IA-32中的按位运算指令

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

开场白

上一节课主要介绍了**IA-32**指令系统中的定点算术运算指令，本节课接着介绍位操作指令。

同样，所有**IA-32**指令的细节内容不需要记忆，只要用到某条指令时，会查手册并理解手册中所描述的内容。

IA-32常用指令类型

（3）按位运算指令

- **逻辑运算**

  NOT：非，包括 notb、notw、notl等

  AND：与，包括 andb、andw、andl等

  OR：或，包括 orb、orw、orl等

  XOR：异或，包括 xorb、xorw、xorl等

  TEST：做“与”操作测试，仅影响标志

仅NOT不影响标志，其他指令OF=CF=0，而ZF和SF则根据结果设置：若全0，则ZF=1；若最高位为1，则SF=1

逻辑运算指令举例

- 假设:

M[0x1000]=00000F89H

M[0x1004]=00001270H

R[eax]=FF000001H

R[ecx]=00001000H

说明以下指令的功能

notw %ax

andl %eax, (%ecx)

orb 4(%ecx), %al

xorw %ax,4(%ecx)

testl %eax, %ecx

指令执行结果如下：

notw %ax

R[ax]=not(0001H)=FFFEH

andl %eax, (%ecx)

M[0x1000]=00000F89H∧FF000001H

=00000001H

orb 4(%ecx), %al

R[al]=01H∨70H=71H

xorw %ax,4(%ecx)

M[0x1004]=1270H⊕0001H

=1271H

testl %eax, %ecx

不改变寄存器和存储单元的内容

因为 00001000H∧FF000001H=0

故 ZF=0

IA-32常用指令类型

（3）按位运算指令

- 移位运算（左/右移时，最高/最低位送CF）

SHL/SHR: 逻辑左/右移，包括 shlb、shrw、shrl等

SAL/SAR: 算术左/右移，左移判溢出，右移高位补符

（移位前、后符号位发生变化，则OF=1 ）

包括 salb、sarw、sarl等

ROL/ROR: 循环左/右移，包括 rolb、rorw、roll等

RCL/RCR: 带进位循环左/右移，即：将CF作为操作数

一部分循环移位，包括 rclb、rcrw、rcll等

RCL:

按位运算指令举例

假设short型变量x被编译器分配在寄存器AX中，R[ax]=FF80H，则以下汇编代码段执行后变量x的机器数和真值分别是多少？

```
movw %ax, %dx
salw  $2, %ax
addl  %dx, %ax
sarw  $1, %ax
```

R[dx]←R[ax]

1111 1111 1000 0000<<2　　算术左移，OF=0

1111 1110 0000 0000+1111 1111 1000 0000

1111 1101 1000 0000>>1=1111 1110 1100 0000

sarw $1,%ax 可简写成 sarw %ax

解：$2和$1分别表示立即数2和1 。

x是short型变量，故都是算术移位指令，并进行带符号整数加。

上述代码段执行前R[ax]=x，则执行((x<<2)+x)>>1后，

R[ax]=5x/2。算术左移时，AX中的内容在移位前、后符号未发生变化，故OF=0，没有溢出。最终AX的内容为FEC0H，解释为short型整数时，其值为-320。验证：x=-128，5x/2=-320。经验证，结果正确。

逆向工程：从汇编指令推断出高级语言程序代码

移位指令举例

```
#include <stdio.h>
void main()
{
    int a = 0x80000000;
    unsigned int b = 0x80000000;
    printf("a=   0x%X\n", a >>1);
    printf("b=   0x%X\n", b >>1);
}
```

```
                                    push   %ebp
                                    mov    %esp,%ebp
                                    and    $0xfffffff0,%esp
                                    sub    $0x20,%esp
00 00 00                            movl   $0x80000000,0x1c(%esp)

                                    movl   $0x80000000,0x18(%esp)
00 00 00

19:   8b 44 24 1c                   mov    0x1c(%esp),%eax
1d:   d1 f8                         sar    %eax                      算术
1f:   89 44 24 04                   mov    %eax,0x4(%esp)
23:   c7 04 24 00 00 00 00          movl   $0x0,(%esp)
2a:   e8 fc ff ff ff                call   2b <main+0x2b>
2f:   8b 44 24 18                   mov    0x18(%esp),%eax
33:   d1 e8                         shr    %eax                      逻辑
35:   89 44 24 04                   mov    %eax,0x4(%esp)
39:   c7 04 24 0b 00 00 00          movl   $0xb,(%esp)
40:   e8 fc ff ff ff                call   41 <main+0x41>
45:   c9                            leave
46:   c3                            ret
```