CS作业

> 丁海桐——软件 2203——202226010304

3. 34：

- `%ebx` 储存的是传入的参数 `x`

```
int rfun(unsigned x) {
        if (!x) return 0;
        unsigned nx = x >> 1,
        int rv = rfun(nx);
        return (x & 1) + rv;
}
```

- 此函数的作用是统计 `x` 中 `1` 的个数

3. 56：

```
        movl 8(%ebp), %esi        //%esi = x
        movl 12(%ebp), %ebx       //%ebx = n
        movl $1431655765, %edi    //%edi = 1431655765
        movl $-2147483648, %edx   //%edx = -2147483648
.L2:
        movl %edx, %eax           //%eax = edx = -2147483648  //int result = -2147483648 & x
        andl %esi, %eax           //%eax = -2147483648 & x
        xorl %eax, %edi           //%edi = 1431655765 ^ (-2147483648 & x)
        movl %ebx, %ecx           //%ecx = n;
        shrl %cl, %edx            //%edx = -2147483648 >> (n & 0xff)
        testl %edx, %edx
        jne  .L2
        movl  %edi, %eax
```

```
result -> %edi; mask -> %edx
```

1. `result = 1431655765, mask = -2147483648`
2. `mask != 0`
3. `mask = (unsigned)(mask >>(n & 0xff))`
4. `result ^= (mask & x)`

```
int loop(int x, int n)
{
        int result = 1431655765 ;
        int mask;
        for (mask = -2147483648; mask != 0 ;mask = mask >>((unsigned)n & 0xff); ) {
```

```
                result ^= (mask & x);
            }
            return result;
}
```

3. 59：

```
1   08048420 <switch_prob>:
2    8048420:  55                    push   %ebp
3    8048421:  89 e5                 mov    %esp,%ebp
4    8048423:  8b 45 0c              mov    0xc(%ebp),%eax
5    8048426:  83 e8 28              sub    $0x28,%eax
6    8048429:  83 f8 05              cmp    $0x5,%eax
7    804842c:  77 07                 ja     8048435 <switch_prob+0x15>
8    804842e:  ff 24 85 f0 85 04 08  jmp    *0x80485f0(,%eax,4)
9    8048435:  8b 45 08              mov    0x8(%ebp),%eax
10   8048438:  eb 24                 jmp    804845e <switch_prob+0x3e>
11   804843a:  8b 45 08              mov    0x8(%ebp),%eax
12   804843d:  8d 76 00              lea    0x0(%esi),%esi
13   8048440:  eb 19                 jmp    804845b <switch_prob+0x3b>
14   8048442:  8b 45 08              mov    0x8(%ebp),%eax
15   8048445:  c1 e0 03              shl    $0x3,%eax
16   8048448:  eb 17                 jmp    8048461 <switch_prob+0x41>
17   804844a:  8b 45 08              mov    0x8(%ebp),%eax
18   804844d:  c1 f8 03              sar    $0x3,%eax
19   8048450:  eb 0f                 jmp    8048461 <switch_prob+0x41>
20   8048452:  8b 45 08              mov    0x8(%ebp),%eax
21   8048455:  c1 e0 03              shl    $0x3,%eax
22   8048458:  2b 45 08              sub    0x8(%ebp),%eax
23   804845b:  0f af c0              imul   %eax,%eax
24   804845e:  83 c0 11              add    $0x11,%eax
25   8048461:  5d                    pop    %ebp
26   8048462:  c3                    ret
```

```
(gdb) x/6w 0x80485f0
0x80485f0: 0x08048442 0x08048435 0x08048442 0x0804844a
0x8048600: 0x08048452 0x0804843a
```

```
int switch_prob(int x, int n)
{
    int result = x;
    switch(n)
    {
        case 0x28:
        case 0x2a:
            result <<= 3; break;
        case 0x2b:
            result >>= 3; break;
        case 0x2c:
            result <<= 3;
            result -= x;
        case 0x2d:
            result *= result;
        case 0x29: //也可以不要
```

```
        default:
            result += 0x11;
    }
    return result;
}
```

3. 66:

```
1   typedef struct {
2       int left;
3       a_struct a[CNT];
4       int right;
5   } b_struct;
6
7   void test(int i, b_struct *bp)
8   {
9       int n = bp->left + bp->right;
10      a_struct *ap = &bp->a[i];
11      ap->x[ap->idx] = n;
12  }
```

编译时常数 CNT 和结构 a_struct 的声明在一个你没有访问权限的文件中。幸好，你有代码的'.o'版本，可以用 OBJDUMP 程序来反汇编这些文件，得到如图 3-45 所示的反汇编代码。

```
1   00000000 <test>:
2      0:   55                      push   %ebp
3      1:   89 e5                   mov    %esp,%ebp
4      3:   53                      push   %ebx
5      4:   8b 45 08                mov    0x8(%ebp),%eax
6      7:   8b 4d 0c                mov    0xc(%ebp),%ecx
7      a:   6b d8 1c                imul   $0x1c,%eax,%ebx
8      d:   8d 14 c5 00 00 00 00    lea    0x0(,%eax,8),%edx
9     14:   29 c2                   sub    %eax,%edx
10    16:   03 54 19 04             add    0x4(%ecx,%ebx,1),%edx
11    1a:   8b 81 c8 00 00 00       mov    0xc8(%ecx),%eax
12    20:   03 01                   add    (%ecx),%eax
13    22:   89 44 91 08             mov    %eax,0x8(%ecx,%edx,4)
14    26:   5b                      pop    %ebx
15    27:   5d                      pop    %ebp
16    28:   c3                      ret
```

图 3-45　家庭作业 3.66 的反汇编代码

运用你的逆向工程技术，推断出下列内容：

A. CNT 的值。

B. 结构 a_struct 的完整声明。假设这个结构中只有字段 idx 和 x。

```
%ecx -> bp
%eax -> i
%ebx = 0x1c * i -> &a[i]                              //sizeof(a[i]) = 0x1c
%edx = 7 * i + 4 + bp + 0x1c * i = bp + 4 + 0x23 * i //sizeof(x[i]) = 7
bp->right = bp + 0xc8                                 //sizeof(a) = 0xc4 = 196  //CNT = 7
&ap->x[ap->idx] = 0x8 + bp + 4 * %edx
```

```
typedef b_struct = {
        int           left;
        a_struct[CNT] a;
```

```
        int           right;
}

typedef a_struct = {
    int            idx;
    c_struct[CNT_1]   x;
}
```

```
根据11行可得：&bp->right = bp + 0xc8
根据12行可得：&bp->left  = bp
//所以sizeof(a) = 0xc8 - 0x4 = 0xc4 = 196

根据13行可得：&bp->a[i]->x[idx] = bp + 4 * (7 * i + (4 + bp + 0x1c * i)) + 8 ,   这里的8一半分给了b_struct->left, 一半分给了a[i]->idx
                                = bp + 4 + i * sizeof(a_struct) + 4 + idx * sizeof(c_struct)
                                = bp + 4 + 28 * i + 4 + idx * 4
根据10行可得：%edx = (0x4 + bp + 0x1c * i) + 7 * i
                  = (&bp->a[i])           + 7 * i
                  = (&bp->a[i]->idx)      + 7 * i
                  = bp->a[i]->idx         + 7 * i
//所以sizeof(a_struct) = 0x1c = 28, CNT = 0xc4 / 0x1c = 7
//所以sizeof(c_struct) = 28 - 4 = 24
```

- CNT = 7

```
typedef a_struct{
        int idx;
        int x[6];
}
```