CA_Assignment12

Assign	
tag	homework
姓名	
学号	

1. 写两个测试程序，分别用于测量一台计算机系统最大的 MIPS 和最大的 MFLOPS 的值。

```
#ifndef ARRLEN #define ARRLEN 0x1000 #endif

#ifndef K #define K 3 #endif

int arr[ARRLEN];

int fib(int k, int m) {
  if (m < k - 1) { return 0; }

  if (m == k - 1) { return 1; }

  int i, cnt;

  for (i = 0; i < k; ++i) {

  arr[i] = i == k - 1 ? 1 : 0;

  }

  for (cnt = 1; i <= m; ++i) {

  arr[i] = cnt;

  cnt += (arr[i] - arr[i - k]);

}

  return arr[m];

}

int main() {
  fib(K, ARRLEN - 1); return 0;
}
```

结果为：

```
        0.64 msec task-clock                #    0.515 CPUs utilized
           0      context-switches          #    0.000 K/sec
           0      cpu-migrations            #    0.000 K/sec
          49      page-faults               #    0.076 M/sec
     697,492      cycles                    #    1.088 GHz
     635,609      instructions              #    0.91  insn per cycle
     120,268      branches                  #  187.647 M/sec
       4,290      branch-misses             #    3.57% of all branches

 0.001244670 seconds time elapsed

 0.001313000 seconds user
 0.000000000 seconds sys
```

故其MIPS为(635609/0.00124467)/1000000=510.7

MFLOPS 的测定采用如下方式：

```
#define MAX 1000000.0

int main() {
  double tmp = 1.1;
  for (float i = 1.0; i < MAX; i = i + 1.0) {
    tmp = tmp * tmp;
    }
  return 0;
}
```

结果为：

```
       10.32 msec task-clock                #    0.941 CPUs utilized
           0      context-switches          #    0.000 K/sec
           0      cpu-migrations            #    0.000 K/sec
          46      page-faults               #    0.004 M/sec
  10,729,724      cycles                    #    1.040 GHz
  10,613,747      instructions              #    0.99  insn per cycle
   1,118,001      branches                  #  108.361 M/sec
       4,479      branch-misses             #    0.40% of all branches

 0.010966144 seconds time elapsed

 0.011045000 seconds user
 0.000000000 seconds sys
```

故其MFLOPS（可以用MIPS近似）为967.86

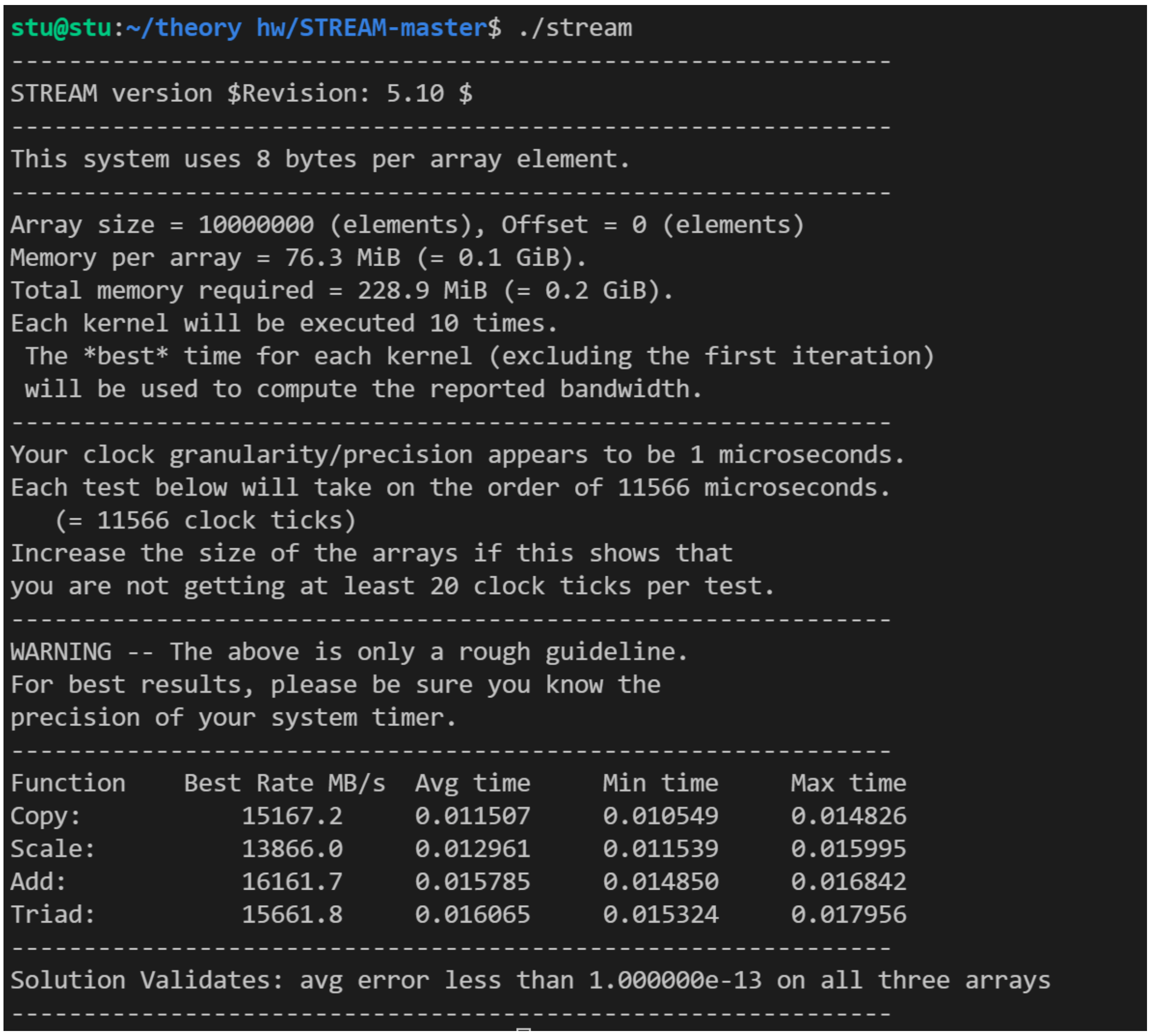
2. 阅读和分析 STREAM v1 基准测试程序：

   • 测出一台计算机上的测试结果并给出分析报告。

   • 调节处理器的频率，看内存的带宽和频率的关系。

   • 修改 STREAM 测试程序，看单精度带宽和双精度带宽的差别。

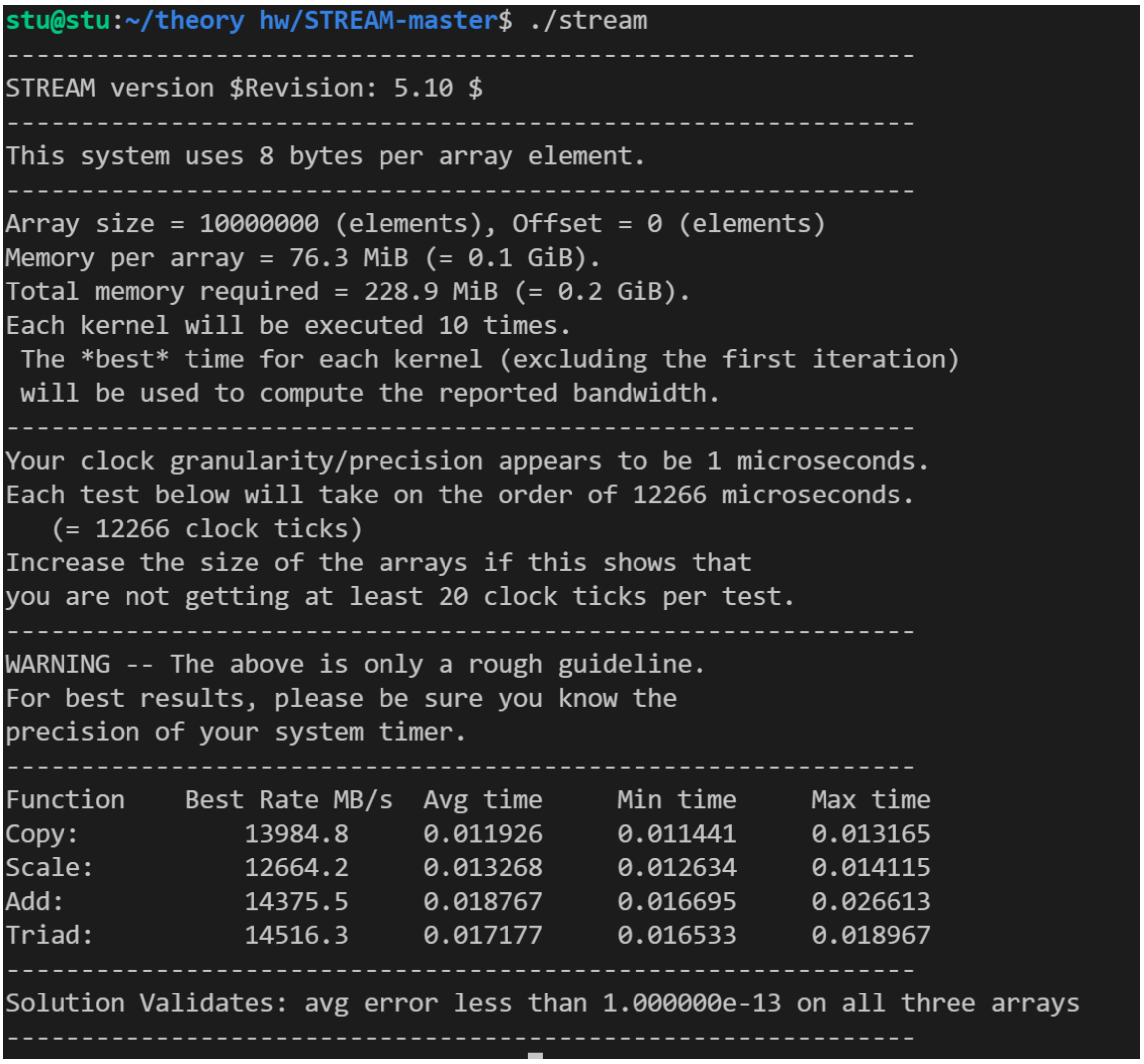
   a.



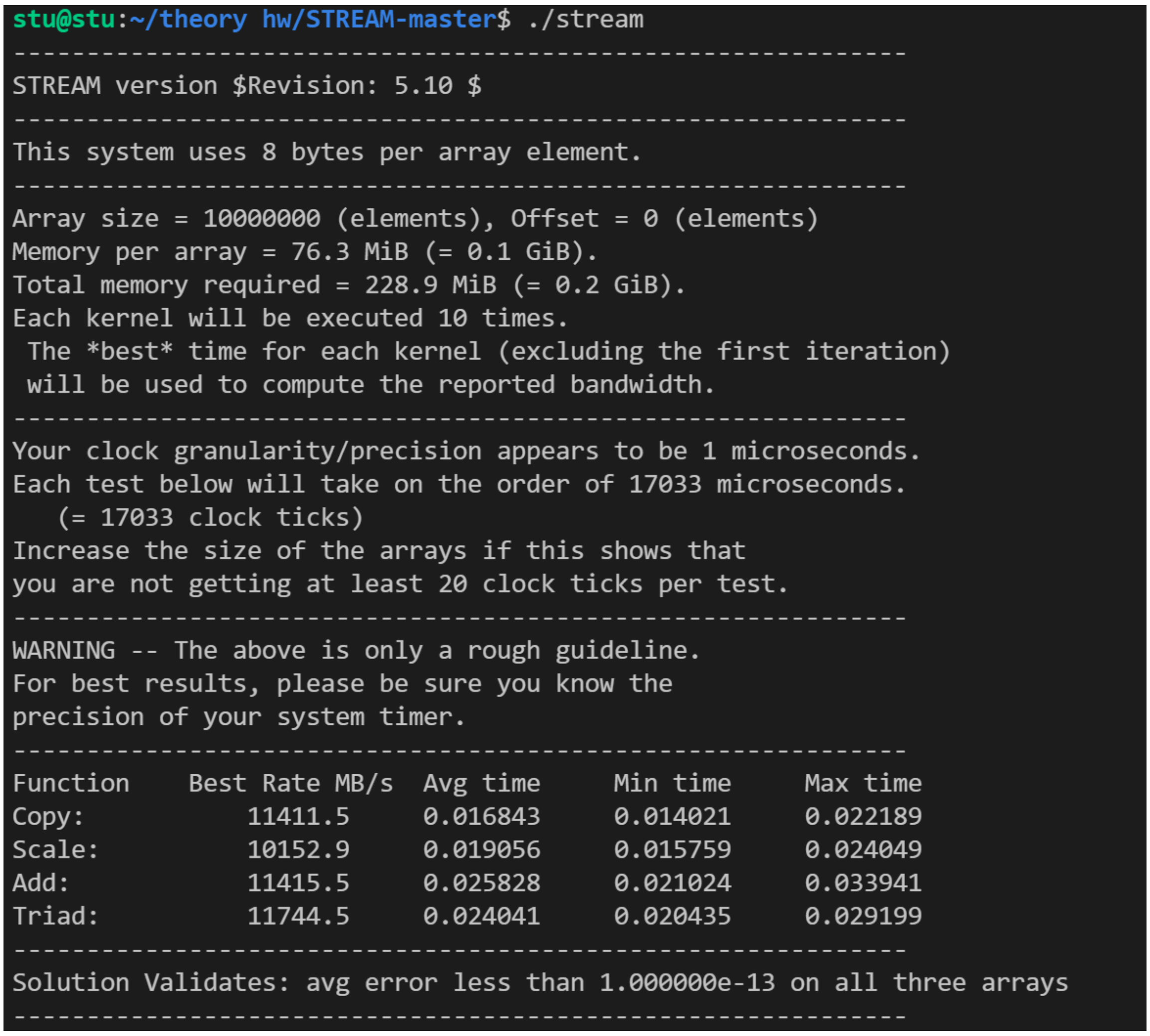
```
stu@stu:~/theory hw/STREAM-master$ ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 11566 microseconds.
   (= 11566 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           15167.2     0.011507     0.010549     0.014826
Scale:          13866.0     0.012961     0.011539     0.015995
Add:            16161.7     0.015785     0.014850     0.016842
Triad:          15661.8     0.016065     0.015324     0.017956
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

可见单次操作需要 3 次访存的 add 与 triad 操作带宽比单次仅需两次访存的 copy 与 scale 要高。 对于 scale 与 triad 而言，其计算分别比 copy、add 复杂，访存在单次操作中的时间占比较低，故而 其带宽相比后者均较低。

b. 在更改运算性能后：



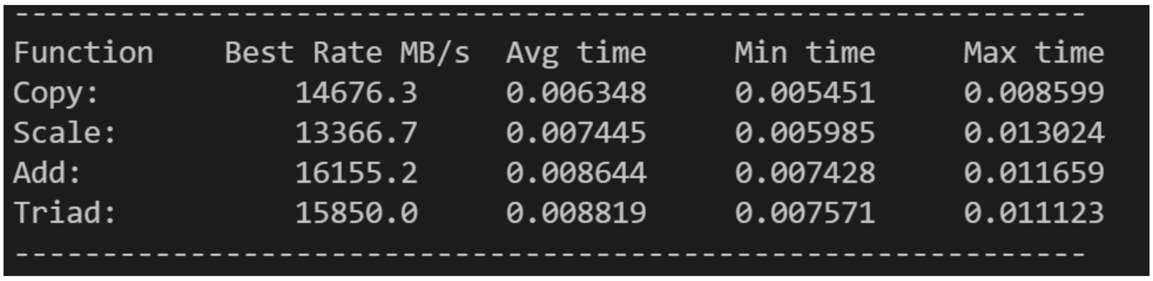
```
stu@stu:~/theory hw/STREAM-master$ ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 12266 microseconds.
   (= 12266 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           13984.8     0.011926     0.011441     0.013165
Scale:          12664.2     0.013268     0.012634     0.014115
Add:            14375.5     0.018767     0.016695     0.026613
Triad:          14516.3     0.017177     0.016533     0.018967
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```



```
stu@stu:~/theory hw/STREAM-master$ ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 17033 microseconds.
   (= 17033 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           11411.5     0.016843     0.014021     0.022189
Scale:          10152.9     0.019056     0.015759     0.024049
Add:            11415.5     0.025828     0.021024     0.033941
Triad:          11744.5     0.024041     0.020435     0.029199
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

可以看出结论非常符合直觉，频率减少，带宽降低。

c.



```
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           14676.3     0.006348     0.005451     0.008599
Scale:          13366.7     0.007445     0.005985     0.013024
Add:            16155.2     0.008644     0.007428     0.011659
Triad:          15850.0     0.008819     0.007571     0.011123
-------------------------------------------------------------
```

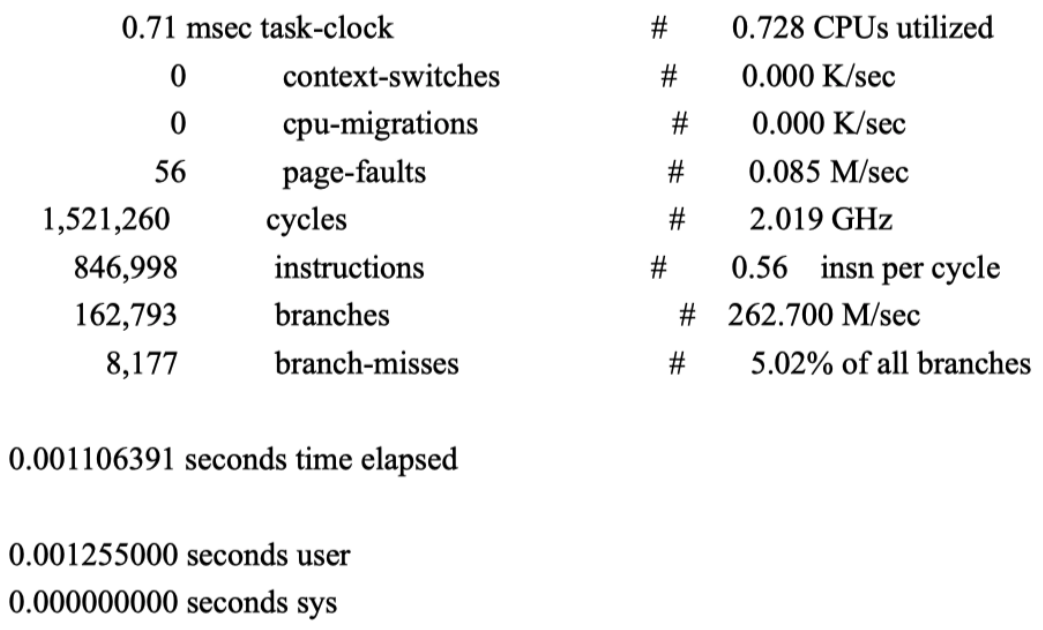
单精度带宽略微小于双精度

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。

libquantum 以串行形式模拟量子计算机上并行过程的 C 语言库，对于处理器串行处理的能力要 求较高。
ICC 可能在对串行并行化上具有一定优势，在高串行要求下表现优秀。

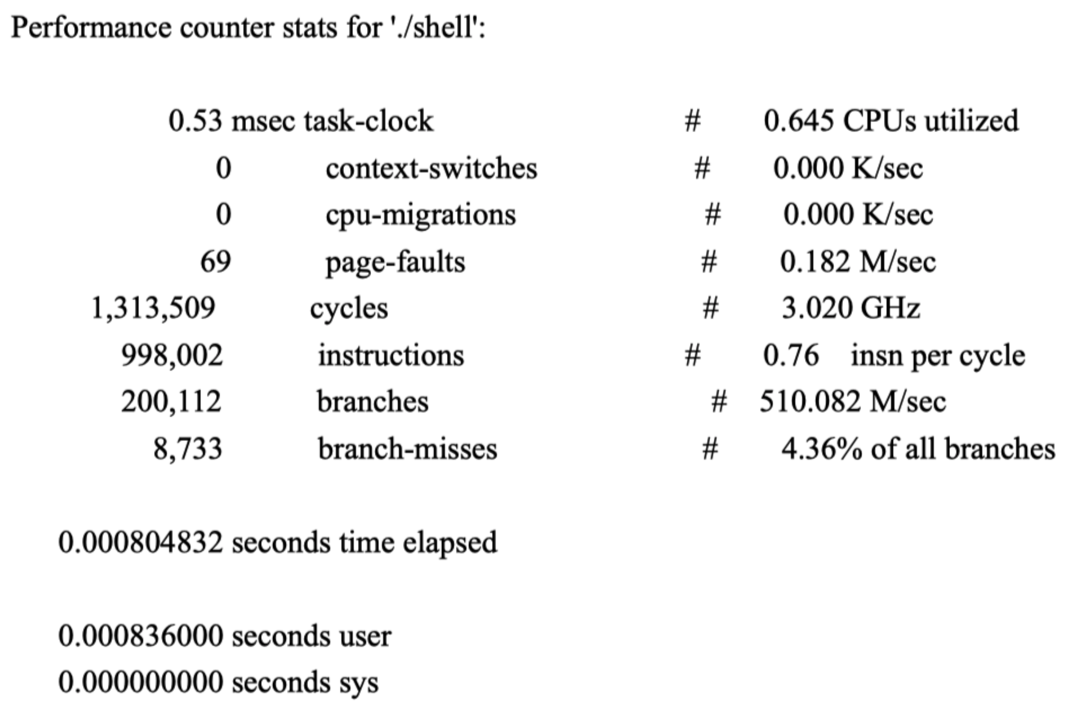
4. 使用 Perf 工具，测量各种排序算法如冒泡排序、希尔排序等算法的 IPC，分析排序算法对处理 器微结构的压力在哪里。

冒泡排序：



```
          0.71 msec task-clock                #    0.728 CPUs utilized
             0      context-switches           #    0.000 K/sec
             0      cpu-migrations             #    0.000 K/sec
            56      page-faults                #    0.085 M/sec
     1,521,260      cycles                     #    2.019 GHz
       846,998      instructions               #    0.56  insn per cycle
       162,793      branches                   #  262.700 M/sec
         8,177      branch-misses              #    5.02% of all branches

   0.001106391 seconds time elapsed

   0.001255000 seconds user
   0.000000000 seconds sys
```

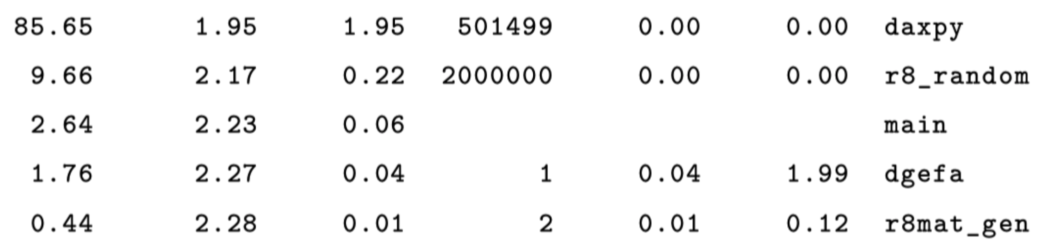
希尔排序5



```
Performance counter stats for './shell':

              0.53 msec task-clock                #     0.645 CPUs utilized
                 0         context-switches        #      0.000 K/sec
                 0         cpu-migrations          #      0.000 K/sec
                69         page-faults             #      0.182 M/sec
         1,313,509        cycles                   #      3.020 GHz
           998,002         instructions           #     0.76    insn per cycle
           200,112         branches                 #    510.082 M/sec
             8,733         branch-misses           #      4.36% of all branches

       0.000804832 seconds time elapsed

       0.000836000 seconds user
       0.000000000 seconds sys
```

5. 使用 gprof 工具，获得 linpack 程序的热点函数。

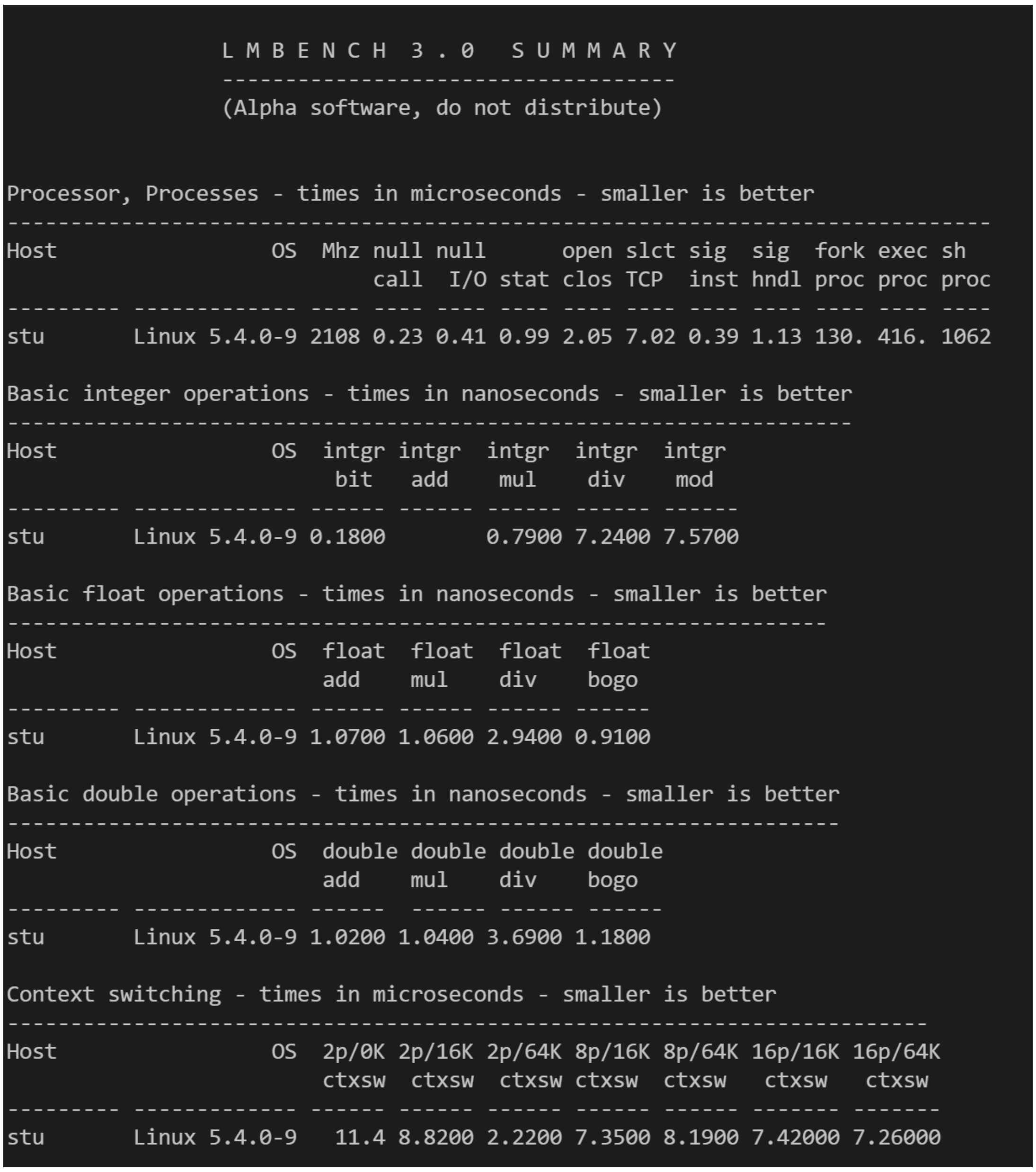
   编译运行后执行 gprof，获取到性能分析结果，可 知程序运行 85.65% 的时间用于函数 daxpy。



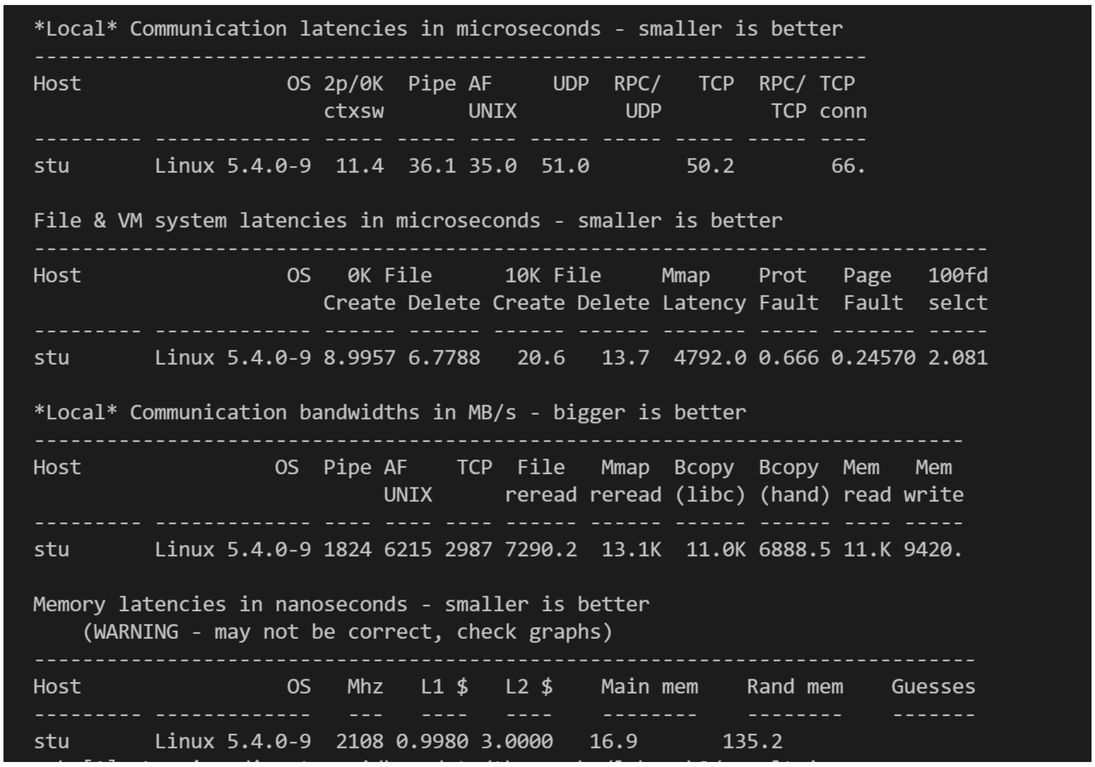
```
  85.65      1.95      1.95    501499      0.00      0.00   daxpy
   9.66      2.17      0.22   2000000      0.00      0.00   r8_random
   2.64      2.23      0.06                                  main
   1.76      2.27      0.04         1      0.04      1.99   dgefa
   0.44      2.28      0.01         2      0.01      0.12   r8mat_gen
```

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

   运行结果如下：



```
                 L M B E N C H  3 . 0   S U M M A R Y
                 ------------------------------------
                 (Alpha software, do not distribute)


Processor, Processes - times in microseconds - smaller is better
------------------------------------------------------------------------------
Host                 OS  Mhz null null      open slct sig  sig  fork exec sh
                             call  I/O stat clos TCP  inst hndl proc proc proc
--------- ------------- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
stu       Linux 5.4.0-9 2108 0.23 0.41 0.99 2.05 7.02 0.39 1.13 130. 416. 1062

Basic integer operations - times in nanoseconds - smaller is better
-------------------------------------------------------------------
Host                 OS  intgr intgr  intgr  intgr  intgr
                          bit   add    mul    div    mod
--------- ------------- ------ ------ ------ ------ ------
stu       Linux 5.4.0-9 0.1800        0.7900 7.2400 7.5700

Basic float operations - times in nanoseconds - smaller is better
-----------------------------------------------------------------
Host                 OS  float  float  float  float
                         add    mul    div    bogo
--------- ------------- ------ ------ ------ ------
stu       Linux 5.4.0-9 1.0700 1.0600 2.9400 0.9100

Basic double operations - times in nanoseconds - smaller is better
------------------------------------------------------------------
Host                 OS  double double double double
                         add    mul    div    bogo
--------- ------------- ------  ------ ------ ------
stu       Linux 5.4.0-9 1.0200 1.0400 3.6900 1.1800

Context switching - times in microseconds - smaller is better
-------------------------------------------------------------------------
Host                 OS  2p/0K 2p/16K 2p/64K 8p/16K 8p/64K 16p/16K 16p/64K
                         ctxsw  ctxsw  ctxsw ctxsw  ctxsw   ctxsw   ctxsw
--------- ------------- ------ ------ ------ ------ ------ ------- -------
stu       Linux 5.4.0-9   11.4 8.8200 2.2200 7.3500 8.1900 7.42000 7.26000
```



```
*Local* Communication latencies in microseconds - smaller is better
---------------------------------------------------------------------
Host                 OS 2p/0K  Pipe AF     UDP  RPC/   TCP  RPC/ TCP
                        ctxsw       UNIX         UDP         TCP conn
--------- ------------- ----- ----- ---- ----- ----- ----- ----- ----
stu       Linux 5.4.0-9  11.4  36.1 35.0  51.0        50.2        66.

File & VM system latencies in microseconds - smaller is better
-------------------------------------------------------------------------------
Host                 OS   0K File      10K File     Mmap    Prot   Page   100fd
                        Create Delete Create Delete Latency Fault  Fault  selct
--------- ------------- ------ ------ ------ ------ ------- ----- ------- -----
stu       Linux 5.4.0-9 8.9957 6.7788   20.6   13.7  4792.0 0.666 0.24570 2.081

*Local* Communication bandwidths in MB/s - bigger is better
-----------------------------------------------------------------------------
Host                OS  Pipe AF    TCP  File   Mmap  Bcopy  Bcopy  Mem   Mem
                             UNIX      reread reread (libc) (hand) read write
--------- ------------- ---- ---- ---- ------ ------ ------ ------ ---- -----
stu       Linux 5.4.0-9 1824 6215 2987 7290.2  13.1K  11.0K 6888.5 11.K 9420.

Memory latencies in nanoseconds - smaller is better
    (WARNING - may not be correct, check graphs)
------------------------------------------------------------------------------
Host                 OS   Mhz   L1 $   L2 $    Main mem    Rand mem    Guesses
--------- -------------   ---   ----   ----    --------    --------    -------
stu       Linux 5.4.0-9   2108 0.9980 3.0000   16.9       135.2
```

可以看到 L1 cache 延迟为 0.998，L2 cache 延迟为 3，主存延迟为 16.9，Rand mem 延迟 为 135.2

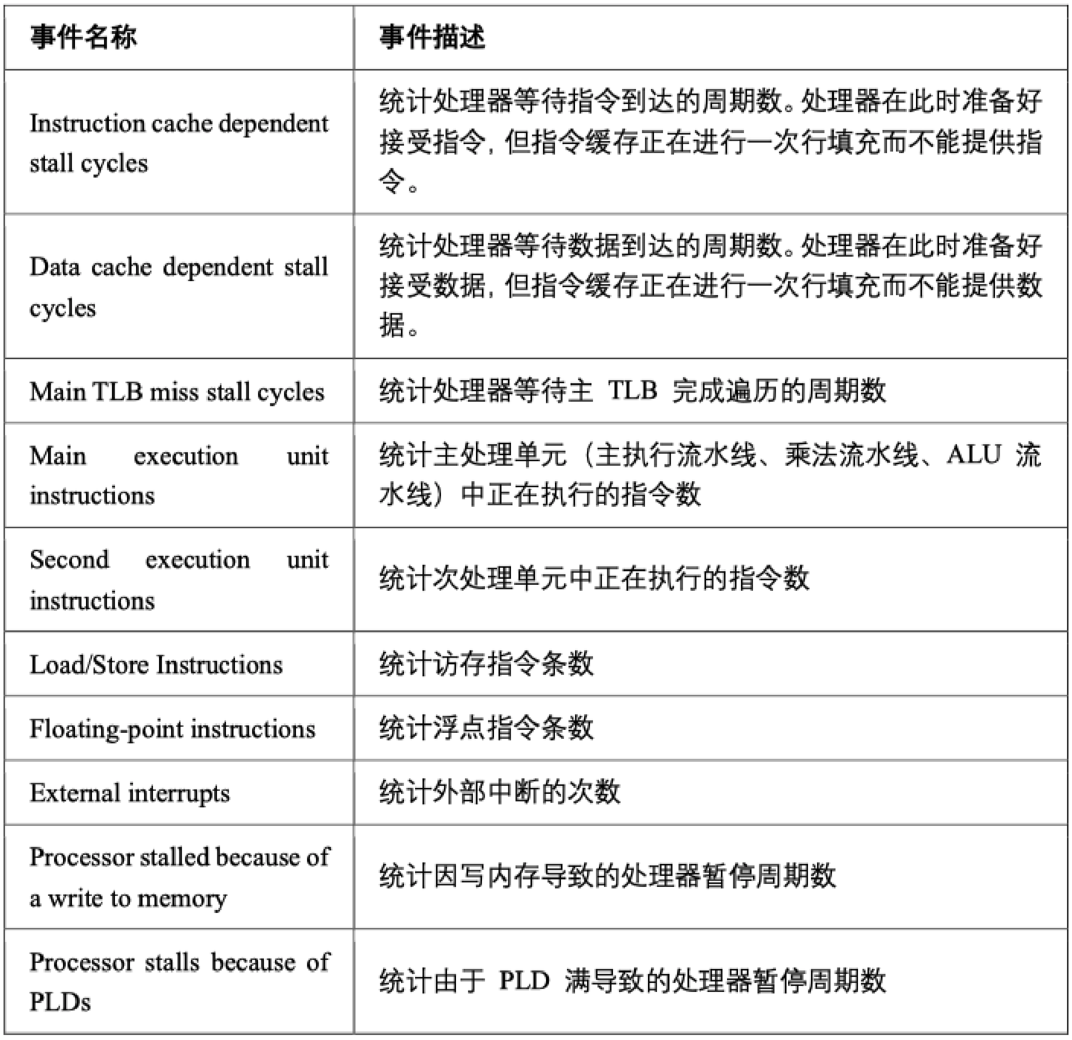
7. 使用 SimpleScalar 模拟器，分析二级 Cache 的延迟对性能的影响（从 24 变到 12 个 假设使用 Alpha 的指令集，测试程序为 SPEC CPU 2000 的 164.bzip 和 253.perlbmk（相应二进制文 件可从网络获取）。

   无力完成

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？

   嵌入式系统与一般桌面系统相比，可用硬件资源更少而对功耗等要求更高，故而嵌入式基准测试 程序可能不过多关注 CPU 设计中的一些优化技术而对功耗等指标进行监测。

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。



事件名称	事件描述
Instruction cache dependent stall cycles	统计处理器等待指令到达的周期数。处理器在此时准备好接受指令，但指令缓存正在进行一次行填充而不能提供指令。
Data cache dependent stall cycles	统计处理器等待数据到达的周期数。处理器在此时准备好接受数据，但指令缓存正在进行一次行填充而不能提供数据。
Main TLB miss stall cycles	统计处理器等待主 TLB 完成遍历的周期数
Main execution unit instructions	统计主处理单元（主执行流水线、乘法流水线、ALU 流水线）中正在执行的指令数
Second execution unit instructions	统计次处理单元中正在执行的指令数
Load/Store Instructions	统计访存指令条数
Floating-point instructions	统计浮点指令条数
External interrupts	统计外部中断的次数
Processor stalled because of a write to memory	统计因写内存导致的处理器暂停周期数
Processor stalls because of PLDs	统计由于 PLD 满导致的处理器暂停周期数

10. 模拟建模的方法和性能测量的方法相比有哪些优点？

    模拟建模以软件的形式模拟计算机系统的运行，在设计阶段即可用于测试分析。性能测量依赖构 建完成的实机系统与受限的计数事件，模拟建模在软件上建模分析，减小测试分析开销并拓宽性能事 件的观测面。

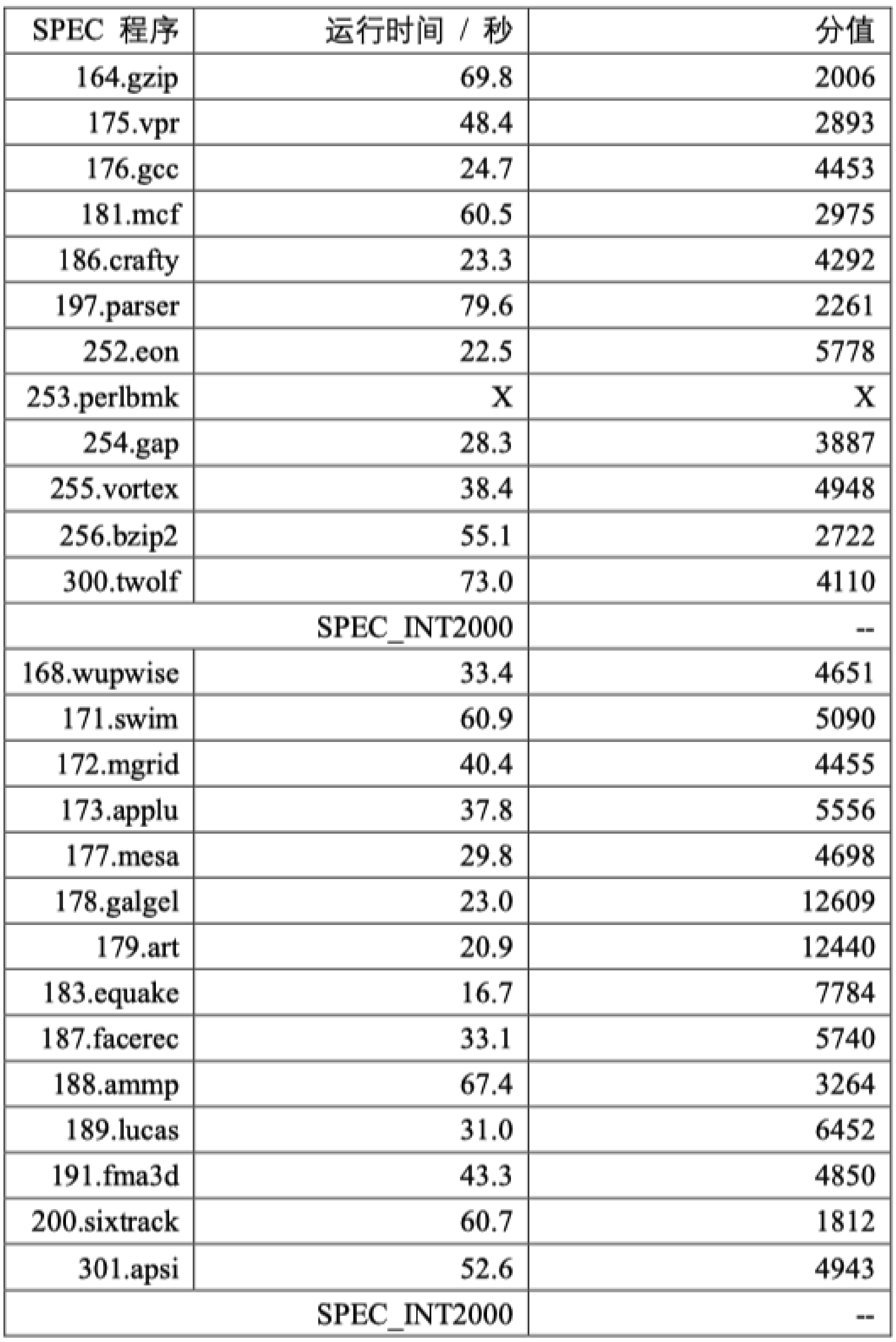
11. SimPoint 的基本原理是什么，为什么其能减少模拟建模的时间？

    SimPoint 为采样模拟技术中的一种采样方法，其首先找到程序执行的相位，然后采样能够代表 每个相位的部分并进行模拟仿真。避免了模拟时的大量重复，从而减少模拟建模的时间。

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？

    校准过程为比较模拟器与实机在运行同一程序时的性能计数器等结果，保证不出现较大、较普遍 的误差。
    评价指标为模拟结果与实机结果的吻合程度。

13. 在你的电脑上运行 SPEC CPU 2000 Rate 并给出分值。



SPEC 程序	运行时间 / 秒	分值
164.gzip	69.8	2006
175.vpr	48.4	2893
176.gcc	24.7	4453
181.mcf	60.5	2975
186.crafty	23.3	4292
197.parser	79.6	2261
252.eon	22.5	5778
253.perlbmk	X	X
254.gap	28.3	3887
255.vortex	38.4	4948
256.bzip2	55.1	2722
300.twolf	73.0	4110
	SPEC_INT2000	--
168.wupwise	33.4	4651
171.swim	60.9	5090
172.mgrid	40.4	4455
173.applu	37.8	5556
177.mesa	29.8	4698
178.galgel	23.0	12609
179.art	20.9	12440
183.equake	16.7	7784
187.facerec	33.1	5740
188.ammp	67.4	3264
189.lucas	31.0	6452
191.fma3d	43.3	4850
200.sixtrack	60.7	1812
301.apsi	52.6	4943
	SPEC_INT2000	--