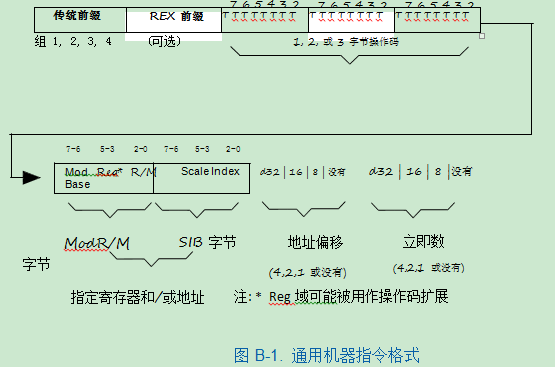
Intel x86的指令格式



b= Hex:0x31

Binary:110001

disas\_insn() at translate.c:4,902 0x55555590a630

第一步，先循环识别指令前缀。

第二步，switch中分析指令。

**switch**(b) {

case 0x30 ... 0x35:/\*XOR\*/

{

f=0

switch(f) {

case 0: /\* OP Ev, Gv \*/

modrm = cpu\_ldub\_code(env, s->pc++);

reg = ((modrm >> 3) & 7) | rex\_r;

mod = (modrm >> 6) & 3;

rm = (modrm & 7) | REX\_B(s);

if (mod != 3) {

gen\_lea\_modrm(env, s, modrm, &reg\_addr, &offset\_addr);//解析mod R/M字段，返回结果

。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

gen\_op\_mov\_TN\_reg(ot, 1, reg);

gen\_op(s, op, ot, opreg);

**break**;

**}**

**static** **inline** **void** **gen\_op\_mov\_TN\_reg**(**int** ot, **int** t\_index, **int** reg)

{

}

**static** **inline** **void** **gen\_op\_mov\_v\_reg**(**int** ot, TCGv t0, **int** reg)

{

}

**static** **inline** **void** **tcg\_gen\_mov\_i32**(TCGv\_i32 ret, TCGv\_i32 arg)

{

}

最终：

**static** **inline** **void** **tcg\_gen\_op2\_i32**(TCGOpcode opc, TCGv\_i32 arg1, TCGv\_i32 arg2)

{/\*：两个操作数指令的生成：第一个是寄存器，第二个 是寄存器\*/

\*tcg\_ctx.gen\_opc\_ptr++ = opc; //操作码的数组

\*tcg\_ctx.gen\_opparam\_ptr++ = GET\_TCGV\_I32(arg1);//操作数的数组

\*tcg\_ctx.gen\_opparam\_ptr++ = GET\_TCGV\_I32(arg2);

}

i386上实例：跳转指令。

b =Hex:0xea；

主体流程：

disas\_insn{

**case** 0xea: /\* ljmp im \*/

{

**unsigned** **int** selector, offset;

**if** (CODE64(s))

**goto** illegal\_op;

ot = dflag ? *OT\_LONG* : *OT\_WORD*;

offset = insn\_get(env, s, ot);//偏移量

selector = insn\_get(env, s, *OT\_WORD*);//选择子

/\*跟踪？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？？/\*/

A gen\_op\_movl\_T0\_im(selector);/\*将 立即数放入 T0寄存器，\*/

gen\_op\_movl\_T1\_imu(offset);/\*将 带符号的立即数放入 T1寄存器，\*/

}

**goto** do\_ljmp;

do\_ljmp:

**if** (s->pe && !s->vm86) {/\*x86默认是运行在保护模式的，pe就是保护模式\*/

} **else** {//？？为什么直接这里执行？？pe= 0 不再保护模式

B gen\_op\_movl\_seg\_T0\_vm(R\_CS); 将t0放入cs段所在的内存

C gen\_op\_movl\_T0\_T1(); 将t1：偏移量放入t0

D gen\_op\_jmp\_T0(); 将t0放入EIP 寄存器中

}

gen\_eob(s);

**break**;

}