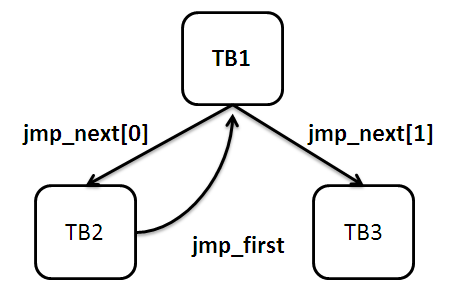
# CPU指令频度统计文档

## 1 对QEMU的修改

QEMU使用了块链接的机制，这样避免了在QEMU主控循环和代码缓存之间跳转的消耗后，运行速度大大加快了，也是QEMU模拟性能较高的一个原因。



而为了得到完整的指令信息，还需要对块链接信息的分析。不过最简单的方法是将这种链接打破。反映到代码中，就是将执行块链接的语句删除。以下的代码块就是要删除的部分：

if (next\_tb != 0 && tb->page\_addr[1] == -1) {

tb\_add\_jump((TranslationBlock \*)(next\_tb & ~TB\_EXIT\_MASK),

next\_tb & TB\_EXIT\_MASK, tb);

}

还需要对输出语句增加一些内容，即一个翻译块的大小，这样可以在事后将记录的翻译块序列还原为对应的指令序列。

if (qemu\_loglevel\_mask(CPU\_LOG\_EXEC)) {

qemu\_log("Trace %p [" TARGET\_FMT\_lx "] %s size %d\n",

tb->tc\_ptr, tb->pc, lookup\_symbol(tb->pc), tb->size);

}

格式为”Trace 翻译代码对应指针（暂无用） [“ 翻译块起始地址 “] 翻译块大小”

使用-d exec参数启动时就会在日志文件中输出如下信息：

Trace 0xb4456000 [fffffff0] size 5

Trace 0xb4456040 [000fe05b] size 11

Trace 0xb44560f0 [000fe066] size 4

Trace 0xb4456140 [000fe06a] size 6

Trace 0xb4456180 [000fe070] size 9

Trace 0xb44561b0 [000fd12a] size 43

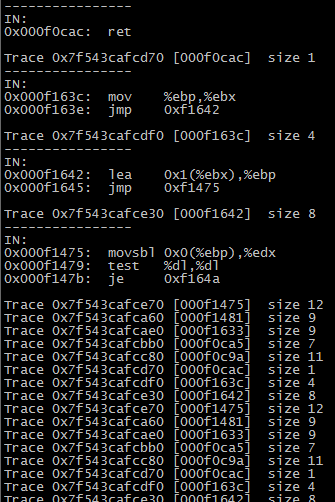
## 2． 获取指令流的方法

### 2.1. 记录qemu运行日志

qemu-system-i386 –hda ubuntu-10.04.vmdk -d exec,in\_asm -D qemu\_log3.txt

即可启动待测试系统，并且记录指令流到qemu\_log3.txt文件当中

qemu\_log3.txt文件格式如下：



其中IN：开头的表示qemu第一次执行该指令块时翻译时的记录情况。

以Trace 开头的表示将要执行的指令块已经翻译过，其中0xxxxxxxx可以唯一的标记一个指令块

### 2.2. 处理日志文件

#### 2.2.1 展开日志，获取指令流

要获取指令流，需要将Trace 0x7f543cafcd70 [000f0cac] size 1 这样的标记还原成

----------------

IN:

0x000f163c: mov %ebp,%ebx

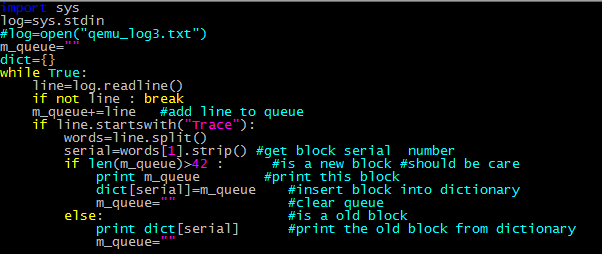
0x000f163e: jmp 0xf1642

Trace 0x7f543cafcdf0 [000f163c] size 4

这样的标记。

还原的方法是从搜索之前的翻译历史找出对应的翻译记录即可

python处理程序如下



日志文件中每一个IN都对应一个Trace，对于没有IN与之对应的Trace，表示这个代码块已经被翻译过，只需要从之前的翻译历史中找出与序列号对应的翻译块即可

运行方法

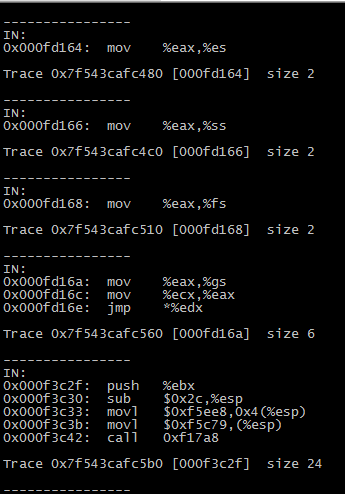
qemu\_log\_chuli.py < qemu\_log3.txt > qemu\_log3\_zhankai.txt

其中qemu\_log3.txt是要处理的日志的文件名

qemu\_log3\_zhankai.txt是qemu\_log3.txt的展开结果

qemu\_log3\_zhankai.txt的内容如下：

这个时候每个Trace都有一个IN与之配对



因为我们关系的是指令流所以，只需要保留

0x000fd16a: mov %eax,%gs

0x000fd16c: mov %ecx,%eax

0x000fd16e: jmp \*%edx

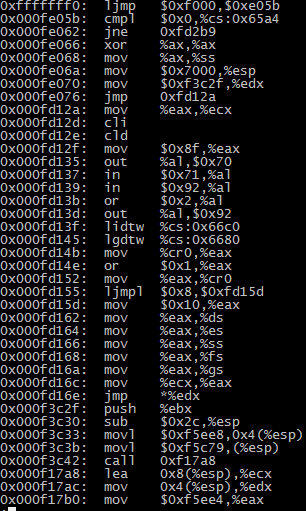
这样的信息即可别的都不需要

grep "^0x" qemu\_log3.txt qemu\_log3\_zhankai.txt > qemu\_log3.txt qemu\_log3\_zhankai\_zhilin.txt

这样可以将没用的信息过滤掉，只剩下要统计的指令流。

qemu\_log3\_zhankai\_zhilin.txt

内容如下：

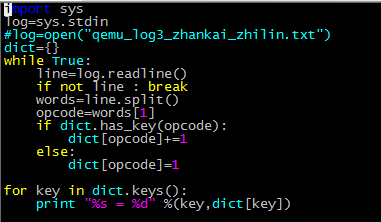


#### 2.2.2统计指令频度

建立一个字典结构，其中key为指令，value为对应指令的频度，只需要遍历

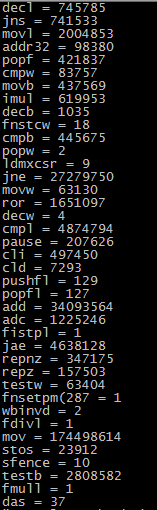
qemu\_log3\_zhankai\_zhilin.txt

遇到一条指令使其key对应的value加1即可。



statistics.py < qemu\_log3\_zhankai\_zhilin.txt > qemu\_log3\_zhankai\_zhilin\_statistics.txt

qemu\_log3\_zhankai\_zhilin\_statistics.txt内容如下：

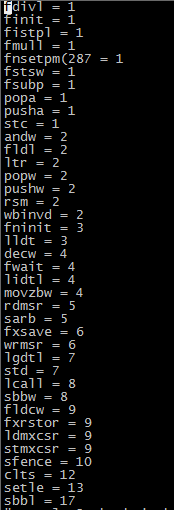


上述结果没有按指令使用的频度进行排序。

下面进行排序：

sort -n -k3 qemu\_log3\_zhankai\_zhilin\_statistics.txt > qemu\_log3\_zhankai\_zhilin\_statistics\_sort.txt

排序后的结果如下：



## 3. 自动化测试方法：

运行

instruction\_statistics目录下面的start.sh脚本即可，自动完成所有测试流程

