# 第一次实习任务报告

## 1 任务一

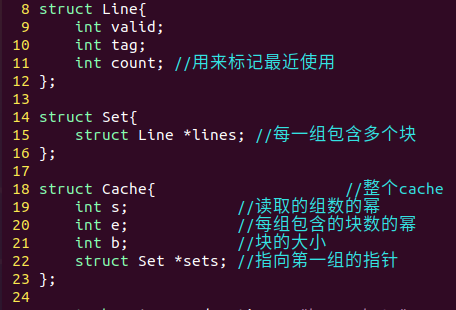
* 1. **任务简述**

该任务需要我们使用C语言编写一个cache的模拟器，最后会读取不同的文件并且显示命中次数。

**1.2实现方式**

**1.2.1 定义的数据结构**

这里我一共定义了三个数据结构，分别是cache，set，line。

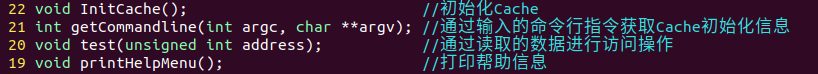


**图1 定义的数据结构**

根据cache的定义，在cache结构体中定义了三个整型变量用来存储输入的三个参数s，e，b，最后定义的时指向第一组的结构体指针。Set结构体代表了一个组，里面定义了一个line结构体指针用来指向组内第一行数据。Line结构体代表了一行数据，里面定义了一个valid位代表存储的数据是否有效，tag位用来判断是否命中，最后的count用来进行LRU算法的计数器。

**1.2.2 实现函数介绍**

我一共定义了四个功能函数具体定义以及功能如下：

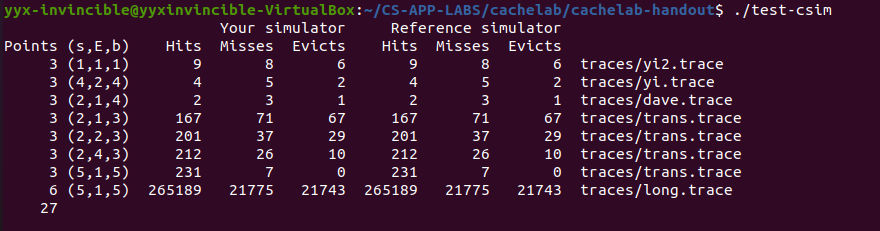


**图2 定义的功能函数**

在main函数里面依次对以上的四个函数进行调用就可以完成cache的模拟工作流程。特别说明的是，在初始化的时候，由于我调用了malloc函数申请了一段地址连续的存储空间。为了访问方便，我之后访问每一行使用的是数组的访问方式。

**1.2.3 实验结果以及分析**

在终端中执行测试文件可以得到模拟结果如下：



**图3 任务一测试结果**

从图中可以看到我编写的csim和资源中所给的csim-ref文件的执行结果各项参数是一样的，从而说明了我编写的程序的正确性。

这个任务乍看上去比较复杂，但是由于我在计算机组成原理的课程中已经学习过了cache的全相联组相联的存储格式以及各种淘汰算法，因此在编写程序的时候我思路十分的清晰，基本上没有遇到什么困难。但是在调试的过程中，由于我是在vim中编写的程序，没有了windows中的IDE环境，我只能先熟悉gdb的使用。我遇到的报错是段错误（核心已转储）但是单是根据这个报错很难看出究竟有什么问题，只有先使用gcc编译后再在gdb中设置断点进行单步调试我才发现了之前的逻辑错误导致了我的数组访问越界了，linux对于访问越界会出现上面的报错。

## 2 任务二

**2.1 任务简述**

通过数组按行存储的特性，根据cache访问的特性对一个矩阵完成转置操作。

2.2 实现方式

这里我只用修改trans.c文件中的transpose\_submit函数。

**2.2.1 32\*32的矩阵**

第一个任务是32\*32的矩阵，一个int型变量占用了四个字节，题目中说明了使用的cache类型，我们可以计算出一个cache行可以存储8个int型变量。又因为s=5，e=1代表了对于一个32\*32的矩阵来说，前八行可以一次性把整个cache装满因此我选择的策略是把32\*32的矩阵分割成8\*8的小矩阵，依次按行访问每一个小矩阵，完成整个大矩阵的转置。

**2.2.2 64\*64的矩阵**

第二个任务是64\*64的矩阵，由于前四行就可以装满整个cache了，这里我首先使用8\*8的切割法进行了尝试，效果和不切割的几乎没太大差别，接着我使用了4\*4切割法，但是最少只能到1656次左右，无法拿到满分要求的1300以内，于是我想到了结合我上面的两种尝试。依旧是采用8\*8进行大的分割，但是会把8\*8分成上半部分和下半部分4\*8两块，再在每个4\*4的小矩阵中完成内部的转置，最后再把左下角的4\*4矩阵和右上角的4\*4矩阵进行交换来完成最终的转置。

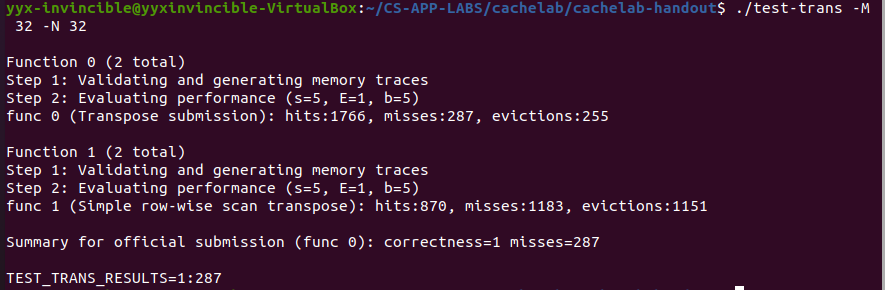
这样的设计保证了在读取的时候利用了8\*8矩阵按行读取时命中率高的特点，但是我最后通过对8\*8矩阵内部的两个4\*4矩阵进行交换实现了彻底的转置。

2.2.3 61\*67的矩阵

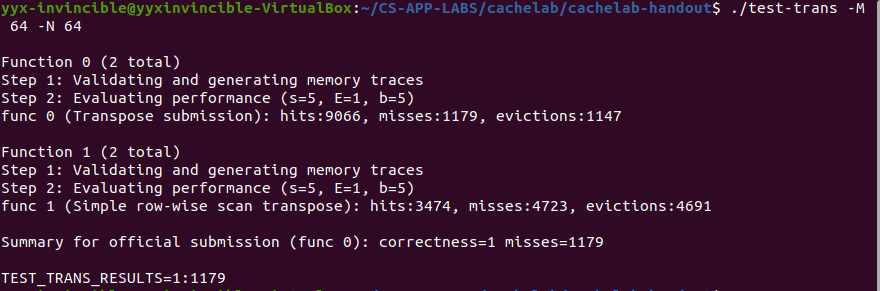
**2.2.3 61\*67的矩阵**

第三个任务是不规则的矩阵转置，这个我直接使用了尝试法，发现分割成16\*16时的效果已经可以达到了满分misses低于2000的要求。

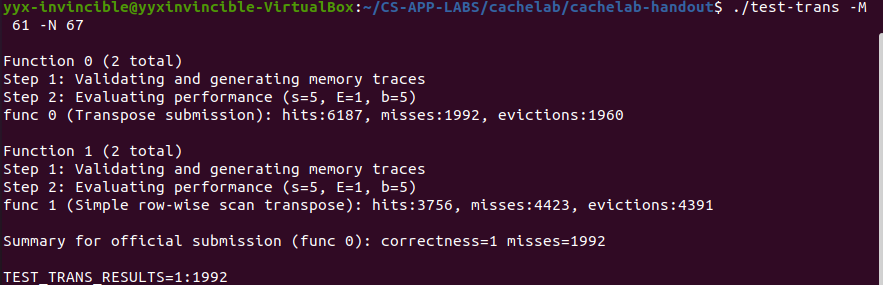
**2.2.4 实验结果以及分析**



**图4 32\*32的矩阵测试结果**



**图5 64\*64的矩阵测试结果**



**图6 61\*67的矩阵测试结果**

通过以上的测试结果可以看到我完成了所有的实验内容并且都达到了满分要求。

任务二的代码量很少，但是需要我动脑子想的时间更多，尤其时再任务二的第二个64\*64矩阵，我原本准备直接用4\*4切割方案先完成了再说，但是后来一直动脑子考虑了很久之后才想到了把两种方式结合起来的方案，最终达到了满分的要求，感觉整个过程还是十分有趣的。

源代码见：https://github.com/yux20000304/CS-APP-LABS/tree/master/cachelab