# 第十二章作业

钟赟 2016K8009915009

1. **写两个简单的测试程序，用于测量一台计算机系统最大的 MIPS 和最大的 MFLOPS 的值。**

我的思路是写一个整数运算操作的循环，记录它完成所需的时间，算出 MIPS，计算 MFLOPS 只需将整型操作换成浮点操作即可。程序如下：

```
#include<stdio.h>
#include<sys/time.h>
#include<sys/syscall.h>
#define RUN_TIMES 1000000000
int main() {
    struct timeval t1, t2, t3, t4, t5, t6;
    int i, j = 0;
    float k = 0.0;

    //执行 1000000000 次整数运算指令
    gettimeofday(&t1, NULL);
    for(i = 0; i < RUN_TIMES; i ++)          j += 1;
    gettimeofday(&t2, NULL);

    //执行 1000000000 次整数运算指令
    gettimeofday(&t3, NULL);
    for(i = 0; i < RUN_TIMES; i ++)          k += 1.2;
    gettimeofday(&t4, NULL);

    double MIPS_time = (t2.tv_sec - t1.tv_sec)*1000000 + (t2.tv_usec - t1.tv_usec);
    double MFLOPS_time = (t4.tv_sec - t3.tv_sec)*1000000 + (t4.tv_usec - t3.tv_usec);
    printf("MIPS: %lf\nMFLOPS: %lf\n", RUN_TIMES/MIPS_time, RUN_TIMES/MFLOPS_time);
    return 0;
}
```

运行结果：

```
stu@stu-VirtualBox:~/Documents$ gcc -o mips mips.c
stu@stu-VirtualBox:~/Documents$ ./mips
MIPS: 439.781974
MFLOPS: 353.936070
```

2. **阅读和分析 STREAM v1 基准测试程序：**

   a) **测出一台计算机上的测试结果并给出分析报告。**

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           5495.6     0.033453     0.029114     0.045249
Scale:          5855.0     0.029661     0.027327     0.031001
Add:            8106.2     0.033333     0.029607     0.036168
Triad:          7805.2     0.035832     0.030749     0.044500
-------------------------------------------------------------
```

   b) **调节处理器的频率，看内存的带宽和频率的关系。**

调大处理器的频率，带宽没有明显增大。测试结果如下：

```
-------------------------------------------------------------
```

```
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           5671.8      0.030092     0.028210     0.033411
Scale:          6747.1      0.028143     0.023714     0.033163
Add:            8343.2      0.030672     0.028766     0.033845
Triad:          8106.7      0.033743     0.029605     0.037196
-------------------------------------------------------------
```

**c) 修改 STREAM 测试程序，看单精度带宽和双精度带宽的区别。**

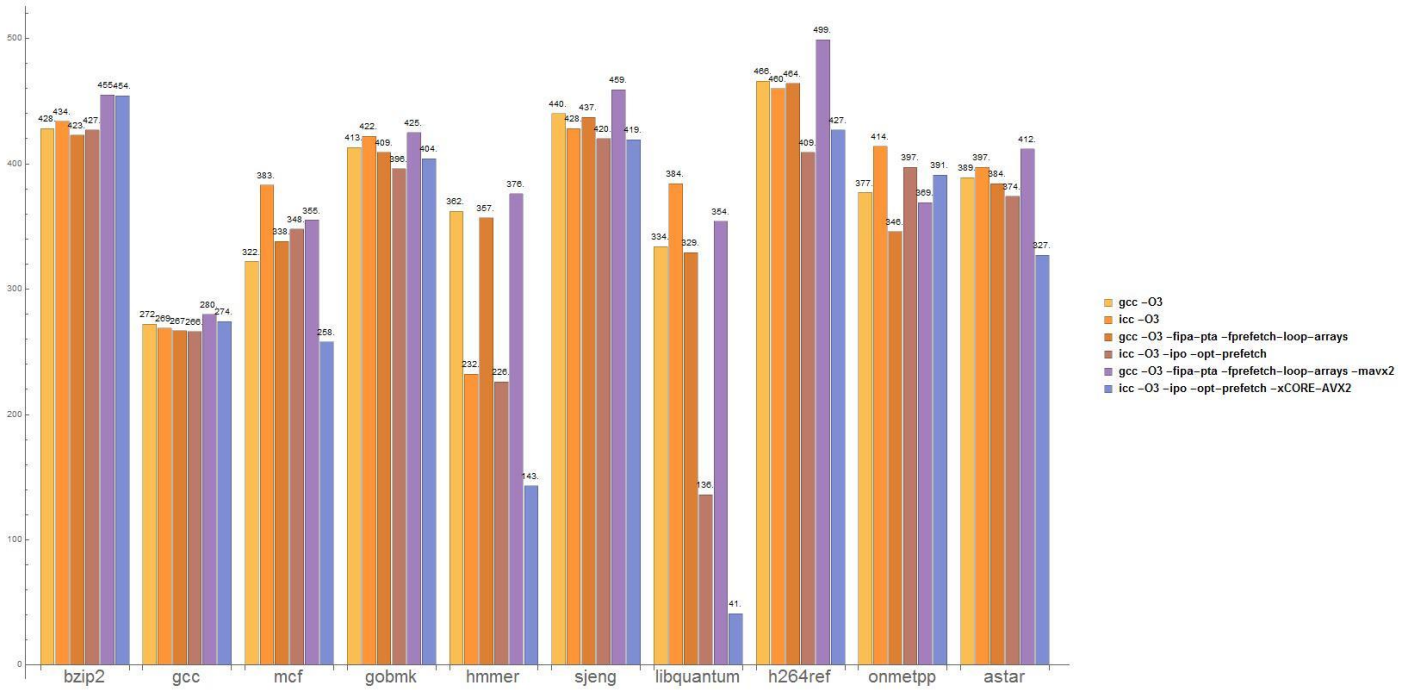
修改为双精度带宽，带宽有明显增大。测试结果如下：

```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           7360.4      0.022212     0.021738     0.022533
Scale:          8274.3      0.020000     0.019337     0.021543
Add:           10944.9      0.023155     0.021928     0.028491
Triad:         10435.7      0.024811     0.022998     0.029545
-------------------------------------------------------------
```

**3. 分析 SPEC CPU 2006 中 462.libquantum 程序，看它对处理器微结构的压力在哪里。查阅 spec.org 网站，看不同编译器对 462.libquantum 的分值的影响，猜测 Intel 编译器 icc 采用了什么编译技术使得其分值能达到上百分。**

462.libquantum 程序对处理器微结构的压力可能在与并行操作性能。

不同编译器对 462.libquantum 的分值的影响如下：



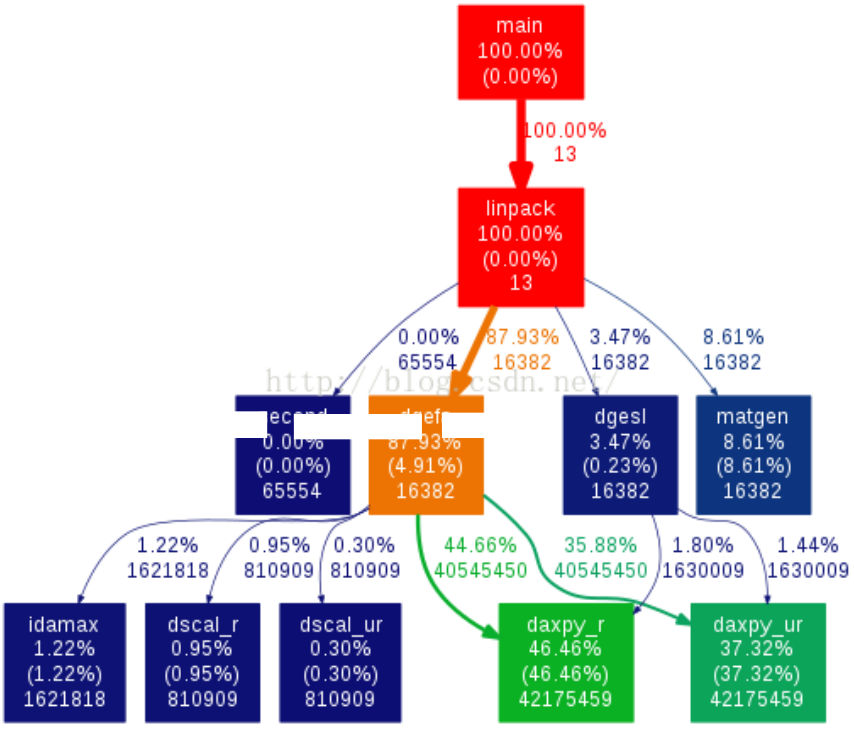

由图可见 icc 比 gcc 更有优势。猜测 Intel 编译器 icc 采用了向量指令优化的方法来提高并行性能。

**4. 使用 Perf 工具测量各种排序算法如冒泡排序、希尔排序等算法的 IPC，分析排序算法对处理器微结构的压力在哪里。**

Perf 无法正常运行：进入/tools/perf 文件夹后，执行 make 指令失败，报错如下：

```
stu@stu-VirtualBox:/usr/src/linux-headers-3.11.0-15/tools/perf$ make
Makefile:6: config/utilities.mak: No such file or directory
config/Makefile:52:  /usr/src/linux-headers-3.11.0-15/tools/perf/config/feature-tests.mak:  No such file or directory
config/Makefile:53:  /usr/src/linux-headers-3.11.0-15/tools/perf/config/utilities.mak:  No  such file or directory
config/Makefile:56: *** Error: flex is missing on this system, please install it.  Stop.
```

**5. 使用 gprof 工具，获得 linpack 程序的热点函数。**




6. **使用 LMbench 测试程序，获得 CPU 的一级，二级、三级 cache 和内存的访存延迟。**

   使用 LMbench 测试程序进行测试时，分配空间时总是报错：空间不足，该问题还未解决。

7. **使用 SimpleScalar 模拟器，分析二级 cache 的延迟对性能的影响（从 24 变到 12 个时钟周期），假设使用 Alpha 的指令集，测试程序为 SPEC CPU 2000 的 164.bzip 和 253.perlbmk。**

   未完成。

8. **嵌入式基准测试程序如 EEMBC 和桌面基准测试程序在行为特性上有什么差别?**

   嵌入式基准测试程序如 EEMBC 主要是验证现实世界的基准测试和基准测试得分，用于选择与系统适用的嵌入式处理器。需要更多的考虑到嵌入式设备实际使用时的场景，关注功耗与性能等等指标。测试程序不一定是对于 CPU 的计算性能有着很高的要求，而是会模拟嵌入式实际使用时"不时被唤醒"，"连接传感器"等情况。

   桌面基准测试程序是针对桌面通用处理器，统一的为现代计算机系统建立基准测试程序集。一般会执行一些科学计算操作，常用算法和常用程序的性能测试，测试程序需要长时间的占用 CPU，测试程序对于 CPU 的性能(主频, IPC)有着很高的要求，其目的是测试桌面处理器的极限性能。

9. **查找 ARM Cortex A 系列处理器的用户手册，列出你认为比较重要的硬件性能计数器的 10 个性能事件，并给出事件描述。**

   下表为 ARM Cortex A73 处理器的用户手册中 PMU 性能事件的前十个:

**Table 11-26 PMU events**

| Event number | Event mnemonic | PMU event bus (to external) | PMU event bus (to trace) | Event name |
|---|---|---|---|---|
| 0x00 | SW_INCR | - | - | Software increment. Instruction architecturally executed (condition check pass). |
| 0x01 | L1I_CACHE_REFILL | [0] | [0] | L1 instruction cache refill. |
| 0x02 | L1I_TLB_REFILL | [1] | [1] | L1 instruction TLB refill. |
| 0x03 | L1D_CACHE_REFILL | [2] | [2] | L1 data cache refill. |
| 0x04 | L1D_CACHE | [4:3] | [4:3] | L1 data cache access. |
| 0x05 | L1D_TLB_REFILL | [5] | [5] | L1 data TLB refill. |
| 0x08 | INST_RETIRED | [16:10] | [16:10][di] | Instruction architecturally executed. |
| 0x09 | EXC_TAKEN | [17] | [17] | Exception taken. |
| 0x0A | EXC_RETURN | [18] | [18] | Instruction architecturally executed, condition code check pass, exception return. |
| 0x0B | CID_WRITE_RETIRED | [19] | [19] | Instruction architecturally executed, condition code check pass, write to CONTEXTIDR. |

分别为:

1) 软件自增：体系结构执行条件检查通过的指令数；
2) 一级指令 Cache 重填例外；
3) 一级指令 TLB 烈爱；
4) 一级数据 Cache 重填例外；
5) 一级数据 Cache 访问；
6) 一级数据 TLB 例外；
7) 体系结构上执行的指令数；
8) 例外；
9) 条件检查通过、例外返回的指令数；
10) 条件检查通过并回写 CONTEXTIDR 寄存器的指令数。

**10. 模拟建模的方法和性能测量的方法相比有哪些优点？**

性能测量：用于理解已经搭建好的系统或者原型系统，主要包括片上的硬件检测器、片外硬件检测器、软件检测器、微码插桩。

模拟建模：使用软件的方法来模拟计算机系统硬件再体系结构层面的功能和性能特性，保罗踪迹驱动模拟、执行驱动模拟、全系统模拟、事件驱动模拟、统计方法模拟。

相比性能测试，模拟建模具有灵活和开销小的优点。模拟器在初期可以用来对各种设计方案进行粗粒度模拟，来选择最优方案；在中期阶段用来对各种微结构设计进行评估，对一些为结构参数的选择进行折中；后期用来与目标系统进行性能验证，保证性能模型与实际机器的吻合；系统完成后，模拟器可以产生踪迹信息，对系统进行瓶颈分析和性能优化。相对来说，性能测量的方法更加精准。一般模拟器比真实硬的速度慢几个数量级，模拟结果会受到人为因素的干扰。

**11. SimPoint 的基本原理是什么，为什么能减少模拟建模时间？**

SimPoint 的原理：找到程序执行的相位，采样能够代表每个相位的部分，并进行模拟仿真，把它作为整个程序模拟结果的一个近似。SimPoint 通过找出测试程序中具有代表性的样本进行精确的时序模拟，减少了指令数，进而减少模拟时间。

**12. 模拟器和真实的机器怎么校准，校准的评价指标通常是什么？**

模拟器使用 C、C++、Python 等高级语言开发，利用这些串行结构化语言的函数或者类来模拟计算机系统部件的功能和行为。模拟器最直接的方法就是将真实机器的二进制代码中的每一条指令都转换为同样语义的宿主机器的指令，在宿主机器上执行。将目标机器的每一个寄存器和标识位对应一个变量，目标机器的所有逻辑操作都可以被间接地翻译成变量的运算，目标机器上程序的所有操作都可以在原始机器上以软件模拟的形式复现出来。

校准的标准：在进行微结构级的详细模拟时，模拟器需要在时钟周期级别上记录每条动态指令的运行结果、每一集流水线触发器的状态、内部各种队列的状态一级体系结构寄存器的状态、内存和 Cache 的行为、分支预测器的状态等。

**13. 在你的电脑上运行 SPEC CPU2000 的 rate 并给出分值。**

```
[Result]:
Success 164.gzip ratio=1612.75, runtime=86.808345
Success 175.vpr ratio=2458.94, runtime=56.935060
Success 176.gcc ratio=2904.36, runtime=37.874131
Success 181.mcf ratio=5980.15, runtime=30.099575
Success 186.crafty ratio=2409.40, runtime=41.504100
Success 197.parser ratio=2163.44, runtime=83.200674
Success 253.perlbmk ratio=1846.04, runtime=97.506179
Error 254.gap ratio=50659.70, runtime=2.171351
Error 255.vortex ratio=1835376.40, runtime=0.103521
Success 256.bzip2 ratio=2165.75, runtime=69.260127
Success 300.twolf ratio=3902.42, runtime=76.875446
Success 177.mesa ratio=2442.46, runtime=57.319146
Success 179.art ratio=6228.98, runtime=41.740393
Success 183.equake ratio=7550.62, runtime=17.217128
Success 188.ammp ratio=2621.16, runtime=83.932373
```