추측 가능.



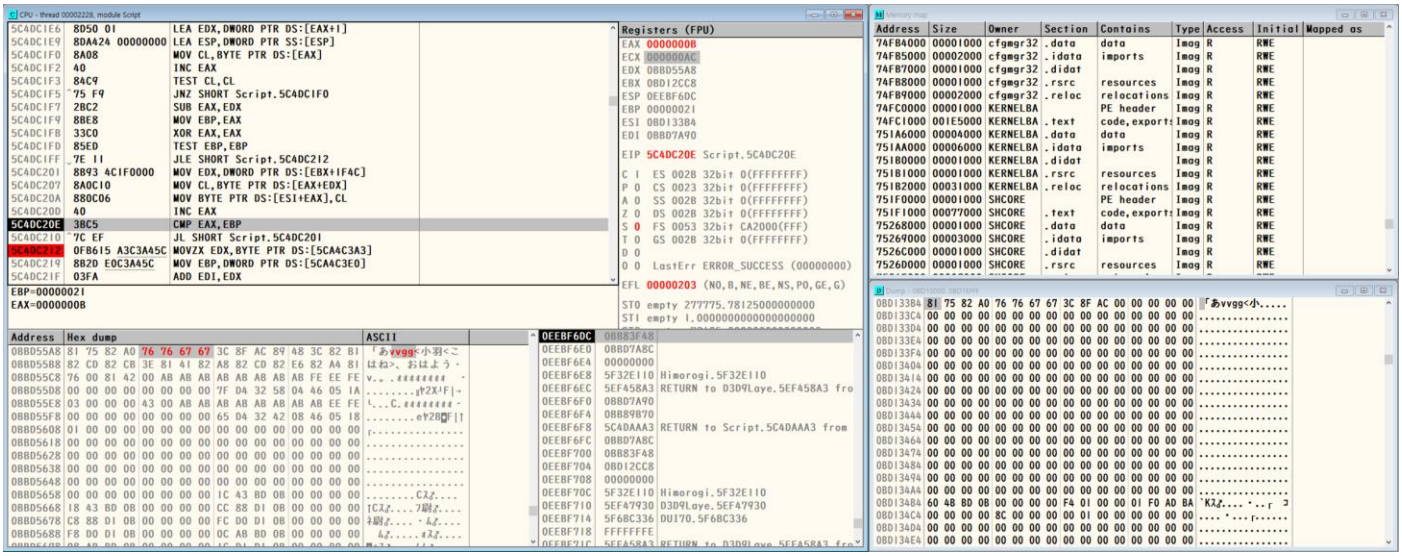
본격적으로 CL로 한글자씩 뽑아내는 것을 확인가능. 사진상에서 81이 ECX의 마지막(CL)부분에 들어갔고, 이는 문장 시작지점의 WORD값인 81과 동일한 것이 보인다.



Dump창에서 메모리에 81이 들어갔다. INC EAX로 1씩 증가시키면서 EBP와 비교하는데, 이전에 문자길이 쟀을 때 21을 저장한 곳이 바로 이 때를 위함이다.



한번 원본값을 바꿔보자.



아주 잘 들어간다.

이로써 EDX로부터 받아오는 것이 맞음을 확신할수 있다.

5C4DC201 MOV EDX,DWORD PTR DS:[EBX+1F4C]

이곳을 낚아채버리면 될듯하다.

결론2 : 대사추출은 5C4DC201, EDX 인줄 알았는데 아닌가봄 ㅠㅠㅠㅠㅠ2

정리

1. 이름추출은 5C4DC201, EDX
2. 대사추출은 5C4DC201, EDX
3. 어라? 같은코드 쓰는구먼...

HOOK(대충 스크립트.dll에서연결!0x5C4DC201,TRANS(???EDX???,TWOBYTE,OVERWRITE(IGNORE)),RETNPOS(COPY))

이런식으로 나오긴 할 듯..

추가) 프로세스 새로 열어도 똑같이 잘 작동함