

**计算机系统结构实验报告**

|  |
| --- |
| 姓 名：梁一飞 |
| 学 院：计算机科学与技术 |
| 专 业：计算机科学与技术 |
| 班 级：CS1706 |
| 学 号：U201714762 |
| 指导教师：万继光 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2020 年 4月23日

**目 录**

[1. Cache模拟器实验 3](#_Toc38569577)

[**1.1.** **实验目的** 3](#_Toc38569578)

[**1.2.** **实验环境** 3](#_Toc38569579)

[**1.3.** **实验思路** 3](#_Toc38569580)

[**1.4.** **实验结果和分析** 5](#_Toc38569581)

[2. 总结和体会 6](#_Toc38569582)

[3. 对实验课程的建议 6](#_Toc38569583)

# Cache模拟器实验

* 1. **实验目的**

1理解cache工作原理。  
2加深cache缓存组成结构对c程序性能的影响的理解。

* 1. **实验环境**

操作系统类型：Ubuntu 18.04.4 LTS

操作系统版本：linux4.4.214

CPU：Intel（R） Core（TM） i7-6700HQ

* 1. **实验思路**

1.cache的模拟：

给出cache行结构定义如图1所示：

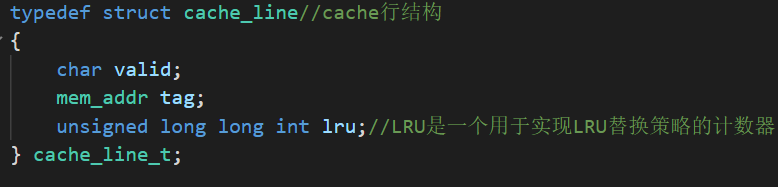


图1.cache行结构体定义

其中valid表示cache行内数据是否有效，tag：内存地址的高位部分，表示cache行中数据对应哪一个内存区。

同时定义cache组和cache如下：

typedef cache\_line\_t\* cache\_set\_t;//表示一个cache组

typedef cache\_set\_t\* cache\_t;//表示多个cache组构成的一个完整cache

实现模拟cache 的逻辑是：用一个cache\_line的数组来模拟一个cache组：cache\_set，然后用一个cache\_set的数组来模拟整个cache。

2.访存地址：

对于访存的一个地址，一个内存地址具有如表1所示形式， index用于确定内存地址映射到的组，tag用于在组里用全相联的方式寻找所在的cache行， offset为行内偏移地址

|  |  |  |
| --- | --- | --- |
| tag | index | offset |

表1.访存地址

3.输入输出参数：

给出的代码中对于程序执行所需的输入输出参数的定义如图2所示：

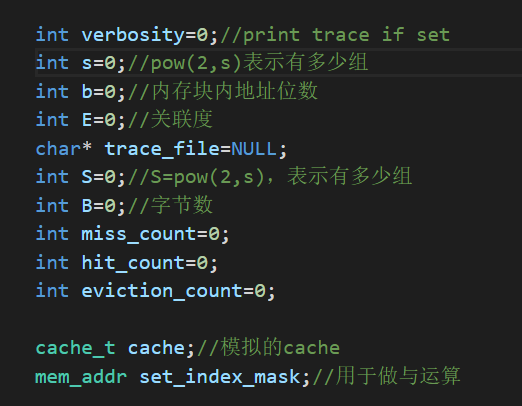


图2.参数

trace\_file是一个文件指针。指向需要测试的轨迹文件,miss\_count,hit\_count和eviction\_count分别表示缺失次数，命中次数和淘汰次数。cache全局变量是模拟的cache，空间需要根据输入的参数动态的分配。set\_index\_mask是为了通过与运算得到某个内存地址对应的cache组,其余见注释。

3.初始化和清零：

分别有initCache函数和freeCache函数实现。在initCache中，先分配了一个cache组指针构成的数组，然后每一个cache数组在分配物理空间。在freeCache中，需要先释放每一个cache组的物理空间，然后释放存放每一个cache组指针的空间。

4.cache访问数据过程模拟：

由accessData函数实现，函数流程图如图3所示：

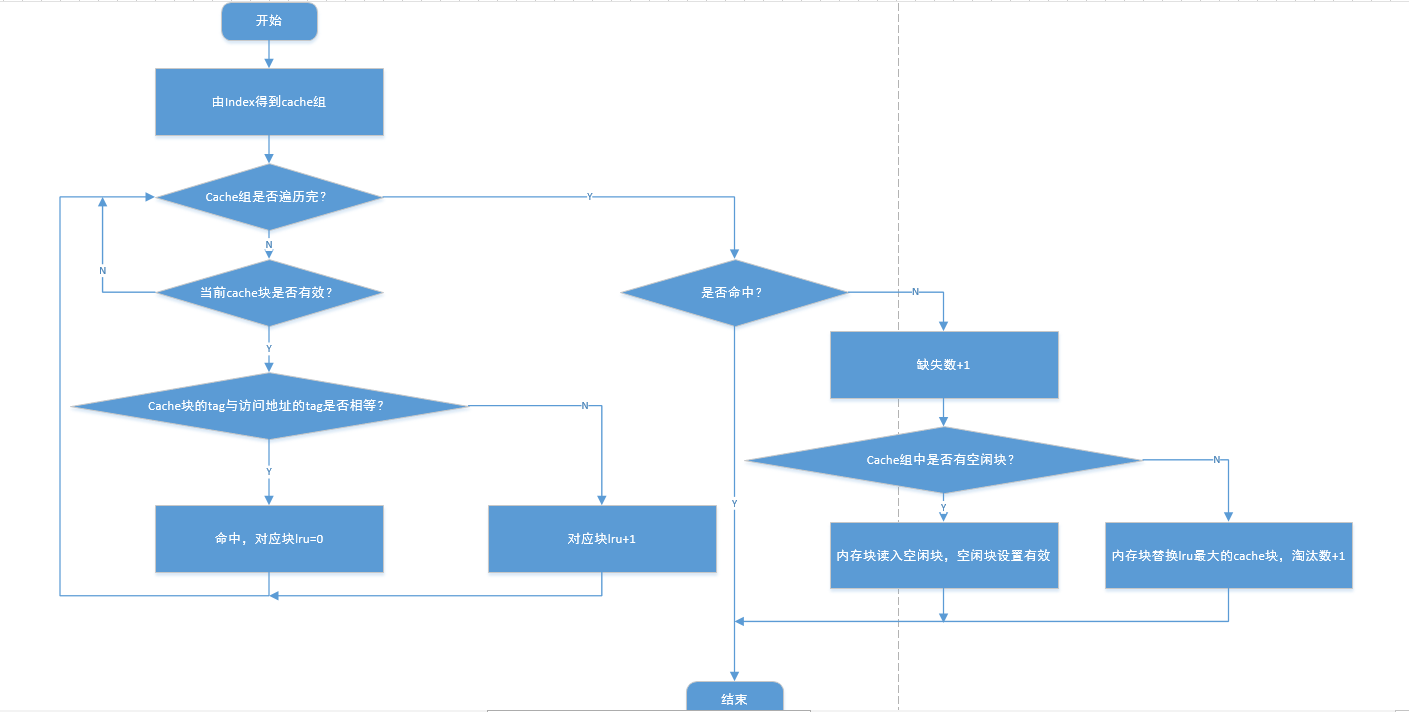


图3.accessData流程图

当需要访问一个内存地址时，根据内存地址的index可以找到其cache映射块的组号，然后在组内遍历每一个cache块。如果遇到有效的cache块，则比较cache块的tag位与访问地址的tag位是否相等，相等则说明cache命中，这时需要将对应cache块的计时器lru重新清零，如果不命中则将该cache块的计时器lru加1，如果是无效的cache块则看不对其进行任何操作。  
如果在整个cache组内没有找到对应的数据块。则说明cache块缺失，需要从内存中将数据块调入，如果cache组内有空闲的块，则选择一块空闲块将内存数据存入，然后将这个cache快标记为有效块。如果cache组满了，则需要淘汰掉最近未使用的cache块，选出计时器lru最大的一块，然后将数据替换

5. 文件操作

由replayTrace函数实现，用来读取trace轨迹文件并根据文件内容模拟内存访问的过程。其中有三个参数，operation：表示需要执行的操作指令，addr：表示指令所访问的内存地址，len：表示指令访问的地址空间的长度。依次读取每一行的三个参数，根据进来的参数进行处理：由于内存访问不会超过边界，所以可以忽略第三个参数len。如果operation是I，说明会访问数据cache，这时不会访问数据cache。如果是M，表示需要进行依次读操作和依次写操作，调用两次accessData函数即可。如果是S或者L，则只需要进行一次数据访问，调用一次accessData即可。

* 1. **实验结果和分析**

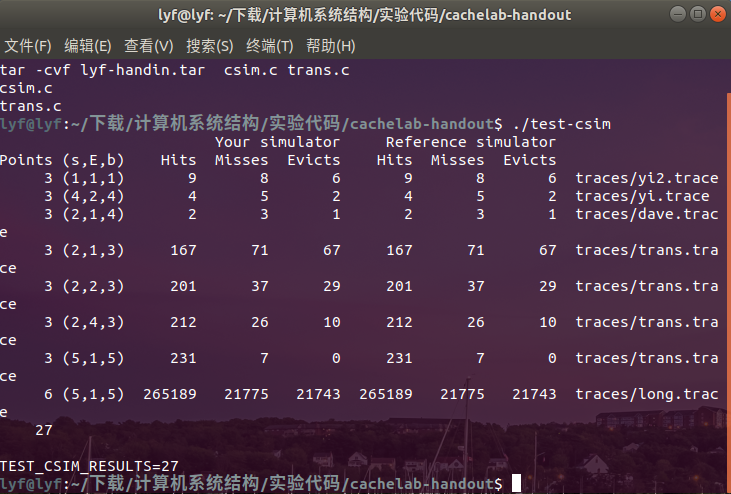
在中断make编译，运行test-csim测试程序测试，运行结果如图4所示：  


图4.实验结果

由图可知：测试结果与标答一致，故实现的cache模拟程序是正确的

# 总结和体会

这次实验进一步加深了我对cache的理解，复习了cache工作的基本工作原理和流程。之前在组成原理课程中用logisim实现过cache，这次用代码形式模拟实现，感觉比直接画电路要轻松一些，整个实验有很多收获，既结合了课堂上的理论知识，又通过动手加深了理解，我觉得是一个非常好的实验。

# 对实验课程的建议

线上教学不如线下实验指导充分，希望给出更详细的实验文档。