

**实验报告**

**实 验（六）**

题 目 Cache Lab

高速缓冲器模拟

专 业 计算机专业

学　　 号 1190200501

班　　 级 1903002

学 生 林燕燕

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.05.28

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc500230541)

[1.1 实验目的 - 3 -](#_Toc500230542)

[1.2 实验环境与工具 - 3 -](#_Toc500230543)

[1.2.1 硬件环境 - 3 -](#_Toc500230544)

[1.2.2 软件环境 - 3 -](#_Toc500230545)

[1.2.3 开发工具 - 3 -](#_Toc500230546)

[1.3 实验预习 - 3 -](#_Toc500230547)

[第2章 实验预习 - 4 -](#_Toc500230548)

[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 -](#_Toc500230549)

[2.2用CPUZ等查看你的计算机Cache各参数，写出各级Cache的C S E B s e b（5分） - 4 -](#_Toc500230550)

[2.3写出各类Cache的读策略与写策略（5分） - 5 -](#_Toc500230551)

[2.4 写出用gprof进行性能分析的方法（5分） - 5 -](#_Toc500230552)

[2.5写出用Valgrind进行性能分析的方法（（5分） - 6 -](#_Toc500230553)

[第3章 Cache模拟与测试 - 7 -](#_Toc500230554)

[3.1 Cache模拟器设计 - 7 -](#_Toc500230555)

[3.2 矩阵转置设计 - 8 -](#_Toc500230556)

[第4章 总结 - 10 -](#_Toc500230557)

[4.1 请总结本次实验的收获 - 10 -](#_Toc500230558)

[4.2 请给出对本次实验内容的建议 - 10 -](#_Toc500230559)

[参考文献 - 11 -](#_Toc500230560)

# 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；1.6GHz；8G RAM；256G SSD Disk；1T HDD Disk

### 1.2.2 软件环境

Windows10 64位；Vmware 14pro；Ubuntu 20.04.2 LTS 64位

### 1.2.3 开发工具

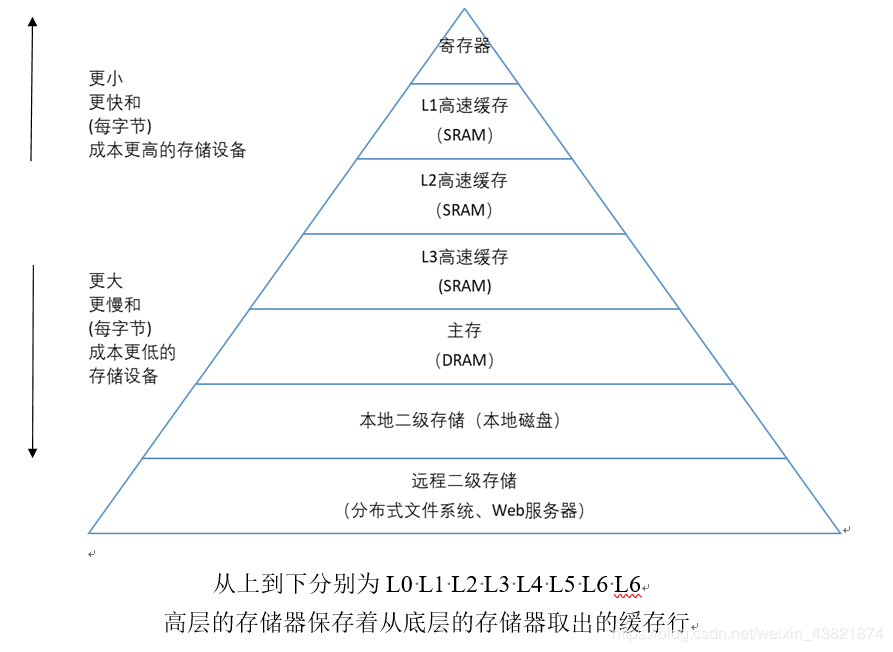
Visual Studio Code 64位；vim/gpedit+gcc

## 1.3 实验预习

* 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、 e 、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

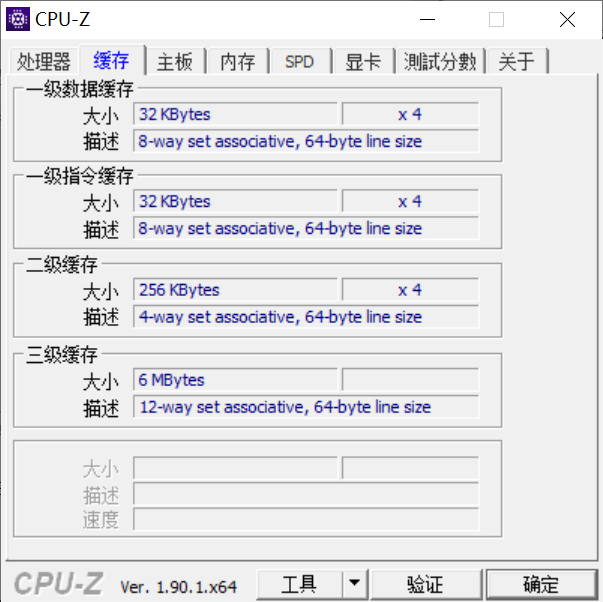
# 第2章 实验预习

## 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。



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 各级缓存 | C  大小 | S  组数 | E  路数 | B  块大小 | s  组编码位数 | b  块内偏移地址位数 |
| 一级数据缓存 | 128KB | 256 | 8 | 64 | 8 | 6 |
| 一级指令缓存 | 128KB | 256 | 8 | 64 | 8 | 6 |
| 二级缓存 | 1024KB | 4096 | 4 | 64 | 12 | 6 |
| 三级缓存 | 6MB | 8192 | 12 | 64 | 13 | 6 |

## 2.3写出各类Cache的读策略与写策略（5分）

一、Cache读策略：

1：命中，则从cache中读相应数据到CPU或上一级cache中。

2：未命中，则从主存或下一级cache中读取数据，存储在当前层级cache。

二、Cache写策略：

1：命中：

（1）写回法：只写本级cache，暂时不写数据到主存或下一级cache，将该行替换出去后，将数据写回到主存或下一级cache。

（2）写直达：写本级cache，同时写数据到主存或下一级cache，将该行替换出去后，不用写回数据。

2：未命中：

（1）按写分配，载入到缓存中，并更新缓存。

（2）写不分配：直接写数据到主存或下一级cache，不载入到缓存。

## 2.4 写出用gprof进行性能分析的方法（5分）

通过分析gprof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，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，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，可以在显示器上看到函数调用相关的统计、分析信息。

## 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：

-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。

# 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，完成void initCache(), void freeCache(), void accessData(mem\_addr\_t addr)的代码并在main函数中计算S、E、B的值。

1、void initCache()：

为cache分配内存malloc(S \* sizeof(cache\_set\_t))，将valid、lru、tag写为0，并计算set\_index\_mask。

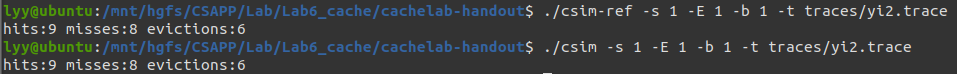
2、void freeCache()：

将initCache()中申请的空间全部释放，先释放每一组的空间，再释放整个cache的空间。

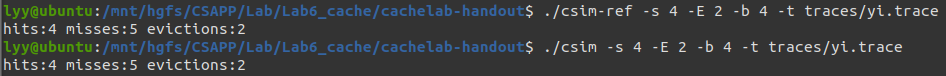
3、void accessData(mem\_addr\_t addr)：

访问内存地址中的数据，若命中即valid=1且tag相等，hit\_count++；若不命中， miss\_count++；若有一个有效位为0，则未满，若有效位全为1，则要驱逐，eviction\_count++。

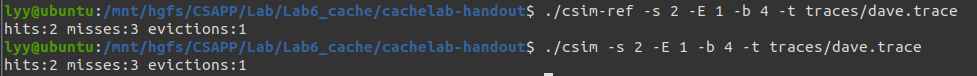
测试用例1的输出截图（5分）：



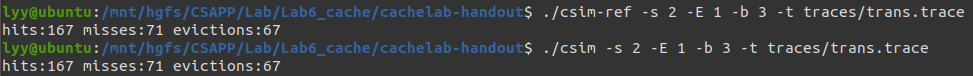
测试用例2的输出截图（5分）：



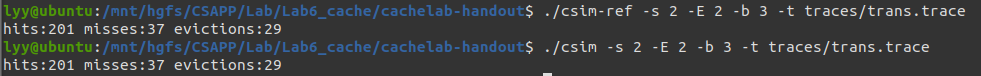
测试用例3的输出截图（5分）：



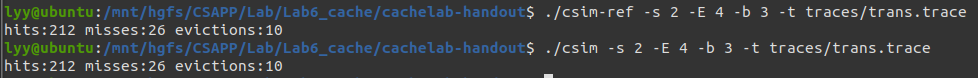
测试用例4的输出截图（5分）：



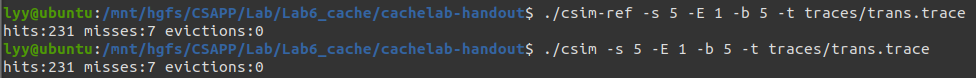
测试用例5的输出截图（5分）：



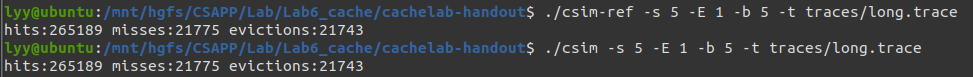
测试用例6的输出截图（5分）：



测试用例7的输出截图（5分）：

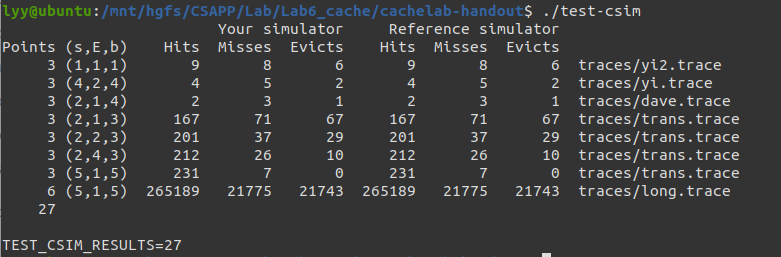


测试用例8的输出截图（10分）：



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

执行test-csim结果：



## 3.2 矩阵转置设计

提交trans.c

程序设计思想：

cache 的参数为s=5、E=1、b=5，即对于这个缓存，有32个组，每组有一个块，块的大小为32个字节，一个块共能装入8个int变量，cache能装入8\*32=256个int变量。

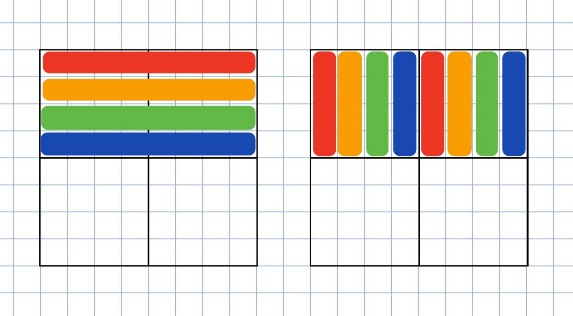
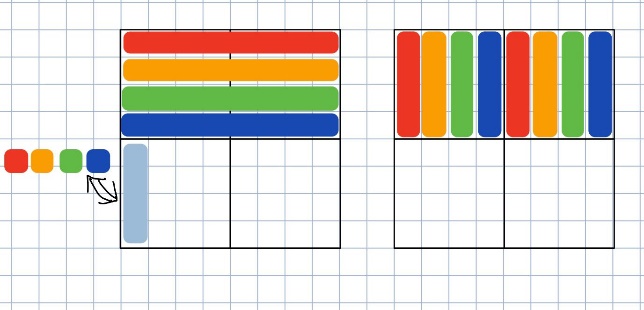
1、32\*32矩阵

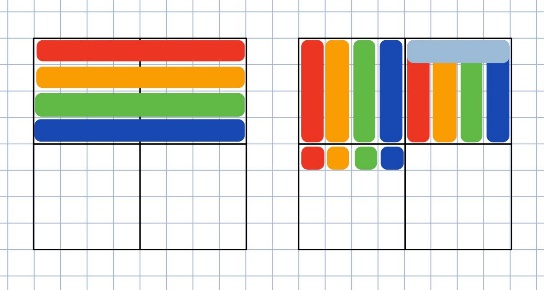
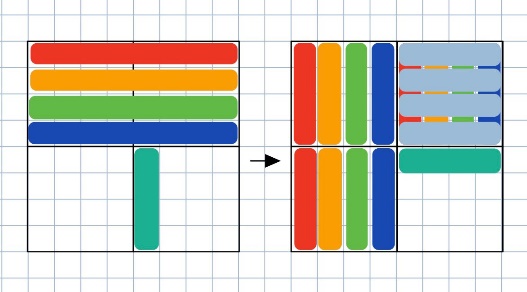
为了最小化缓存缺失的数量，考虑将数组分块成8\*8，将每块依次转置。

2、64\*64矩阵

8\*8与4\*4相结合，将矩阵按8\*8分块，将每一块按4\*4分块，分成左上、右上、左下、右下。

将A的8\*8分块的左上进行转置操作放在B的左上，将A的右上转置放在B的右上；将A的左下依次存入t[0]- t[4]，t[0]- t[4]与B的右上进行交换，再将交换后的t[0]- t[4]存入B的左下；将A的右下转置存入B的右下。

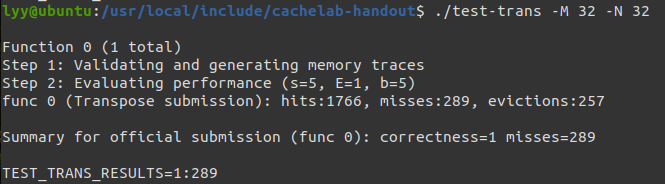
 

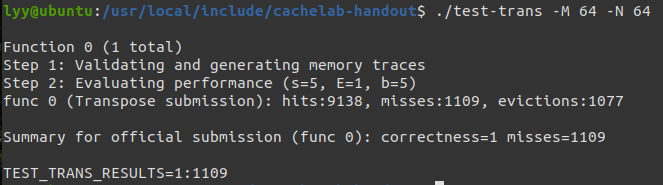
3、61\*67矩阵

将矩阵分块，分块大小可变，经测试，设置为23，miss大小符合要求。

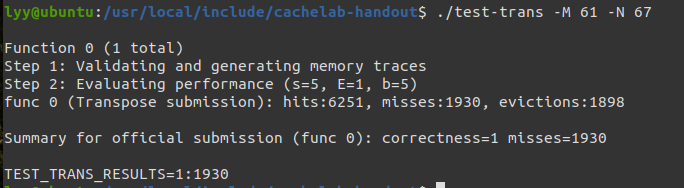
**32×32（10分）：运行结果截图**

****

**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

了解了缓存的相关结构及知识，对缓冲命中的原理有了深入理解；

学会了通过对代码的优化实现增加缓存命中率的方法。

## 4.2 请给出对本次实验内容的建议

希望ppt可以更详细一些。

注：本章为酌情加分项。

# 参考文献

[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.