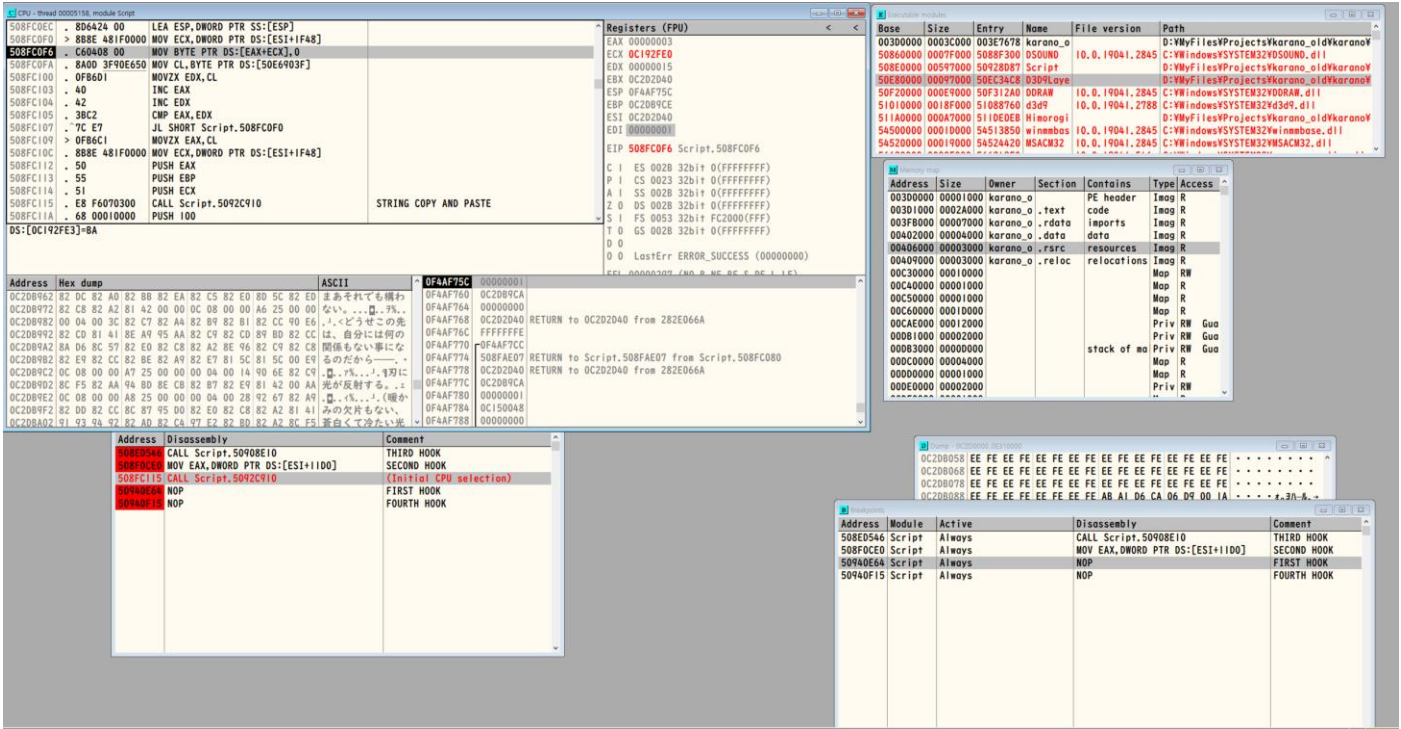
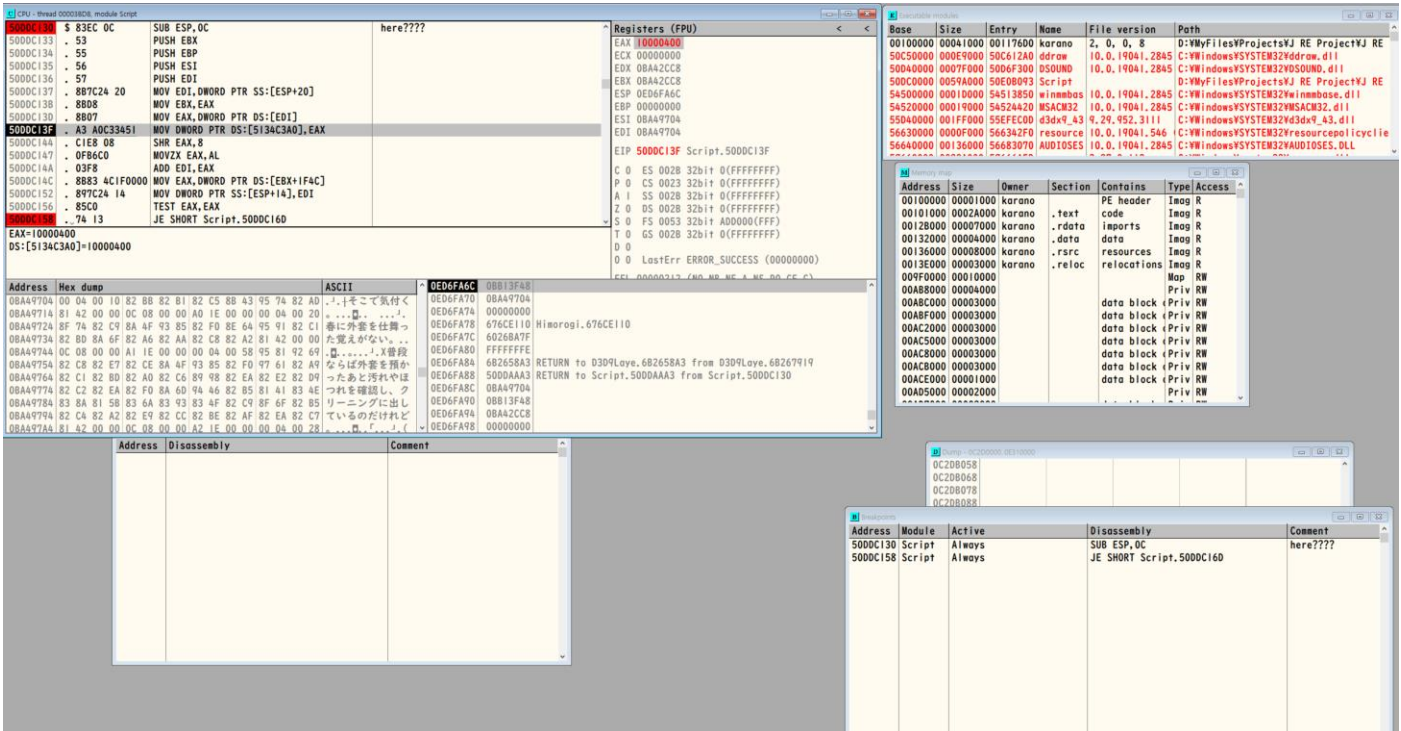


첫페이지는 메모...



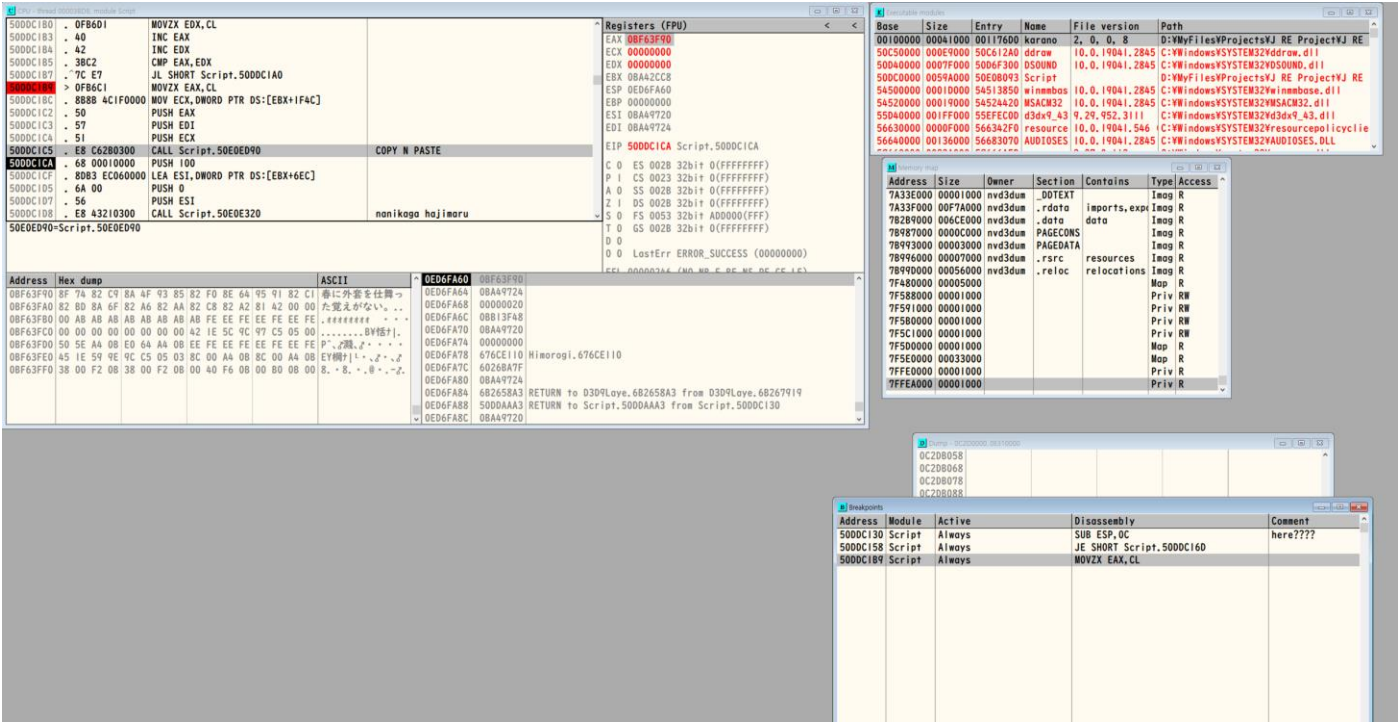
위 사진은 구버전..



EBX 붙여넣기하는곳

EDI 원본주소

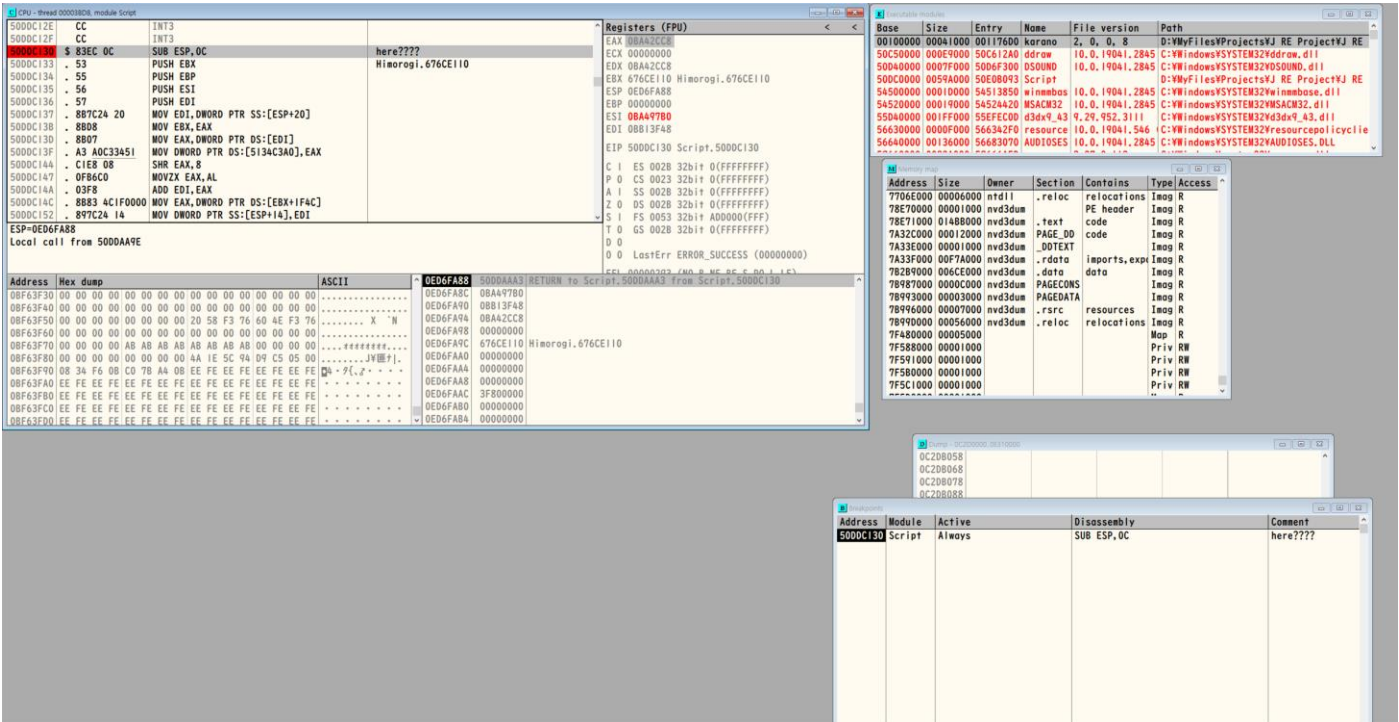
이렇게 테스트해보다가 갑자기 뭔가 떠올라서 기록 남기기 시작



드디어 문자열 copy and paste를 시작하는 최상위 지점을 찾은듯하다..

50DDC1C5 (0018C1C5) = 문자열 붙여넣는 스크립트 이동지점

이 위쪽으로 거슬러 올라가보면...



50DDC130 (0018C130) = 시작지점(?)

이곳이 시작점같다. 그렇다면 이곳 언저리에 후킹하면 되지 않을까 싶은데..

조금 테스트를 해보니 무언가 수상하다





이런식으로 스프라이트가 표시되는 화면에서 브포에 두번 걸린다.

좀더 테스트를 해보며 정확히 어느때 걸리는건지 확인해보자.

테스트 시작은 ??? なにかかしら 가 출력되어있는 상태인데 엄청 많이 봐왔으니 알겠지요?

테스트 시작 화면은 스프라이트 없고, 이름+대사 출력된 상태..



그리고 클릭하여 대사를 넘기면 위 상태가 되며 브포에 걸린다.





다시 F9 를 눌러 진행하면 바로 대사가 출력되며 상태는 Running 이 된다.

→ 대사만 출력되는 상황에서는 브포 한번걸림.

그렇다면 스프라이트가 나오면 어찌될지 보기위해 계속 대사를 넘겨보자

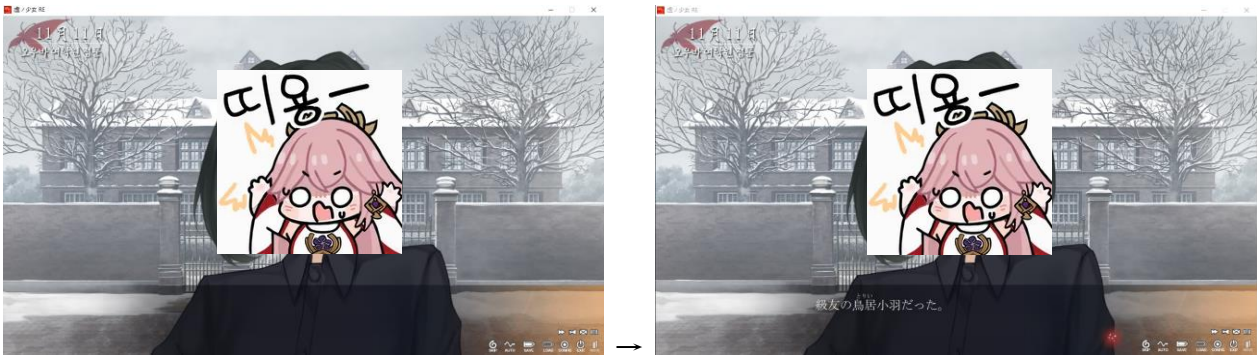
아래는 위 상태에서 클릭을 한번하여 브포에 걸린상태,

이 아래는 왼쪽에서 F9 한번 누른 것



또 F9 를 누르니 이름+대사가 출력됨 (상태는 Running)

→ 아하! 스프라이트 출력시 한번, 이름+대사 출력시 한번이 확실하다.



참고로 스프라이트 표시는 이렇게 커다랗게 튀어나올때도 한번 걸린다..(집중해서 테스트해보다가 갑자기 튀어나와서 깜짝놀람;)

올리씨를 잠시 쉬게 돌려보내고 아랄씨를 불러오자.

라고 생각하며 올리디버거를 종료했는데 아이고 어떤 레지스터를 납치해야하는지 안보고 켜네~

다시 키기도 귀찮고 이미 켜으니 무지성으로 이건가 싶은거 가져와봤는데 아무래도 이건 아닌거같다.



## <여담 1> - 원본데이터구조

The screenshot displays a debugger interface with three main panels:

- Assembly:** Shows instructions such as `SUB ESP, 0C` at address `50BDC12E` and `MOV EAX, 00000000` at `50BDC130`. A comment "here???" is present.
- Registers:** Shows the state of CPU registers, with `EAX` containing `00000000`.
- Memory Dump:** Shows a hex dump of memory starting at `5114C3A0`. The ASCII column shows characters like "冷たくなっていた" and "。...".

올리씨를 켜서 레지스터를 확인하던중, 코드에 [5114C3A0]이라는 의문의 절대주소가 눈에 띈다.

EAX 를 그대로 옮겨갔다놓는데.. 주시해보자.

SHR EAX,8      MOVZX EAX,AL

3400 0400      → 0034 0004      → 0000 0004

그리고 EDI 에 EAX 를 더한다. 이때 EDI 는 PASTE 하는주소값인데, 그럼 1 WORD 만큼 EDI 가 오른것이다.

다시 정리해보면...

- EAX 는 절대주소[5114C3A0]에의해 3400 0400 으로 고정
- EAX 는 Shift Right 이후 AL 만 추출되어 0000 0004 로 변환
- 붙여넣기할 주소를 가리키는 EDI 에 EAX 가 더해져 다음 EDI 로 1WORD 를 추출해올수 있는 준비 완료

Address	Hex dump	ASCII
0BCC9649	97 E2 82 BD 82 AD 82 C8 82 C1 82 C4 82 A2 82 BD	冷たくなっていた
0BCC9659	81 42 00 00 0C 08 00 00 9D IE 00 00 00 04 00 34	。... 4
0BCC9669	82 BB 82 EB 82 BB 82 EB 8C 5A 82 CC 8A 4F 93 85	そろそろ兄の外套
0BCC9679	82 E0 97 70 88 D3 82 B5 82 C4 82 A0 82 B0 82 E9	も用意してあげる
0BCC9689	82 D7 82 AB 82 A9 82 E0 82 B5 82 EA 82 C8 82 A2	べきかもしれない
0BCC9699	81 42 00 42 B4 04 00 0C 79 75 6B 66 30 37 30 36	。BI..yukf0706
0BCC96A9	2E 70 6E 67 14 08 00 00 FA 00 00 00 00 04 00 08	.png 4
0BCC96B9	81 94 8E 87 00 1D 46 00 0C 08 00 00 9E IE 00 00	#紫.F... 4
0BCC96C9	27 08 00 00 00 00 00 07 79 75 6B 30 30 30 32 00	'.....yuk0002.
0BCC96D9	04 00 10 81 75 81 63 81 63 82 A0 82 C1 81 76 00	。+「.....あっ」
0BCC96E9	E2 46 00 11 08 00 00 00 00 00 00 14 08 00 00 FA	礎。 4

이제 이쪽을 한번 보자..

Hex dump 와 ASCII 를 보며 비교해보는거다.

대사, 스포라이트파일명, 인물명 등등 직전에 각 문자열의 길이가 나와있다.

### 〈여담 2〉 - 구버전과의 비교

**Disassembly**

Address	Hex dump	ASCII	Comments
00401000	7E 26	..	JMP SHORT Script.6660C0F0
00401002	00 00 00 00	....	CALL Script.6660C0F0
00401004	8B 4E 4D 4C	..	MOV EAX, EDI
00401006	8B 4E 4D 4C	..	MOV EAX, EDI
00401008	8B 4E 4D 4C	..	MOV EAX, EDI
0040100A	8B 4E 4D 4C	..	MOV EAX, EDI
0040100C	8B 4E 4D 4C	..	MOV EAX, EDI
0040100E	8B 4E 4D 4C	..	MOV EAX, EDI
00401010	8B 4E 4D 4C	..	MOV EAX, EDI
00401012	8B 4E 4D 4C	..	MOV EAX, EDI
00401014	8B 4E 4D 4C	..	MOV EAX, EDI
00401016	8B 4E 4D 4C	..	MOV EAX, EDI
00401018	8B 4E 4D 4C	..	MOV EAX, EDI
0040101A	8B 4E 4D 4C	..	MOV EAX, EDI
0040101C	8B 4E 4D 4C	..	MOV EAX, EDI
0040101E	8B 4E 4D 4C	..	MOV EAX, EDI
00401020	8B 4E 4D 4C	..	MOV EAX, EDI
00401022	8B 4E 4D 4C	..	MOV EAX, EDI
00401024	8B 4E 4D 4C	..	MOV EAX, EDI
00401026	8B 4E 4D 4C	..	MOV EAX, EDI
00401028	8B 4E 4D 4C	..	MOV EAX, EDI
0040102A	8B 4E 4D 4C	..	MOV EAX, EDI
0040102C	8B 4E 4D 4C	..	MOV EAX, EDI
0040102E	8B 4E 4D 4C	..	MOV EAX, EDI
00401030	8B 4E 4D 4C	..	MOV EAX, EDI
00401032	8B 4E 4D 4C	..	MOV EAX, EDI
00401034	8B 4E 4D 4C	..	MOV EAX, EDI
00401036	8B 4E 4D 4C	..	MOV EAX, EDI
00401038	8B 4E 4D 4C	..	MOV EAX, EDI
0040103A	8B 4E 4D 4C	..	MOV EAX, EDI
0040103C	8B 4E 4D 4C	..	MOV EAX, EDI
0040103E	8B 4E 4D 4C	..	MOV EAX, EDI
00401040	8B 4E 4D 4C	..	MOV EAX, EDI
00401042	8B 4E 4D 4C	..	MOV EAX, EDI
00401044	8B 4E 4D 4C	..	MOV EAX, EDI
00401046	8B 4E 4D 4C	..	MOV EAX, EDI
00401048	8B 4E 4D 4C	..	MOV EAX, EDI
0040104A	8B 4E 4D 4C	..	MOV EAX, EDI
0040104C	8B 4E 4D 4C	..	MOV EAX, EDI
0040104E	8B 4E 4D 4C	..	MOV EAX, EDI
00401050	8B 4E 4D 4C	..	MOV EAX, EDI
00401052	8B 4E 4D 4C	..	MOV EAX, EDI
00401054	8B 4E 4D 4C	..	MOV EAX, EDI
00401056	8B 4E 4D 4C	..	MOV EAX, EDI
00401058	8B 4E 4D 4C	..	MOV EAX, EDI
0040105A	8B 4E 4D 4C	..	MOV EAX, EDI
0040105C	8B 4E 4D 4C	..	MOV EAX, EDI
0040105E	8B 4E 4D 4C	..	MOV EAX, EDI
00401060	8B 4E 4D 4C	..	MOV EAX, EDI
00401062	8B 4E 4D 4C	..	MOV EAX, EDI
00401064	8B 4E 4D 4C	..	MOV EAX, EDI
00401066	8B 4E 4D 4C	..	MOV EAX, EDI
00401068	8B 4E 4D 4C	..	MOV EAX, EDI
0040106A	8B 4E 4D 4C	..	MOV EAX, EDI
0040106C	8B 4E 4D 4C	..	MOV EAX, EDI
0040106E	8B 4E 4D 4C	..	MOV EAX, EDI
00401070	8B 4E 4D 4C	..	MOV EAX, EDI
00401072	8B 4E 4D 4C	..	MOV EAX, EDI
00401074	8B 4E 4D 4C	..	MOV EAX, EDI
00401076	8B 4E 4D 4C	..	MOV EAX, EDI
00401078	8B 4E 4D 4C	..	MOV EAX, EDI
0040107A	8B 4E 4D 4C	..	MOV EAX, EDI
0040107C	8B 4E 4D 4C	..	MOV EAX, EDI
0040107E	8B 4E 4D 4C	..	MOV EAX, EDI
00401080	8B 4E 4D 4C	..	MOV EAX, EDI
00401082	8B 4E 4D 4C	..	MOV EAX, EDI
00401084	8B 4E 4D 4C	..	MOV EAX, EDI
00401086	8B 4E 4D 4C	..	MOV EAX, EDI
00401088	8B 4E 4D 4C	..	MOV EAX, EDI
0040108A	8B 4E 4D 4C	..	

String copy and paste 콜 하는부분은 상대주소로 0004 C910 에 존재하나보다.

〈여담 3〉 - string copy and paste 이전 상위 코드를 찾아보자.

Script: 50BDA66

```

$ 81EC 10020000 SUB ESP,210
50BDA66 . AI 680CC350 MOV EAX,DWORD PTR DS:[50C30C68]
50BDA6B . 33C4 XOR EAX,ESP
50BDA6D . 878424 0C0200 MOV DWORD PTR SS:[ESP+20C],EAX
50BDA74 . 0FB68424 1402 MOVZX EAX,BYTE PTR SS:[ESP+214]
50BDA7C . 53 PUSH EBX
50BDA7D . 55 PUSH EBP
50BDA7E . 56 PUSH ESI
50BDA7F . 57 PUSH EDI
50BDA80 . 8BF1 MOV ESI,ECX
50BDA82 . 3D B8000000 CMP EAX,0B8
50BDA87 . 0F87 28080000 JA Script.50BDB285
50BDA8D . 0FB688 84B48D MOVZX ECX,BYTE PTR DS:[EAX+50BDB484]
50BDA94 . FF248D 0CB3D8 JMP DWORD PTR DS:[ECX+4+50BDB300]
> 56 PUSH ESI
50BDA98 . 8BC2 MOV EAX,EDX
50BDA9C . E8 8D160000 CALL Script.50BDC130
50BDC130=Script.50BDC130

```

Registers (FPU)

Base	Size	Entry	Name
00100000	00041000	00117600	karano
509E0000	00063000	50A2FEA0	03D1W700
50A50000	000E9000	50A612A0	drow
50B40000	0007F000	50B6F300	DSOUND
50BC0000	0059A000	50CB0893	Script
51160000	001FF000	5131EC00	d3dx9_43
54500000	0001D000	54513850	winnmbos
54520000	00019000	54524A20	MSACM32
55D10000	00050000	55D190A3	Sound

Script: 50BDA66

```

C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 9A6000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
SEI 00000000 (No NP, NE, A, NS, DF, CF, C)

```

Memory map

Address	Size	Owner	Section
00100000	00001000	karano	
00101000	0002A000	karano	.text
00128000	00007000	karano	.rdata
00132000	00004000	karano	.data
00136000	00008000	karano	.rsrc
0013E000	00003000	karano	.reloc
00690000	00010000		
006A0000	00001000		
006B0000	00001000		
006C0000	00001000		
0070E000	00001200		
00720000	00004000		
00730000	00001000		
00740000	00002000		
00785000	00008000		
00790000	00009000		

Address Hex dump ASCII

```

0BC22C8 00 00 00 00 00 00 AD BA F4 FF FF FF 00 00 00 00 .....3.....
0BC22C8 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....1.....
0BC22CE8 00 00 00 00 20 FF 33 FF 20 00 0E 66 10 67 00 00 ..... 3 .ffg...
0BC22CF8 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA . . . . .
0BC22D08 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA . . . . .
0BC22D18 00 F0 AD BA F4 FF FF FF 00 00 00 00 00 00 00 00 . . . . .
0BC22D28 00 00 00 00 90 01 00 00 00 00 80 04 00 00 00 00 .....0.....
0BC22D38 00 00 00 00 20 FF 33 FF 20 00 0E 66 10 67 00 00 AD BA @. . 3 .ffg...
0BC22D58 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA . . . . .
0BC22D68 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA 00 F0 AD BA . . . . .

```

Registers (FPU)

Base	Size	Entry	Name
00100000	00041000	00117600	karano
509E0000	00063000	50A2FEA0	03D1W700
50A50000	000E9000	50A612A0	drow
50B40000	0007F000	50B6F300	DSOUND
50BC0000	0059A000	50CB0893	Script
51160000	001FF000	5131EC00	d3dx9_43
54500000	0001D000	54513850	winnmbos
54520000	00019000	54524A20	MSACM32
55D10000	00050000	55D190A3	Sound

Script: 50BDA66

```

C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 9A6000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
SEI 00000000 (No NP, NE, A, NS, DF, CF, C)

```

Memory map

Address	Size	Owner	Section
00100000	00001000	karano	
00101000	0002A000	karano	.text
00128000	00007000	karano	.rdata
00132000	00004000	karano	.data
00136000	00008000	karano	.rsrc
0013E000	00003000	karano	.reloc
00690000	00010000		
006A0000	00001000		
006B0000	00001000		
006C0000	00001000		
0070E000	00001200		
00720000	00004000		
00730000	00001000		
00740000	00002000		
00785000	00008000		
00790000	00009000		

이곳이 더 상위지점. 이곳을 브포 걸면 현재 화면을 지울 때도 걸리고 다음화면을 그릴때도 걸린다. 다음 코드파인딩기록은 이것을 중점적으로 살펴보는것으로 시작할듯하다. 이쪽이던가 후킹 지점을 잡으면 되는거 아닐까..?

저기 call susang point(수상한지점) 잡아놓은곳을 타고 가서 50BDC130(0001 C130)에 브포를 걸고 테스트하는건 이번 코드파인딩기록에서 테스트 해본것이다.

그럼 이제 다시한번 정리해보자.







#### <여담 4>

The screenshot shows a debugger window with three main panes:

- Assembly View:** Displays assembly instructions with addresses from 50BDA66 to 50BDA9C. The instruction at 50BDA9C is highlighted in red: `CALL Script.50BDC130`. Comments include "Switch (cases 0..BB)", "Case 0 of switch 50BDA82", and "call susang point".
- Registers (FPU):** Shows the state of various registers. `EAX` is 00000012, `ECX` is 0BCC9B08, `EDX` is 0BCC2CC8, `EBX` is 6B26E110 (labeled "Himorogi.6B26E110"), `ESP` is 0E9BFDB0, `ESI` is 0BCC2CC8, and `EDI` is 0B763F48. The `EIP` register points to `Script.50BDA60`.
- Memory Dump:** Shows a hex dump and ASCII representation of memory starting at address 0BCC998E. The ASCII column contains Japanese text, including "おはようございます" and "ようこそ".

이렇게 브포를 걸면 다음 대사로 넘어갈 때 스프라이트와 대사를 지우는 동작을 하며 걸린다.

참고로 지울때는 위에서 찾은 수상한 지점이 브레이크에 걸리지 않고 화면의 스프라이트와 대사가 지워지고,

새로운 스프라이트와 대사를 출력할 때 그제야 비로소 수상한 지점의 쿨이 호출되어 브레이크가 걸리게 된다.

그렇다면 확실한건 위의 두 브포 사이는 화면 지우기를 담당하고, 화면 출력은 아래쪽의 쿨을 타고 간 이후에 작동한다는것이다!

다음 코드 파인딩 기록은 이곳을 중점적으로 바라보게 될 듯하다.