2021/03/09_x86逆向_第4课_基本运算(除法相关定理和推理)

笔记本: x86逆向-C
创建时间: 2021/3/9 星期二 9:44
作者: ileemi

相关定理和推理

- 定理1
  若x为实数，有:
  (x 向下取整) <= x，且(x 向上取整) >= x
  推导1:
  x -1 < (x) 向下取整 <= x <= (x) 向上取整 < x + 1
  推导2:
  x < (x) 向下取整 + 1
  推导3（当x不为整数时）:
  (x) 向上取整 = (x) 向下取整 + 1
- 定理2
  若x为整数，则:
  (x) 向下取整 = x，且 **(x) 向上取整 = x**
- 定理4
  若x为实数，n为整数，有:
  (x + n) 向下取整 = (x) 向下取整 + n
  (x + n) 向上取整 = (x) 向上取整 + n

无符号整数除以2的幂

> 定式：**shr reg, n**

代码示例:

```
// unsigned int argc
printf("%d\n", argc / 4);
// 对应的汇编代码 Release
mov eax, [esp+argc]
shr eax, 2
```

有符号整数除以2的幂

> 定式1（除数等于2时）：
> **cdq**
> **sub eax, edx**
> **sar eax, 1**

代码示例：

```
// int argc
printf("%d\n", argc / 2);
// 对应的汇编代码 Debug
上层转下层：等价除以2
```

edq：符号扩展，将 eax 最高位填充到 edx 中，如果 eax 的最高位（符号位）为1，则 edx 的值就为 0xFFFFFFFF，也就是 -1，否则为 0。

> 定式2：除数是大于2，且数值为2的幂
> **cdq**
> **and edx, imm ; imm = 2^n -1**
> **add eax, edx**
> **sar eax, n**

代码示例：

```
// int argc
printf("%d\n", argc / 4);
```

> 若能证明 2^n - 1 等于 imm，则除数为：**2的n次方**，也就是 **被除数（有符号数整型） / 2^n**

C语言的右移位运算（无符号）相当于数学除法并求下整（向下取整）运算：
(x / 8 向下取整) = x >> 3
负数进行移位操作需要进行调整（负数区间）：
除法：10 / 8 = 1，移位：10 >> 3 = 1
除法：-10 / 8 = -1，移位：-10 >> 3 = -10+8-1 >> 3 = -1

无符号整数除以非2的幂

> 如果除数是变量，则只能使用除法指令，只要除数为常量该程序的汇编代码就有优化的空间。根据除数的数值相关特性，编译器有对应的处理方式。

> 无符号变量除以常量定式：
> mov eax, MgcNum
> mul [rsg+argc]
> shr edx, 2

在乘法指令中，由于edx存放乘积数据的高4字节，所以直接使用edx就相当于乘积右移了32位，再算上 shr edx, 2 就相当于移动移动了34位。

除数 = (2^(32+2) / MgcNum) 向上取整
代码示例：

```
// unsigned int argc
printf("%d\n", argc / 5);

// Release对应汇编
mov eax, 0CCCCCCCDh ; 3435973837
mul [esp+argc]
shr edx, 2 ; 2^34 =

除数 = (17179869184 / 3435973837 ) 向上取整 = (4.9999...) 向上取整 = 5
```

Debug：没有优化
Release：不使用div相关除法指令，优先考虑使用其他运算指令进行代替。