리버스 엔지니어링

6장 -흔히 사용하는 패턴-

4/14

발표자 : 김정우

OVERVIEW

- 흔히 사용하는 패턴?

  - 리버싱을 하면서 반복해서 나오는 패턴에 대해 어셈블리를 분석

  - 반복해서 나오는 패턴? -> 조건문! 반복문!

조건문의 기본

main.c

Project4 (전역 범위)

```
#include <stdio.h>

void main() {
    int x = 0;
    if (x > 0x100) {
        printf("x > 100₩n");
    }
}
```

위의 코드를 디스어셈블 하면?

조건문의 기본

왼쪽 인자의 값이 오른쪽보다 작으면 분기!
If 문의 조건에 따라 해당 명령어가 변경됨!

ex) if (x == 0) → (jnz short loc_101)

```
.text$mn:000000E4                mov     [ebp+var_8], 0
.text$mn:000000EB                cmp     [ebp+var_8], 100h
.text$mn:000000F2                jle     short loc_101
.text$mn:000000F4                push    offset ??_C@_0BOGOFFHHO@x?5?$DO?5100?6@ ; "x > 100%n"
.text$mn:000000F9                call    _printf
.text$mn:000000FE                add     esp, 4
.text$mn:00000101
.text$mn:00000101 loc_101:                               ; CODE XREF: _main+36↑j
.text$mn:00000101                xor     eax, eax
.text$mn:00000103                pop     edi
.text$mn:00000104                pop     esi
.text$mn:00000105                pop     ebx
.text$mn:00000106                add     esp, 0CCh
.text$mn:0000010C                cmp     ebp, esp
.text$mn:0000010E                call    __RTC_CheckEsp
.text$mn:00000113                mov     esp, ebp
.text$mn:00000115                pop     ebp
.text$mn:00000116                retn
.text$mn:00000116 _main          endp
```

심화된 조건문 – if else 문

main.c

Project4 (전역 범위)

```
#include <stdio.h>

void main() {
    int x = 0;
    if (x > 0x100)
        printf("x > 100₩n");
    else
        printf("x = else₩n");
}
```

위의 코드를 디스어셈블 하면?

심화된 조건문 – if else 문

```
.text$mn:000000E4                mov     [ebp+var_8], 0
.text$mn:000000EB                cmp     [ebp+var_8], 100h
.text$mn:000000F2                jle     short loc_103
.text$mn:000000F4                push    offset ??_C@_08OGOFFHHO@x?5?$DO?5100?6@ ; "x > 100¥n"
.text$mn:000000F9                call    _printf
.text$mn:000000FE                add     esp, 4
.text$mn:00000101                jmp     short loc_110
.text$mn:00000103 ; ---------------------------------------------------------------------------
.text$mn:00000103
.text$mn:00000103 loc_103:                               ; CODE XREF: _main+36↑j
.text$mn:00000103                push    offset ??_C@_09GDONEFOA@x?5?$DN?5else?6@ ; "x = else¥n"
.text$mn:00000108                call    _printf
.text$mn:0000010D                add     esp, 4
.text$mn:00000110
.text$mn:00000110 loc_110:                               ; CODE XREF: _main+45↑j
.text$mn:00000110                xor     eax, eax
.text$mn:00000112                pop     edi
.text$mn:00000113                pop     esi
.text$mn:00000114                pop     ebx
.text$mn:00000115                add     esp, 0CCh
.text$mn:0000011B                cmp     ebp, esp
.text$mn:0000011D                call    __RTC_CheckEsp
.text$mn:00000122                mov     esp, ebp
.text$mn:00000124                pop     ebp
.text$mn:00000125                retn
.text$mn:00000125 _main          endp
```

이전 if문과 별차이없이 else에 해당되는 코드만 추가 됨!

IF ELSE문이 추가될 때마다 분기문이 추가됨!

심화된 조건문 – 조건문에 &&, || 들어가면?

```
.text$mn:000000E4                mov     [ebp+var_8], 0
.text$mn:000000EB                cmp     [ebp+var_8], 100h
.text$mn:000000F2                jle     short loc_10C
.text$mn:000000F4                cmp     [ebp+var_8], 200h
.text$mn:000000FB                jge     short loc_10C
.text$mn:000000FD                push    offset ??_C@_08OGOFFHHO@x?5?$DO?5100?6@ ; "x > 100\n"
.text$mn:00000102                call    _printf
.text$mn:00000107                add     esp, 4
.text$mn:0000010A                jmp     short loc_119
.text$mn:0000010C ; ---------------------------------------------------------------------------
.text$mn:0000010C
.text$mn:0000010C loc_10C:                               ; CODE XREF: _main+36↑j
.text$mn:0000010C                                        ; _main+3F↑j
.text$mn:0000010C                push    offset ??_C@_09GDONEFOA@x?5?$DN?5else?6@ ; "x = else\n"
.text$mn:00000111                call    _printf
.text$mn:00000116                add     esp, 4
.text$mn:00000119
.text$mn:00000119 loc_119:                               ; CODE XREF: _main+4E↑j
.text$mn:00000119                xor     eax, eax
.text$mn:0000011B                pop     edi
.text$mn:0000011C                pop     esi
.text$mn:0000011D                pop     ebx
.text$mn:0000011E                add     esp, 0CCh
.text$mn:00000124                cmp     ebp, esp
.text$mn:00000126                call    __RTC_CheckEsp
.text$mn:0000012B                mov     esp, ebp
.text$mn:0000012D                pop     ebp
.text$mn:0000012E                retn
.text$mn:0000012E _main          endp
```

Cmp + jmp 코드가 연속으로 등장!

조건문 이후에 action이 나오는지 아닌지가 핵심

반복문

main.c

Project4 (전역 범위)

```c
#include <stdio.h>

void main() {
    int x = 0;

    for (int i = 0; i <= 0x100; i++)
    {
        printf("loop : %d₩n",x);
        x++;
    }
}
```

위의 코드를 디스어셈블 하면?

반복문

1

3

2

```
.text$mn:000000E4                mov     [ebp+var_8], 0
.text$mn:000000EB                mov     [ebp+var_14], 0
.text$mn:000000F2                jmp     short loc_FD
.text$mn:000000F4 ; ---------------------------------------------------------------------------
.text$mn:000000F4
.text$mn:000000F4 loc_F4:                                ; CODE XREF: _main+64↓j
.text$mn:000000F4                mov     eax, [ebp+var_14]
.text$mn:000000F7                add     eax, 1
.text$mn:000000FA                mov     [ebp+var_14], eax
.text$mn:000000FD
.text$mn:000000FD loc_FD:                                ; CODE XREF: _main+36↑j
.text$mn:000000FD                cmp     [ebp+var_14], 100h
.text$mn:00000104                jg      short loc_122
.text$mn:00000106                mov     eax, [ebp+var_8]
.text$mn:00000109                push    eax
.text$mn:0000010A                push    offset ??_C@_0L@EIJACMNN@loop?5?3?5?$CFd?6@ ; "loop : %d\n"
.text$mn:0000010F                call    _printf
.text$mn:00000114                add     esp, 8
.text$mn:00000117                mov     eax, [ebp+var_8]
.text$mn:0000011A                add     eax, 1
.text$mn:0000011D                mov     [ebp+var_8], eax
.text$mn:00000120                jmp     short loc_F4
.text$mn:00000122 ;
.text$mn:00000122
.text$mn:00000122 loc_122:                               ; CODE XREF: _main+48↑j
.text$mn:00000122                xor     eax, eax
.text$mn:00000124                pop     edi
.text$mn:00000125                pop     esi
.text$mn:00000126                pop     ebx
.text$mn:00000127                add     esp, 0D8h
.text$mn:0000012D                cmp     ebp, esp
.text$mn:0000012F                call    __RTC_CheckEsp
.text$mn:00000134                mov     esp, ebp
.text$mn:00000136                pop     ebp
.text$mn:00000137                retn
.text$mn:00000137 _main          endp
.text$mn:00000137
```

Int x , int i 의 값을 0으로 초기화

Int i의 값을 증가

조건식을 확인하고 X의 값을 출력

X의 값에 1을 더하고 분기

반복문 – break가 들어가는 경우

Project4

```c
#include <stdio.h>

void main() {
    int c = 0;
    int d = 0;

    for (int i = 0; i <= 0x100; i++)
    {
        if (i==c)
        {
            break;
        }


        printf("loop : %d₩n",d);
        d++;
    }
}
```

```
.text$mn:000000E4                mov     [ebp+var_8], 0
.text$mn:000000EB                mov     [ebp+var_14], 0
.text$mn:000000F2                mov     [ebp+var_20], 0
.text$mn:000000F9                jmp     short loc_104
.text$mn:000000FB ; ---------------------------------------------------------------------------
.text$mn:000000FB
.text$mn:000000FB loc_FB:                                ; CODE XREF: _main+75↓j
.text$mn:000000FB                mov     eax, [ebp+var_20]
.text$mn:000000FE                add     eax, 1
.text$mn:00000101                mov     [ebp+var_20], eax
.text$mn:00000104
.text$mn:00000104 loc_104:                               ; CODE XREF: _main+3D↑j
.text$mn:00000104                cmp     [ebp+var_20], 100h
.text$mn:0000010B                jg      short loc_133
.text$mn:0000010D                mov     eax, [ebp+var_20]
.text$mn:00000110                cmp     eax, [ebp+var_8]
.text$mn:00000113                jnz     short loc_117
.text$mn:00000115                jmp     short loc_133
.text$mn:00000117 ; ---------------------------------------------------------------------------
.text$mn:00000117
.text$mn:00000117 loc_117:                               ; CODE XREF: _main+57↑j
.text$mn:00000117                mov     eax, [ebp+var_14]
.text$mn:0000011A                push    eax
.text$mn:0000011B                push    offset ??_C@_0L@EIJACMNN@loop?5?3?5?$CFd?6@ ; "loop : %d₩n"
.text$mn:00000120                call    _printf
.text$mn:00000125                add     esp, 8
.text$mn:00000128                mov     eax, [ebp+var_14]
.text$mn:0000012B                add     eax, 1
.text$mn:0000012E                mov     [ebp+var_14], eax
.text$mn:00000131                jmp     short loc_FB
.text$mn:00000133 ; ---------------------------------------------------------------------------
.text$mn:00000133
.text$mn:00000133 loc_133:                               ; CODE XREF: _main+4F↑j
.text$mn:00000133                                        ; _main+59↑j
.text$mn:00000133                xor     eax, eax
```

Var_8 => c
Var_14 => d
Var_20 => i

i와 c 를 비교하고 분기!

반복문 – continue가 들어갈 경우

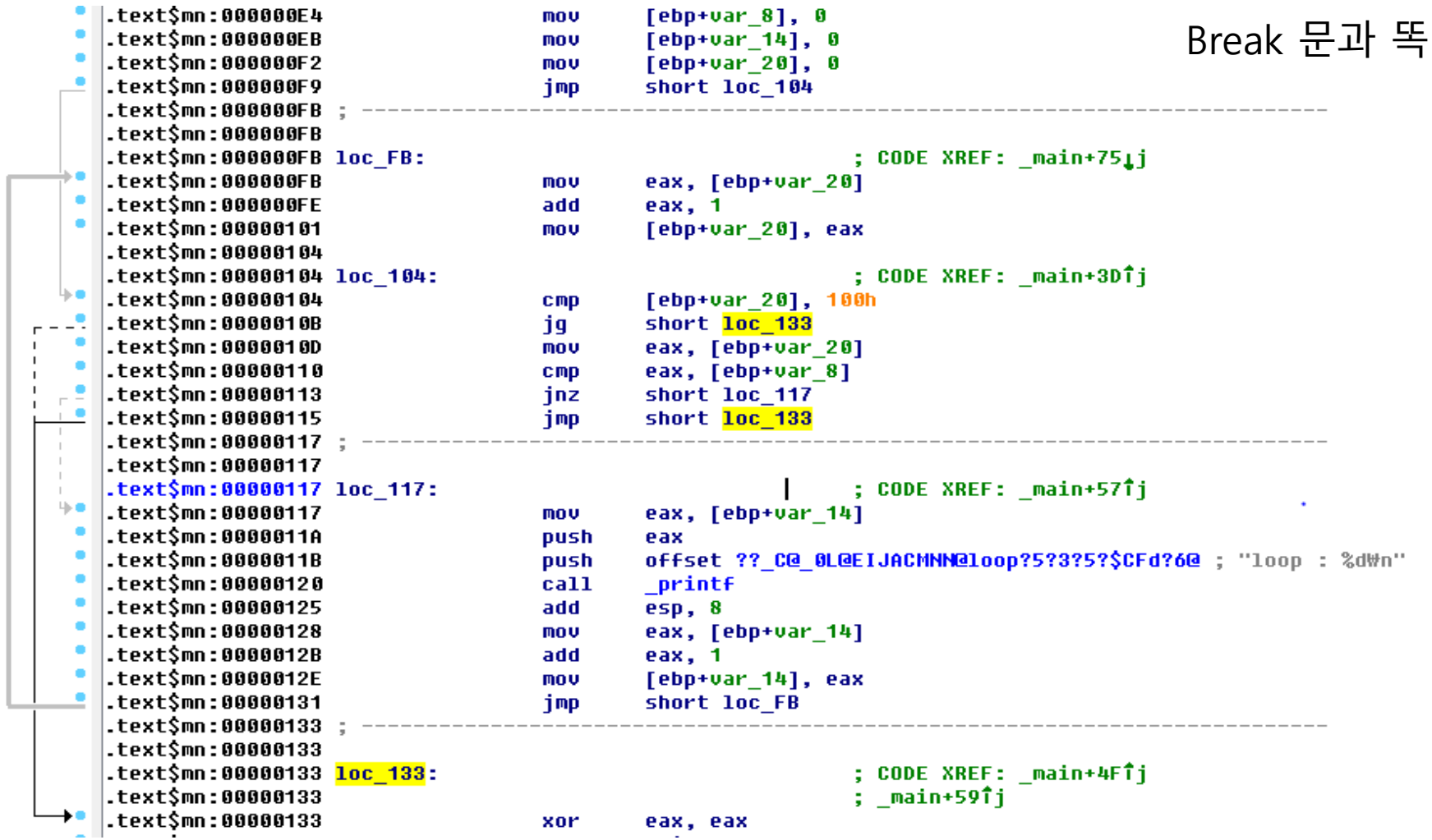
main.c

Project4

```c
#include <stdio.h>

void main() {
    int c = 0;
    int d = 0;

    for (int i = 0; i <= 0x100; i++)
    {
        if (i==c)
        {
            continue;
        }


        printf("loop : %d₩n",d);
        d++;
    }
}
```

```
.text$mn:000000F9                   jmp     short loc_104
.text$mn:000000FB ; ---------------------------------------------------------------------------
.text$mn:000000FB
.text$mn:000000FB loc_FB:                               ; CODE XREF: _main+59↓j
.text$mn:000000FB                                       ; _main+75↓j
.text$mn:000000FB                   mov     eax, [ebp+var_20]
.text$mn:000000FE                   add     eax, 1
.text$mn:00000101                   mov     [ebp+var_20], eax
.text$mn:00000104
.text$mn:00000104 loc_104:                              ; CODE XREF: _main+3D↑j
.text$mn:00000104                   cmp     [ebp+var_20], 100h
.text$mn:0000010B                   jg      short loc_133
.text$mn:0000010D                   mov     eax, [ebp+var_20]
.text$mn:00000110                   cmp     eax, [ebp+var_8]
.text$mn:00000113                   jnz     short loc_117
.text$mn:00000115                   jmp     short loc_FB
.text$mn:00000117 ; ---------------------------------------------------------------------------
.text$mn:00000117
.text$mn:00000117 loc_117:                              ; CODE XREF: _main+57↑j
.text$mn:00000117                   mov     eax, [ebp+var_14]
.text$mn:0000011A                   push    eax
.text$mn:0000011B                   push    offset ??_C@_0L@EIJACMNN@loop?5?3?5?$CFd?6@ ; "loop : %d₩n"
.text$mn:00000120                   call    _printf
.text$mn:00000125                   add     esp, 8
.text$mn:00000128                   mov     eax, [ebp+var_14]
.text$mn:0000012B                   add     eax, 1
.text$mn:0000012E                   mov     [ebp+var_14], eax
.text$mn:00000131                   jmp     short loc_FB
.text$mn:00000133 ; ---------------------------------------------------------------------------
.text$mn:00000133
.text$mn:00000133 loc_133:                              ; CODE XREF: _main+4F↑j
.text$mn:00000133                   xor     eax, eax
```

반복문 – return이 생기는 경우



Break 문과 똑같이 분기!

문자열 컨트롤

- C언어에서 사용되는 문자열 라이브러리가

  디스어셈블되는 구조를 확인하자!

strcpy

main.c

Project4

```c
#include <stdio.h>

void main() {
    char *szSource = "Hello World !";
    char szTarget[15];

    strcpy(szTarget, szSource);
    printf("%s₩n", szTarget);
}
```

위의 코드를 디스어셈블 하면?

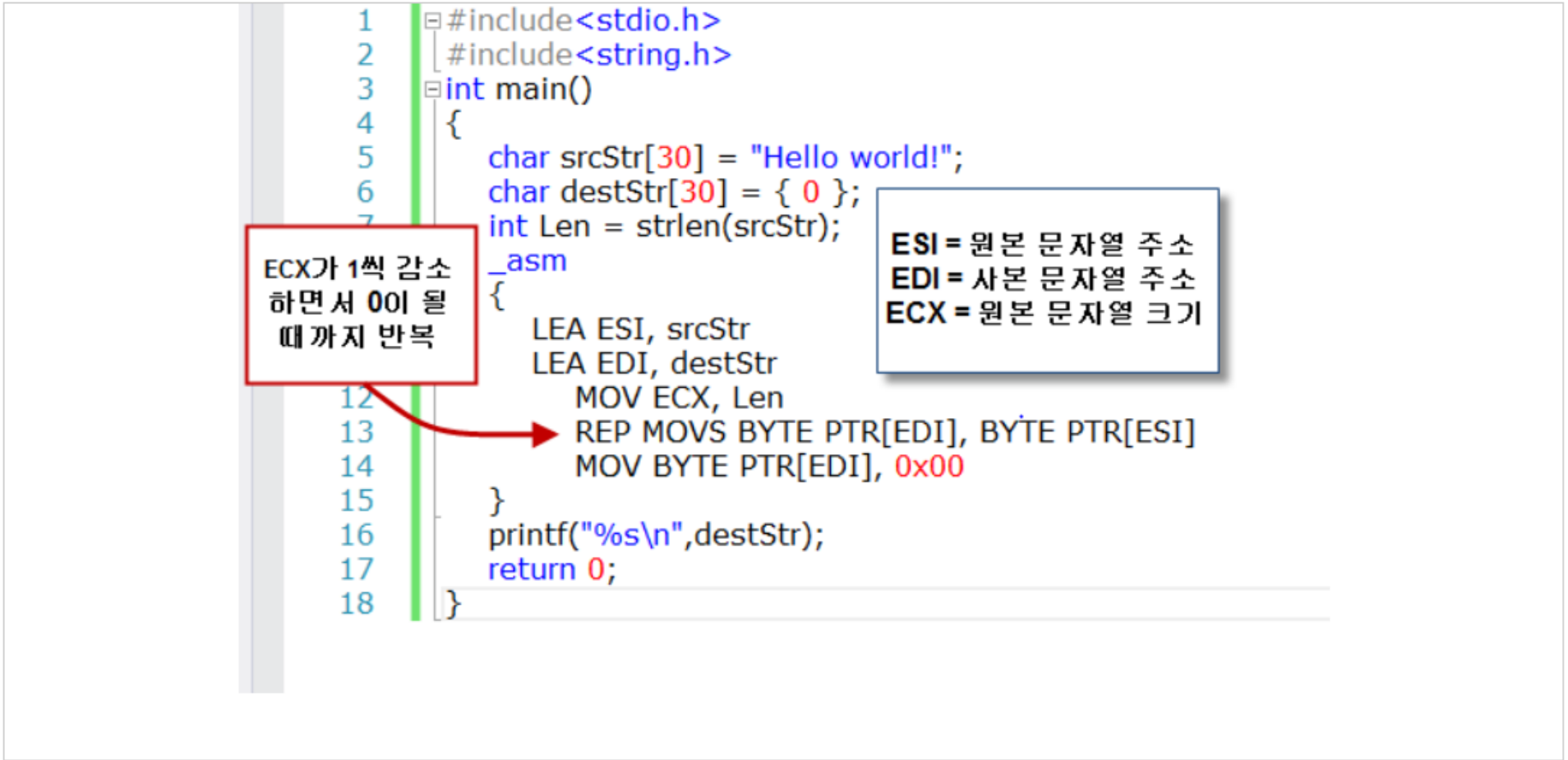
strcpy

```
.text$mn:000000BC push    ebp
.text$mn:000000BD mov     ebp, esp
.text$mn:000000BF sub     esp, 0E8h
.text$mn:000000C5 push    ebx
.text$mn:000000C6 push    esi
.text$mn:000000C7 push    edi
.text$mn:000000C8 lea     edi, [ebp+var_E8]
.text$mn:000000CE mov     ecx, 3Ah ; ':'
.text$mn:000000D3 mov     eax, 0CCCCCCCCh
.text$mn:000000D8 rep stosd
.text$mn:000000DA mov     eax, dword ptr ds:___security_cookie
.text$mn:000000DF xor     eax, ebp
.text$mn:000000E1 mov     [ebp+var_4], eax
.text$mn:000000E4 mov     ecx, offset __3F388996_main@c
.text$mn:000000E9 call    @__CheckForDebuggerJustMyCode@4 ; __CheckForDebuggerJustMyCode(n)
.text$mn:000000EE mov     [ebp+Source], offset ??_C@_0O@KCFADNJI@Hello?5World?5?$CB@ ; "Hello World !"
.text$mn:000000F5 mov     eax, [ebp+Source]
.text$mn:000000F8 push    eax             ; Source
.text$mn:000000F9 lea     ecx, [ebp+Dest]
.text$mn:000000FC push    ecx             ; Dest
.text$mn:000000FD call     strcpy
.text$mn:00000102 add     esp, 8
.text$mn:00000105 lea     eax, [ebp+Dest]
.text$mn:00000108 push    eax
.text$mn:00000109 push    offset ??_C@_03OFAPEBGM@?$CFs?6@ ; "%sWn"
.text$mn:0000010E call    _printf
```

Strcpy 함수호출여부를 알려줌!

책에 나온 다른 문자열함수도
마찬가지로 호출여부를 알려줌!

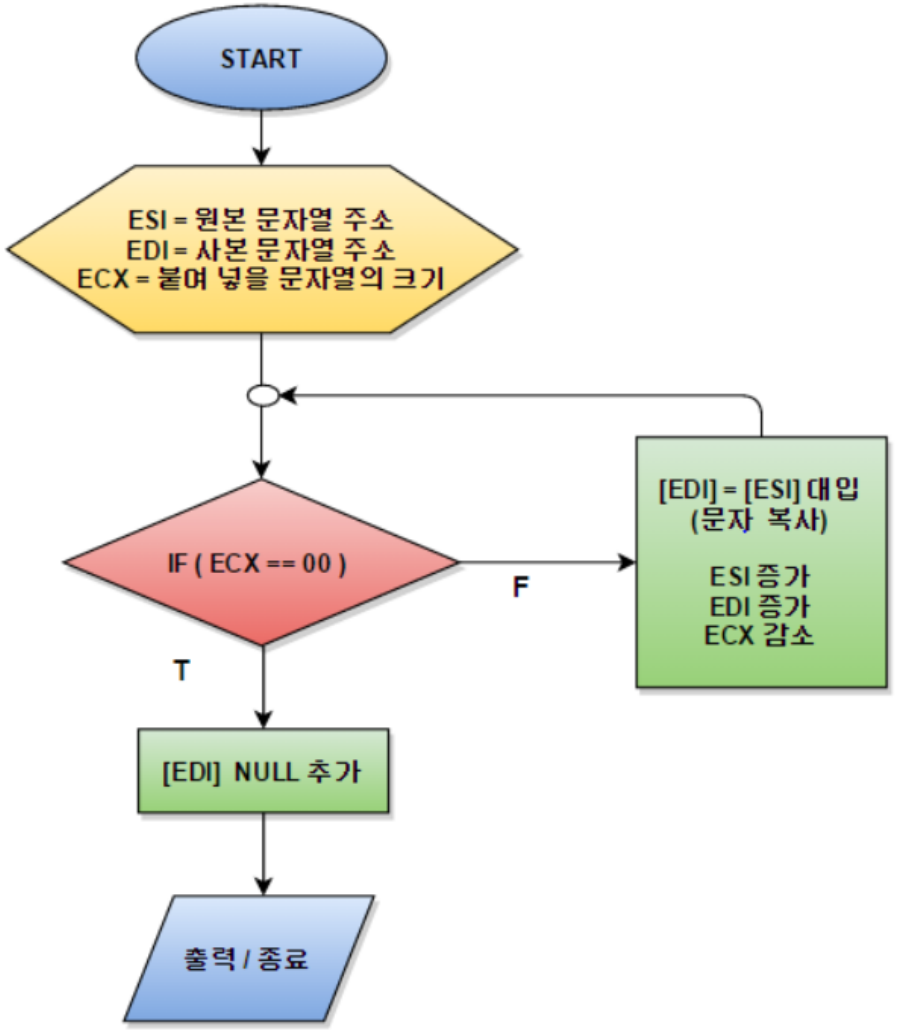
Strcpy
이해를 위한 어셈블리 소스




```
1  #include<stdio.h>
2  #include<string.h>
3  int main()
4  {
5      char srcStr[30] = "Hello world!";
6      char destStr[30] = { 0 };
       int Len = strlen(srcStr);
       _asm
       {
           LEA ESI, srcStr
           LEA EDI, destStr
12             MOV ECX, Len
13             REP MOVS BYTE PTR[EDI], BYTE PTR[ESI]
14             MOV BYTE PTR[EDI], 0x00
15     }
16     printf("%s\n",destStr);
17     return 0;
18 }
```

[strcpy 어셈블리 소스]

Strcpy 순서도


START
ESI = 원본 문자열 주소
EDI = 사본 문자열 주소
ECX = 붙여 넣을 문자열의 크기
IF (ECX == 00)
F
T
[EDI] = [ESI] 대입
(문자 복사)
ESI 증가
EDI 증가
ECX 감소
[EDI] NULL 추가
출력 / 종료

strcat

```
1   #include<stdio.h>
2   #include<string.h>
3   int main()
4   {
5       char destStr[20] = "Hello";
6       char srcStr[10] = " world!";
7       int srcLen, destLen;
8       destLen = strlen(destStr);
9       srcLen = strlen(srcStr);
10
11      _asm
12      {
13          LEA ESI, srcStr
14          LEA EDI, destStr
15          ADD EDI, destLen // 문자열 마지막 주소로 이동
16          MOV ECX, srcLen
17          REP MOVS BYTE PTR[EDI], BYTE PTR[ESI]
18          // ECX 크기 만큼 [ESI] 를 [EDI]로 복사
19          MOV BYTE PTR[EDI] , 0x00
20      }
21
22      printf("%s\n", destStr);
23      return 0;
24  }
```

EDI = 문자열의 마지막주소
ECX = 붙여넣을 문자열 크기

반복하면서 EDI, ESI증가.

[strcat 어셈블리 소스]

Q&A