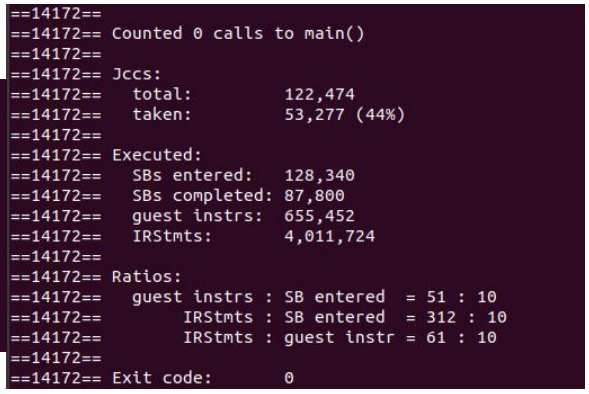
实验日志

一，valgrind 的安装

执行命令：sudo apt-get install valgrind 安装（左图）



安装完成（左图）

执行命令：valgrind --log-fd=1 --tool=lackey -v --trace-mem=yes ls -l（右图）

验证成功（右图）。

二，学会 PartA 部分参考引用 csim-ref 模拟器使用命令行参数及使用方法

格式：./csim-ref [-hv] -s <s> -E <E> -b <b> -t <tracefile>，详细介绍见参考书

使用方法：

linux> ./csim-ref -s 4 -E 1 -b 4 -t traces/yi.trace 打印出 hit、miss、eviction 次数

linux> ./csim-ref -v -s 4 -E 1 -b 4 -t traces/yi.trace 打印出详细信息

三，认识 I、L、S、M 操作含义，并运行在缓存模型上，逐条分析文件操作且对比 LRU 替换策略，能分析 hit、miss、eviction 结果产生过程

“I”表示指令加载，“L”表示数据加载，“S”表示数据存储，“M”表示数据修改（即数据存储之后的数据加载）

分析：

执行的指令设置偏移位的宽度为 4，故可计算出组索引s 的值

✧ **L 10,1 对于地址 0x10 进行访问：**

x10=0001 0000，s=1;

初始时 cache 是空的，第一次访问为 miss。

✧ **M 20,1 数据修改（连续对地址 0x20 进行连续两次访问）：**

0x20=0010 0000，s=2;

第一次访问时没有要的内容，miss；然后cache 从低一级存储器读取第一次访问需要的内容。

第二次访问时有需要的内容且标记位相等，hit；

✧ **L 22,1 对地址 0x22 进行访问：**

0x22=0010 0100，s=2;

由于 M 20,1 以将该块存入高速缓存且标记位都相等，故 hit;

✧ S 18,1 对地址 0x18 进行访问：

0x18=0001 0100,s=1；

由于之前的操作，该块已存入高速缓存且标记位都为 0，hit

✧ L 110,1 对地址 0x110 进行访问：

0x110=0001 0001 0000 s=1；

由于 S 18,1 将该块存入高速缓存了但标记位不相等，故 miss,发生一次 eviction;

✧ L 210,1 对地址 0x210 进行访问：

0x210=0010 0001 0000， s=1，

这里标记位为 2 跟L 110,1 操作读取新的行的标记位不匹配，miss,cache 读取新的行，发生一次 eviction。

✧ M 12,1 对地址 0x12 进行连续两次访问：

0x12=0001 0010,s=1;

第一次 miss,因为操作 L 210,1 时发生了一次行替换把该块驱逐了(即使标记位相等),cache 重新读取该块，发生一次eviction，第二次肯定为 hit。

四，编写函数体get_Opt()、helpPrint()、checkOptArg()

```c
//解析输入函数
int get_Opt(int argc, char **argv, int *s, int *E, int* b,
                        char *tracefileName, int *isVerbose)
{
        int opt;
        for(;((opt=getopt(argc,argv,"hvs:E:b:t:"))!=-1);)
        {
        switch(opt)
                {
                        case 'h':
                                printHelpMenu();
                                exit(0);
                        case 'v':
                                *isVerbose=1;break;
                        case 's':
                                checkOptarg(optarg);
                                *s=atol(optarg);break;
                        case 'E':
                                checkOptarg(optarg);
                                *E=atol(optarg);break;
                        case 'b':
                                checkOptarg(optarg);
                                *b=atol(optarg);break;
                        case 't':
                                checkOptarg(optarg);
                                strcpy(tracefileName, optarg);break;
                        default:
                                printf("invalid option -- '%c'\n", opt);
                                printHelpMenu();
                                exit(0);
                }
        }
        return 1;
}
```

```c
//检查命令行输入是否合法函数
void checkOptarg(char *curOptarg)
{
        if(curOptarg[0]=='-')
        {
                printf("./csim  : Missing required command line argument\n");
                printHelpMenu();
                exit(0);
        }
}
```

```c
//帮助信息打印函数
void printHelpMenu()
{
    printf("Usage: ./csim-ref [-hv] -s <num> -E <num> -b <num> -t <file>");
        printf("\n\nOptions:\n");
        printf("    -h         Print this help message.\n");
        printf("    -v         Optional verbose flag.\n");
        printf("    -s <num>   Number of set index bits.\n");
        printf("    -E <num>   Number of lines per set.\n");
        printf("    -b <num>   Number of block offset bits.\n");
        printf("    -t <file>  Trace file.\n\n");
        printf("Examples:\n");
        printf("linux>  ./csim-ref -s 4 -E 1 -b 4 -t traces/yi.trace\n");
        printf("linux>  ./csim-ref -v -s 8 -E 2 -b 4 -t traces/yi.trace\n");
}
```

五、编写创建并释放部分代码的程序：initCache()、freeCache()

```c
//获取组数行数，分配组内存，行内存，清零有效位和Lru计数值
void init_SimCache(int s, int E, int b, Sim_Cache *cache)
{
        cache -> set_num = 2<<s;
        cache -> line_num = E;
        cache -> sets = (Set *)malloc(cache -> set_num * sizeof(Set));
        for(int i = 0; i < cache -> set_num; i++)
        {
                cache -> sets[i].lines = (Line *)malloc(E * sizeof(Line));
                for(int j = 0; j < cache -> line_num; j++)
                {
                        cache -> sets[i].lines[j].valid = 0;
                        cache -> sets[i].lines[j].tag = 0;
                        cache -> sets[i].lines[j].LruNumber = 0;
                }
        }
}
```

```c
void free_cache()
{
    int S = (1 << s);

    setptr = (struct oneSet*)malloc(sizeof(struct oneSet) * S);
    for (int ind = 0; ind < S; ++ind)
    {
        free(setptr[ind].v);
        free(setptr[ind].last_access_time);
        free(setptr[ind].tag);
    }

    free(setptr);
}
```

六、检验主函数代码以及结果分析

分析：右图可以看到我编写的缓存模拟器与样例得分一致，验证正确。

```c
int main(int argc,char **argv)
{
        misses = 0; hits = 0; evictions = 0;
        int s = 0, E = 0, b = 0, isVerbose = 0;
        char *tracefileName = (char *)malloc(100);
        Sim_Cache cache;
        get_Opt(argc, argv, &s, &E, &b,tracefileName, &isVerbose);
        init_SimCache(s,E,b,&cache);
        FILE *Path = fopen(tracefileName,"r");
        if(s<=0||E<=0||b<=0)
        {
                printf("Please check the range about 's' 'E' or 'b'\n");
                printf("%d %d %d\n",s,E,b);
                exit(0);
        }
        if(Path==NULL)
        {
                printf("%s : No such file or directory\n",tracefileName);
                exit(0);
        }
        int addr, number;
        char c;
        char opt[3];
        for(;(fscanf(Path,"%s %x%c%d",opt,&addr,&c,&number)!=EOF);)
        {
                if(opt[0]=='I')
                        continue;
                int setBits = getSet(addr, s, b);
                int tagBits = getTag(addr, s, b);
                //printf("setBits is %x, tagBits is %x\n", setBits, tagBits);

                if(isVerbose == 1)
                        printf("%s %x,%d ", opt, addr, number);
                if(opt[0] == 'L')
                        loadData(&cache, setBits, tagBits, isVerbose);
                else if(opt[0] == 'S')
                        storeData(&cache, setBits, tagBits, isVerbose);
                else if(opt[0] == 'M')
                        modifyData(&cache,setBits, tagBits, isVerbose);
                else
                        printf("Error!!!\n");

                if(isVerbose == 1)
                        printf("\n");
        }
        printSummary(hits,misses,evictions);

        //free_SimCache(&cache);
    return 0;
}
```

```
zhangjiwei@zhangjiwei-virtual-machine:~/cachelab-handout$ make
gcc -g -Wall -Werror -std=c99 -m64 -o csim csim.c cachelab.c -lm
gcc -g -Wall -Werror -std=c99 -m64 -O0 -c trans.c
gcc -g -Wall -Werror -std=c99 -m64 -o test-trans test-trans.c cachelab.c trans.o
gcc -g -Wall -Werror -std=c99 -m64 -O0 -o tracegen tracegen.c trans.o cachelab.c
# Generate a handin tar file each time you compile
tar -cvf zhangjiwei-handin.tar  csim.c trans.c
csim.c
trans.c
zhangjiwei@zhangjiwei-virtual-machine:~/cachelab-handout$ ./test-csim.c
bash: ./test-csim.c: 没有那个文件或目录
zhangjiwei@zhangjiwei-virtual-machine:~/cachelab-handout$ ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     3 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     3 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     3 (2,1,4)       2       3       1       2       3       1  traces/dave.trace
     3 (2,1,3)     167      71      67     167      71      67  traces/trans.trace
     3 (2,2,3)     201      37      29     201      37      29  traces/trans.trace
     3 (2,4,3)     212      26      10     212      26      10  traces/trans.trace
     3 (5,1,5)     231       7       0     231       7       0  traces/trans.trace
     6 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trace
    27

TEST_CSIM_RESULTS=27
zhangjiwei@zhangjiwei-virtual-machine:~/cachelab-handout$
```