컴퓨터시스템개론 Lab2 보고서(report)

전공: 수학과 학년: 3학년 학번: 20191274 이름: 장유빈

2-1

'problem1.bin'의 어셈블리 코드를 하나하나씩 차근차근 분석하자.

```
sub     $0x28,%rsp
```

이 부분은 스택에 40바이트 할당하는 것이다.

```
mov     $0x402004,%edi
```

puts 함수호출시 들어갈 첫 번째 인자다. 내용은 "Provide your input:"이다.

```
callq  0x401030 <puts@plt>
```

puts 함수를 호출한다.

```
mov     %rsp,%rsi
mov     $0x402018,%edi
mov     $0x0,%eax
callq  0x401040 <__isoc99_scanf@plt>
```

세 번째 줄의 코드는 'int temp1 = 0;'으로 이해했다.

그리고 1, 2번째 줄이 scanf 함수호출 시 들어갈 인자를 셋업하는 것이다.

여기서 알 수 있는 점은 %rsp의 값이 buf의 시작 주소 값임을 알 수 있다.

```
mov     $0x0,%eax
```

'int temp1 = 0;'으로 이해했다.

```
movslq %eax,%rdx
```

'long temp2 = temp1 (= 0);'으로 이해했다.

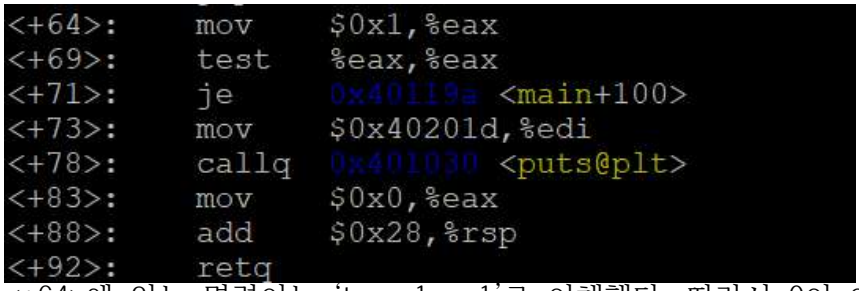
```
movzbl 0x404038(%rdx),%edx
```

현재 temp2의 값이 0이므로 0x404038 주소에 있는 값은 temp2에 대입하는 것임을 알 수 있다. 'x/1xb 0x404038' 명령어를 써서 살펴보면 '0x43'으로 대문자 'C'가 있는 것을 알 수 있다. 또한 msg라는 문자열의 시작부분임을 알 수 있다. 따라서 temp2에 대문자 'C'를 대입하는 것이다.

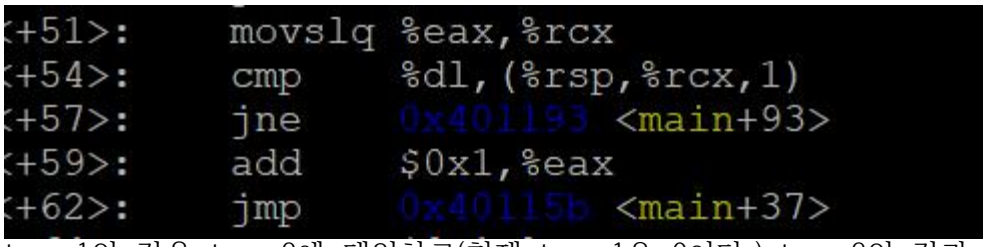
```
test   %dl,%dl
je     0x401176 <main+64>
```

그런데, temp2가 0이 되는 상황은 '0x404038 + %rdx' 주소에 있는 값이 널문자가 되는 상황일 것이다. 먼저 0이라고 가정해보자. 그러면 <main+64>로 점프할 것이다.

-->> "다음 장에 계속됩니다..!!"



<+64>에 있는 명령어는 'temp1 = 1'로 이해했다. 따라서 0이 아니므로 점프를 안할 것이다. 'x/s 0x40201d' 명령어를 통해 출력해보면 성공 문자열이라는 것을 알 수 있고 'return 0;'를 할 준비를 하고 스택 영역을 반환하고 함수를 내가 원하는 대로 종료하는 것을 알 수 있다.
다시 돌아와서 temp2가 'C'인 상황을 살펴보자.



temp1의 값을 temp3에 대입하고(현재 temp1은 0이다.) temp2의 값과 buf[temp3]을 비교한다. 즉, msg의 첫 문자와 우리가 입력한 문자열의 첫 문자를 비교하는 것이다. 같다고 가정해보자. 같으면 temp1에 1을 더한다. 그리고 <main+37>로 점프한다. 그러면, 우리가 위에서 봤던 명령어들이 다시 보일 것이다. 그러면 temp2에 'msg+1'의 내용을 대입하게 될 것이고, 즉, 다음은 두 번째 문자를 서로 비교하는 것임을 알 수 있다. 즉, msg의 널 문자가 나올 때까지 같다면 성공 메시지를 보내는 것이다. 따라서 우리는 msg에 저장된 문자열과 반드시 같은 문자를 입력해야 한다.
'x/1xb 주소' 명령어를 통해서 널 문자가 나올 때까지 확인해본 결과 "CSE3030@Sogang" 문자열임을 알 수 있고 우리는 이 문자열을 입력하면 됩니다..!!

2-2

problem2.bin의 소스파일을 봐보자.

<+0> 명령어는 스택에 24바이트 할당하는 것이다.

<+4> 명령어는 puts함수호출을 하는 것이고 입력을 하라는 의미이다.

다음은 x, y, z 세 정수를 입력받는 어셈블리 코드이다.

```
lea    0x4(%rsp),%rcx
lea    0x8(%rsp),%rdx
lea    0xc(%rsp),%rsi
mov    $0x402018,%edi
mov    $0x0,%eax
callq  0x401040 <__isoc99_scanf@plt>
```

C코드와 이 코드의 비교를 통해 '0xc(%rsp)'에 x가 의미하고, '0x8(%rsp)'에 y가 할당되고, '0x4(%rsp)'에 z가 할당되는 것을 알 수 있다.

5번째 줄은 'int temp1 = 0;'으로 이해했다.

```
 <+44>:     mov    0xc(%rsp),%edx
```

이 코드는 'int temp2 = x;'로 이해했다.

```
cmp    $0x40,%edx
jle    0x4011be <main+136>
```

이 코드는 x와 60을 비교하는 코드이다. x가 60보다 큰 경우를 생각하자.

```
mov    $0x1,%ecx
cmp    $0x60,%edx
jle    0x40117a <main+68>
```

첫 번째 줄은 'int temp3 = 1;'로 이해했다.

두 번째 줄은 x와 96을 비교하는 것이다. x가 96보다 작은 경우를 생각해보자. 그렇다면 <main+68>로 점프할 것이다. 점프해보자.

```
a <+68>:    mov    0x4(%rsp),%eax
e <+72>:    cmp    $0x200,%eax
3 <+77>:    jg     0x40118a <main+84>
```

첫 번째 줄은 'temp1 = z;'로 이해했다.

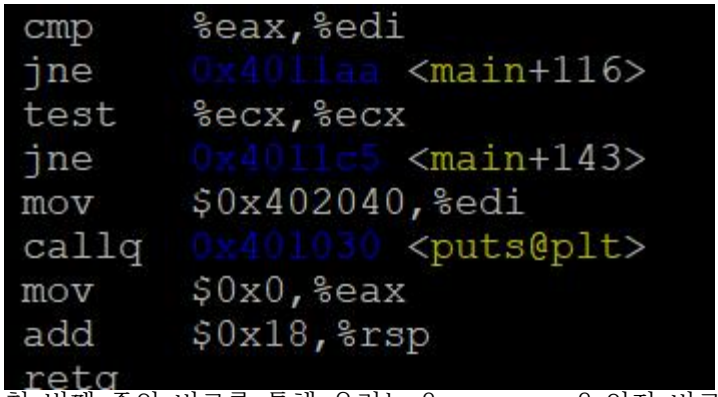
두 번째 줄은 z와 512를 비교하는 것이다. 크다고 가정해보자. 그러면 <main+84>로 점프할 것이다. 점프해보자.

```
<+84>:    cmp    $0x230,%eax
<+89>:    jle    0x401196 <main+96>
```

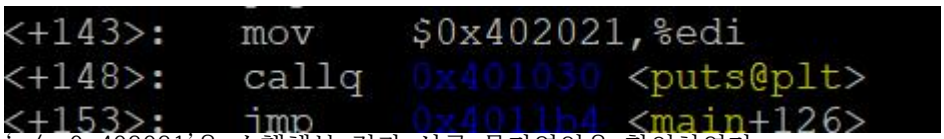
z와 560을 비교하는 것이다. 작거나 같다고 가정하고 <main+96>으로 점프해보자.

```
<+96>:    mov    0x8(%rsp),%esi
<+100>:   mov    %esi,%edi
<+102>:   sub    %edx,%edi
<+104>:   sub    %esi,%eax
<+106>:   add    %eax,%eax
```

이 4줄을 통해 %eax의 값은 2*(z-y)가 되고, %edi는 y-x가 될 것이다.



첫 번째 줄의 비교를 통해 우리는 3*y == x + 2z인지 비교하는 것이다.
같다고 가정하고 3번째 줄로 가보자. 아까 temp3는 1로 설정했으므로, <main+143>으로 점프할 것이다. 점프해보자.



'x/s 0x402021'을 수행해본 결과 성공 문자열임을 확인하였다.
나머지 부분은 스택을 반환하고 'return 0;'을 하기위한 설정을 하고 ret하는 것이다.
따라서, 결론적으로 우리는 '64 < x <= 96', '512 < z <= 560', 'x + 2*z == 3*y'를 만족하는 정수를 입력으로 넣어야 한다. 나는 (80, 400, 560)을 넣을 것이다.
확인해본 결과 정상적으로 수행되는 것을 확인할 수 있다.

2-3

problem3.bin의 assembly code를 살펴보자.

첫 두 줄은 Callee-saved registers를 백업 해놓는 작업을 하는 것이다.

<+2>는 스택에 24바이트 할당하는 것이다.

```
<+6>:     mov    $0xa,%ebp
<+11>:    mov    $0x0,%ebx
```

첫 번째 줄은 'int temp1 = 10;' 그리고 두 번째 줄은 'int temp2 = 0;'으로 이해했다.

```
jmp    0x40114e <main+24>
```

<main+24>로 점프해보자.

```
<+24>:    cmp    $0x2,%ebx
<+27>:    jg     0x401198 <main+98>
```

temp2가 2보다 크면 <main+98>로 점프한다. 일단 2보다 큰 상황을 가정하고, <main+98>로 점프해보자.

```
<+98>:    cmp    $0x191,%ebp
<+104>:   je     0x4011b6 <main+128>
```

temp1이 401과 같은지 비교하고 있다. 같다고 가정해보고 <main+128>로 점프해보자.

```
<+128>:   mov    $0x40201b,%edi
<+133>:   callq  0x401030 <puts@plt>
<+138>:   jmp    0x4011aa <main+116>
```

'x/s 0x40201b'를 해본 결과 성공 메시지인 것을 확인하였거 따라서 우리는 temp1을 401로 만들어야하는 의무가 생김을 알았다.

다시 temp2가 0이라고 가정해보자.

<+29>와 <+34>는 입력하라는 메시지를 출력하는 것이다.

```
<+39>:    lea    0xc(%rsp),%rsi
<+44>:    mov    $0x402018,%edi
<+49>:    mov    $0x0,%eax
<+54>:    callq  0x401040 <__isoc99_scanf@plt>
```

위의 코드를 통해 우리는 0xc(%rsp)에 x가 할당된 것을 알 수 있다. 위의 코드는 x를 입력받는 명령어이다.

```
<+59>:    mov    0xc(%rsp),%eax
```

'int temp3 = x;'로 이해했다.

```
cmp    $0x7,%eax
ja     0x40114b <main+21>
```

위의 코드는 x가 7보다 큰지 비교하는 명령어이다.

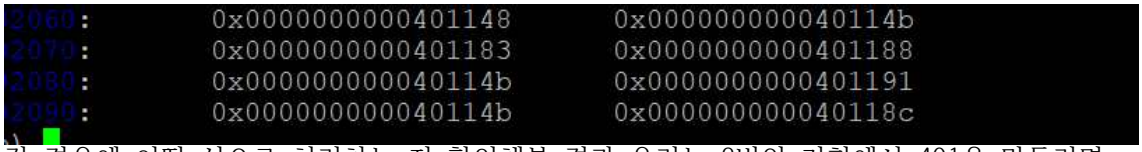
```
<+21>:    add    $0x1,%ebx
```

커버리면 아무런 동작도 없이 temp2만 1 커지는 꼴이 되어버리므로 우리가 입력해야 하는 정수는 0 ~ 7임을 알 수 있다.

```
<+70>:    jmpq   *0x402060(,%rax,8)
```

점프 테이블이 있는 것으로 보아 switch문임을 알 수 있다.

0 ~ 7까지 8개의 경우이므로 각 경우마다 8바이트 씩이므로, ‘x/8xg 0x402060’ 명령어를 입력해 보았다.



각 경우에 어떤 식으로 처리하는 지 확인해본 결과 우리는 3번의 기회에서 401을 만들려면 처음에 x에 3을 입력하여 20을 만들고, 그다음 입력 때 x에 7을 입력하여 400을 만들고, 마지막 입력 때 x에 0을 입력하여 401을 만들어야 한다. 따라서, 우리는 순서대로 3, 7, 0을 x에 입력해야 한다.

2-4

마지막 문제인 2-4는 주어진 어셈블리 코드를 통해 제가 핵심 부분만 c언어로 복원한 코드를 보여드리겠습니다. 편의상 알파벳 등장 여부 카운팅 배열의 이름을 스택포인터의 이름을 따 rsp라 가정했습니다. 수도 코드를 제가 구현해보았다고 생각하시면 될 것 같습니다. 문제의 조건을 만족하기 위한 경우만 고려하여 수도 코드를 제작했다고 생각하시면 될 것 같습니다.

```
int main(void)
{
    char buf[32];
    long temp1 = 0; // %eax
    // ....
    // ....
    // ....
    int n = strlen(buf); // %ebx

    if (11 == n) // 다른 경우 만족 못함.
    {
        int temp2 = 1; // %ebp
        memset(%rsp, 0, 26); // 알파벳 등장 여부 체킹하는 배열
        int temp3 = 0; // %esi
        int temp4 = 0; // %ecx
        while (1)
        {
            temp4 += 1;

            if (temp4 >= n) // 입력한 문자열을 한바퀴 다 순회했다면,
            {
                if (temp3 == 5) // temp3의 값이 5가 되야함.
                {
                    if (temp2 != 0) // temp2의 값이 1이어야함.
                    {
                        // Success..!! <--- "여길 와야 합니다..!!"
                    }
                }
            }
            else if (temp4 < n) // 입력한 문자열 아직 순회 중
            {
                temp1 = buf[temp4]; // %rax
                temp5 = temp1 - 97; // %edx, 물론 97이상이어야 함..!!
                if (temp5 <= 25)
                {
                    temp5 = n - temp4 - 1;
                    if (buf[temp5] == buf[temp4])
                    {
                        temp5 = buf[temp4] - 97;

                        if (rsp[temp5] != 0)  // 편의상 알파벳 등장 여부 체킹 배열을 rsp라 하겠음.
                        {
                            continue;
                        }
                        else if (rsp[temp5] == 0)
                        {
                            rsp[temp5] = 1;
                            temp3 += 1; // 여기를 5번 와야함..!!
                        }
                    }
                }
            }
        }
    }

    return 0;
}
```

이 코드를 통해서 알 수 있듯이 소문자 알파벳으로 구성된 길이 11인 문자열을 입력해야 합니다. 또한 이 코드를 통해서 우리는 다음과 같은 사실을 알 수 있습니다.

입력한 문자열을 한 바퀴 순회하면서 ‘temp3 = 5’가 될려면 “abcddeddbca”와 같은 문자열

을 입력해야 합니다. 앞의 “abcd” 부분에서 temp3이 4가 되고, “e”에서 temp3이 5가 되면서 문제의 조건을 만족하기 때문입니다. 물론 여러 예는 많겠지만, 저는 “abcddeddbca”를 선택했습니다.

이상입니다.
읽어주셔서 감사드리고 좋은 강의 항상 감사드립니다.
-20191274 장유빈-