**cpufreq**

**2018K8009918002 陈彦帆**

C嵌入式汇编 freq.c

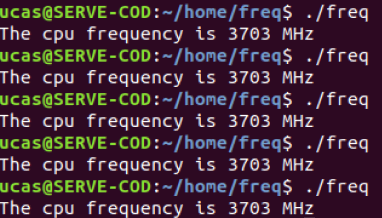
1. 编译命令 （若无法运行请联系我）

gcc freq.c -o freq

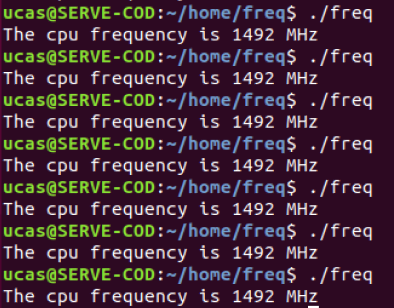
./freq

2. 测试结果

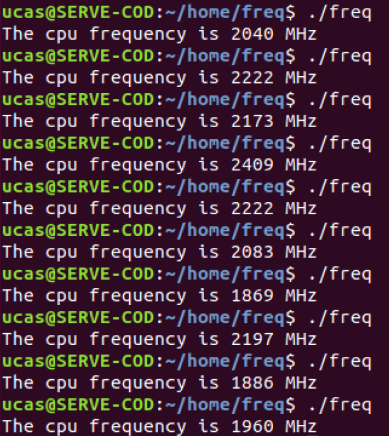
测试1： 运行环境 Intel Core i7 8550u 基频1.99GHz， 最大睿频4GHz

 (1) 笔记本插电状态：

测量时cpu自动睿频，频率上升。测量值在3703MHz。符合睿频频率。

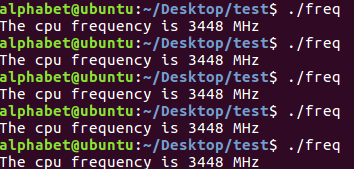
 (2)笔记本不插电，节能模式

测量值在1492MHz，符合预期。

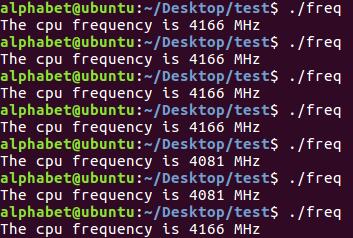
 (3)笔记本插电，使用软件烤机。烤机后，从任务管理器看出，cpu频率在2000MHz上下浮动，与测试结果相符。

由于笔记本频率浮动过大，难以评估准确性，下面用台式机测试。

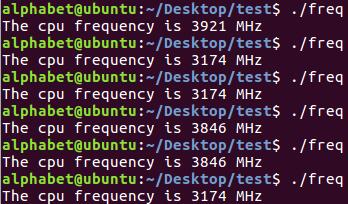
测试2： 运行环境 AMD R5-3600

(1)锁频至3500MHz

多次测试结果为3448MHz，误差1.5%

(2) 不锁频，开启自动睿频。基频为3600MHz，最大睿频4200GHz

测试结果接近最大睿频。

 (3) 不锁频，用软件烤机。烤机后，从任务管理器看出，全核频率在3.85GHz左右

多次测试出现了3个值。其中3921和3846与实际结果接近。而3174与任务管理器数据偏离较大，猜想可能是 (1)单核降频。(2)烤机软件占用了部分运行时间。

测试3： 运行环境 AMD A6-3620 基频2.2GHz

这是一台2012年的电脑。采用32位操作系统（以上均为64位）。

测试结果：2197MHz，与基频误差0.1%。

3. 算法描述

直接用rdtsc测时钟周期是不可行的，因为它只能反映基频的时钟周期，无法反应频率变化。

考虑用一连串已知周期的指令。

最终选定的指令是一连串的add $5, %eax。因为根据锁频测试，从8年前的cpu到最新的cpu，均为一拍执行一条指令。而且这一连串指令是串行的，不能通过多个加法器并行工作来完成。其他一些指令也能达到类似效果，比如xor, shr, inc。

但在操作过程中，遇到了一些问题，连同解决方案如下：

(1)操作系统调度，在测试程序运行的时间内测试程序与其他程序共同占用cpu内核，会和其他程序分摊时间，使测试结果偏低。

改进办法是：增加一次性执行的指令条数和循环周期，使测试程序具有更高的优先级，尽可能在绝大部分时间内独占某一内核。

(2)频率的微小浮动与测试的偶然性

解决办法是，循环550次，每次执行20万条上述add指令，每循环一次记录一次时间，最后取后50次的众数时间来算频率，减小偶然性。每次循环的条数越多，精度越大，但测试时发现，执行条数太多会使A6-3620电脑测得的频率大幅减少，故最终选定每次循环为20万条。

(3)测试程序会使cpu自动睿频，使频率上升，就不是测量前的频率了。

暂时没有想到合适的解决方案。因此在测试具有睿频的处理器时，测得的结果一般接近自动睿频的典型值。但在满载的电脑上（比如烤机状态），cpu频率不会进一步上升，测量的结果会接近或者小于实际值。