实验过程

本次的任务分为三个部分，前两个部分就是进行学习一些Linux的基本知识和之后代码会使用的一些基础工具，之前我一直使用Linux作为主要使用的系统，这些知识并不困难，然后Git也一直有在用，本来就很熟悉了，这一次实验3的代码部分就已经使用Git进行版本管理而且也已经传上github远程仓库进行保存了。

实验的第三个部分才是重点要进行讲解的，中途确实出现过不少问题。实验过程还算是顺利，而且代码部分本来就比较简单，所以主要是思路要正确，编程部分不困难，出错概率不大，但是不同的思路对实验结果的影响非常大。

实验分析

实验分为两个部分，首先是模拟cache的工作方式，通过给予地址和操作方式来统计命中，非命中，替换的次数；第二部分是通过矩阵转置实验来写cache友好的代码，尽量减少cache miss的次数。

Part A

首先，cache的本体构造使用的是二维结构数组，由于实验不要求存储数据，所以就没有数据存储部分。每一个cache行为一个数据结构，存储tag，valid和lru信息。Tag存放实际的地址，valid表示该行是否有效，lru用于替换算法，每访问一次该cache行就lru就加一，若到了要替换的时候就取lru最大的行进行替换。每E行形成一个cache组，每S组组成cache本身，所以使用二维结构数组进行构造。

|  |  |  |
| --- | --- | --- |
| t bits | s bits | b bits |
| Tag:标记位 | Cache set:组索引 | 块偏移 |

图1 地址划分示意

其次，对输入的文件进行解析，输入的为操作数和地址，由于不用进行数据存储，所以size可以忽略，若操作数为I，可以忽略；为M，默认命中，然后访问cache，统计hit，miss，eviction；为L和S，访问cache，统计hit，miss，eviction。

最后在程序中对输入指令进行解析，转化为输入文件路径，构建cache的S和E。

Part B

这一部分要求对给定的矩阵进行转置操作，使用的cache是一个32行的直接相联映射的cache，为了尽可能减少cache的miss率，可以尽量提高代码的空间局部性。

32\*32的矩阵：

每行cache可以存储64位的数据，也就是8个int类型数据，所以矩阵前8行的内容刚好可以把cache占满不会发生替换，而且B矩阵和A矩阵共用一个cache，也就是，那么也就是对A矩阵进行分块处理，将其分为8\*8的小矩阵，依次进行转置，即可达到满分的要求。

64\*64的矩阵：

由于矩阵A一行有64个int数据，所以4行就可以讲cache占满，若此时还像32\*32一样通过8\*8的分块进行处理，前四行和后四行会发生write miss，然后减少cache的命中率。所以一开始尝试使用4\*4的分块，虽然会有一半的空间浪费，但是不会发生write miss，但是最后miss数为1699超标了，然后尝试4\*8的分块，空间浪费的程度减少了一点点，miss数减少了50，仍然超标。

那么为了减少空间的浪费，首先分块处理，分为8\*8的分块，并且在处理的时候再次分块，分成4\*8的小块进行处理，第一次处理将A的0~3\*0~3进行转置存储到B的0~3\*0~3，然后将A的0~3\*4~7转置后逆行序存储到B的0~3\*4~7，演示过程如图2。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  | 0 | 8 | 16 | 24 | 7 | 15 | 23 | 31 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  | 1 | 9 | 17 | 25 | 6 | 14 | 22 | 30 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |  | 2 | 10 | 18 | 26 | 5 | 13 | 21 | 29 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |  | 3 | 11 | 19 | 27 | 4 | 12 | 20 | 28 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |  |  |  |  |  |  |  |  |  |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |  |  |  |  |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |  |  |  |  |  |  |  |  |  |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |  |  |  |  |  |  |  |  |  |

图2 左为矩阵A，右为矩阵B示意

然后进行矩阵A下四行的处理，从中间向两边依次处理，如第一次处理35，36，43，44，51，52，59，60这8个数据，放置到矩阵B中的第4行并将第3行的4~7列重新放置到第4行相应的位置，一共依次处理四次，第一，二次处理后如图3所示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 8 | 16 | 24 | 7 | 15 | 23 | 31 |  | 0 | 8 | 16 | 24 | 7 | 15 | 23 | 31 |
| 1 | 9 | 17 | 25 | 6 | 14 | 22 | 30 |  | 1 | 9 | 17 | 25 | 6 | 14 | 22 | 30 |
| 2 | 10 | 18 | 26 | 5 | 13 | 21 | 29 |  | 2 | 10 | 18 | 26 | 34 | 42 | 50 | 58 |
| 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 |  | 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 |
| 4 | 12 | 20 | 28 | 36 | 44 | 52 | 60 |  | 4 | 12 | 20 | 28 | 36 | 44 | 52 | 60 |
|  |  |  |  |  |  |  |  |  | 5 | 13 | 21 | 29 | 37 | 45 | 53 | 61 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

图3 左为第一次处理，右为第二次处理

就这样处理完四次后就将分块8\*8的矩阵A转置到矩阵B了，然后对剩下的矩阵也依次分成8\*8的块如此处理即可。最终miss次数为1243，小于1300，算作达标。

61\*67矩阵

由于这个矩阵并不是标准的8的倍数，不容易按行为标准进行分块，而且要求的miss数比较宽松，所以就随意安排了，最后测试16\*16的分块数即可达标，miss数为1992。

