华中科技大学

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

1.1. 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。

1.2. 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

1.3. 实验思路

1.cache 的模拟：

给出 cache 行结构定义如图 1 所示：

```
typedef struct cache_line//cache行结构
{
    char valid;
    mem_addr tag;
    unsigned long long int lru;//LRU是一个用于实现LRU替换策略的计数器
} cache_line_t;
```

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：

```
int verbosity=0;//print trace if set
int s=0;//pow(2,s)表示有多少组
int b=0;//内存块内地址位数
int E=0;//关联度
char* trace_file=NULL;
int S=0;//S=pow(2,s)，表示有多少组
int B=0;//字节数
int miss_count=0;
int hit_count=0;
int eviction_count=0;

cache_t cache;//模拟的cache
mem_addr set_index_mask;//用于做与运算
```

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：

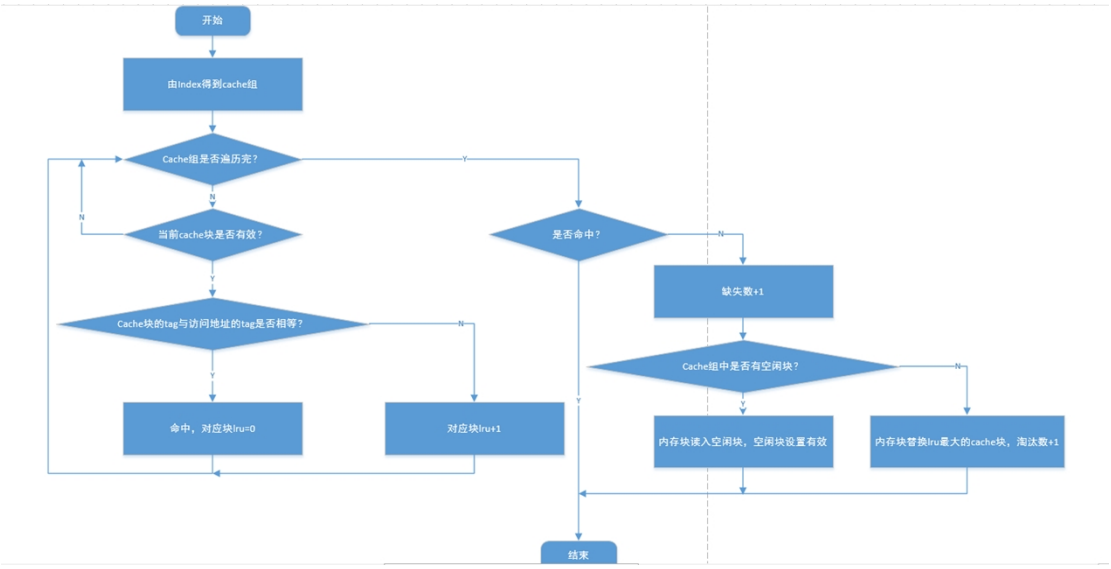

图 3.accessData 流程图

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。

1.4. 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

2. 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。

3. 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。