实验六：cache 模拟器实验

19281171　　王雨潇

实验报告要求：实验报告用 Word 排版，文件内一定要有"学号姓名_学号"。实验报告应提交到课程平台，按时提交。实验报告的内容应尽量全面，避免简单化。简述实验内容，实验过程，实验结果，实验分析，心得体会等。答每道题前应将题目拷贝到实验报告上。

一、 实验目的

熟悉 cache 的原理，加深对 cache 的映像规则、替换方法、cache 命中与缺失的理解。

二、 实验过程

实验内容一：熟悉模拟程序

（1） 阅读给出的 cache 模拟程序 cachesimulator.cpp，理解其中的主要参数与功能。

答：阅读代码得知，程序的主要参数如下

变量定义	功能
cachesize	cache 大小
blocksize	块大小，单位为 word
assoc	n 路组相联的 n
accesscount	总访存次数
hitcount	缓存命中次数
misscount	缓存未命中次数
wordaddress	存放数据的字地址
blockaddress	存放数据的块地址
newarray	模拟缓存的二维数组
lru	辅助最近最少使用算法的统计，二维数组值越大越先被淘汰
c1c, c2c, c3c	强制失效、容量失效、冲突失效的次数

程序功能为根据指定的映像规则、cache、块大小、组相连情况进行 cache 模拟。其细节为对 bytearray 数组逐个访问，计算数据的 index 和 tag，然后分别根据用户之前的映射方式，观察 vaild 和 tag 位判断 cache 是否命中，若未命中则调用 misstype()函数判断失效类型，使用 LRU 算法替换更新。

（2） 修改代码，随机生成 N 个访存地址，运行程序观察并分析结果（例如，可分析其中命中次数，不命中次数，替换次数）。

答：首先修改代码，把从文件 project.txt 读取访存地址的部分修改为生成 500 个具有局部性的随机访存地址的程序：

```
080     // 修改添加的代码：
081     for(i=0; i<500; i+=10) {
082         int temp_var = rand() % 190;
083         for(int k=0; k<10; k++){
084             bytearray[i+k] = temp_var + k;
085         }
086     }
```

以直接映像，cache 大小 64，块大小 4，组数 1 的设置，运行程序，结果如下所示：在这组设置下，计算可知 cache 中一共有 64bytes/16bytes=4 个块，对于程序生成的 500 个范围在[0, 200]的访问地址，缓存命中 452 次，未命中 48 次。在 3C 失效原因中，出现次数最多的是**容量失效**（直接映像不该有容量失效的，见后文分析）共 30 次，其次是强制失效共 12 次，最少的是冲突失效共 6 次。

```
C:\Users\WANGYX\Desktop\计算机体系结构\cachesimulator.exe
Cache Simulation Project:

1. Direct_mapped
2. Set_associate
3. Fully_associate

: 1


Cache Size from range[64/128/256]: 64


Block Size from range[1/2/4]: 4


Enter the value for n-way Set value from[1/2/4/8/16]: 1
        Miss Rate = 0.096000
         Hit Rate = 0.904000
  Compulsory Miss =  12
    Capacity Miss =  30
    Conflict Miss =   6
       hit Number = 452.000000
      miss Number = 48.000000
    Access Number = 500.000000
```

（3） 熟悉 cache 系统的执行过程（可举例详细分析一次命中/不命中/替换过程）。

答：首先修改代码，构造一个不太可能在真实使用场景中出现的用例。

使 bytearray 轮流访问地址 0,256,0,256 交替进行。然后以组相连，cache 大小 256，块大小 4，组数 1 的设置，运行程序。

```
080     // 修改添加的代码：
081     for(i=0; i<500; i++) {
082         if (i%2) bytearray[i] = 0;
083         else bytearray[i] = 256;
084     }
```

计算可知，cache 中的块数是 256bytes/16bytes=16，提前分析 cache 对这组测试用例的执行：

1) 访问 0，内存地址为 0 的字节的块号是 0%16=0，应当映射到 cache 的 0 号块，该块内容第一次来到 cache，强制失效，地址块放入 cache；

2) 访问 256，内存地址为 256 的字节的块号是 256%16=0，应当映射到 cache 的 0 号块，256 所在块内容第一次来到 cache，（程序认定为）强制失效，地址块放入 cache；

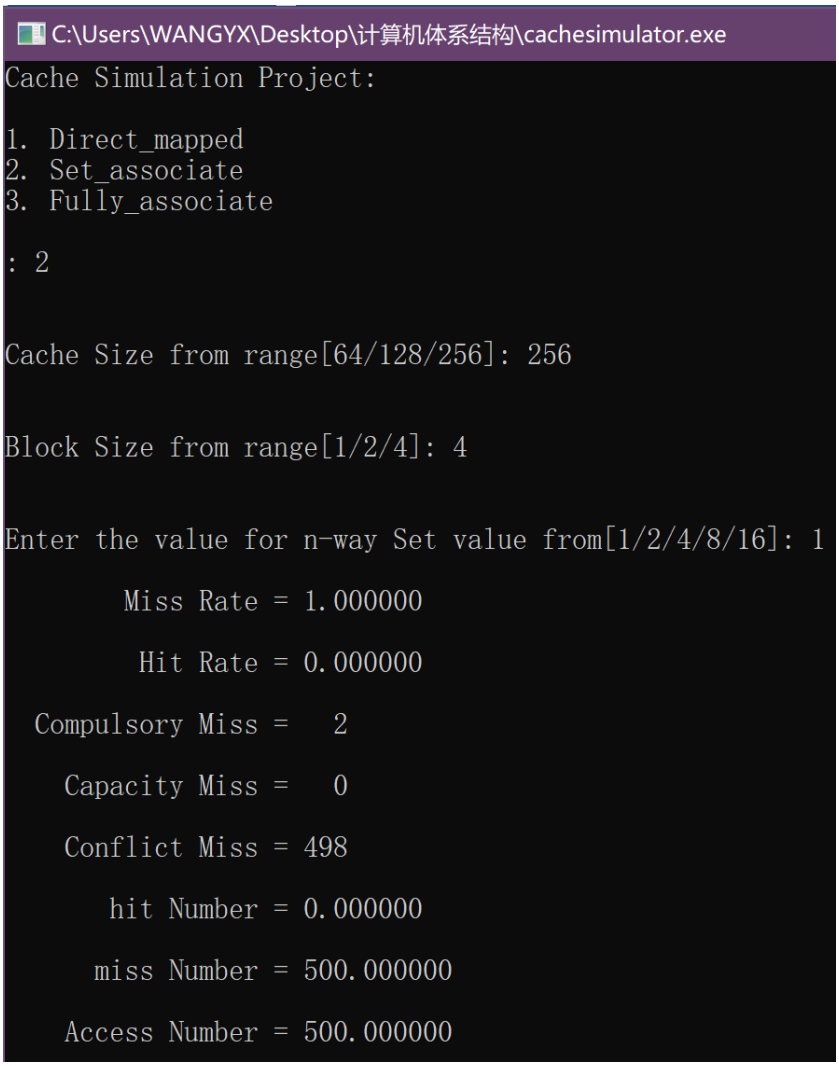
3) 访问 0，此时 cache 的 0 号块放的是 256 处的块内容，属于块和块竞争引起的冲突失效，所在地址块放入 cache；

4) 访问 256，此时 cache 的 0 号块放的是 0 处的块内容，冲突失效；

5) 此后的每次访问重复 3)~4），都是冲突失效。

根据上述推测，这组测试用例会发生 2 次强制失效，498 次冲突失效。

实际结果与分析一致，如下图所示，cache 一次也没命中：



实验内容二：利用该模拟程序仿真课后习题 4.1，并得出结果。

习题 4.1：The following C program is run (with no optimizations) on a machine with a cache that has four-word (16-byte) blocks and holds 256 bytes of data:

```
int i, j, c, stride, array[256];
for (i=0;i<10000;i++)
        for (j=0;j<256;j=j+stride)
                c = array[j]+5;
```

If we consider only the cache activity generated by references to the array and we assume that integers are words.

（1） What is the expected miss rate when the cache is direct-mapped and stride=132? How about if stride=131?

答：已知 cache 大小 256bytes，块大小 16bytes，采用直接映像，计算可得 cache 一共有 16 块。

- 当 stride=132 时，c 在每次循环交替访问 array[0]和 array[132]，

  1) 第 1 次循环访问 0，下标为 0 的整型变量的块号是 0/4%16=0，应当映射到 cache 的 0 号块，该块内容第一次来到 cache，强制失效，地址块放入 cache；

  2) 第 1 次循环访问 132，下标为 132 的整型变量的块号是 132/4%16=1，应当映射到 cache 的 1 号块，该块内容第一次来到 cache，强制失效，地址块放入 cache；

  3) 第 2 次循环访问 0，cache 中第 0 块恰好是需要的数据，cache 命中

  4) 第 2 次循环访问 132，cache 中第 1 块恰好是需要的数据，cache 命中

  5) 此后的 9998 次循环的情况都是重复 3)~4)

  预测的缺失率应为 2/20000=0.0001=0.01%

- 当 stride=131 时，c 在每次循环交替访问 array[0]和 array[131]，

  1) 第 1 次循环访问 0，下标为 0 的整型变量的块号是(0/4)%16=0，应当映射到 cache 的 0 号块，该块内容第一次来到 cache，强制失效，地址块放入 cache；

  2) 第 1 次循环访问 131，下标为 131 的整型变量的块号是(131/4)%16=0，应当映射到 cache 的 0 号块，该块内容第一次来到 cache，强制失效，地址块放入 cache；

  3) 第 2 次循环访问 0，发现 cache 中第 0 块是 131 的数据，发生冲突失效，cache 的 0 号块又换成 0 处数据

  4) 第 2 次循环访问 131，发现 cache 中第 0 块是 0 的数据，发生冲突失效，cache 的 0 号块又换成 131 处数据

  5) 此后的 9998 次循环的情况都是重复 3)~4)

  预测的缺失率应为 20000/20000=1=100%

（2） Would either of these changes if the cache were two-way set associative?
（理论分析该试题，得出上述两问的结果。）

答：已知 cache 大小 256bytes，块大小 16bytes，采用 2 路-组相连，计算可得 cache 一共有 8 组、16 块。

- 当 stride=132 时，c 在每次循环交替访问 array[0]和 array[132]，
  1) 第 1 次循环访问 0，下标为 0 的整型变量的组号是 0/4%8=0，应当映射到 cache 的 0 号组，该块内容第一次来到 cache，强制失效，地址块放入 cache；
  2) 第 1 次循环访问 132，下标为 132 的整型变量的组号是 132/4%8=1，应当映射到 cache 的 1 号组，该块内容第一次来到 cache，强制失效，地址块放入 cache；
  3) 第 2 次循环访问 0，cache 中第 0 组恰好是需要的数据，cache 命中
  4) 第 2 次循环访问 132，cache 中第 1 组恰好是需要的数据，cache 命中
  5) 此后的 9998 次循环的情况都是重复 3)~4)

  预测的缺失率应为 2/20000=0.0001=0.01%
- 当 stride=131 时，c 在每次循环交替访问 array[0]和 array[131]，
  1) 第 1 次循环访问 0，下标为 0 的整型变量的组号是(0/4)%8=0，应当映射到 cache 的 0 号组，该块内容第一次来到 cache，强制失效，地址块放入 cache；
  2) 第 1 次循环访问 131，下标为 131 的整型变量的组号是(131/4)%8=0，应当映射到 cache 的 0 号组，该块内容第一次来到 cache，强制失效，地址块放入 cache；
  3) 第 2 次循环访问 0，发现 cache 中第 0 组的第 0 块是相应数据，命中；
  4) 第 2 次循环访问 131，发现 cache 中第 0 组的第 1 块是相应数据，命中；
  5) 此后的 9998 次循环的情况都是重复 3)~4)

  预测的缺失率应为 2/20000=0.0001=0.01%

（3） 利用 cache 模拟程序仿真该程序的 cache 执行过程，核实实验结果是否与理论分析计算结果一致。

答：修改代码模拟本程序的访存数组，但运行时会出错退出

```
Enter the value for n-way Set value from[1/2/4/8/16]: 1

--------------------------------
Process exited after 7.682 seconds with return value 3221225477

Press ANY key to exit...
```

经过排查发现是模拟程序中默认的数组空间太小，发生了数组越界，修改后如下所示：

```
006 static int blockaddress[50000];
007 static int before[500];
008 static int t=0;
```

```
022     float misscount, accesscount, hitcount;
023     int index, byte, tag, ii;
024     int i=0, j, x, y, z, cc, c, m;
025     int bytearray[50000],wordaddress[50000];
026     int newarray[300][300]={0}, lru[300][300]={0};
027     char ans='y';
028     int c1c=0, c2c=0, c3c=0;
029     float missrate=0, hitrate=0;
```

```
086     // 修改添加的代码：
087     int i2, j2, stride=132, array[256];
088     for (i2=0; i2<10000; i2++) {
089         for (j2=0; j2<256; j2 = j2+stride) {
090             bytearray[i++] = j2*sizeof(int);
091         }
092     }
```

模拟程序验证运行结果：

stride=131，直接映射	stride=131，2 路-组相连
Miss Rate = 1.000000 Hit Rate = 0.000000 Compulsory Miss = 2 Capacity Miss = 0 Conflict Miss = 19998 hit Number = 0.000000 miss Number = 20000.000000 Access Number = 20000.000000	Miss Rate = 0.000100 Hit Rate = 0.999900 Compulsory Miss = 2 Capacity Miss = 0 Conflict Miss = 0 hit Number = 19998.000000 miss Number = 2.000000 Access Number = 20000.000000
stride=132，直接映射	stride=132，2 路-组相连
Miss Rate = 0.000100 Hit Rate = 0.999900 Compulsory Miss = 2 Capacity Miss = 0 Conflict Miss = 0 hit Number = 19998.000000 miss Number = 2.000000 Access Number = 20000.000000	Miss Rate = 0.000100 Hit Rate = 0.999900 Compulsory Miss = 2 Capacity Miss = 0 Conflict Miss = 0 hit Number = 19998.000000 miss Number = 2.000000 Access Number = 20000.000000

发现与预测一致。

(4) 思考：模拟程序中对 3C 类型的判断逻辑

答：下图是模拟器代码的 misstype 函数，该函数对失效的原因判断存在逻辑问题。

```
int misstype(int ba, int nb, int l)
// this function is used to decide which miss type it's belong to
{
    int u,k=0,b=0,n=0,m,ii;
    int blarray[500];
    int type;
    for (ii=0;ii<500;ii++)          // initila the array
        blarray[ii]=9999;
    for(u=0;u<=t;u++)           // check if the block address already in the array
    {
        if(before[u]==ba)       //if the block address in the before array
        {
            k=0;
            break;
        }
        else
            k=1;
    }
    if(k==1)                        // compulsory miss
    {
        type=1;
        before[t]=ba;
        t++;
    }
    if(k==0)                        //capacity or conflict miss
    {
        for(u=(l-1); u>=0; u--)
        {
            if(blockaddress[u]==ba) // current address is refered before
            {
                break;
            }
            else
            {
                n=0;
                for(m=0;m<=b;m++)
                {
                    n++;
                    if(blarray[m]==blockaddress[u]) //if it's not a distinct address,
then break
                        break;
                }

                if(n==(b+1))
                {
                    blarray[b]=blockaddress[u];  //store the address in the blarray
                    b++;                         //blarray[b+1] to store next address
                }
            }
        }
        if((b)<nb)  //conflict miss
            type=2;
        else        //capacity miss
            type=3;
    }
    return type;
}
```

代码的判断分歧可以概括为如下流程：

1) 如果块之前没有在 cache 出现过（k=1），直接判断为**强制失效** Compulsory Miss；

2) 如果块之前曾经在 cache 出现过（k=0），而且该块地址之前被访问过，或被 blarray 存储了，则判断为**冲突失效** Conflict Miss；

3) 如果前两者都不满足条件，则判断为**容量失效** Capacity Miss；

这个逻辑会把它无法识别出来的 miss 都算成容量失效，而且每次循环并没有重置控制变量 c 的值，以上问题导致实验内容 1 中直接映像缓存出现了 30 次容量失效。

三、 心得体会

通过观察本次实验的模拟程序运行，我加深了对缓存机制和映射策略的理解。本次实验不仅考察了我们对体系结构知识的掌握和运用，还考验了我们阅读代码，修改代码，排查问题的能力。通过动手操作，我不仅巩固了课上所学知识，也在实践中进一步理解了 cache 的运行过程。

注意：

1.希望通过实际操作熟悉我们课程上学习的 cache 原理。

2.实验报告的内容应该充实，细致，不要过于简单化。要注明实验名称、姓名、班级、学号等必要信息。

3.实验报告可提交 word 或者 pdf 格式文档,文档命名为:学号姓名_第四章实验，并在截止日期前提交到课程平台。