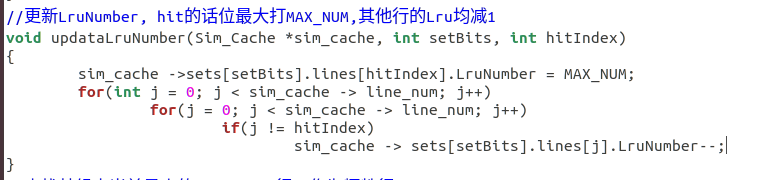
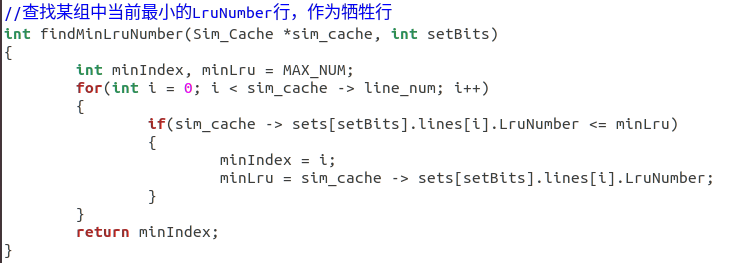
实验日志

1. **编写更新Lru数值的代码（10%）**



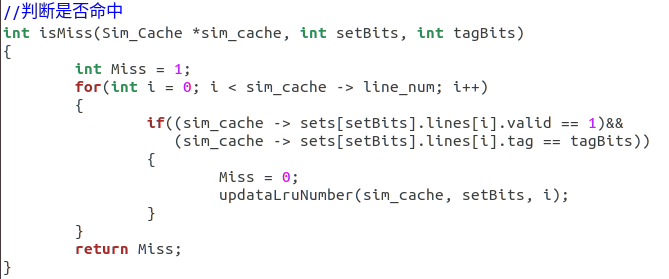
如图：当某组中的某行hit或加载缓存成功时，将这一行的Lru计数值赋值为最大值，而该组中其他行Lru数值全部减1。

1. **编写查找某组牺牲行函数的代码。（10%）**



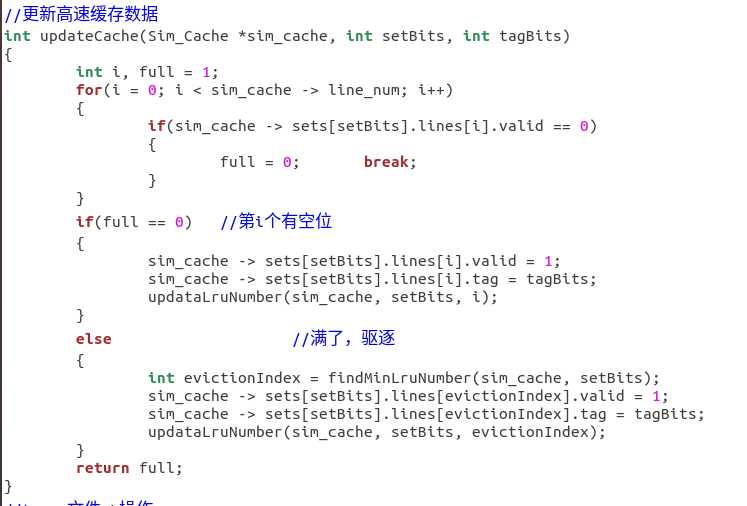
如图：遍历该组中的所有行，找出这组的Lru计数值最小的一行，作为牺牲行。

1. **编写命中判断的函数代码(10%）**



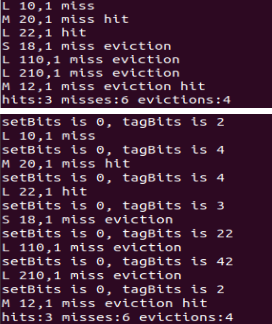
如图：命中要符合：组索引匹配，标记位（tag）相同该行有效位为1，否则就是不命中。命中后更新Lru计数值。

1. **编写更新高速缓存cache的函数代码（10%)**



如图：新加入的数据如果在组索引匹配位置有空位，就直接加入，否则找到Lru值最小的执行驱逐操作。组的每行的有效位全为1表示全满。加载新的数据块要更新有效位为1，更新标记位，更新Lru计数值。

1. **检验LRU主函数代码编写与结果分析（20%）**



如图:上图为正确的csim-ref,下图为自己写的csim，在csim中添加了一个组序列和标志位的输出语句，现在在结果正确的前提下可以通过输出信息进行Lru算法行为分析

L 10,1 中， setBits为0，tagBits 为2 此时没有满，miss后直接存入

M 20,1 中， setBits为0，tagBits 为4 此时没有满，miss后直接存入，第二次hit

L 22,1 中， setBits为0，tagBits 为4 此时tag与第二条相同，hit

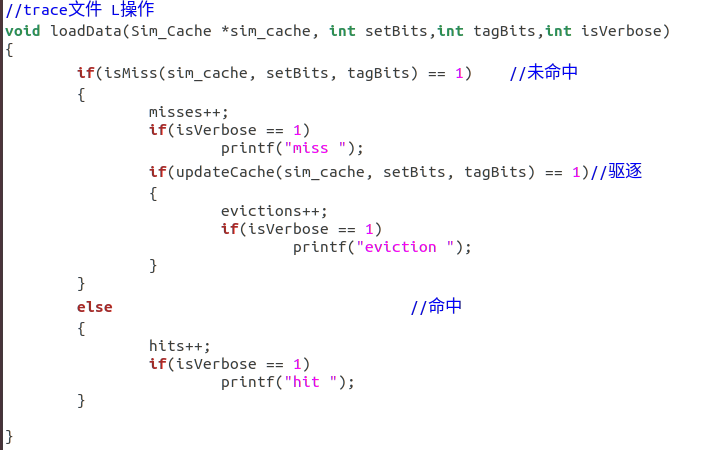
S 18,1 中， setBits为0，tagBits 为3 此时满员，miss后驱逐第一条

L 110,1 中， setBits为0，tagBits 为22 此时满员，miss后驱逐第二条

L 210,1 中， setBits为0，tagBits 为42 此时满员，miss后驱逐第四条

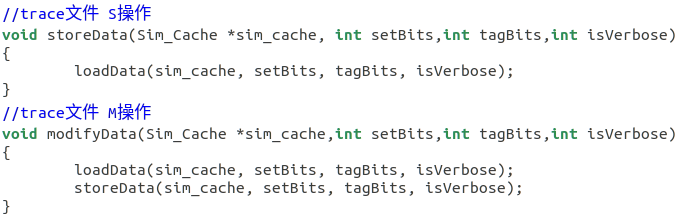
M 12,1 中， setBits为0，tagBits 为2 此时满员，miss后驱逐第五条，第二次hit

1. **编写加载数据的L命令处理函数代码（10%）**



如图：L指令，本质是对数据的一次访问，首先在cache中检索是否命中，命中直接输出hit，否则输出miss，接着判断是否需要驱逐。

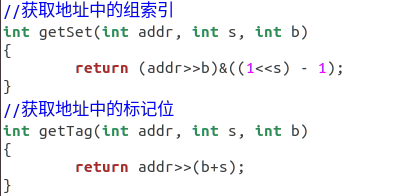
1. **编写存储数据的S命令处理函数和修改数据的M命令的处理函数（5%）**



S也相当于对数据的一次访问，因此在S指令中调用一次L函数

而M指令相当于对数据的两次访问，因此在M指令中调用一次L函数和一次S函数

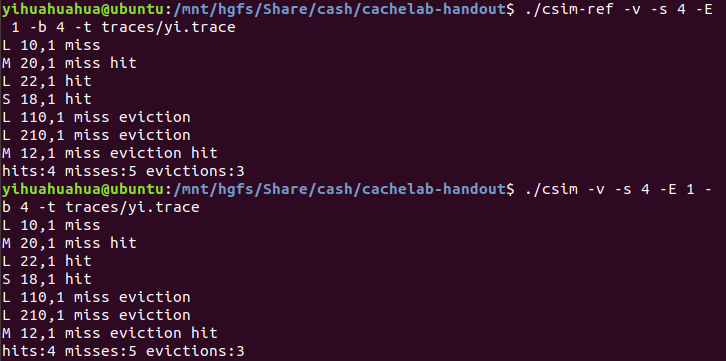
1. **编写获取trace脚本操作地址中的组索引与标记位的函数（10%）**

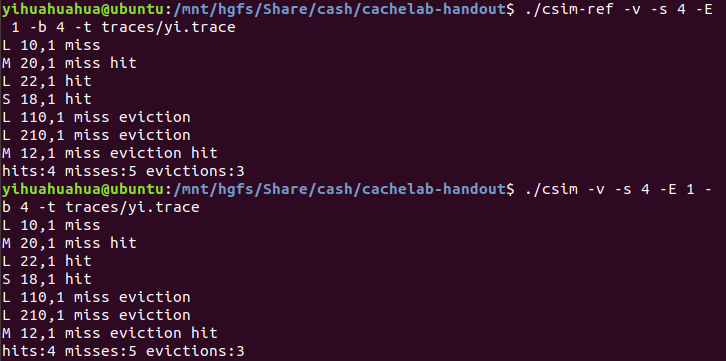


函数如图所示，getSet的原理是将数据右边b位的无关数据剔除，接着与s位长的”111…111”做与运算，保留了底s位。

getTag的原理是将数据右边b+s位的无关数据剔除即可。

1. **检验LRU主函数代码编写与结果分析。（15%）**





如图：上图为cism-ref下图为csim在之前的csim.c基础上，进一步编写主函数，完全正确!