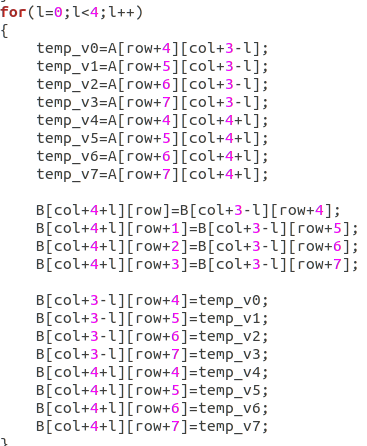
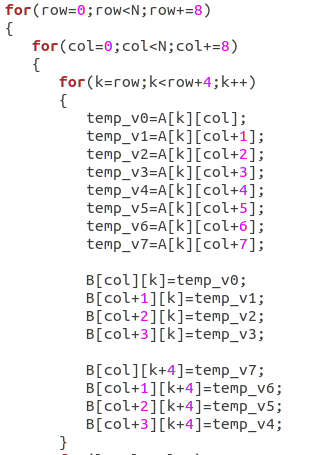
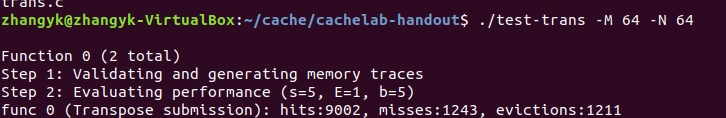
实验日志

2.7：

1. 实验test-trans以M64N64矩阵为例，编写代码优化trans.c中转置函数，采用8分块处理过程和记录miss结果。





1. 追踪分析M64N64的组索引，分析8分块优化处理过程。

（1）8分块时，除了A[0][0]冲突miss，其余情况都命中，存入矩阵b的第一列前四个未命中，即将a的第一行四个元素赋值给b的第一列四个元素，b的前四行存入cache，然后存入b的第4列的四个元素，hit，完成a的第一行转置。

（2）重复（1）， 直到a的前四行转置存入b。

（3）从cache中载入a的第三列后四个元素，miss，从第四列后四个开始，hit。

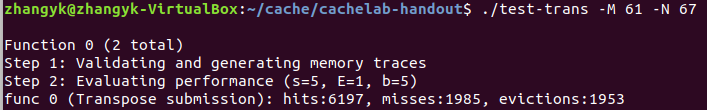
（4）向cache存入b的第四行前三个元素，实现b的右上角的4\*4存入左下角4\*4。

（5）cache存入b的右上、右下4\*4时，hit。

重复此过程完成矩阵转置。

1. 编写代码优化M61N67的trans.c，尝试更多分块并记录miss数目减少结果，选取其中较好方案。





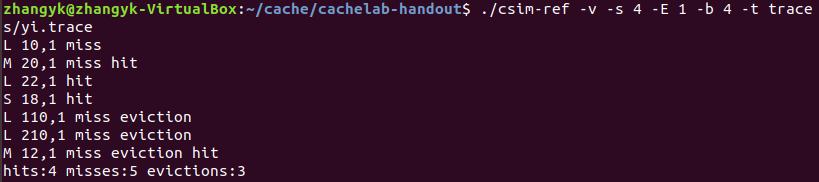
2.2：

1. 整理实验信息。回顾个人日志2.4~2.7，描述实验目标、资源、实验步骤(含关键命令行及说明)。

第一周任务回顾：

安装了valgrind，搭建实验环境；打开cachelab，找到可运行的文件并输入命令，尝试不同参数；利用valgrind输入，打印出最终的hit（命中）、miss（不命中）、evictions（驱逐数）。

运行缓存模型-v -s 4 -E 1 -b 4：



编写parta中csim文件的部分函数，使自己的模拟器可呈现部分结果，初始化及帮助信息。

第二周任务：

编写模拟器计算部分的函数，实现更新lru计数值，更新cache；指令如下：



观察每一行的有效值、标记位、lru，查看更新。

编写加载数据l，存储数据s，修改数据m命令的函数，并添加函数，输出组索引和标记位，检查lru主函数代码并与参考的csim-ref的结果对比，结果相同则正确。

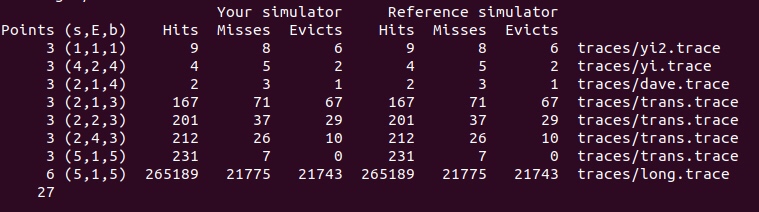
第三周任务：

通过csim-ref跟踪文件trace.f文件（设置M32N32）,命令行./test-trans -M 32 -N 32，生成跟踪文件trace.f1；使用csim跟踪trace.f1，命令行./csim -v -s 5 -E 1 -b 5 -t trace.f1，观察结果。分别以4\*4，32\*32阵为例分析参数变化原因。并对4分块，8分块的代码进行对角线优化，最终提高命中率。

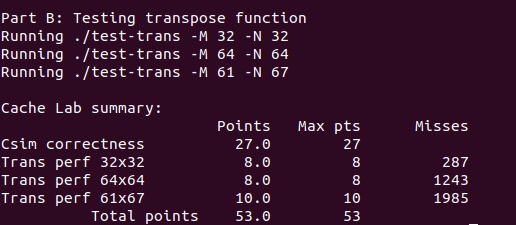
第四周任务2.7:

以64\*64,61\*67矩阵为例，分析结果，并找到最佳优化分块方式。

1. 实验结果。检验(driver. py) PartA和PartB两部分参照评分标准的结果。解释实验现象截图:如何看出实验正确?



设置s，E，b为不同参数，对文件进行跟踪，对比我的模拟器数据与参考模拟器的数据，判断hits，misses，evicts结果是否正确。



测试32\*32,64\*64，61\*67的评分情况，均为最高分，结果显示都有优化效果。

1. 实验总结。在实验室做实验过程中，你获取的课堂外的新知识、学习方法、技能、人文生活等方面。

在这次实验中，对cache有了更加深入的了解，对于cache的存储方式有了清晰的认识，在自己编写模拟器时，对于cache的运行方式有所了解，对相关学习和接下来的实验内容有很大的帮助，测试了不同的参数后，验证实验正确就可以证明模拟器的基本功能满足实验要求。

接下来的partB部分，用不同的方式实现矩阵转置，并分析原理，在此过程中，对于cache的运行方式有了具体并且直观的认识，生成跟踪文件，用csim追踪时，结果在不同的分块优化条件下都有不同，这也说明了不同的优化方式中，不一定每一种都有很高的效率，这与cache的运行有关，找到有效优化方式的过程就是建立在理解cache的基础上，提高对它的利用率，从而起到优化作用。

实验也让我熟悉了虚拟机的更多操作以及对c语言的应用，实验过程中也有很多与同学的交流，这是我同样收获了很多不同的知识和解决问题的思路。