5주차 2차시. 리버스엔지니어링 툴

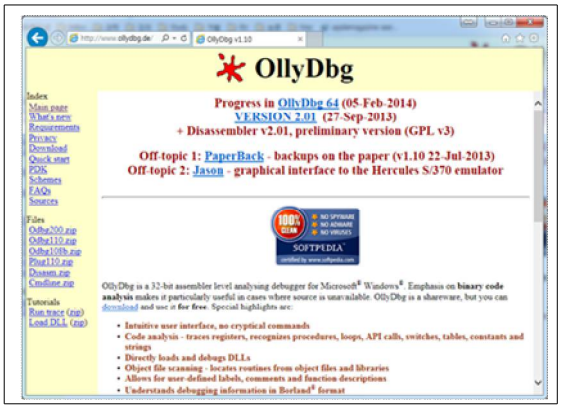
【학습목표】

1. 리버스 엔지니어링 툴에 대해 알고, 리버스 엔지니어링을 위한 함수찾기에 대해 설명할 수 있다.

학습내용1 : 리버스 엔지니어링 툴 다루기

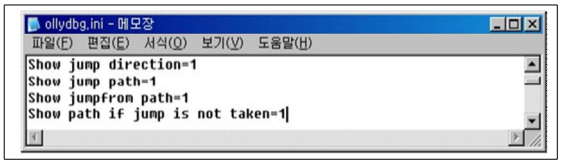
1. 올리 디버거

* http://www.ollydbg.de



2. 올리 디버거 환경설정(ollydbg.ini)

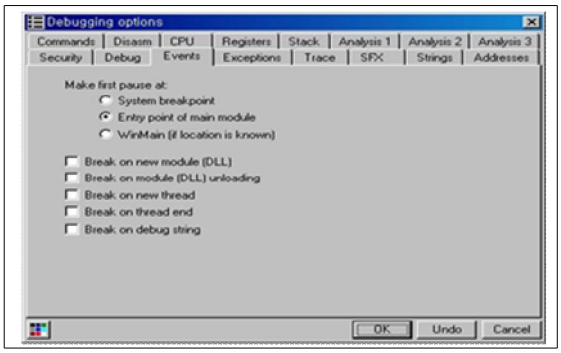
이동하는 곳을 표시하기 위해서 모두 1로 설정



[그림 5-18] ollydbg.ini 설정

3. [Options]-[Debugging options] 메뉴 선택

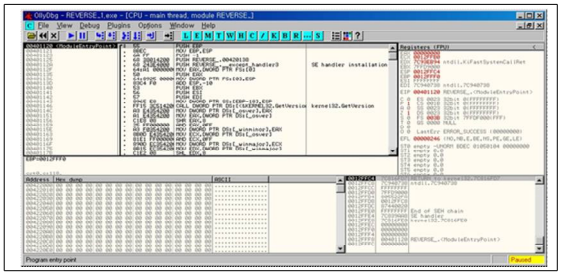
* [Event] 탭에서'Entry point of main module'로 선택
처음 올리 디버거로 파일을 열었을 때화면에 표시할 지점을지정



[그림 5-19] 파일 분석 시올리 디버거 시작 지점 설정

4. 올리 디버거의 기본 구조와 사용법

* REVERSE_1.EXE 파일 열기



[그림 5-20] 올리 디버거 인터페이스

* 올리 디버거 인터페이스 설명
① 메모리 주소
프로그램을 메모리에 로드 시메모리상에 로드된 프로그램의 명령어
코드 및 데이터의 가상 주소와 라벨 표시

② 기계어 코드와 어셈블리어 코드
프로그램의 기계어 코드와 어셈블리어 코드 출력

③ 주석 창
올리 디버거가 분석한 API 함수, 문자열, 함수 인자 확인
사용자가 주석 입력

④ 레지스터와 플래그 창
주요 레지스터와 플래그의 상태 값 확인

⑤ 변수 값 출력 창
디버깅 중에 주요 변수 값 확인

⑥ HEX 덤프 창
프로그램을 HEX 값으로 덤프하여 보여줌
바이너리 파일을 직접 수정
* 올리 디버거 인터페이스 설명

⑦ 스택
스택의 현재 상태 등 확인

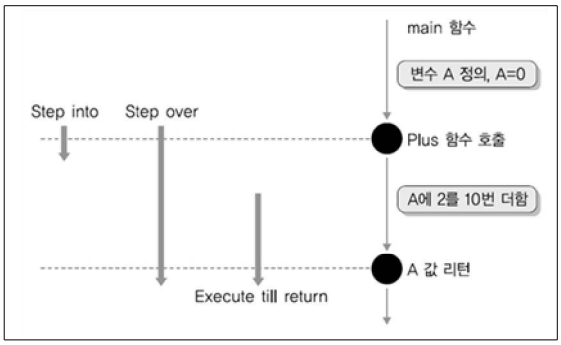
5. 올리 디버거 키와 기능

* 올리 디버거 키와 기능 설명

키	기능	설명
F2	Toggle	브레이크 포인트를 지정
F9	Run	브레이크 포인트까지 실행
Ctrl + F2	Restart	코드 처음 부분으로 회귀
F7	Step into	Call함수나 반복적으로 수행하는 Rep 명령을 만났을 때 함수 내부로 추적 또는 Rep 조건을 만족할 때까지 계속 수행

6. Plus를 함수를 통한 올리 디버거 기능 비교

* Step into, Step over, Execute till return

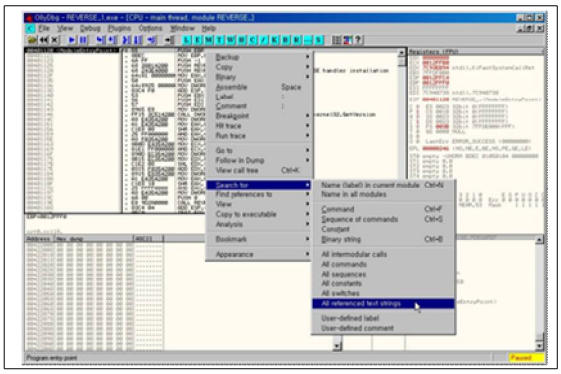


[그림 5-21] Step into, Step over, Execute till return

학습내용2 : 리버스 엔지니어링을 위한 함수 찾기

1. 문자열 찾기

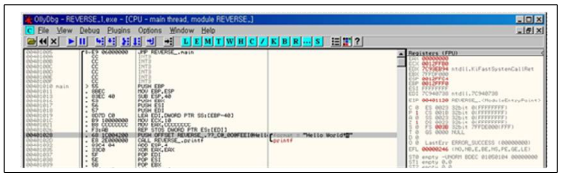
함수 찾기의 가장 일반적 방법



[그림 5-22] 프로그램에 존재하는 모든 문자열을 찾는 메뉴 실행

2. 문자열이 위치한 주소(Address)와어셈블리 명령어, 찾은 문자열의 내용 확인

* [그림 5-23] 'Hello World!'문자열을 통한main 함수 추적 과정

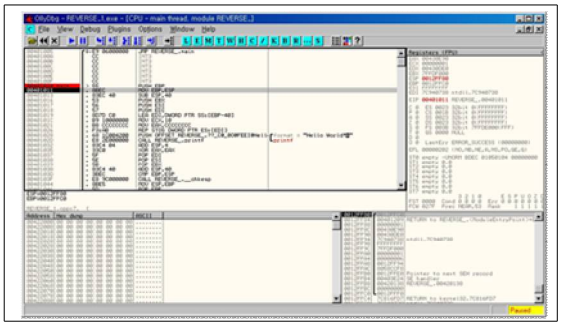


학습내용3 : 함수의 로직 확인하기

* 해당 코드의 의미와 동작을 이해하는 과정
* 00401010 주소 라인에서F2(Toggle)를 눌러 브레이크 포인트 설정
* [그림 5-24] 브레이크 포인트 설정

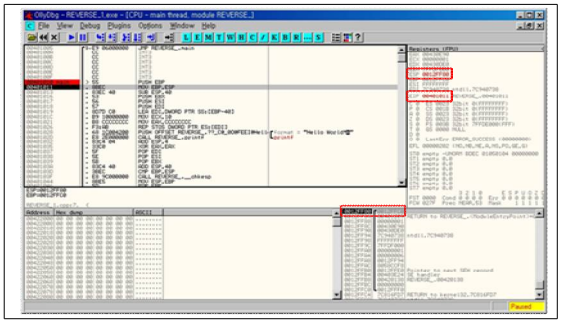
1. F9(Run) 클릭

F9(Run)를 누르면 브레이크 포인트를설정한 라인 바로 위 라인까지 실행
[그림 5-25] 'PUSH EBP'를 실행한 화면



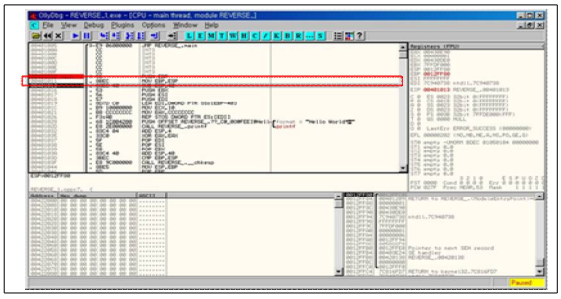
2. cF7(Step into) 클릭

* [그림 5-25] 'PUSH EBP'를 실행한 화면



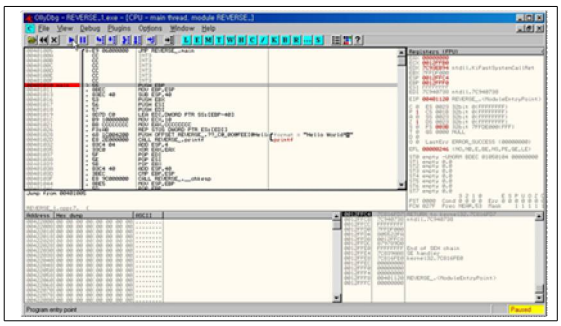
3. F7 클릭

* [그림 5-26] 'MOV EBP, ESP'를 실행한 화면 : [그림 5-25] + F7 1번



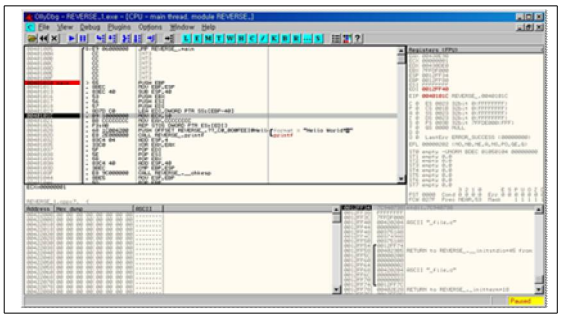
4. F7 4번 클릭

* 커서가 LEA EDI,DWORD PTR SS:[EBP-40] 라인에 위치
* SS(Stack Segment)는 0023, EBP-40은 0012FF40(=0012FF80-40)
* [그림 5-27]‘ PUSH EBX’,‘ PUSH ESI’,‘ PUSH EDI’를 실행한 화면 : [그림 5-26]+ F7 4번



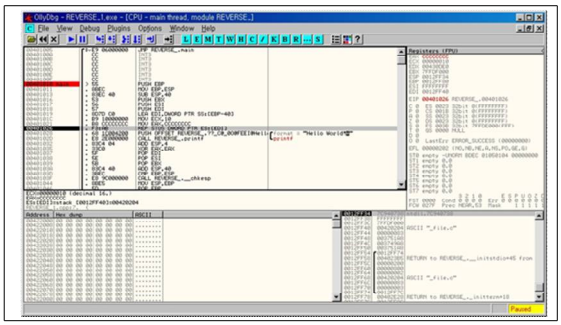
5. F7 클릭

* LEA EDI, DWORD PTR SS:[EBP-40] 실행
* [그림 5-28] ‘LEA EDI, DWORD PTR SS:[EBP-40]’을 실행한 화면 : [그림 5-27]+ F7 1번



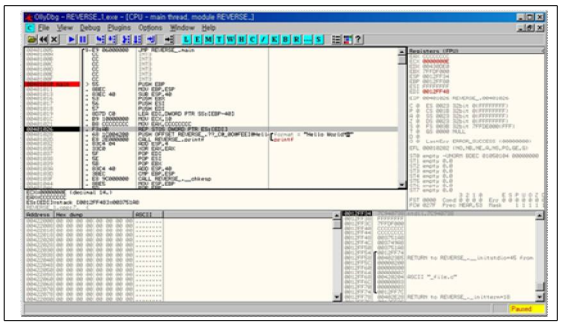
6. F7 2번 클릭

* MOV ECX,10’라인과‘MOV EAX,CCCCCCCC’라인 실행
* ECX 값 00000010, EAX 값 CCCCCCCC
* 00401026 주소 라인에서‘REP STOS DWORD PTR ES:[EDI]’실행되도록 설정
* 실행화면



7. F7 2번 더 클릭

* 레지스터 창에서 ECX 값 0000000E(2감소)



8. F7 클릭 시

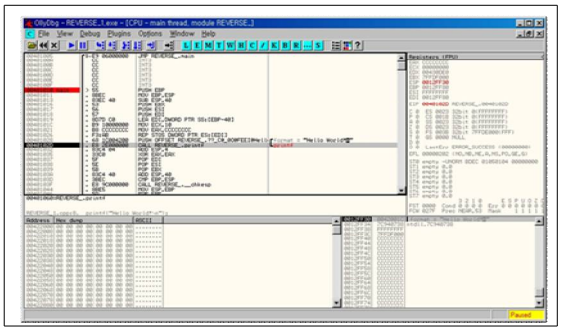
* ECX 1씩 감소
* EDI 4씩 증가
* EDI가 가리키는 스택 CCCCCCCC로 채워짐

9. F7 을 계속 누름

* ‘REP STOS DWORD PTR ES:[EDI]’ 모두 실행
* 커서가 ‘CALL REVERSE_.printf’라인에 위치토록 함
* 커서가 위치한 라인이‘Hello World!’ 실행 위한 명령 라인

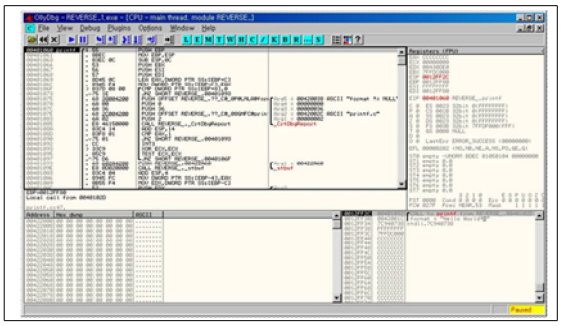
10. ‘Hello World!’ 실행 위한 명령 라인

* ‘Hello World!’ 실행 위한 명령 라인 화면



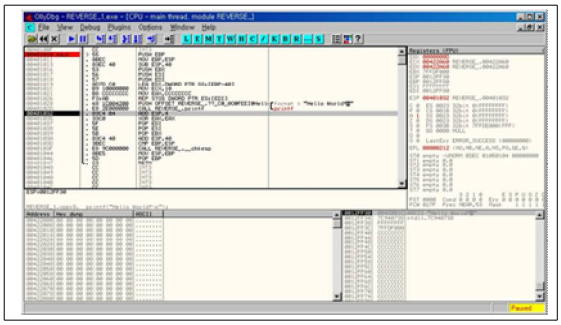
11. F7 누름

* printf가 구현된 어셈블리어 코드로 화면이 넘어감



11. F8 누름

* 원래 화면에서'CALL REVERSE_.printf'가실행된 바로 다음 라인으로 넘어감



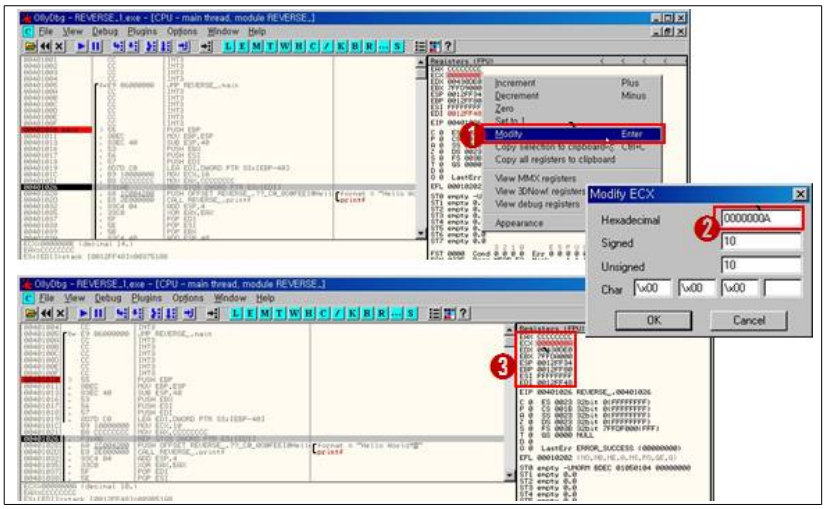
학습내용4 : 프로그램의 변수 값 변경 후 디버깅하기

1. 레지스터 값 고치기(레지스터를 변경하여 디버깅)

① ECX가 0000000E일 때, ECX에서 마우스 오른쪽 버튼 눌러 [Modify] 메뉴 선택
- 팝업창이 뜨면 HEX나 10진수값 등을 넣을 수 있음

② 0000000E를 0000000A로 변경
- ECX 레지스터 값이 0000000A로 변경되며, 'REP STOS DWORD PTR ES:[EDI]'문도

③ 0000000A번만 수행됨

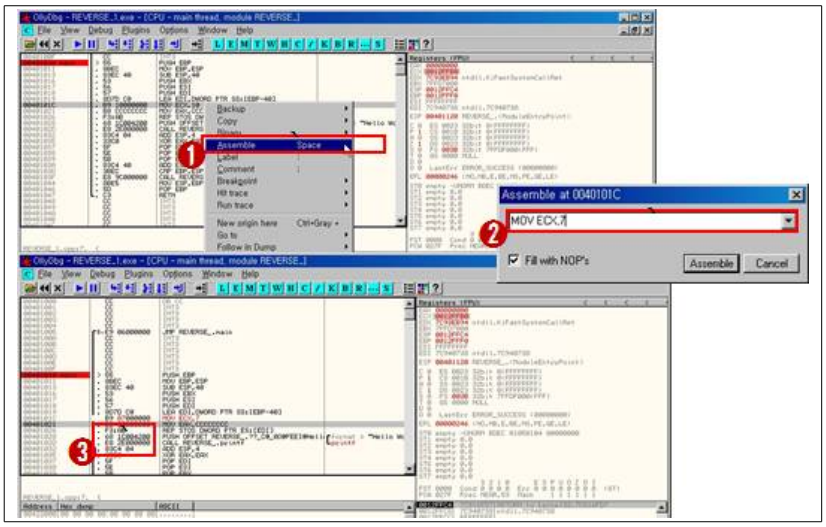


2. 어셈블리어 코드 고치기 (디버거에서 어셈블리어를 고쳐 프로그램 실행)

① ‘ MOVECX, 10’실행되는 0040101C 주소 라인에서 마우스 오른쪽 버튼 눌러 [Assemble] 메뉴 선택

② 팝업창이 뜨면‘MOV ECX, 10’를 ‘MOV ECX, 7’로 수정
- 〈Assemble〉 버튼을 누르면 해당 라인의 변경이 적용되고, 붉은색으로 표시

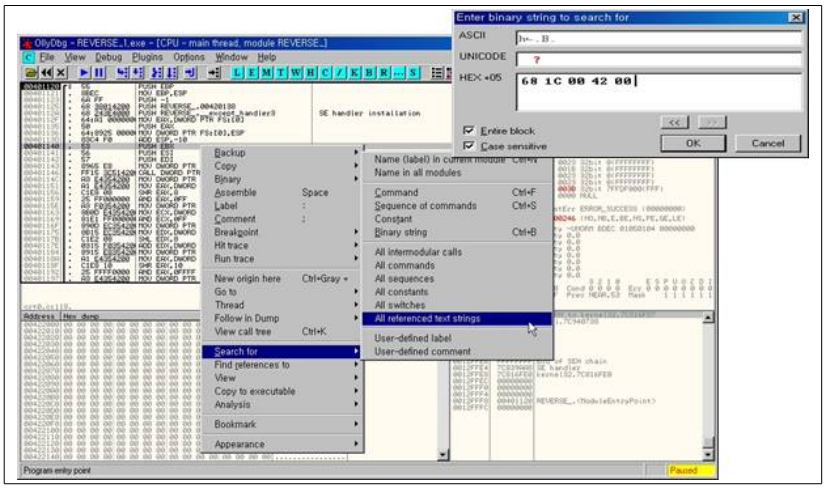
③ 실행하면 ‘MOV ECX, 7’로 적용되어 프로그램 실행



학습내용5 : 값 찾기

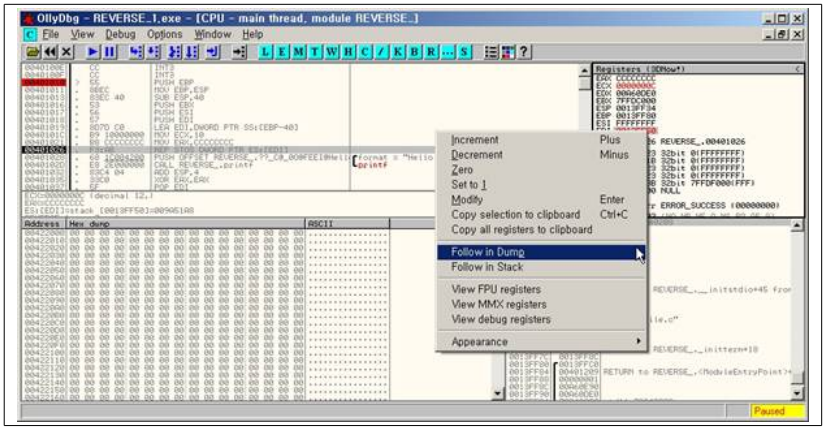
1. 문자열 찾기

- 분석할 포인트를 찾기 위해 가장 많이 사용하는 기능
- 찾는 문자열은 화면에 출력되는 문자열일 수도 있고, 함수의 이름일 수도 있음
- 어셈블리어 창에서 마우스 오른쪽 버튼 클릭 [Search for]-[All referenced text strings] 이용
- [Searchfor]-[Binary String] : 특정 값을 찾기 위해 많이 사용
- ASCII 문자열 찾기
- UNICODE 찾기
- 찾고자 하는 HEX 값을 직접 입력해 찾기도 가능

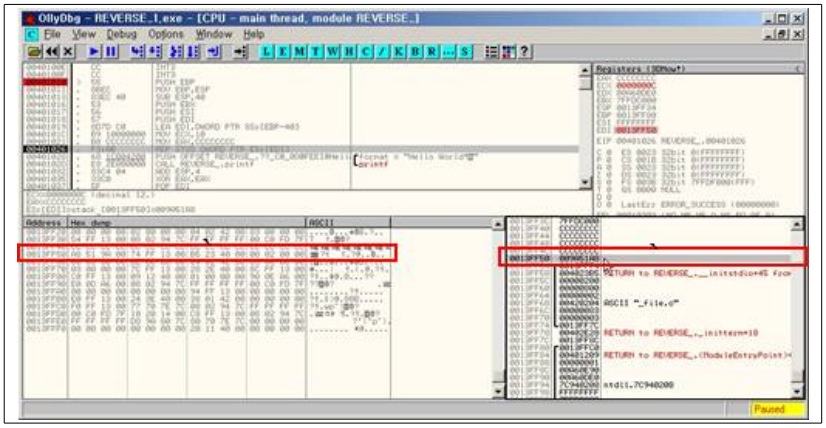


2. 스택과 파일 덤프 위치 찾기

- 'REP STOS DWORD PTR ES:[EDI]'라인에서 F7을 눌러 두세 번 실행
- EDI 값에서 마우스 오른쪽 버튼 클릭 : [Follow in Dump]과 [Follow in Stack] 메뉴 확인
- 파일 덤프 창과 스택에서 해당 주소 값으로 위치 찾아갈 때 사용



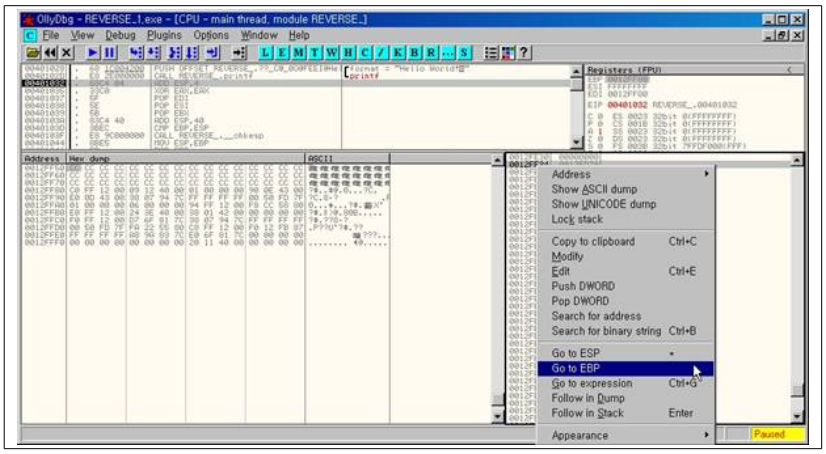
- ‘Follow in Dump’와‘Follow in Stack’명령을 통해 파일 덤프 창과 오른쪽의 스택 창에서 0013FF50 주소의 위치 이동 가능



학습내용6 : 스택 창 기능

1. EBP, ESP 주소 값으로 스택 창 이동하기

- EBP : 함수가 호출되어 시작되는 위치의 주소 값
- 스택에서 마우스 오른쪽 버튼 클릭 : [Go to EBP] 메뉴 실행
- 현재 EBP 레지스터가 가리키는 주소 값에 해당하는 위치로 스택 이동

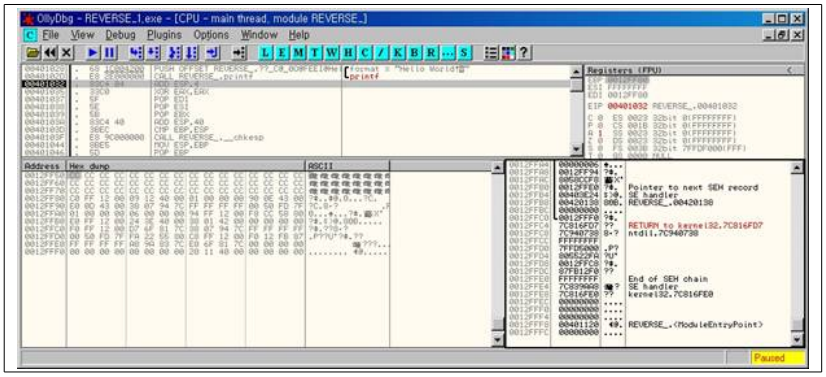


2. 스택 고정하기

- [Lock stack] : 스택이 표시되는 윈도우가 ESP의 값에 따라서 변하는 것을 막을 때 사용

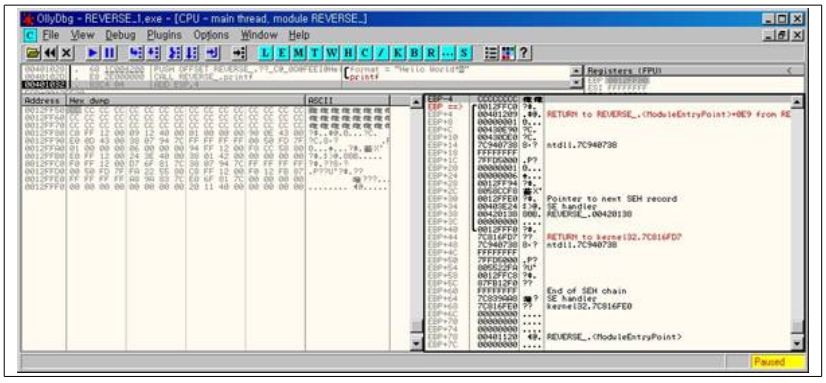
3. 스택 값 확인하기

- [Show ASCII dump]나 [Show UNICODE dump] 메뉴 실행
- 스택에 들어 있는 데이터를 아스키 값이나 유니코드 값으로 확인



4. 스택 상대 주소 확인하기

- [Address]-[Relative toEBP] 메뉴 : EBP를 기준으로 스택의 상대 주소 확인



【학습정리】

1. 리버스 엔지니어링을 위해 올리 디버거, IDA, Process Monitor와 같은 다양한 툴이 사용되고 있다.