### 2021/03/10 x86逆向 第5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:** 2021/3/10 星期三 10:08

作者: ileemi

- 无符号除法
  - 无符号除以2的幂
  - 无符号除以非2的幂
    - <u>MagicNumber无进位</u>
    - <u>MagicNumber有进位</u>
- 有符号除法
  - 除以正数
    - 除以2的幂 -- 除数为正
    - <u>除以非2的幂(除数为正) -- MagicNumber 为正数</u>
    - <u>除以非2的幂(除数为正) -- MagicNumber 为负数</u>
  - 除以负数
    - 除以2的幂 -- 除数为负
    - <u>除以非2的幂(除数为负) -- MagicNumber为正</u>
    - <u>除以非2的幂(除数为负) -- MagicNumber为负</u>
  - 当满足除以常量的定式时,要判定以下情况

# 无符号除法

- 1. 除以正数
  - 1.1 除以2的幂
  - 1.2 除以非2的幂

MagicNumber无进位

MagicNumber有进位

# 无符号除以2的幂

除数为2的幂,定式如下:

除数: 2<sup>N</sup> mov reg, 被除数 shr reg, N

## MagicNumber无进位

### 有以下定式:

```
mov eax, MagicNumber
mul 被除数
shr edx, N

除数 = ceil(2^(32+N) / MagicNumber)
```

## MagicNumber有进位

定式如下: 乘、减、移、加、移

```
mov reg, 被除数
mov eax, MagicNumber
mul reg
sub reg, edx
shr reg, N1
add reg, edx
shr reg, N2

// 求除数:

除数 = ceil(2^(32+N2+N1) / (2^32 + M))
```

#### 无符号整型变量除以常量7(以及7的倍数)

代码示例:

```
printf("%d\n", (unsigned int )argc / 7);
// 对应的汇编代码 (Release)
// MagicNumber 是有进位情况的,需要加上 2^32 (100000000H)
// 右移的次数需要相加
mov ecx, [esp+argc]
mov eax, 24924925h
mul ecx
sub ecx, edx
shr ecx, 1
add ecx, edx
shr ecx, 2
push ecx

MsgNum = 100000000H + 24924925H = 124924925H = 4908534053
```

## 求除数: $(2^{32} + 1 + 2) / 4908534053)$ 向上取整 = 7

#### 推导过程:

设 c 为常量 24924925h, 以上代码等价于:

. text:0040108B 处的 sub ecx, edx 直接减去了乘法结果的高 32 位, 数学表达式等价

$$\mp ecx - \frac{ecx * c}{2^{32}};$$

其后 shr ecx, 
$$1$$
 相当于是除以  $2$ : 
$$\frac{ecx - \frac{ecx * c}{2^{32}}}{2}$$
:

其后 add ecx,edx 再次加上乘法结果的高 32 位:  $\frac{ecx - \frac{ecx*c}{2^{32}}}{2} + \frac{ecx*c}{2^{32}};$ 

其后 shr ecx, 
$$2$$
 等价于把加法的结果再次除以  $4$ : 
$$\frac{ecx-\frac{ecx*c}{2^{32}}}{2}+\frac{ecx*c}{2^{32}}$$

最后直接使用 ecx, 乘法结果低 32 位 eax 弃而不用。

先简化表达式:

$$\frac{ecx - \frac{ecx * c}{2^{32}}}{2} + \frac{ecx * c}{2^{32}} = > \frac{2^{32} * ecx - ecx * c}{2^{33}} + \frac{ecx * c}{2^{32}} = > \frac{2^{32} * ecx - ecx * c + 2 * ecx * c}{2^{35}} = > \frac{2^{32} * ecx + ecx * c}{2^{35}} = > \frac{ecx * (2^{32} + c)}{2^{35}} = > \frac{e$$

为了规避大数运算,编译器的作者其实是将结果变形成了上图中的最初式子。**所以在** 还原代码时,如果汇编代码中遇到 "乘、减、右移、加、右移" 运算指令序列的,基 本可以判定时除法优化后的代码,其除法原型为 "a" 除以常量 "o", "mul" 可表明是 无符号计算,其操作数是优化前的被除数 "a",接下来统计右移的次数以便确定公式 中的 "n" 值, 然后使用公式 "o = 2^(32+n) / 2^32 + c" 并将 "MagicNumber" 做为 c 值带入公式求解常量除数 o (运算结果后需要向上取整) , 即可恢复除法原 型。

公式: o = 2^(32+n) / 2^32 + c

除数 2^32 + c 表示 2^32 + MagicNumber, 接就是在无符号 MagicNumber 最高位前添加一个1参与后面的运算。

# 有符号除法

- 1. 除以正数
  - 1.1 除以2的幂
  - 1.2 除以非2的幂

MagicNumber为正数

MagicNumber为负数

- 2. 除以负数
  - 2.1 除以2的幂
  - 2.2 除以非2的幂

MagicNumber为正数

MagicNumber为负数

# 除以正数

### 除以2的幂 -- 除数为正

除数为2时,有以下定式:

```
mov eax,被除数
cdq
sub eax,edx
sar eax,1
```

### 除数为2^N时,有以下定式:

```
mov eax,被除数
cdq
and edx,2^N - 1
add eax,edx
sar eax,N
```

# 除以非2的幂(除数为正) -- MagicNumber 为正数

当汇编代码中出现 "**乘、右移、mov、右移、加**" 运算,按照正常方法进行还原,后面的 "mov、右移、加" 下整转上整的三条指令不用管,按照之前的定式求除数即可。

### 定式1:MagicNumber 为正时,下整转上整,调整代码不做还原

```
mov eax, MagicNumber // MagicNumber 转十进制 imul 被除数 sar edx, N mov reg, 被除数 // 下整转上整 shr reg, 1FH add edx, reg // 商在edx中,加符号位; usr edx

除数 = ceil(2^(32+N) / MagicNumber)
```

```
x 不为整数:
floor(x) + 1 = ceil(x)

gcc版: 下整转上整
mov eax, edx
cqd
sub eax, edx // -(-1)

mov eax, ecx
sar eax, 1FH // 使用sar高位补符号位 if ecx >= 0, eax = 0; else eax =
FFFFFFFF
sub edx, eax // floor(x) + 1 = ceil(x), edx = edx - (-1)
```

#### 有符号常量除以5

### 代码示例:

```
printf("%d\n", (int)arge / 5);
// 对应的汇编代码
mov ecx, [esp+arge]
mov eax, 66666667h // 1717986919
imul ecx
sar edx, 1
// 负数区间 下整转上整 调整代码不做还原
mov eax, edx
shr eax, 31 ; 获取有符号数的符号位
add edx, eax ; edx + 符号位
push edx
// 求除数:
// (2 (32 +1) / 1717986919) 向上取整 = 5
```

# 除以非2的幂(除数为正) -- MagicNumber 为负数

定式2: 乘法和下整转上整间缺少移位, N值为32, 默认右移32位

```
mov eax, MagicNumber
imul reg
mov eax, edx
shr eax, 1FH // 31
add edx, eax
; usr edx
```

代码示例:

```
printf("%d\n", argc / 3);
// 对应的汇编代码
mov ecx, [esp+argc]
mov eax, 55555556h // 1431655766
imul ecx // edx. eax
// 这里缺少移位,后面的代码有直接使用了 edx
// 说明 N值为32,没有移位就表示默认移32位
// 负数区间 下整转上整 调整代码不做还原
mov eax, edx
shr eax, 1Fh
add edx, eax
push edx
// 求除数:
// (2^(32) / 1431655766) 向上取整 = 3
```

汇编指令中,"乘"下面没有出现右移位情况,后面的汇编指令直接使用edx后,说明其只移动了32位,按照定式进行正常还原即可。

### 定式3:MaigcNumber小于0时, imul和sar之间有针对edx的调整指令

```
mov 被除数, [esp+argc]
mov eax, MagicNumber ; MagicNumber 为负数
imul 被除数
add edx, 被除数 // 有符号整数乘以无符号整数的调整, 调整 edx 的值
sar edx, N
mov eax, edx // 上整转下整
shr eax, 1FH
add edx, eax // 商在edx中
push edx

除数 = ceil(2^(32+N) / MagicNumber)
```

有符号整型变量除以常量7

代码示例:

```
printf("%d\n", argc / 7);

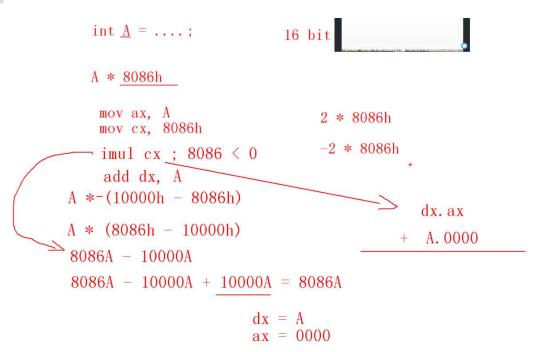
// 对应的汇编代码 (Release)

mov ecx, [esp+argc]

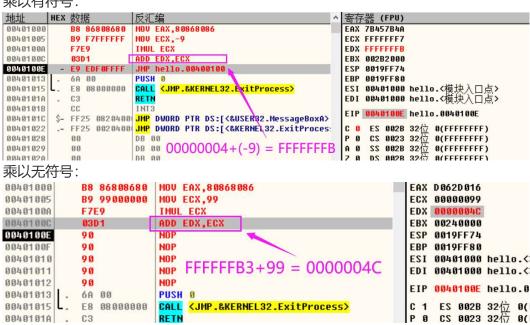
mov eax, 92492493h
```

```
imul ecx
// 所以就需要add + argc 进行调整
add edx, ecx
sar edx, 2
mov eax, edx
shr eax, 1Fh
add edx, eax
push edx
(2^{32+2}) / 2454267027) 向上取整 = 7
```

MaigcNumber大于0时, imul和sar之间没有针对edx的调整指令; 当 MaigcNumber小于0时,imul和sar之间有针对edx的调整指令(最高位加上乘 数),则可以判定除数为正数。这种指令集,按照定式直接还原即可。



#### 乘以有符号:



# 除以负数

### 除以2的幂 -- 除数为负

同正数定式,最后补充对结果求补neg eax。

代码示例:

```
printf("%d\n", argc / -8);

// 对应的汇编代码

mov eax, [esp+argc]
cdq
and edx, 7
add eax, edx
sar eax, 3
neg eax

// 当作正数处理,结果取neg
printf("%d\n", -(argc / 8));
```

### 除以非2的幂(除数为负) -- MagicNumber为正

定式: MagicNumber 为正数且乘法、移位间有对乘积结果高位进行减调整(减去乘数),说明该定式为除法,且除数为负数,该 MagicNumber 是求补后的结果(源MagicNumber 为先结果取反+1)。

```
mov reg, [esp+argc]
mov eax, MagicNumber // MagicNumber为正数
imul reg
sub edx, reg // MagicNumber求补后为正数, 故需要调整
sar edx, N
mov eax, edx // 上整转下整
shr eax, 1Fh
add edx, eax // 商在edx中
push edx

除数 = ceil(2^(32+N) / neg(MagicNumber))
```

#### 代码示例:

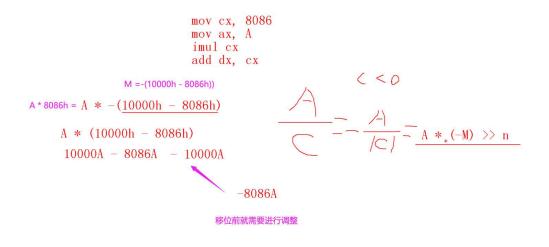
```
printf("%d\n", argc / -7);
// 对应的反汇编代码 (Release)
mov ecx, [esp+argc]
```

```
mov eax, 6DB6DB6Dh
imul ecx
sub edx, ecx
sar edx, 2
mov eax, edx
shr eax, 1Fh
add edx, eax
push edx

// 求除数:

源MagicNumber: (6DB6DB6Dh) NOT + 1 = 92492493H = 2454267027
(2^(32+2) / 2454267027) 向上取整 = 7
| c | = 7, c < 0
c = -7
```

### 定式推导过程:



# 除以非2的幂(除数为负) -- MagicNumber为负

定式:MagicNumber 为负数且乘法、移位间没有进行调整,说明除数为负常量, 且该MagicNumber是求补后的结果。

```
mov reg, [esp+argc]
mov eax, MagicNumber // MagicNumber 为负数
imul reg
sar edx, N // 没有调整直接移位,说明除数为负常量
mov eax, edx
shr eax, 1Fh
add edx, eax
push edx

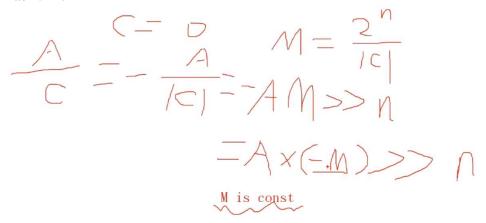
除数 = ceil(2^(32+N) / neg(MagicNumber))
```

代码示例:

```
printf("%d\n", argc / -9);
// 对应的汇编代码 (Release)
mov ecx, [esp+argc]
mov eax, 0C71C71C7h;
imul ecx
sar edx, 1 // 没有调整直接移位,说明除数为负数
mov eax, edx
shr eax, 1Fh
add edx, eax
push edx

// 求除数c, c < 0
源M: (0C71C71C7h) NOT + 1 = 38E38E38H + 1 = 38E38E39H
( 2 (32+1) / 954437177) 向上取整 = 9
| c | = 9, c < 0
c = -9
```

#### 定式推导过程:



源M: (-M) NOT + 1

# 当满足除以常量的定式时, 要判定以下情况

- 1. MaigcNumber 大于0, imul 和 sar 之间没有针对 edx 的调整指令; MaigcNumber 小于0, imul 和 sar 之间有针对 edx 的加乘数的调整,则判定除数为正数。
- 2. MaigcNumber 小于0, imul 和 sar 之间没有针对 edx 的调整指令; MaigcNumber 大于0, imul 和 sar 之间有针对 edx 的减乘数的调整,则判定 除数为负数。对 MaigcNumber 求补后计算得到除数的绝对值,从而得知源 值。