

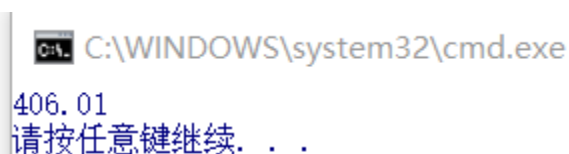
Q1.测量一台计算机系统最大的 MIPS 和最大的 MFLOPS 的值。

测量 MFLOPS:

```
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <algorithm>
#include <cstring>

using namespace std;
const int T=1000000000;
int main()
{
    float a=3.14159265359,b=10.23456789765,c=200.842793857468,d;
    for(int i=0;i<T;i++){
        d=a*b+c;
    }
    printf("%.2f\n",(double)T/(1000000*(double)clock()/CLOCKS_PER_SEC));

    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
406.01
请按任意键继续. . .
```

测试结果:

至于 MIPS 的测量，随便写一个程序（运算由加减法完成），让其循环运行一定的次数（比如 10000 次），统计总用时（记为 T，单位是 s），再反汇编查看共有多少条指令（记为 N），则 $MIPS = 10000 \times N / (1000000 \times T) = N / (10T)$ 。

Q2.

1. 测试结果

```
sai@Computer:~$ ./stream_exe
-----
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 20308 microseconds.
(= 20308 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:          9385.3    0.018933    0.017048    0.024651
Scale:         9094.5    0.019518    0.017593    0.024441
Add:          10676.6    0.024637    0.022479    0.027268
Triad:         8114.7    0.031510    0.029576    0.037256
-----
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-----
sai@Computer:~$
```

2. 理论上，带宽=内存频率*位数/8。

测试中发现，修改 cpu 频率对内存带宽影响不大。查询相关资料后发现，修改 cpu 频率一般来说并不会影响内存频率。CPU 的工作频率总是基于主板总线频率的倍数而倍频得来的，系统运行时，不管主板总线频率和 CPU 的频率有多高，其运行速度总是受限于最小速度的内存频率

3.修改为单精度后的测试结果：

```
precision of your system timer.
-----
Function      Best Rate MB/s  Avg time     Min time     Max time
Copy:          8581.0    0.012221    0.009323    0.018404
Scale:         4810.9    0.018167    0.016629    0.021033
Add:          6799.6    0.020959    0.017648    0.026386
Triad:         4036.7    0.031217    0.029727    0.034840
-----
Solution Validates: avg error less than 1.000000e-06 on all three arrays
-----
sai@Computer:~$
```

可以看出 copy 测试的结果对于时间项减少到三分之二，考虑到复制内容长度变为一半，带宽较原来略微下降。

但是 scale、add、triad 测试时间没有明显变化，而带宽降到原先的一半。说明内存操作是以双精度为单位进行的。

Q3. 462.libquantum:C 语言实现 (C99) 。libquantum 是模拟量子计算机的库文件, 用来进行量子计算机应用的研究。

libquantum 是一个用于模拟量子计算机的库。量子计算机以量子力学原理为基础, 可以在多项式时间内解决某些计算上的难题。1994 年, Peter Shor 发现了一个多项式时间算法用于数字的因式分解, 这是密码分析中一个特别有趣的问题, 因为广泛使用的 RSA 密码系统依赖于质因数分解是一个只能在指数时间内解决的问题。在 libquantum 中包含 Shor 的因数分解算法的实现。

libquantum 提供了一种表示量子寄存器和一些基本门的结构。测量可以用来从系统中提取信息。此外, libquantum 还提供退相干模拟, 这是构建实用量子计算机的最重要障碍。因此不仅可以模拟任何量子算法, 而且可以开发量子误差校正算法。由于 libquantum 允许添加新的门, 因此可以很容易地扩展它以适应正在进行的研究, 例如, 它已被部署用于分析量子密码术。

462 号测试中的重要测试对象是 SIMD 性能。SIMD (单指令流多数据流) 是一种采用一个控制器来控制多个处理器, 同时对一组数据 (又称“数据矢量”) 中的每一个分别执行相同的操作从而实现空间上的并行性的技术。在微处理器中, 单指令流多数据流技术则是一个控制器控制多个平行的处理微元, 例如 Intel 的 MMX 或 SSE 以及 AMD 的 3D Now! 技术。462 测试对微结构的压力也主要是对平行处理微元的测试。

ICC 编译器编译后的运行效果较 gcc 明显加快。查阅相关文档后, 发现在并行运算的程序编译上, ICC 有着更为良好的效果。这与 462 测试中所需要的并行性技术测试也相吻合。

Q4.

使用 perf 工具测量冒泡排序和希尔排序的 IPC

测量结果:

冒泡排序: 2.37

希尔排序: 1.24

排序算法对微结构的压力主要是分支预测。每一次比较的结果基本上来说都是随机的, cpu 基本不可能做到准确率很高的预测。但是, 如果某排序算法能迅速实现局部的有序化, 那么将大大降低预测分支错误的情况。

Q5.linpack 的热点函数:

dgemm 函数: 实现 LU 分解中 A 及 panel 内部矩阵的更新操作, 本质是双精度矩阵间的乘加运算。

dtrsm 函数: 实现 LU 分解中矩阵 U 的更新。

dgemv 函数: 实现对列向量的求解和更新操作, 本质是列向量与矩阵的乘加操作。

Q6.

访存 Lmbench 测试结果:

Memroy latencies in nanoseconds – smaller is better

(WARNING – may not be correct, check graphs)

Host	OS	Mhz	L1 \$	L2 \$	L3 \$	Main mem	Rand mem
192.168.0	Linux 4.4.0	2808	2.9	12.8	25.8	90.6	187.2

192.168.0 Linux 4.4.0	2808	2.8	14.3	26.1	90.2	193.2
192.168.0 Linux 4.4.0	2808	2.5	13.7	26.1	91.4	185.8
192.168.0 Linux 4.4.0	2808	2.6	13.8	25.6	89.2	189.3
192.168.0 Linux 4.4.0	2808	2.6	13.6	25.9	96.1	188.5

Q7.

（由于 simplescalar 一直安装失败，就没有得到测试结果，只能从理论角度分析）

降低二级 cache 的延迟可以减少一级 cache miss，二级 cache hit 时的延迟，可以提高访存带宽。

Q8.

桌面基准测试分为两大类：处理器密集型基准测试和图形密集型基准测试，不过许多图形基准测试中包含大量处理器行为。SPEC 基准测试是一些实际应用程序，这些应用程序经过修改就能够移植，并能在最大程度上降低 I/O 对性能的影响。事实上，桌面基准测试所面向的测试对象主要为通用计算机系统，系统实现的功能较多，面向对象广泛。

嵌入式基准测试（如 EEMBC）测试的对象为嵌入式计算机系统。嵌入式计算机系统同通用型计算机系统相比具有以下特点：

- 1.嵌入式系统通常是面向特定应用的。嵌入式 CPU 与通用型的最大不同就是嵌入式 CPU 大多工作在为特定用户群设计的系统中，它通常都具有低功耗、体积小、集成度高等特点。
- 2 嵌入式系统的硬件和软件都必须高效率地设计，量体裁衣、去除冗余，力争在同样的硅片面积上实现更高的性能
- 3.为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存贮于磁盘等载体中。

因此,嵌入式基准测试更注重于机器在某一特定领域上的功能。对于桌面基准测试程序而言，通用机如果对基准程序的测试方法进行特定的优化，性能未必提升，但是最终得分很可能会有较大提高。而对于嵌入式来说，针对测试程序的优化就是对问题处理真实情况的优化,所以不存在虚假测试的问题。

对于关注重点，桌面基准测试更注重全面性，嵌入式更注重专一性。

Q9.

下表为 ARM CortexA-57 的硬件性能计数器表格

Table 11-5 Common Event Identification Register 0 bit assignments

Bit	Name	Event number	Value	Event implemented if bit set to 1 or not implemented if bit set to 0
[31]	-	0x1F	0	Reserved, RES0.
[30]	CH	0x1E	1	Chain. ^a An odd-numbered counter increments when an overflow occurs on the preceding even-numbered counter. For even-numbered counters, does not count.

Bit	Name	Event number	Value	Event implemented if bit set to 1 or not implemented if bit set to 0
[29]	BC	0x1D	1	Bus cycle.
[28]	TW	0x1C	1	TTBR write, architecturally executed, condition check pass - write to translation table base.
[27]	IS	0x1B	1	Instruction speculatively executed.
[26]	ME	0x1A	1	Local memory error.
[25]	BA	0x19	1	Bus access.
[24]	DC2W	0x18	1	Level 2 data cache Write-Back.
[23]	DC2R	0x17	1	Level 2 data cache refill.
[22]	DC2A	0x16	1	Level 2 data cache access.
[21]	DC1W	0x15	1	Level 1 data cache Write-Back.
[20]	IC1A	0x14	1	Level 1 instruction cache access.
[19]	MA	0x13	1	Data memory access.
[18]	BP	0x12	1	Predictable branch speculatively executed.
[17]	CC	0x11	1	Cycle.
[16]	BM	0x10	1	Mispredicted or not predicted branch speculatively executed.
[15]	UL	0x0F	0	Instruction architecturally executed, condition check pass - unaligned load or store.
[14]	BR	0x0E	0	Instruction architecturally executed, condition check pass - procedure return.
[13]	BI	0x0D	0	Instruction architecturally executed - immediate branch.
[12]	PW	0x0C	0	Instruction architecturally executed, condition check pass - software change of the PC.
[11]	CW	0x0B	1	Instruction architecturally executed, condition check pass - write to CONTEXTIDR.
[10]	ER	0x0A	1	Instruction architecturally executed, condition check pass - exception return.

Bit	Name	Event number	Value	Event implemented if bit set to 1 or not implemented if bit set to 0
[9]	ET	0x09	1	Exception taken.
[8]	IA	0x08	1	Instruction architecturally executed.
[7]	ST	0x07	0	Instruction architecturally executed, condition check pass - store.
[6]	LD	0x06	0	Instruction architecturally executed, condition check pass - load.
[5]	DT1R	0x05	1	Level 1 data TLB refill.This event is implemented.
[4]	DC1A	0x04	1	Level 1 data cache access.
[3]	DC1R	0x03	1	Level 1 data cache refill.
[2]	IT1R	0x02	1	Level 1 instruction TLB refill.
[1]	IC1R	0x01	1	Level 1 instruction cache refill.
[0]	SI	0x00	1	Instruction architecturally executed, condition check pass - software increment.

表中列出了所有可被记录的性能事件。其中比较重要的有：
访存错误、分支指令被正确推测的执行、分支指令被错误推测的执行、例外次数、一、二级 cache 访问、一、二级 cache 重填等等。

Q10.

性能分析的目的是测量平台及微结构和软件及数据结构之间的交互行为。
模拟建模的途径有踪迹驱动模拟、执行驱动模拟、全系统模拟、事件驱动模拟和统计方法模拟等。性能测量的方法有片上硬件检测、片外硬件监测、软件检测等。
模拟建模最大的优点是开销小。由于不需要实现真实的测试对象和测试器具，模拟建模可以省下大量的人力物力和时间。并且模拟建模能很好的捕捉用户态和内核态的行为特性。另外，建模可以忽略硬件实现细节的情况下考虑其对性能的影响, 从而量化的分析影响处理器性能的因素，具有速度快、灵活性高的特点。

Q11.

SimPoint 是离线阶段聚类分析的一个特定用途，其动机是执行高效和准确的程序分析和架构模拟，学术界和 Intel 的一些研究人员正在使用 SimPoint 来准确地指导架构模拟研究。
离线阶段分析的第一步是快速分析正在执行的代码的频率，以创建代码签名，这些签名代表了在执行过程中的不同点上程序的行为。然后将这些签名(向量)与聚类分析中的一些技术一起使用，以简明地将程序执行的相似部分分组到各个阶段。然后可以使用这些信息来了解整

个程序的行为、特定执行阶段的目标优化，并通过只使用有代表性的样本来大大减少模拟时间。

SimPoint 使用离线分析生成的相位聚类来智能地选择在何处使用模拟时间。有许多不同的方法来使用 SimPoint 和提供的一些算法。在最高级别，SimPoint 计算程序/输入对的阶段，然后从每个阶段中选择一个代表。然后，用户只能对这些选定的代表执行程序分析或执行详细的模拟。然后，将这些统计信息组合在一起，就可以完整而准确地表示程序的全部执行情况。每个阶段(即一组间隔)的代表是通过找到最接近 clusterA 中心(形心)的间隔来选择的。此阶段的选定间隔称为该阶段的模拟点。然后，整个程序的行为是通过权衡每个仿真点的性能结果，通过它所来自的相位的大小(间隔数)来估计的。SimPoint 能够显著减少程序分析和仿真时间，并提供完整程序的准确描述。

Q12.

模拟器和真实机器的校验：

对于已有系统，校验精度可以通过和实际系统运行结果对比；对于正在实现的系统可以通过和 RTL 代码模拟的结果进行对比。经过比较之后可以得到模拟器的精度，可以结合真实结果对模拟器的参数进行重新调整。这样校正过后，模拟器在模拟未来的系统时，虽不能做到完全准确，但是和当今已有系统的相对性能比较还是可以反映的出来。

Q13.

UX.

Benchmarks	Base Ref Time	Base Run Time	Estimated Base Ratio	Peak Ref Time	Peak Run Time	Estimated Peak Ratio
168.wupwise	1600	--	X			
171.swin	3100	--	X			
172.ngrid	1800	--	X			
173.applu	2100	--	X			
177.mesa	1400	47.6	2941	*		
177.mesa	1400	45.6	3073	X		
177.mesa	1400	47.7	2936	X		
178.galgel	2900	--	X			
179.art	2600	21.8	11981	*		
179.art	2600	21.6	12038	*		
179.art	2600	21.1	12318	*		
183.equake	1300	17.8	7293	*		
183.equake	1300	17.7	7332	*		
183.equake	1300	17.7	7336	*		
187.facerec	1900	--	X			
188.amp	2200	61.6	3573	X		
188.amp	2200	61.7	3567	*		
188.amp	2200	61.7	3568	*		
189.lucas	2000	--	X			
191.fna3d	2100	--	X			
200.sixtrack	1100	--	X			
301.apsl	2600	--	X			
168.wupwise			X			
171.swin			X			
172.ngrid			X			
173.applu			X			
177.mesa	1400	47.6	2941	*		
178.galgel			X			
179.art	2600	21.6	12038	*		
183.equake	1300	17.7	7332	*		
187.facerec			X			
188.amp	2200	61.7	3568	*		
189.lucas			X			
191.fna3d			X			
200.sixtrack			X			
301.apsl			X			
Est. SPECfp_base2000			--			
Est. SPECfp2000			--			
Benchmarks	Base Ref Time	Base Run Time	Estimated Base Ratio	Peak Ref Time	Peak Run Time	Estimated Peak Ratio
164.gzip	1400	75.3	1858	*		
164.gzip	1400	83.3	1681	*		
164.gzip	1400	90.8	1556	*		
175.vpr	1400	54.1	2588	*		
175.vpr	1400	51.1	2737	*		
175.vpr	1400	56.6	2472	*		
176.gcc	1100	33.3	3306	*		
176.gcc	1100	33.4	3292	*		
176.gcc	1100	33.4	3289	*		
181.mcf	1800	64.2	2805	*		
181.mcf	1800	61.5	2927	*		
181.mcf	1800	65.8	2734	*		
186.crafty	1000	39.2	2553	*		
186.crafty	1000	38.6	2590	*		
186.crafty	1000	41.8	2437	*		
197.parser	1800	100	1796	*		
197.parser	1800	100	1794	*		
197.parser	1800	102	1757	*		
252.eon	1300	49.1	2646	*		
252.eon	1300	51.1	2545	*		
252.eon	1300	50.6	2567	*		
253.perlbmk	1800	--	X			
254.gap	1100	71.9	1530	*		
254.gap	1100	72.0	1528	*		
254.gap	1100	71.4	1540	*		
255.vortex	1900	63.2	3008	*		
255.vortex	1900	62.8	3024	*		
255.vortex	1900	61.6	3083	*		
256.bzip2	1500	64.5	2324	*		
256.bzip2	1500	64.8	2344	*		
256.bzip2	1500	64.8	2315	*		
300.twolf	3000	82.2	3652	*		
300.twolf	3000	82.7	3630	*		
300.twolf	3000	82.6	3631	*		
164.gzip	1400	83.3	1681	*		
175.vpr	1400	54.1	2588	*		
176.gcc	1100	33.4	3292	*		
181.mcf	1800	64.2	2805	*		
186.crafty	1000	39.2	2553	*		
197.parser	1800	100	1794	*		
252.eon	1300	50.6	2567	*		
253.perlbmk			X			
254.gap	1100	71.9	1530	*		
255.vortex	1900	62.8	3024	*		
256.bzip2	1500	64.5	2324	*		

```

HARDWARE
-----
Hardware Vendor:
Model Name: i7
CPU: i7
CPU MHz: 2400
FPU: Integrated
CPU(s) enabled: 1
CPU(s) orderable: 1
Parallel: No
Primary Cache: 64KBI + 64KBD on chip
Secondary Cache: 8192KB(I+D) on chip
L3 Cache: N/A
Other Cache: N/A
Memory: 2 x 512 PC3200 DDR SDRAM CL2.0
Disk Subsystem: IDE, WD2000
Other Hardware: None

SOFTWARE
-----
Operating System: Ubuntu for x86
Compiler: --
File System: Linux/ext3
System State: Multi-user SuSE Run level 3
```

SPEC 2000在VBOX Ubuntu虚拟机下
运行 all 的时间测试结果.