汇编：第六讲作业

刘泓尊 2018011446 计84 liu-hz18@mails.tsinghua.edu.cn

---

linux中使用`gcc`的`__buildin_expect(long exp, long c)`扩展 进行程序分支预测功能，意为`exp==c`的概率很大。它通过改变生成的汇编指令的顺序，来优化处理器流水线的运行，以使得最可能执行的分支无需执行任何跳转指令。

在多级流水线处理器中，同时有多条指令位于流水线中。处理器采用分支预测逻辑来猜测每条跳转指令是否会被执行。如果错误预测一个跳转，那么cpu必须清空流水线，重新从正确位置起始的指令取填充流水线。这将带来很严重的惩罚，浪费大约15-30个时钟周期。

当设置为`likely(EXPRESSION)`时，编译器会将第一个分支的代码放在前面；当设置为`unlikely(EXPRESSION)`时，编译器会将第二个分支的代码放在前面。这种安排保证了对应的高概率情况不存在跳转，从而充分利用流水线上的结果，提高程序性能。

注:

`__builtin_expect (exp, c)`要求`exp`必须为整数，两个叹号`!!`能够保证`x`的值不是`0`就是`1`

具体在本例中

由`if(unlikely(a == 2)) {...} else {...}`编译出的汇编代码中

```
leal -1(%rax), %ecx
leal -1(%rsi), %edx
(%rax <- a, %rsi <- b)
```

位于前面，即执行`a--;b--;`将不存在跳转。

相反地，`if(likely(a == 2)) {...} else {...}`编译出的汇编代码中

```
leal 1(%rcx), %eax
movl $3, %edx
```

在前面，即`b+1`放在`%eax`中，并将`3`放入`%edx`中，随后`main`函数返回`a+b`. 所以`a++; b++;`这段代码将不存在跳转。

在我的机器上，使用`unlikely (a == 2)`指令，执行`./main 1`和`./main 2`各`1000`次的时间分别为

```
//unlikely (a == 2)的运行时间
./main 1:  0.006542205810546875
./main 2:  0.007308006286621094
```

可见确实有性能提升。

综上所述，上述编译指示改变了汇编指令顺序，充分利用现代处理器的流水线机制，通过避免分支预测错误引起的惩罚，来实现性能优化。