

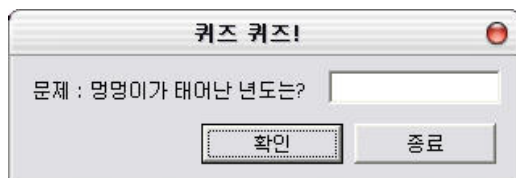
Olly Debugger 사용 방법 강좌 2부

2005.4.14

이번엔 올리 디버거의 사용 실습에 대해 알아보겠습니다.
다루고자하는 내용들은 다음과 같습니다.

- 브레이크 포인트와 실행 재개 방법의 이해.
- 특정 함수가 실행되는 위치 찾기.
- 특정 문자열이 참조되는 위치 찾기.
- 바이너리의 기계어 값을 수정하여 프로그램 흐름 변경하기.

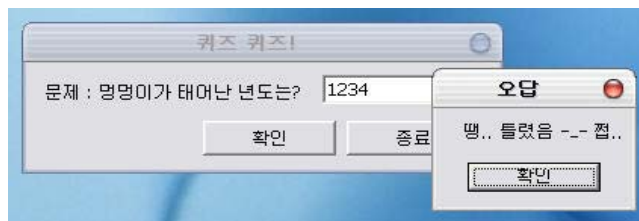
쉬운 이해를 위하여 가장 단순한 Windows 프로그램을 만들어 보았습니다. 프로그램의 역할은 어떤 문제가 주어지고 그것에 대한 답을 맞추는 것입니다.



[그림1]

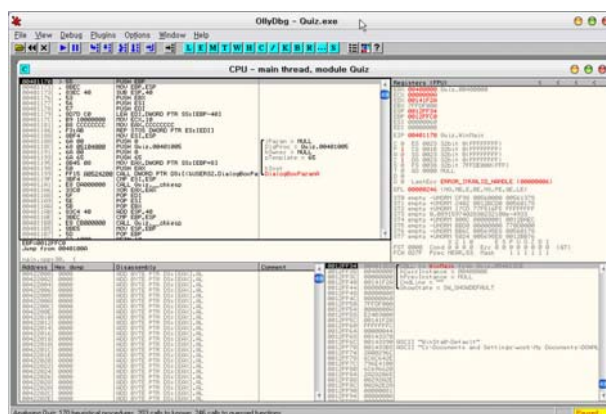
이제 우리가 해 볼 것은 Reverse Engineering을 통해 답이 맞건 틀리건 상관없이 맞은 것으로 처리하도록 하는 것입니다.

일단 이 프로그램을 한 번 실행해 보겠습니다.



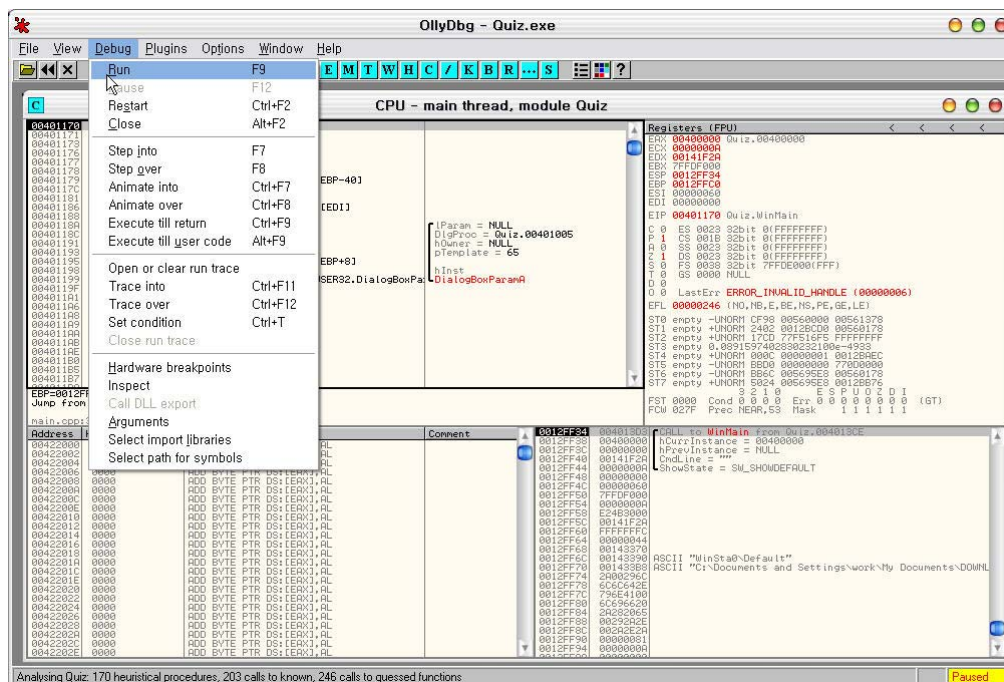
[그림2]

그럼 이제부터 RE(Debugging)를 시작해 보겠습니다.
먼저, 올리 디버거를 실행하시고요. 대상 파일을 불러옵니다.



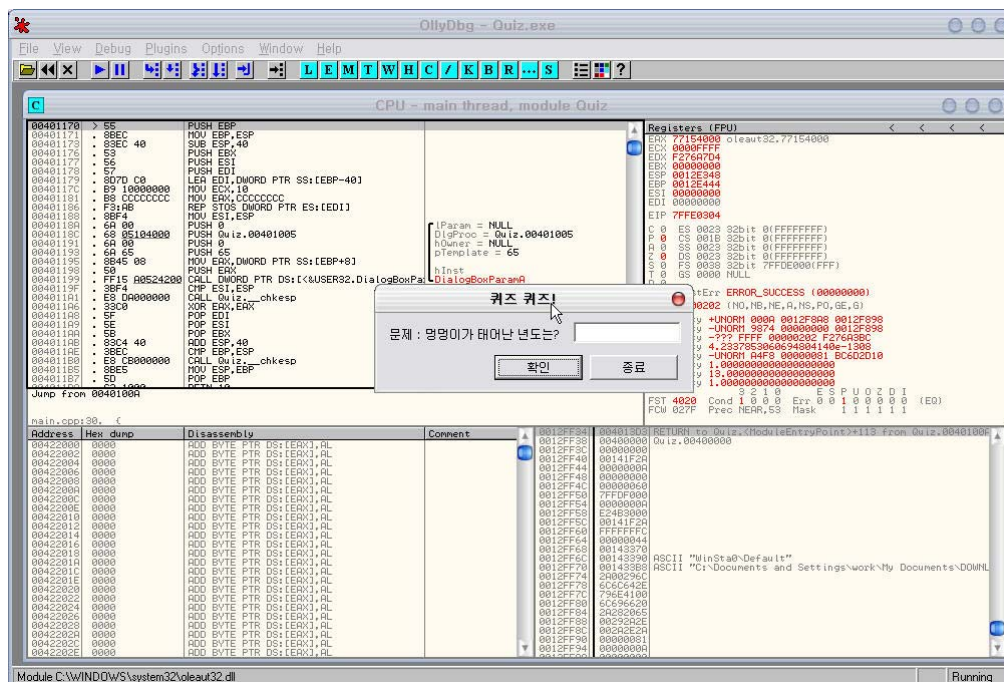
[그림3]

프로그램의 시작 부분인 0x00401170에 자동으로 break 되었습니다.
이제 프로그램을 한 번 계속 실행시켜 봅시다. Debug -> Run을 클릭합니다.



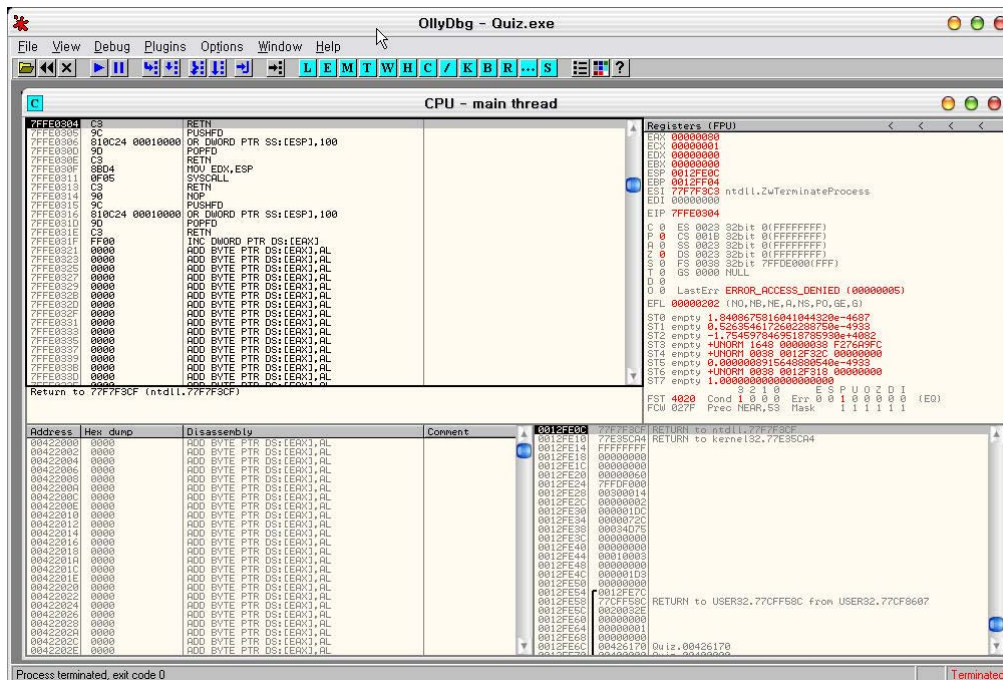
[그림4]

그럼 다음과 같이 디버깅 상태임과 동시에 프로그램이 실행되어 나타납니다.
이제 프로그램을 이래저래 사용해 봅니다. 따로 브레이크 포인트를 설정한 곳이 없기 때문에 디버깅 창들에는 아무 변화가 없는 것을 확인할 수 있습니다.



[그림5]

이제 프로그램을 종료해봅시다.
다음과 같이 프로그램이 종료된 후엔 0x7FFE0304로 이동됩니다.



[그림6]

Address	Size	Owner	Section	Contains	Type	Access	Initial
77CF0000	00001000	USER32		PE header	Image 01001002	R	RWE
77CF1000	00005000	USER32	.text	code, import	Image 01001002	R	RWE
77CF2000	00002000	USER32	.data	data	Image 01001002	R	RWE
77D4F000	0002B000	USER32	.rsrc	resources	Image 01001002	R	RWE
77D7A000	00003000	USER32	.reloc	relocations	Image 01001002	R	RWE
77D80000	00001000	ADVAPI32		PE header	Image 01001002	R	RWE
77D81000	00005000	ADVAPI32	.text	code, import	Image 01001002	R	RWE
77DE6000	00005000	ADVAPI32	.data	data	Image 01001002	R	RWE
77DEB000	0002B000	ADVAPI32	.rsrc	resources	Image 01001002	R	RWE
77E13000	00005000	ADVAPI32	.reloc	relocations	Image 01001002	R	RWE
77E20000	00001000	kernel32		PE header	Image 01001002	R	RWE
77E21000	00075000	kernel32	.text	code, import	Image 01001002	R	RWE
77E96000	00003000	kernel32	.data	data	Image 01001002	R	RWE
77E99000	000A0000	kernel32	.rsrc	resources	Image 01001002	R	RWE
77F39000	00006000	kernel32	.reloc	relocations	Image 01001002	R	RWE
77F50000	00001000	ntdll		PE header	Image 01001002	R	RWE
77F51000	0006F000	ntdll	.text	code, export	Image 01001002	R	RWE
77FC0000	00005000	ntdll	ECODE	code	Image 01001002	R	RWE
77FC5000	00005000	ntdll	.data	data	Image 01001002	R	RWE
77FCA000	0002C000	ntdll	.rsrc	resources	Image 01001002	R	RWE
77FF6000	00005000	ntdll	.reloc	relocations	Image 01001002	R	RWE
78000000	00001000	RPCRT4		PE header	Image 01001002	R	RWE
78001000	00062000	RPCRT4	.text	code, import	Image 01001002	R	RWE
78063000	00006000	RPCRT4	.orpc	code	Image 01001002	R	RWE
78069000	00001000	RPCRT4	.data	data	Image 01001002	R	RWE
7806A000	00001000	RPCRT4	.rsrc	resources	Image 01001002	R	RWE
7806B000	00004000	RPCRT4	.reloc	relocations	Image 01001002	R	RWE
7F6F0000	00007000				Map 00041020	R E	R E
7FA00000	00033000				Map 00041002	R	R
7FFDE000	00001000			data block	Priv 00021840	RWE	RWE
7FFDF000	00001000				Priv 00021840	RWE	RWE
7FFE0000	00001000				Priv 00021840	R	R

[그림7]

자, 이제 오답도 정답이 되도록 작업을 해야겠는데 어느 부분을 어떻게 수정해야 할까요? 일단 어느 부분을 수정해야 할지를 차근차근 생각해 봅시다.

- 가상 메모리의 0x00401000 ~ 0x0041FFFF 까지는 코드 영역이다.
- 이 코드 영역 중에는 분명 정답/오답을 판단하는 루틴이 있다.
- 그럼 이 루틴의 위치를 어떻게 찾을까?
가) 무식하게 모든 코드 영역을 훑어본다. --
나) 특정 함수가 실행되는 순간을 추적한다.
다) 특정 문자열이 참조되는 순간을 추적한다.
- 해당 루틴을 분석하여 RE 작업 수행.

위 가), 나), 다)의 순서로 한번 진행을 해봅시다.
 이 프로그램은 단순하기 때문에 가) 방법이 쉽게 적용됩니다.
 하지만 일반적인 프로그램을 대상으로는 인간이 할 것이 못되겠죠.

먼저 코드 영역의 시작 부분인 0x00401000으로 이동해 봅시다.
 그리고 아래로 조금만 이동해 보면 다음과 같은 부분을 볼 수 있습니다.

0040101C	CC	INT3	
0040101D	CC	INT3	
0040101E	CC	INT3	
0040101F	CC	INT3	
00401020	> 55	PUSH EBP	
00401021	. 8BEC	MOV EBP,ESP	
00401023	. 81EC 48010000	SUB ESP,148	
00401029	. 53	PUSH EBX	
0040102A	. 56	PUSH ESI	
0040102B	. 57	PUSH EDI	
0040102C	. 8DBD B8FEFFFF	LEA EDI,DWORD PTR SS:[EBP-148]	
00401032	. B9 52000000	MOV ECX,52	
00401037	. B8 CCCCCCCC	MOV EAX,CCCCCCCC	
0040103C	. F3AB	REP STOS DWORD PTR ES:[EDI]	
0040103E	. 8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
00401041	. 8985 FCFEFFFF	MOV DWORD PTR SS:[EBP-104],EAX	
00401047	. 81BD FCFEFFFF	CMP DWORD PTR SS:[EBP-104],111	
00401051	. < 74 05	JE SHORT Quiz.00401058	
00401053	. < E9 B7000000	JMP Quiz.0040110F	
00401058	> 8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
0040105B	. 898D F8FEFFFF	MOV DWORD PTR SS:[EBP-108],ECX	
00401061	. 83BD F8FEFFFF	CMP DWORD PTR SS:[EBP-108],1	
00401068	. < 74 12	JE SHORT Quiz.0040107C	
0040106A	. < 83BD F8FEFFFF	CMP DWORD PTR SS:[EBP-108],2	
00401071	. < 0F84 83000000	JE Quiz.004010FA	
00401077	. < E9 93000000	JMP Quiz.0040110F	
0040107C	> 8BF4	MOV ESI,ESP	
0040107E	. 68 FF000000	PUSH 0FF	Count = FF (255.)
00401083	. 8D95 00FFFFFF	LEA EDX,DWORD PTR SS:[EBP-100]	Buffer
00401089	. 52	PUSH EDX	ControlID = 3E8 (1000.)
0040108A	. 68 E8030000	PUSH 3E8	hWnd
0040108F	. 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	GetDlgItemTextA
00401092	. 50	PUSH EAX	
00401093	. FF15 A4524200	CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA]	
00401099	. 3BF4	CMP ESI,ESP	ASCII "1981"
0040109B	. E8 E0010000	CALL Quiz.00401280	
004010A0	. 68 70004200	PUSH Quiz.00420070	
004010A5	. 8D8D 00FFFFFF	LEA ECX,DWORD PTR SS:[EBP-100]	
004010AB	. 51	PUSH ECX	
004010AC	. E8 3F010000	CALL Quiz.004011F0	
004010B1	. 83C4 08	ADD ESP,8	
004010B4	. 85C0	TEST EAX,EAX	
004010B6	. < 75 21	JNZ SHORT Quiz.004010D9	
004010B8	. < 8BF4	MOV ESI,ESP	
004010BA	. 6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
004010BC	. 68 68004200	PUSH Quiz.00420068	Title = "3월 3일"
004010C1	. 68 40004200	PUSH Quiz.00420040	Text = "3월 3일! 1981! 축하합니다"
004010C6	. 8B55 08	MOV EAX,DWORD PTR SS:[EBP+8]	

[그림8]

004010C1	. 68 40004200	PUSH Quiz.00420040	Text = "3월 3일! 1981! 축하합니다"
004010C6	. 8B55 08	MOV EAX,DWORD PTR SS:[EBP+8]	hOwner
004010C9	. 52	PUSH EDX	MessageBoxA
004010CA	. FF15 A8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA]	
004010D0	. 3BF4	CMP ESI,ESP	
004010D2	. E8 A9010000	CALL Quiz.00401280	
004010D7	. < EB 1F	JMP SHORT Quiz.004010F8	
004010D9	> 8BF4	MOV ESI,ESP	
004010DB	. 6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
004010DD	. 68 38004200	PUSH Quiz.00420038	Title = "3월 3일"
004010E2	. 68 1C004200	PUSH Quiz.0042001C	Text = "3월 3일! 1981! 축하합니다"
004010E7	. 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	hOwner
004010EA	. 50	PUSH EAX	MessageBoxA
004010EB	. FF15 A8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA]	
004010F1	. 3BF4	CMP ESI,ESP	
004010F3	. E8 88010000	CALL Quiz.00401280	
004010F8	. < EB 15	JMP SHORT Quiz.0040110F	
004010FA	> 8BF4	MOV ESI,ESP	
004010FC	. 6A 00	PUSH 0	Result = 0
004010FE	. 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	hWnd
00401101	. 51	PUSH ECX	EndDialog
00401102	. FF15 AC524200	CALL DWORD PTR DS:[<&USER32.EndDialog]	
00401108	. 3BF4	CMP ESI,ESP	
0040110A	. E8 71010000	CALL Quiz.00401280	
0040110F	. < 3CC0	XOR EAX,EAX	
00401111	. 5F	POP EDI	
00401112	. 5E	POP ESI	
00401113	. 5B	POP EBX	
00401114	. 81C4 48010000	ADD ESP,148	
0040111A	. 3BEC	CMP EBP,ESP	
0040111C	. E8 5F010000	CALL Quiz.00401280	
00401121	. 8BE5	MOV ESP,EBP	
00401123	. 5D	POP EBP	
00401124	. C2 1000	RETN 10	
00401127	. CC	INT3	
00401128	. CC	INT3	
00401129	. CC	INT3	
0040112A	. CC	INT3	
0040112B	. CC	INT3	
0040112C	. CC	INT3	
0040112D	. CC	INT3	
0040112E	. CC	INT3	
0040112F	. CC	INT3	
00401130	. CC	INT3	
00401131	. CC	INT3	
00401132	. CC	INT3	
00401133	. CC	INT3	
00401134	. CC	INT3	

[그림9]

코드를 조금만 훑어보면 원 소스에서 다음에 해당하는 부분임을 추측할 수 있습니다.

```

=====
BOOL CALLBACK MainProc(HWND hDlg, UINT iMSG, WPARAM wParam, LPARAM lParam)
{
    char Answer[255];

    switch(iMSG)
    {
    case WM_COMMAND:
        switch(wParam)
        {
        case IDOK:
            GetDlgItemText(hDlg, IDC_EDIT1, Answer, 255);
            if(strcmp(Answer, "1981") == 0)
                MessageBox(hDlg, "딩동댕! 맞았음! 축하합니다. ^,^", "
정답", MB_OK);
            else
                MessageBox(hDlg, "땡.. 틀렸음 -_- 꺾..", "오답",
MB_OK);

            break;
        case IDCANCEL:
            EndDialog(hDlg, 0);
            break;
        }
        break;
    }

    return 0;
}
=====

```

그리고 또 조금 내려 보면 또 다른 루틴이 나오는데, 이는 WinMain 함수임을 역시 분석을 통해 추측할 수 있습니다.

00401165	CC	INT3	
00401166	CC	INT3	
00401167	CC	INT3	
00401168	CC	INT3	
00401169	CC	INT3	
0040116A	CC	INT3	
0040116B	CC	INT3	
0040116C	CC	INT3	
0040116D	CC	INT3	
0040116E	CC	INT3	
0040116F	CC	INT3	
00401170	> 55	PUSH EBP	
00401171	8BEC	MOV EBP,ESP	
00401173	83EC 40	SUB ESP,40	
00401176	53	PUSH EBX	
00401177	56	PUSH ESI	
00401178	57	PUSH EDI	
00401179	8D7D C0	LEA EDI,DWORD PTR SS:[EBP-40]	
0040117C	B9 10000000	MOV ECX,10	
00401181	B8 CCCCCC	MOV EBX,CCCCC	
00401186	F3AB	REP STOS DWORD PTR ES:[EDI]	
00401188	8BF4	MOV ESI,ESP	
0040118A	6A 00	PUSH 0	
0040118C	68 05104000	PUSH DWORD 00401005	
00401191	6A 00	PUSH 0	
00401193	6A 65	PUSH 65	
00401195	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
00401198	5B	PUSH EBX	
00401199	FF15 A0524200	CALL DWORD PTR DS:[<USER32.DialogBoxPa	
0040119F	3BF4	CMF ESI,ESP	
004011A1	E8 DA000000	CALL DWORD 00401280	
004011A6	33C0	XOR EAX,EAX	
004011A8	5F	POP EDI	
004011A9	5E	POP ESI	
004011AA	5B	POP EBX	
004011AB	83C4 40	ADD ESP,40	
004011AE	3BEC	CMF EBP,ESP	
004011B0	E8 CB000000	CALL DWORD 00401280	
004011B5	3BEC	CMF EBP,ESP	
004011B7	5D 1000	POP EBP	
004011B8	C2 1000	RET 10	
004011B9	CC	INT3	
004011BC	CC	INT3	
004011BD	CC	INT3	
004011BE	CC	INT3	
004011BF	CC	INT3	
004011C0	CC	INT3	
004011C1	CC	INT3	

[그림10]

```

=====
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPInst, LPSTR Cmd, int ShowCmd)
{
    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG1), HWND_DESKTOP, MainProc);
    return 0;
}
=====

```

여기까지 이해를 했다면 이제 이런 생각이 들 것입니다.

'오호.. MainProc 함수 내의 어딘가를 수정하면 되겠군!'

이렇게 파악을 하였다면 다음에 해야 할 것은 의심가는 이 부분을 차근차근 분석해 나가는 것입니다. 그럼 분석이 진행되면서 자연스레 수정해야할 부분을 알게 됩니다.

00401071	> 0F84 83000000	JE Quiz.004010FA	
00401077	> E9 93000000	JMP Quiz.0040110F	
0040107C	> 8BF4	MOV ESI,ESP	
0040107E	> 68 FF000000	PUSH OFF	Count = FF (255.)
00401083	> 8D95 00FFFFFF	LEA EDX,DWORD PTR SS:[EBP-100]	
00401089	> 52	PUSH EDX	Buffer
0040108B	> 68 E8030000	PUSH 3E8	ControlID = 3E8 (1000.)
0040108F	> 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
00401092	> 50	PUSH EAX	hWnd
00401093	> FF15 A4524200	CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA	GetDlgItemTextA
00401099	> 3BF4	CMP ESI,ESP	
0040109B	> E8 E0010000	CALL Quiz.00401280	
004010A0	> 68 70004200	PUSH Quiz.00420070	ASCII "1981"
004010A5	> 8D8D 00FFFFFF	LEA ECX,DWORD PTR SS:[EBP-100]	
004010AB	> 51	PUSH ECX	
004010AC	> E8 3F010000	CALL Quiz.004011F0	
004010B1	> 83C4 08	ADD ESP,8	
004010B4	> 85C0	TEST EAX,EAX	
004010B6	> 75 21	JNZ SHORT Quiz.004010D9	
004010B8	> 8BF4	MOV ESI,ESP	
004010BA	> 6A 00	PUSH 0	
004010BC	> 68 68004200	PUSH Quiz.00420068	Style = MB_OK!MB_APPLMODAL
004010C1	> 68 40004200	PUSH Quiz.00420040	Title = "오답"
004010C6	> 8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	Text = "오답! 맞았는! 축하"
004010C9	> 52	PUSH EDX	hOwner
004010CA	> FF15 A8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA	MessageBoxA
004010D0	> 3BF4	CMP ESI,ESP	
004010D2	> E8 A9010000	CALL Quiz.00401280	
004010D7	> EB 1F	JMP SHORT Quiz.004010F8	
004010D9	> 3BF4	MOV ESI,ESP	
004010DB	> 6A 00	PUSH 0	
004010DD	> 68 38004200	PUSH Quiz.00420038	Style = MB_OK!MB_APPLMODAL
004010E2	> 68 1C004200	PUSH Quiz.0042001C	Title = "오답"
004010E7	> 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	Text = "오답! 맞은! 축하"
004010EA	> 50	PUSH EAX	hOwner
004010EB	> FF15 A8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA	MessageBoxA
004010F1	> 3BF4	CMP ESI,ESP	
004010F3	> E8 88010000	CALL Quiz.00401280	
004010F8	> EB 15	JMP SHORT Quiz.0040110F	
004010FA	> 8BF4	MOV ESI,ESP	
004010FC	> 6A 00	PUSH 0	
004010FE	> 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	Result = 0
00401101	> 51	PUSH ECX	hWnd
00401102	> FF15 AC524200	CALL DWORD PTR DS:[<&USER32.EndDialog>]	EndDialog
00401105	> 3BF4	CMP ESI,ESP	
0040110A	> E8 71010000	CALL Quiz.00401280	
0040110F	> 33C0	XOR EAX,EAX	
00401111	> CF	POP ESI	

[그림11]

그럼 분석을 해보겠습니다.

```

0040107E |. 68 FF000000    PUSH OFF                      ; /Count = FF (255.)
00401083 |. 8D95 00FFFFFF   LEA EDX,DWORD PTR SS:[EBP-100] ; |
00401089 |. 52              PUSH EDX                      ; |Buffer
0040108A |. 68 E8030000     PUSH 3E8                      ; |ControlID = 3E8
(1000.)
0040108F |. 8B45 08         MOV EAX,DWORD PTR SS:[EBP+8]   ; |
00401092 |. 50              PUSH EAX                      ; |hWnd
00401093 |. FF15 A4524200   CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA
-----> GetDlgItemText 함수 호출 부분입니다.

```

```

00401099 |. 3BF4           CMP ESI,ESP
0040109B |. E8 E0010000    CALL Quiz.00401280
004010A0 |. 68 70004200    PUSH Quiz.00420070           ; ASCII "1981"
004010A5 |. 8D8D 00FFFFFF   LEA ECX,DWORD PTR SS:[EBP-100]
004010AB |. 51             PUSH ECX
004010AC |. E8 3F010000    CALL Quiz.004011F0
004010B1 |. 83C4 08        ADD ESP,8
004010B4 |. 85C0           TEST EAX,EAX

```

-----> strcmp 함수로 "1981"과 버퍼의 값을 비교합니다.

```
004010B6 |. 75 21          JNZ SHORT Quiz.004010D9
-----> 리턴 값이 0이 아니라면 0x004010D9로 점프합니다.
```

```
004010B8 |. 8BF4          MOV ESI,ESP
004010BA |. 6A 00          PUSH 0                      ; /Style =
MB_OK|MB_APPLMODAL
004010BC |. 68 68004200    PUSH Quiz.00420068         ; |Title = "정답"
004010C1 |. 68 40004200    PUSH Quiz.00420040         ; |Text = "딩동댕! 맞았
음! 축하합니다. ^,^"
004010C6 |. 8B55 08        MOV EDX,DWORD PTR SS:[EBP+8] ; |
004010C9 |. 52            PUSH EDX                    ; |hOwner
004010CA |. FF15 A8524200  CALL DWORD PTR DS:[<&USER32.MessageBoxA>; WMessageBoxA
004010D0 |. 3BF4          CMP ESI,ESP
004010D2 |. E8 A9010000    CALL Quiz.00401280
004010D7 |. EB 1F          JMP SHORT Quiz.004010F8
-----> 답이 맞았을 때의 처리입니다.
```

```
004010D9 |> 8BF4          MOV ESI,ESP
004010DB |. 6A 00          PUSH 0                      ; /Style =
MB_OK|MB_APPLMODAL
004010DD |. 68 38004200    PUSH Quiz.00420038         ; |Title = "오답"
004010E2 |. 68 1C004200    PUSH Quiz.0042001C         ; |Text = "땡.. 틀렸음
-_- 찼.. "
004010E7 |. 8B45 08        MOV EAX,DWORD PTR SS:[EBP+8] ; |
004010EA |. 50            PUSH EAX                    ; |hOwner
004010EB |. FF15 A8524200  CALL DWORD PTR DS:[<&USER32.MessageBoxA>; WMessageBoxA
-----> 틀렸을 때의 처리입니다.
```

자, 위의 분석 내용을 차분하게 생각해보면 어떤 부분을 어떻게 수정해야할지 답이 나옵니다.

머리를 어떻게 굴리느냐에 따라서 수정 방법은 다양해질 수 있겠지만 저에겐 다음 부분이 가장 눈에 띄었습니다.

```
004010B6 |. 75 21          JNZ SHORT Quiz.004010D9
-----> 리턴 값이 0이 아니라면 0x004010D9로 점프합니다.
```

답이 틀릴 경우 0x004010D9로 점프하는 부분입니다. 이 부분을 이렇게 수정할 수 있겠죠.

- 1) JNZ(Jump Not Zero)를 반대 의미인 JZ(Jump Zero)로 바꿈.
- 2) 0x004010D9를 0x004010B8(맞았을 때의 루틴)로 바꿈.

이제 이 부분을 수정할 것을 생각하면서 브레이크 포인트를 겁니다.
F2 키를 누르면 이 부분의 주소가 빨강색으로 바뀝니다.

0040107C	> 8BF4	MOV ESI,ESP	
0040107E	. 68 FF000000	PUSH 0FF	Count = FF (255.)
00401083	. 3D95 00FFFFFF	LEA EDX,DWORD PTR SS:[EBP-100]	Buffer
00401089	. 52	PUSH EDX	ControlID = 3E8 (1000.)
0040108A	. 68 E8030000	PUSH 3E8	hWnd
0040108F	. 3B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	GetDlgItemTextA
00401092	. 50	PUSH EAX	
00401093	. FF15 B4524200	CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA@4]	
00401099	. 3BF4	CMF ESI,ESP	
0040109B	. E8 E0010000	CALL Quiz.00401280	
004010A0	. 68 70004200	PUSH Quiz.00420070	
004010A5	. 8D8D 00FFFFFF	LEA ECX,DWORD PTR SS:[EBP-100]	ASCII "1981"
004010AB	. 51	PUSH ECX	
004010AC	. E8 3F010000	CALL Quiz.004011F0	
004010B1	. 83C4 08	ADD ESP,8	
004010B4	. 85C0	TEST EAX,EAX	
004010B6	< 75 21	JNZ SHORT Quiz.004010D9	
004010B8	. 8BF4	MOV ESI,ESP	
004010BA	. 6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004010BC	. 68 68004200	PUSH Quiz.00420068	Title = "오답"
004010C1	. 68 40004200	PUSH Quiz.00420040	Text = "오답입니다! 맞았습니까? 축하합니다!"
004010C6	. 8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	hOwner
004010C9	. 52	PUSH EDX	MessageBoxA
004010CA	. FF15 B8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA@4]	
004010D0	. 3BF4	CMF ESI,ESP	
004010D2	. E8 A9010000	CALL Quiz.00401280	
004010D7	> EB 1F	JMP SHORT Quiz.004010F8	
004010D9	> 8BF4	MOV ESI,ESP	
004010DB	. 6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004010DD	. 68 38004200	PUSH Quiz.00420038	Title = "오답"
004010E2	. 68 1C004200	PUSH Quiz.0042001C	Text = "오답입니다! 맞았습니까? 축하합니다!"
004010E7	. 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	hOwner
004010EA	. 50	PUSH EAX	MessageBoxA
004010EB	. FF15 B8524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA@4]	
004010F1	. 3BF4	CMF ESI,ESP	
004010F3	. E8 88010000	CALL Quiz.00401280	
004010F8	> EB 15	JMP SHORT Quiz.0040110F	
004010FA	> 8BF4	MOV ESI,ESP	
004010FC	. 6A 00	PUSH 0	Result = 0
004010FE	. 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	hWnd
00401101	. 51	PUSH ECX	EndDialog
00401102	. FF15 AC524200	CALL DWORD PTR DS:[<&USER32.EndDialog@4]	
00401108	. 3BF4	CMF ESI,ESP	
0040110A	. E8 71010000	CALL Quiz.00401280	
0040110F	> 33C0	XOR EAX,EAX	
00401111	. 5F	POP EDI	
00401112	. 5E	POP ESI	
00401113	. EB	END	

[그림12]

이제 프로그램을 다시 실행해 봅시다.

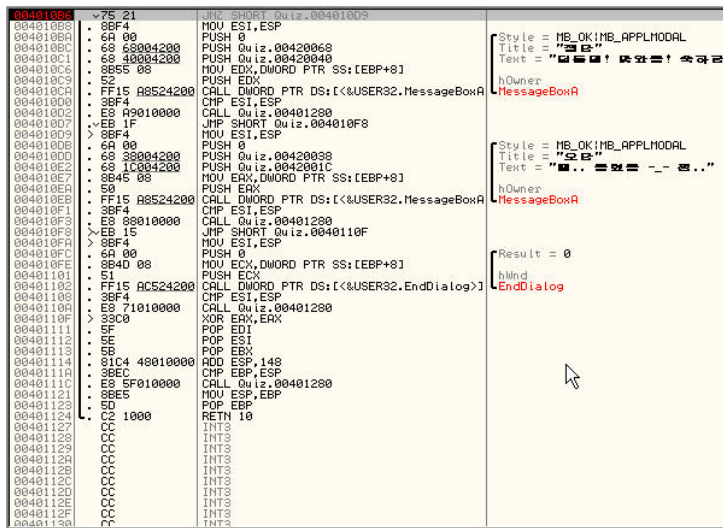
Run	F9
Pause	F12
Restart	Ctrl+F2
Close	Alt+F2
Step into	F7
Step over	F8
Animate into	Ctrl+F7
Animate over	Ctrl+F8
Execute till return	Ctrl+F9
Execute till user code	Alt+F9
Open or clear run trace	
Trace into	Ctrl+F11
Trace over	Ctrl+F12
Set condition	Ctrl+T
Close run trace	
Hardware breakpoints	
Inspect	
Call DLL export	
Arguments	
Select import libraries	
Select path for symbols	

[그림13]

처음 브레이크가 걸리면 Debug-Run을 수행합니다.

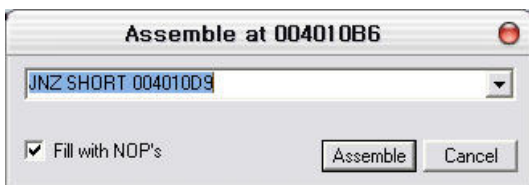
그럼 프로그램이 실행되고, 아무 값을 입력하고 확인을 누릅니다.

이제 다음과 같이 우리가 설정한 브포에서 또 다시 멈추게 될 것입니다.



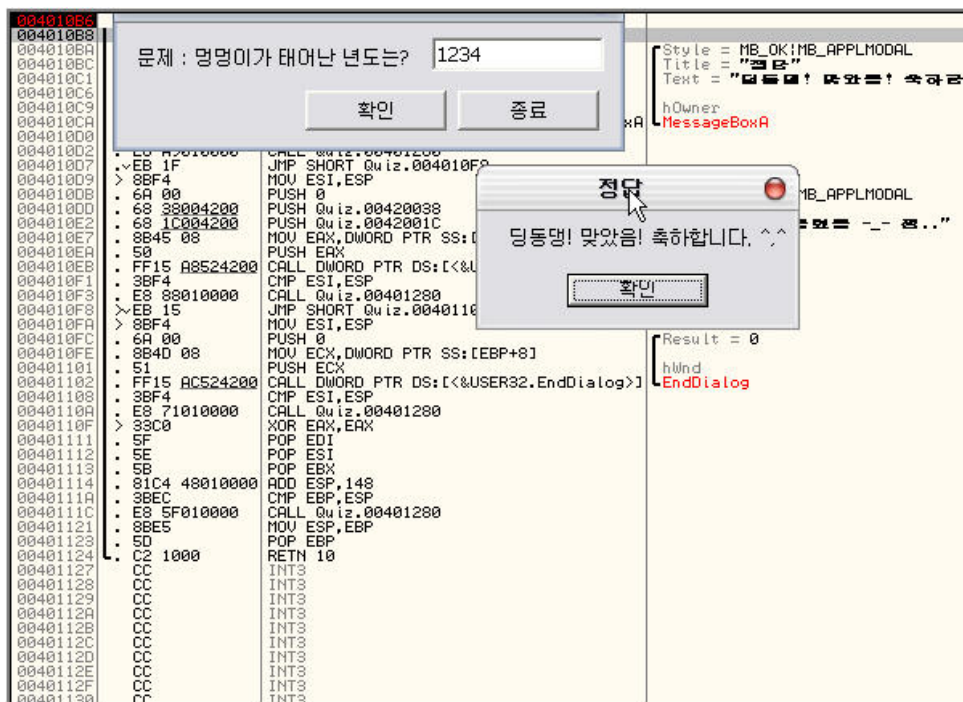
[그림14]

간단하게 JNZ를 JZ로 바꾸겠습니다. 해당 라인을 더블 클릭하면 됩니다.



[그림15]

이제 다시 Debug-Run(F9)를 눌러 프로그램을 재개합니다.
그럼 다음과 같이 우회에 성공한 모습을 볼 수 있을 겁니다.

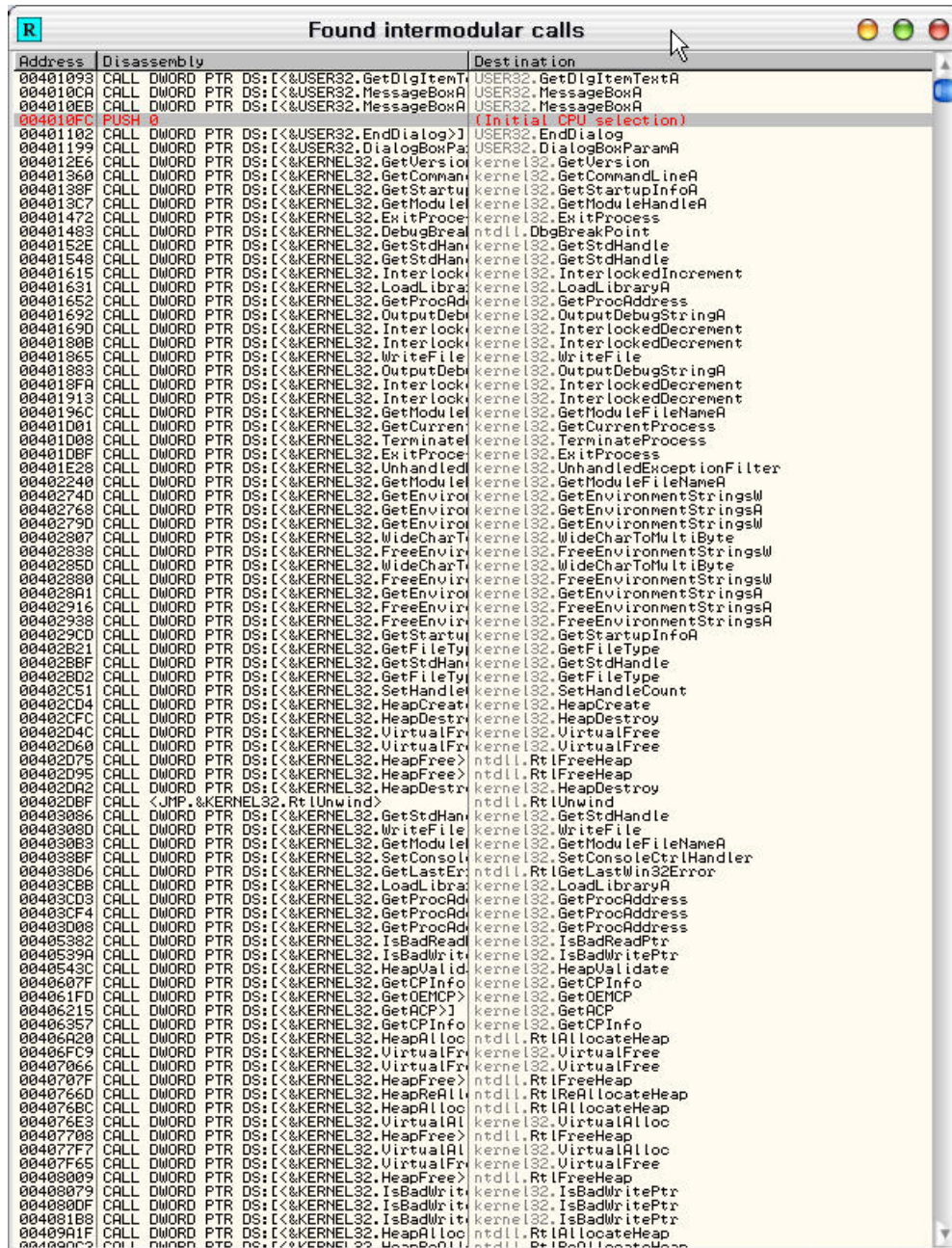


[그림16]

됐죠. 자 이번엔 나) 방법으로 한 번 해보겠습니다.

Search for -> All intermodular calls 메뉴를 클릭합니다.

그럼 다음과 같이 프로그램 내에서 호출된 함수 목록을 볼 수 있습니다.



[그림17]

이들 함수들 중 인증과 관련되었을 법한 함수를 선택합니다.

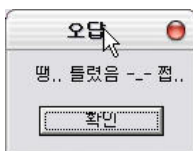
저는 GetDlgItemText 선택하였습니다. 확인 버튼을 클릭하면 이 함수를 이용하여 사용자가 입력한 문자열을 가져오기 때문입니다.



[그림18]

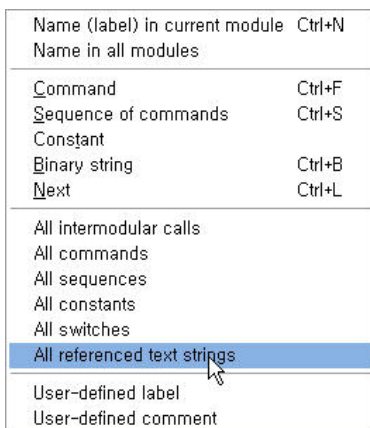
그럼 다음과 같은 주소로 이동하게 되는데, 앞서 봤던 그 부분이지요?
이렇게 정답/오답 판단과 관련된 부분을 쉽게 찾았습니다.
바이너리 수정은 아까 한 것과 동일하게 하면 되겠죠.

이제 마지막으로 다) 방법을 사용해 보겠습니다. 이도 간단합니다.
앞서 오답을 입력하였을 때 "땡.. 틀렸음 -_- 쩌.." 이 문자열을 검색하면
우리가 원하는 루틴을 찾을 수 있겠죠.



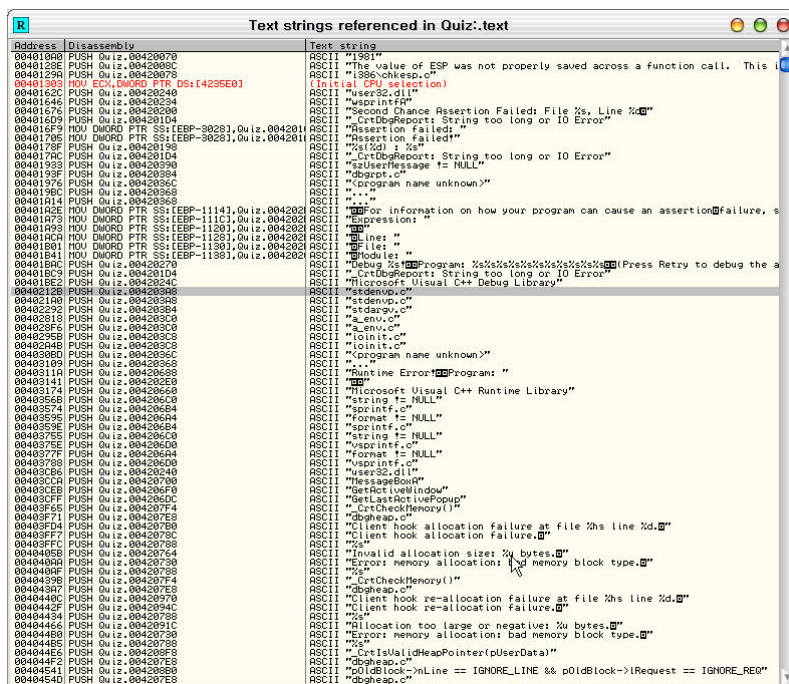
[그림19]

이제 Search for -> All referenced text strings를 클릭합니다.



[그림20]

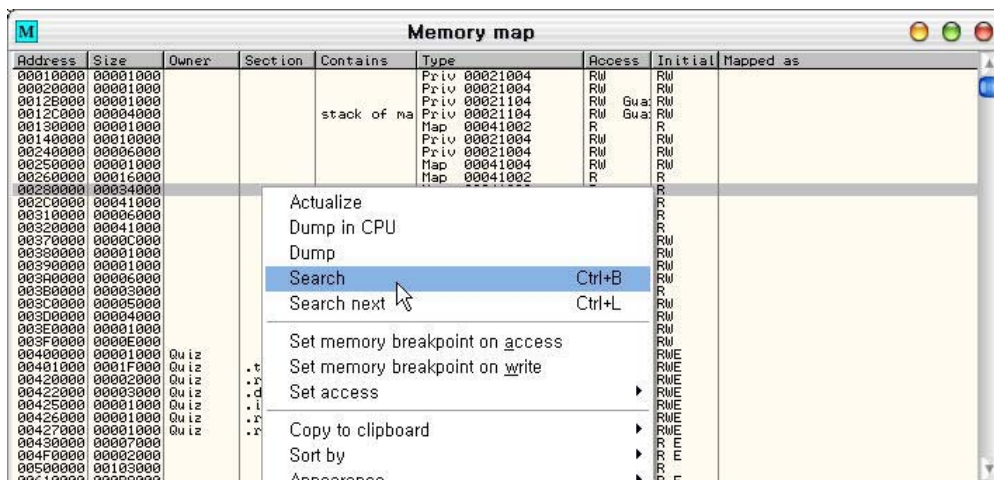
그럼 다음과 같이 프로그램 내의 문자열들이 검색됩니다.



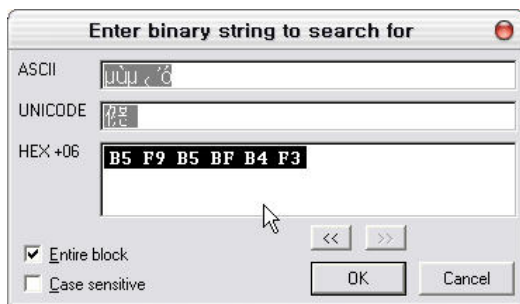
[그림21]

그런데 역시 지난번에 말씀드린 바와 같이 완벽히 검색을 해내지 못하네요.
검색하려는 문자열이 한글이라서 그런 것 같고요. 참고로 W32DASM에선 검색이 됩니다.

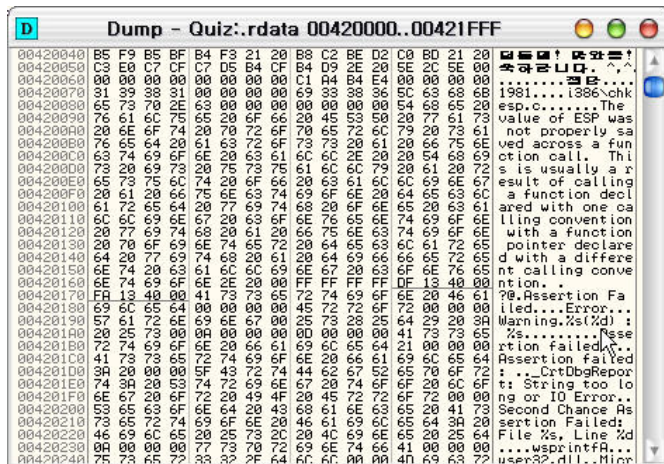
이제 수동 검색을 통해 문자열을 찾아봅시다. View - Memory에서 마우스 오른쪽 버튼을 누른 후 "틀렸음" 이라는 문자열을 검색해 봅시다.



[그림22]

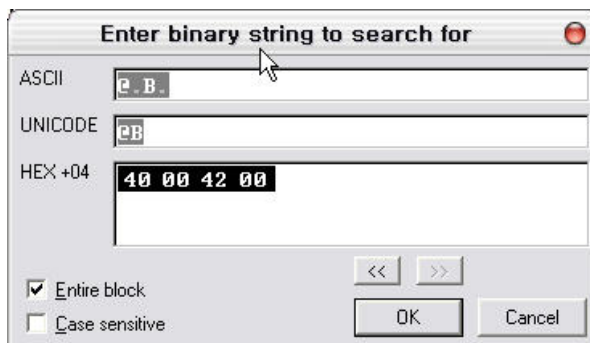


[그림23]



[그림24]

0x00420040 에서 문자열을 찾았습니다. 근데 이 주소로 찾아가는 게 아니고, 이 주소를 참조한 주소를 다시 찾아야겠죠?
다시 검색을 해봅시다. 위 주소가 리틀 엔디안으로 저장된 40004200 으로 검색하면 되겠습니다.
디버깅 윈도우에서 오른쪽 버튼을 누른 후 Search for -> Binary String을 이용합니다.



[그림25]

그럼 이처럼 우리가 찾고자했던 루틴으로 이동됩니다.

00401068	· 74 12	JE SHORT Quiz.0040107C	
0040106A	· 33BD F8FFFFFF	CMPL DWORD PTR SS:[EBP-108],2	
00401071	· 0F84 83000000	JE Quiz.004010FA	
00401077	· E9 93000000	JMP Quiz.0040110F	
0040107C	> 8BF4	MOV ESI,ESP	
0040107E	· 68 FF000000	PUSH 0FF	Count = FF (255.)
00401083	· 3D95 00FFFFFF	LEA EDX,DWORD PTR SS:[EBP-100]	Buffer
00401089	· 52	PUSH EDX	ControlID = 3E8 (1000.)
0040108A	· 68 E8030000	PUSH 3E8	hWnd
0040108F	· 3B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	GetDlgItemTextA
00401092	· 50	PUSH EAX	
00401093	· FF15 84524200	CALL DWORD PTR DS:[<&USER32.GetDlgItemTextA@USER32.dll]	ASCII "1981"
00401099	· 3BF4	CMPL ESI,ESP	
0040109B	· E8 E0010000	CALL Quiz.00401280	
004010A0	· 68 70004200	PUSH Quiz.00420070	
004010A5	· 3D8D 00FFFFFF	LEA ECX,DWORD PTR SS:[EBP-100]	
004010A8	· 51	PUSH ECX	
004010AC	· E8 3F010000	CALL Quiz.004011F0	
004010B1	· 83C4 08	ADD ESP,8	
004010B4	· 35C0	TEST EAX,EAX	
004010B6	· 75 21	JNZ SHORT Quiz.004010D9	
004010B8	· 3BF4	MOV ESI,ESP	
004010BA	· 6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
004010BC	· 68 68004200	PUSH Quiz.00420068	Title = "오답"
004010C1	· 68 40004200	PUSH Quiz.00420040	Text = "오답입니다. 다시 시도하십시오."
004010C6	· 3B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	hOwner
004010C9	· 52	PUSH EDX	MessageBoxA
004010CA	· FF15 85242000	CALL DWORD PTR DS:[<&USER32.MessageBoxA@USER32.dll]	
004010D0	· 3BF4	CMPL ESI,ESP	
004010D2	· E8 A9010000	CALL Quiz.00401280	
004010D7	· EB 1F	JMP SHORT Quiz.004010F8	
004010D9	> 3BF4	MOV ESI,ESP	
004010DB	· 6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
004010DD	· 68 38004200	PUSH Quiz.00420038	Title = "오답"
004010E2	· 68 1C004200	PUSH Quiz.0042001C	Text = "오답입니다. 다시 시도하십시오."
004010E7	· 3B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	hOwner
004010EA	· 50	PUSH EAX	MessageBoxA
004010EB	· FF15 85242000	CALL DWORD PTR DS:[<&USER32.MessageBoxA@USER32.dll]	
004010F1	· 3BF4	CMPL ESI,ESP	
004010F3	· E8 88010000	CALL Quiz.00401280	
004010F8	> EB 15	JMP SHORT Quiz.0040110F	
004010FA	> 3BF4	MOV ESI,ESP	
004010FC	· 6A 00	PUSH 0	Result = 0
004010FE	· 3B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	hWnd
00401101	· 51	PUSH ECX	EndDialog
00401102	· FF15 AC524200	CALL DWORD PTR DS:[<&USER32.EndDialog@USER32.dll]	
00401108	· 3BF4	CMPL ESI,ESP	
0040110A	· E8 71010000	CALL Quiz.00401280	

[그림 26]

이제 마찬가지로 앞서 했던 것처럼 RE 작업을 해나가면 되겠습니다.
여기까지 하고요! 이번 강좌는 여기서 마치도록 하겠습니다.