x64逆向：基本运算和分支结构

通用找main的方法：从start开始进去，第二个call进去找exit函数，一般第一个exit函数上一个函数就是main。高版本vs比较特殊，在exit上面在第二个函数（注意main有三个参数）

基本运算

1. 加减法
   - `lea`比例因子寻址代替`add、sub`，同x86一样
2. 乘法
   - 在x86上优化成多条指令的情况中，x64会直接用`imul`
3. 除法
   - 优化公式

> 在x86下，指数为32。在x64下，指数为64。
> 以下统一以32为例：
>
> 1. 无符号2的幂
>
> ```
> x / 2^n = x >> n
> ```
>
> 2. 无符号非2的幂1
>
> ```
> x / c = x * M >> 32 >> n
> ```
>
> 3. 无符号非2的幂2
>
> ```
> x / c = (((x - (x * M >> 32)) >> n1) + (x * M >> 32)) >> n2
> ```
>
> 4. 有符号正2的幂
>
> ```
> x >= 0
>     x / 2^n = x >> n
> x < 0
>     x / 2^n = (x + (2^n - 1)) >> n
> ```
>
> 5. 有符号负2的幂
>
> 同情况1，`-(x / 2^n)`
>
> 6. 有符号正非2的幂1
>
> ```
> x >= 0
>     x / c = x * M >> 32 >> n
>     此时，M > 0
> x < 0
>     x / c = (x * M >> 32 >> n) + 1
> ```
>
> 7. 有符号正非2的幂2
>
> ```
> x >= 0
>     x / c = (x * M >> 32) + x >> n
> x < 0
>     x / c = ((x * M >> 32) + x >> n) + 1
> ```
>
> 8. 有符号负非2的幂1
>
> ```
> x >=0
>     x / c = x * M >> 32 >> n
> ```

```
            此时，M < 0
        x < 0
            x / c = (x * M >> 32 >> n) + 1

    9. 有符号负非2的幂2
        x >= 0
            x / c = (x * M >> 32) - x >> n
        x < 0
            x / c = ((x * M >> 32) - x >> n) + 1

计算除数c，其中n为移位总次数：
    1. c = 2^n / M
        c > 0 时，使用
    2. c = 2^n / (2^32 - M)
        c < 0 时，使用
    3. c = 2^n / (2^32 + M)
        乘减移加移时，使用
```

- 总结

```
重要公式有三：
    1. x * M >> 32 >> n
        M > 0 时，x / c
        M < 0 时，x / -c
    2. (x * M >> 32) + x >> n
        x / c
    3. (x * M >> 32) - x >> n
        x / -c
```

4. 取模

- 优化公式

```
1. 模2的幂
    x >= 0
        x % 2^n = x & (2^n - 1)
    x < 0
        x % 2^n = (x & (2^n - 1)) - 1 | (~(2^n - 1)) + 1，老公式
        x % 2^n = (x + (2^n - 1) & (2^n - 1)) - (2^n - 1)，新公式
            例如： x % 8 = (x + 7 & 7) - 7

2. 模非2的幂
    x % c = x - x / c * c
        余数 = 被除数 - 商 * 除数
```

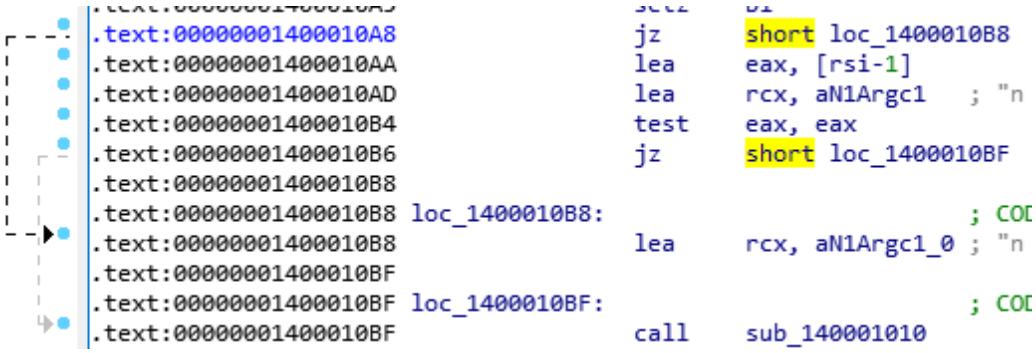
5. 三目运算

- `x == 0 ? 8 : 9` 这种两值相差一的，利用 `sete reg` 条件设置指令
- 其他情况利用 `cmove` 条件传送指令

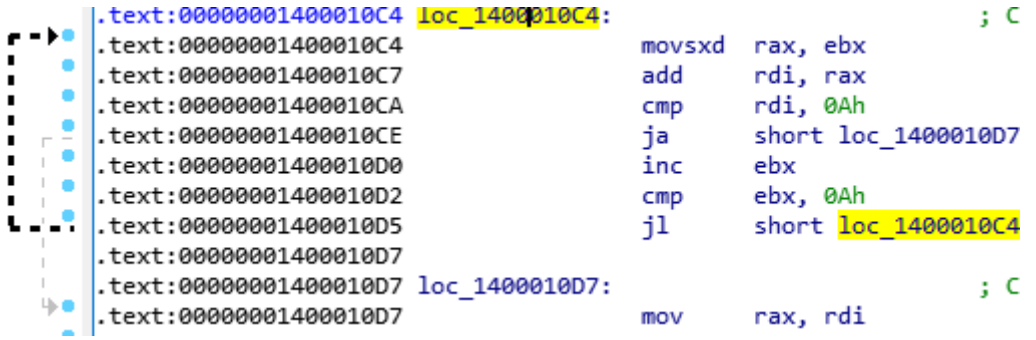
分支结构

看图IDA和x64dbg的跳转线条，实线是`jmp`，虚线是条件跳，中间包含的圆点就是汇编语句，注意线的开头有个判断

1. `if`
    - 单分支`if`，虚线
    - `if-else`，虚线`if`+实线`else`，两部分有重叠
    - `&&`相当于`if`的嵌套
    - 两个虚线交叉是`||`，满足条件1跳，不满足条件2走



2. 循环
    - `do-while`，线条向上
    - `while`优化成`do-while`
    - `for`优化为`if + do-while`
    - `break`线条直接跳出
    - `continue`线条上跳



3. `switch`
    - 线太多，不看线看特征