

**实验报告**

**实 验（六）**

题 目 Cachelab

高速缓冲器模拟

专 业 计算机类

学　　 号 1190200526

班　　 级 1903002

学 生 沈城有

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.5.28

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc73109640)

[1.1 实验目的 - 3 -](#_Toc73109641)

[1.2 实验环境与工具 - 3 -](#_Toc73109642)

[1.2.1 硬件环境 - 3 -](#_Toc73109643)

[1.2.2 软件环境 - 3 -](#_Toc73109644)

[1.2.3 开发工具 - 3 -](#_Toc73109645)

[1.3 实验预习 - 3 -](#_Toc73109646)

[第2章 实验预习 - 4 -](#_Toc73109647)

[2.1 画出存储器层级结构，标识容量价格速度等指标变化（5分） - 4 -](#_Toc73109648)

[2.2计算机Cache的参数查看与分析（5分） - 4 -](#_Toc73109649)

[2.3写出各类Cache的读策略与写策略（5分） - 5 -](#_Toc73109650)

[2.4写出用gprof进行性能分析的方法（5分） - 5 -](#_Toc73109651)

[2.5写出用Valgrind进行性能分析的方法（5分） - 6 -](#_Toc73109652)

[第3章 Cache模拟与测试 - 7 -](#_Toc73109653)

[3.1 Cache模拟器设计 - 7 -](#_Toc73109654)

[3.2 矩阵转置设计 - 7 -](#_Toc73109655)

[第4章 总结 - 8 -](#_Toc73109656)

[4.1 请总结本次实验的收获 - 8 -](#_Toc73109657)

[4.2 请给出对本次实验内容的建议 - 8 -](#_Toc73109658)

[参考文献 - 8 -](#_Toc73109659)

# 第1章 实验基本信息

## 1.1 实验目的

理解现代计算机系统存储器层级结构；

掌握Cache的功能结构与访问控制策略；

培养Linux下的性能测试方法与技巧；

深入理解Cache组成结构对C程序性能的影响。

## 1.2 实验环境与工具

### 1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

### 1.2.2 软件环境

Windows7 64位以上；VirtualBox/Vmware 11以上；Ubuntu 16.04 LTS 64位/优麒麟 64 位

### 1.2.3 开发工具

Visual Studio 2010 64位以上；GDB/OBJDUMP；DDD/EDB等

## 1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT或PDF）。

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

画出存储器的层级结构，标识其容量价格速度等指标变化；

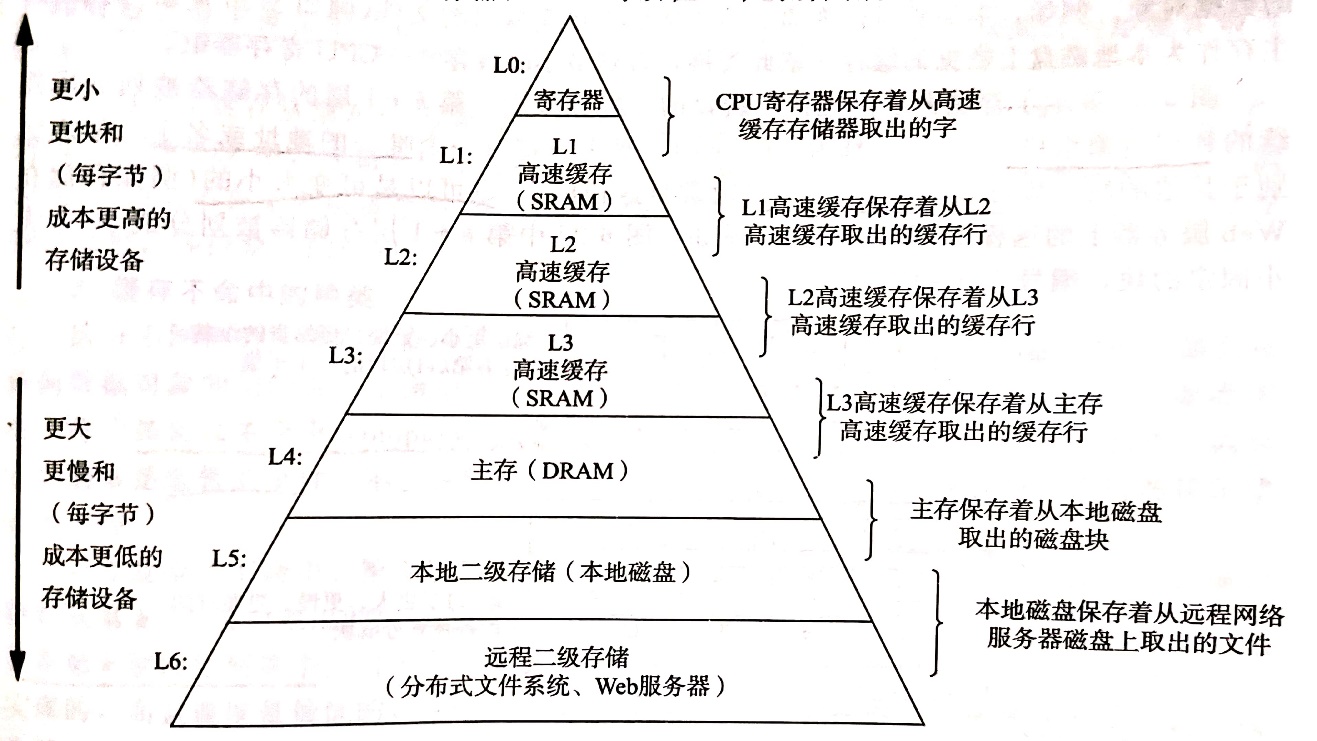
用CPUZ等查看你的计算机Cache各参数，写出Cache的基本结构与参数：缓存大小C、分组数量S、关联度/组内行数E、块大小B，及对应的编码位数：组索引位数s、块内偏移位数b；

写出Cache的各种读策略与写策略；

掌握Valgrind、gprof的使用方法。

# 第2章 实验预习

## 2.1 画出存储器层级结构，标识容量价格速度等指标变化（5分）



## 2.2计算机Cache的参数查看与分析（5分）

用CPUZ等查看你的计算机Cache的参数，写出各级Cache的C(大小)、 S(组数)、 E(路数)、 B(块大小)，并分析相应的s（组编码位数）、b（块内偏移地址位数）数值。

查看到的参数如下图：



L1数据缓存（单核）： C:32KB S:64 E:8 B:64 s:6 b:6

L1指令缓存（单核）： C:32KB S:64 E:8 B:64 s:6 b:6

L2缓存（单核）： C:256KB S:1024 E:4 B:64 s:10 b:6

L3缓存（公共）： C:8MB(8192KB) S:8192 E:16 B:64 s:13 b:6

## 2.3写出各类Cache的读策略与写策略（5分）

**Cache读策略：**

1. 缓存命中，则直接从缓存中读取数据到CPU或上一层cache中；
2. 缓存不命中，则从主存或下一级cache中读取数据，若cache未满，则将取到的数据放在空闲块中；若cache已满，则需要根据替换策略（LRU等）替换一个现有的块。

**Cache写策略：**

如果写一个已经缓存了的字（写命中），有两种策略：

1. 直写：在本级Cache中进行写操作，并立即将修改后的块写回到低一层中；
2. 写回：在本级Cache中修改，并尽可能推迟更新，只有当替换算法要驱逐这个更新过的块时，才将其写到低一层中；

如果写不命中，又分两种策略：

1. 写分配：加载相应的低一层的块到高速缓存中，然后更新这个高速缓存块；
2. 非写分配：避开高速缓存，直接把这个字写到低一层中。

## 2.4写出用gprof进行性能分析的方法（5分）

gprof是GNU profile工具，可以进行C、C++、Pascal、Fortran程序的性能分析，用于程序的性能优化以及程序瓶颈问题的查找和解决。通过分析应用程序运行时产生的“flat profile”，可以得到每个函数的调用次数、每个函数消耗的处理器时间，也可以得到函数的“调用关系图”，包括函数调用的层次关系，每个函数调用花费了多少时间。使用步骤如下：

（1）用gcc、g++、xlC编译程序时，使用-pg参数，如：g++ -pg -o test.exe test.cpp编译器会自动在目标代码中插入用于性能测试的代码片断，这些代码在程序运行时采集并记录函数的调用关系和调用次数，并记录函数自身执行时间和被调用函数的执行时间。

（2）执行编译后的可执行程序，如：./test.exe。该步骤运行程序的时间会稍慢于正常编译的可执行程序的运行时间。程序运行结束后，会在程序所在路径下生成一个缺省文件名为gmon.out的文件，这个文件就是记录程序运行的性能、调用关系、调用次数等信息的数据文件。

（3）使用gprof命令来分析记录程序运行信息的gmon.out文件，如：gprof test.exe gmon.out则可以在显示器上看到函数调用相关的统计、分析信息。上述信息也可以采用gprof test.exe gmon.out > gprofresult.txt重定向到文本文件以便于后续分析。

## 2.5写出用Valgrind进行性能分析的方法（5分）

Valgrind是运行在Linux上一套基于仿真技术的程序调试和分析工具，它包含一个内核——一个软件合成的CPU和一系列的小工具，每个工具都可以完成一项任务——调试，分析，或测试等。

一些常用的选项如下：

-h --help

显示帮助信息。

--version

显示valgrind内核的版本，每个工具都有各自的版本。

-q --quiet

安静地运行，只打印错误信息。

-v --verbose

打印更详细的信息。

--tool= [default: memcheck]

最常用的选项。运行valgrind中名为toolname的工具。如果省略工具名，默认运行memcheck。

--db-attach= [default: no]

绑定到调试器上，便于调试错误。

# 第3章 Cache模拟与测试

## 3.1 Cache模拟器设计

提交csim.c

程序设计思想：

要求设计一个cache模拟器，此模拟器可模拟输入参数s、E、b为合法值时使用LRU策略驱逐块的cache。实验包中已经提供该模拟器的C语言程序主体框架，还需要补充一些函数的代码并在主函数中利用命令行参数计算S、E、B的值。

各部分具体的设计思路如下（代码及注释见提交的源代码文件）：

**（1）void initCache()函数：**

要求为Cache分配内存，并将Cache各块的各项均初始化为0，最后计算用于后续获取组索引的值set\_index\_mask。

为整个Cache分配内存，即分配S \* sizeof(cache\_set\_t)字节（S组的指针）；为每一组分配内存，每组E \* sizeof(cache\_line\_t)字节（每组E行）；

随后均初始化为0。

最后计算set\_index\_mask，即为(1 << s) – 1，产生低s位均为1的值。

**（2）void freeCache()函数：**

要求释放分配的空间，反向对应initCache()分配过程即可，先释放每一组，最后释放整个Cache。

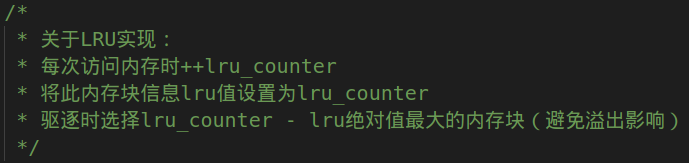
**（3）void accessData()函数：**

此函数模拟访问内存的过程，主要考虑三种情况：

* 命中
* 不命中，有空闲块可用
* 不命中，无空闲块，使用LRU策略驱逐块

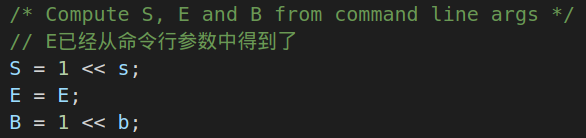
首先是遍历所有组寻找匹配的块，如果命中，则为第一种情况，更新lru，++hit\_count并输出信息（可选）后退出；如果未命中，++miss\_count，输出信息（可选）并遍历所有组检查是否有空闲块(valid = 0)可用，若有，则为第二种情况，将此空闲块设为有效，保存tag信息并更新lru，退出程序。若没有，则为第三种情况，要使用LRU策略选择块驱逐，++enviction\_count，同样更新此块的信息，最后退出。

我认为重点在于LRU策略的编程实现，我的思路如下：

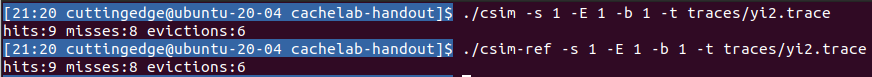


（4）计算S、E、B：

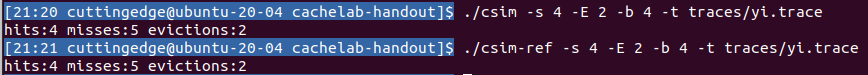
此过程较简单，简单移位即可，如下图：



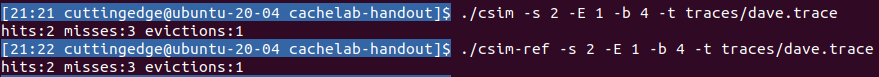
测试用例1的输出截图（5分）：



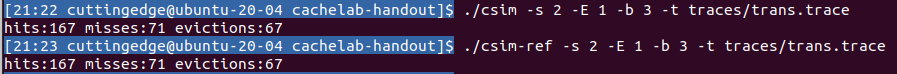
测试用例2的输出截图（5分）：



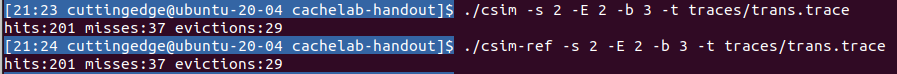
测试用例3的输出截图（5分）：



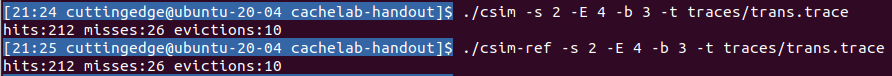
测试用例4的输出截图（5分）：



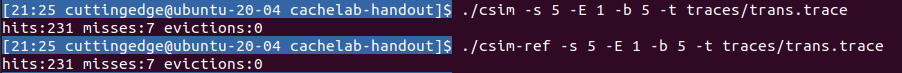
测试用例5的输出截图（5分）：



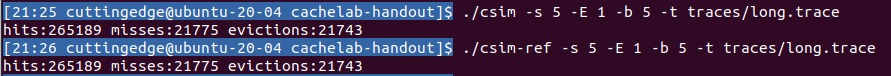
测试用例6的输出截图（5分）：



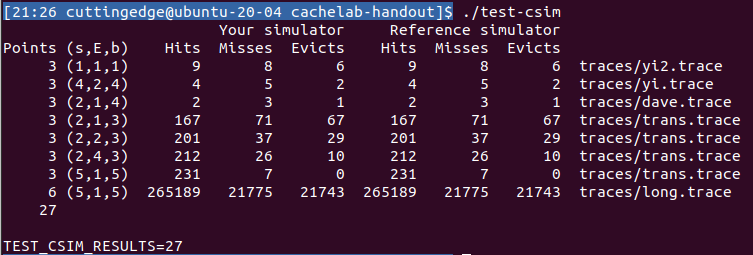
测试用例7的输出截图（5分）：



测试用例8的输出截图（10分）：



附test-csim（自动评分程序）运行结果：



**注：每个用例的每一指标5分（最后一个用例10分）**——与参考csim-ref模拟器输出指标相同则判为正确。

## 3.2 矩阵转置设计

提交trans.c

程序设计思想：

**32×32（10分）：运行结果截图**

**64×64（10分）：运行结果截图**

**61×67（20分）：运行结果截图**

# 第4章 总结

## 4.1 请总结本次实验的收获

## 4.2 请给出对本次实验内容的建议

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.