

计算机组成原理实验四实验报告

**学 院： 计 算 机 工 程 学 院**

**班 级： 计算2114**

**姓 名: 庄佳强 章立早**

**学 号： 202121331104 202121331118**

二〇二二 年 11 月26 日

# 实验四.A

# 一 题目要求

* 1. 在这部分实验中，你需要利用上课介绍过的OpenMP技术对程序进行加速，并且使用阿姆达尔定律对其进行分析。
  2. 分析矩阵乘法的运行时间占比  
      在对矩阵乘法进行优化之前，我们可以对其运行时间进行分析。具体做法如下：（1）统计整个图像处理程序运行所需的**总时间**（你可能需要多次运行以便取得一个可靠的数值）；（2）统计矩阵乘法的程序运行时间（同理，你可能需要多次运行以便取得一个可靠的数值）；（3）估算矩阵乘法的程序运行时间在程序总运行时间中的占比。
  3. 使用OpenMP对矩阵乘法代码进行加速  
      根据上课所讨论的OpenMP使用方法以及网络上提供的参考资料，对矩阵乘法进行优化。如果你的实现是正确的，你将可以同时看到：（1）图像能够正确地被上下颠倒；（2）优化后的运行时间更短。
  4. 使用阿姆达尔定律进行验证  
      阿姆达尔定律为我们提供了一种在优化前就能够大致估算出加速比的方法。你可以使用你设置的OpenMP线程数作为改进量，并结合你之前分析过的矩阵乘法运行时间占比，估算出性能加速比，并与你真正测量到的性能加速比做对比。

# 二 设计思路

具体的实验代码已经给好，我只需要在基础上补写使用多线程的代码和开启OpenMP就可以实现OpenMP多线程工作。

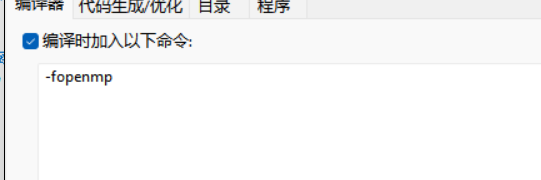
加上这串代码就可以实现多线程。

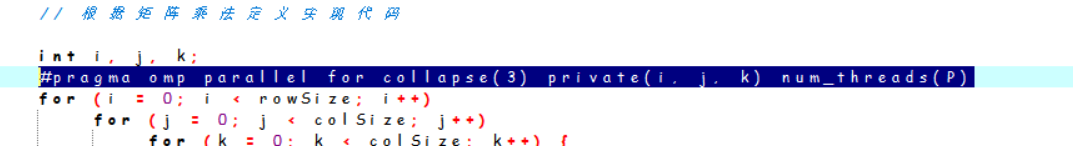
#pragma omp parallel for collapse(3) private(i, j, k) num\_threads(P)。

接着就可以记录数据在EXcel上计算就可以了。

# 三 成果展现

**开启OpenMP**

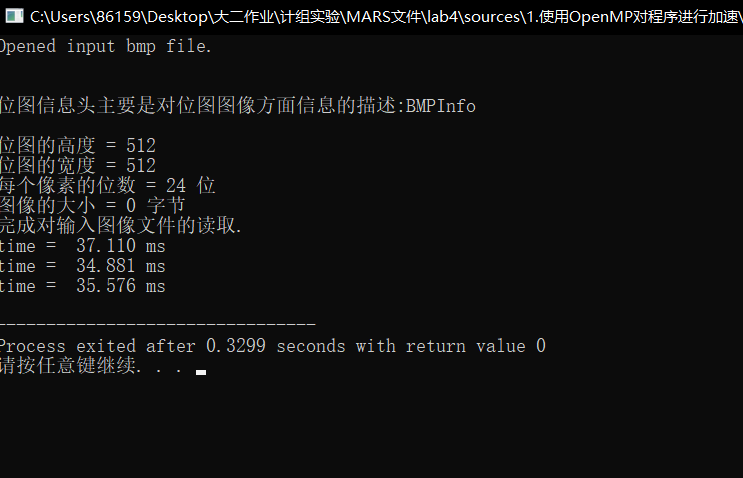




**未开启OpenMP时：**



**开始OpenMp后：**



**数据的记录：**



# 四 实验细节

在本次测量中，我把线程设置为12，每个线程完成工作量为1000000。

阿姆达尔定律为：

其中a为矩阵乘法程序的占比，k为线程的数量。因此得到S≈4.72,但实际的加速比为：3.20。

结论：看出即使一个系统的主要部分(main part)性能提升了很多，整个系统的性能提升远远小于此部分的提升。

# 实验四.B

# 一 题目要求

* 1. 介绍  
      在过去的实验当中，你已使用Logisim设计出了一个能够运行程序的32位的MIPS 处理器。在你设计的处理器中，你使用的存储器是一个理想的存储器：速度可以媲美处理器的其它模块，容量可以很大。  
      在这个实验当中，你将深入了解存储器层次结构（特别是Cache）对于程序的影响。你会利用MARS仿真器当中的小工具观察Cache的行为。
  2. 熟悉并理解测试程序  
      在这部分实验中，你将拿到一个MIPS汇编程序*cache.S* 。首先，你需要阅读该代码头部的伪代码，以便了解程序的基本功能，并弄清楚关键参数在汇编代码中的位置。

# 二 设计思路

本实验重在分析，通过改变a0,a1,a2,a3的值来观察变化，分析原因。

a0是字节存储栈的大小。

a1是每次存储占几个字节。

a2是外循环的次数。

a3是模式的选择。0是为每次存储0到字节中，1为每次存储在原来的基础上加1（访问两次）。

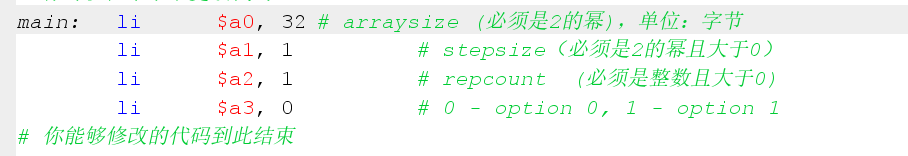
# 三 成果展现和分析

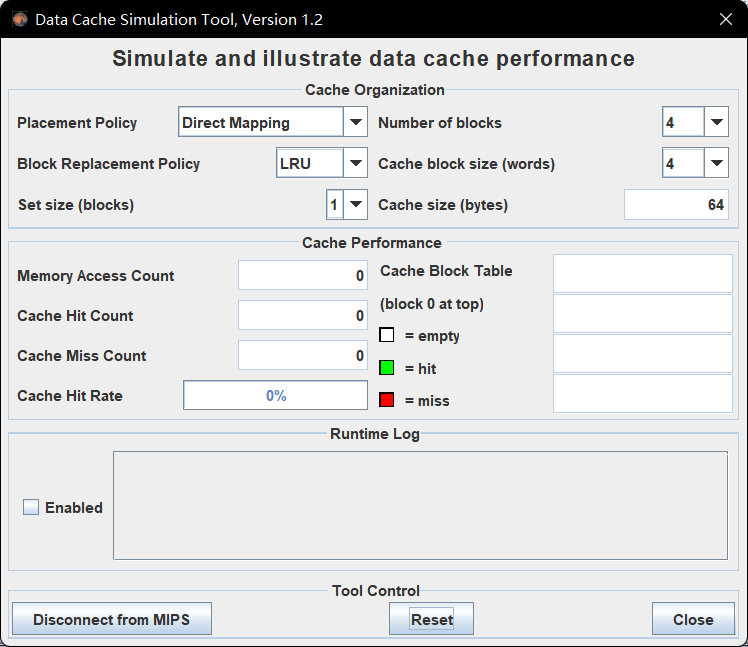
**影响Rate率：**

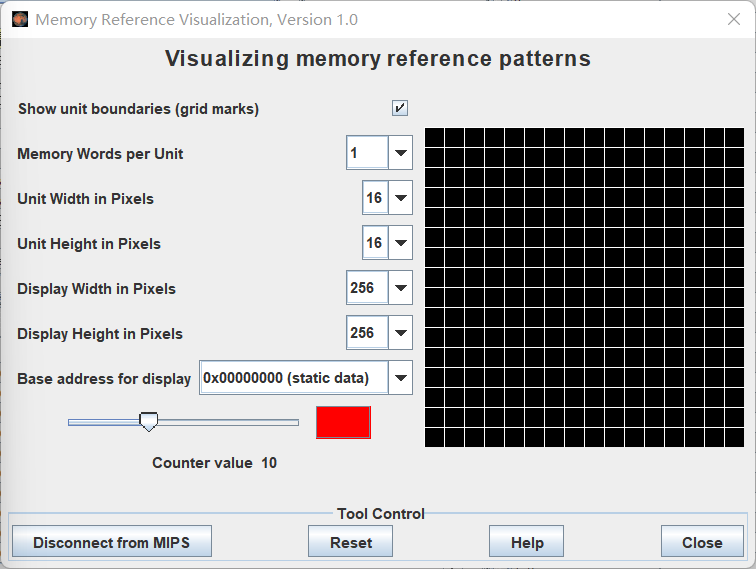
每个Block访问次数：在其他条件不变时，随着访问一个Block的数量增加时，Hit会增加，所有Rate会提升。

Block的大小，数量：在Cache的大小无法把所有的数据都存入时，提升Block的大小和数量可以使Cache存入更多的数据，从而减少反复占用的情况，从而miss,提升Rate。

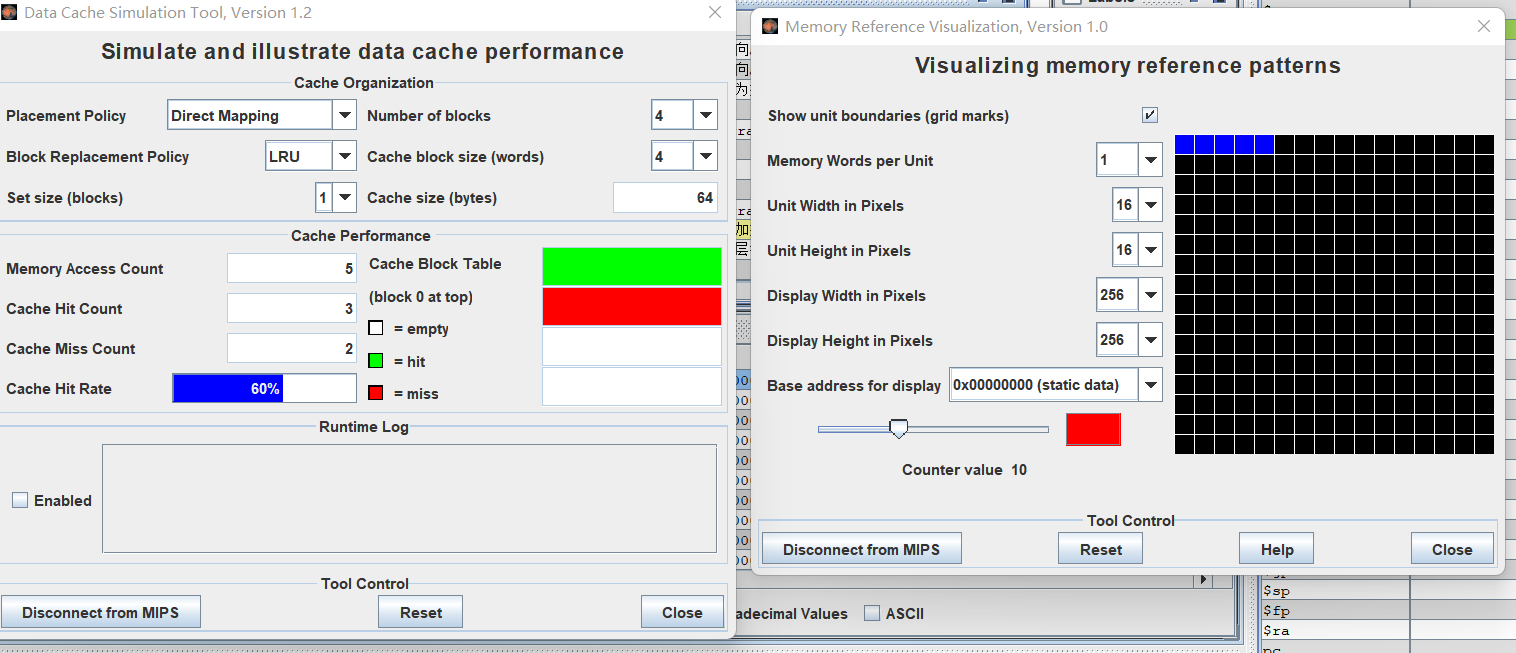
**情况1：**

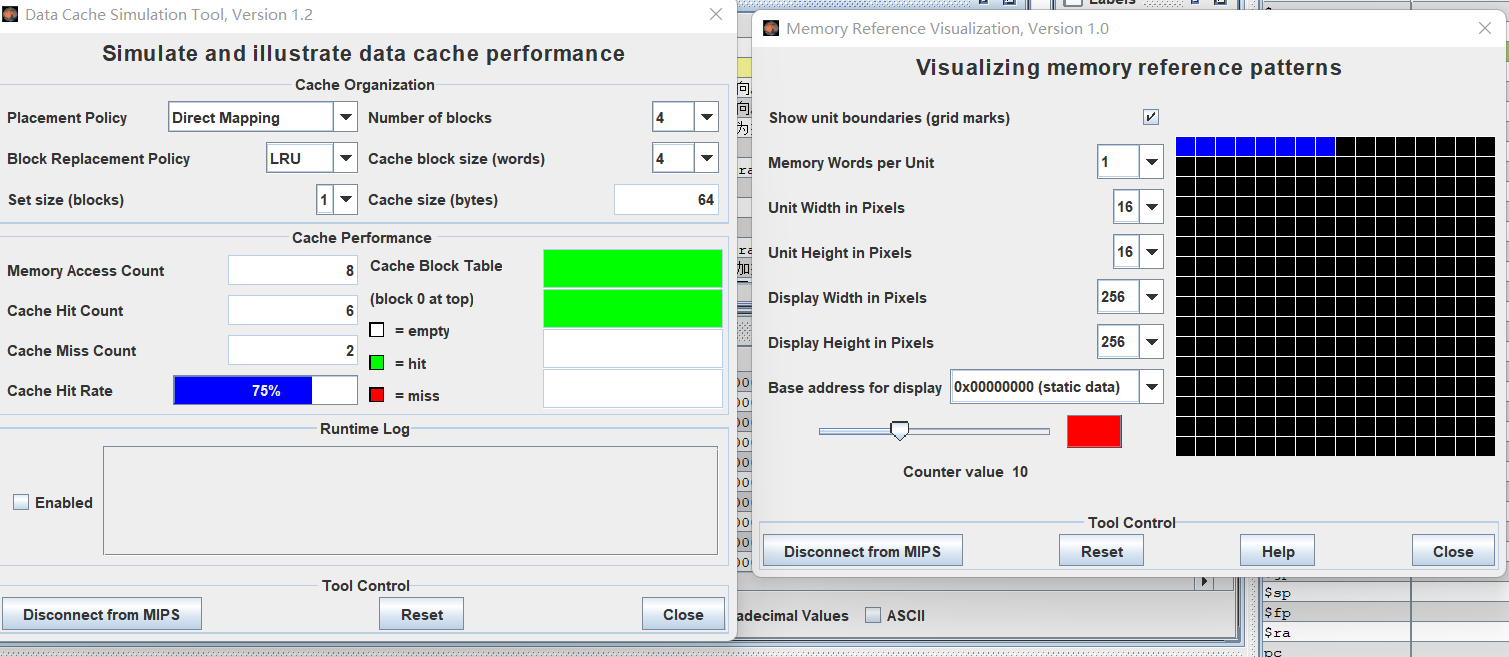


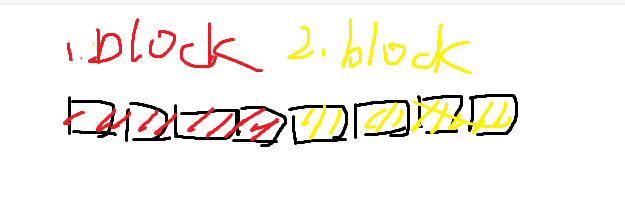


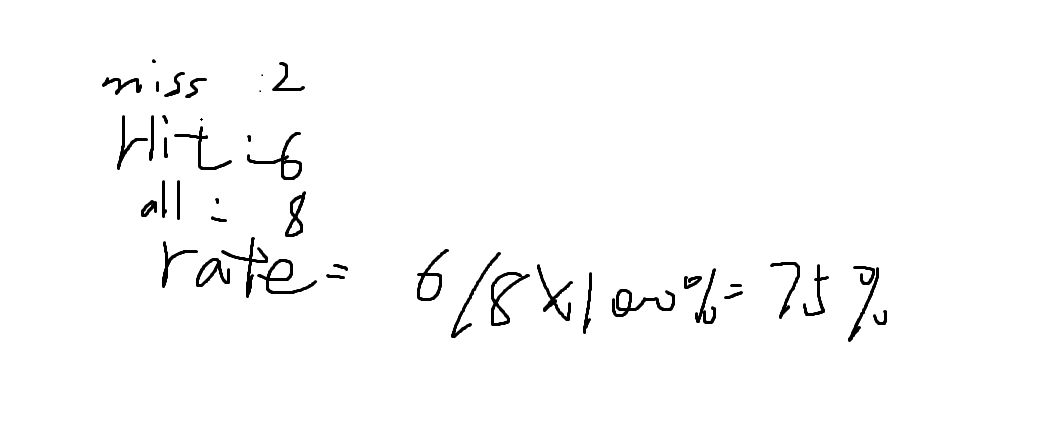


刚开始没有Cache中没有元素，第一次注定miss。然后Cache存入之后四个到第一个Blocks中，所以后三个为Hit, 第二次中的第一个也是miss, 再存四个到第二个Blocks中，后三个为Hit。所有整体有2次miss,六次Hit, 所以整体命中率为75%。

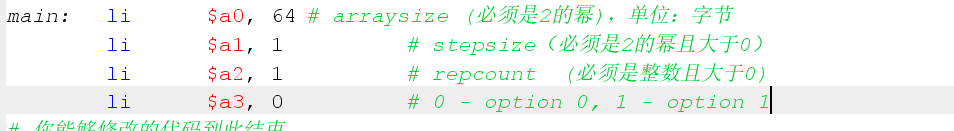






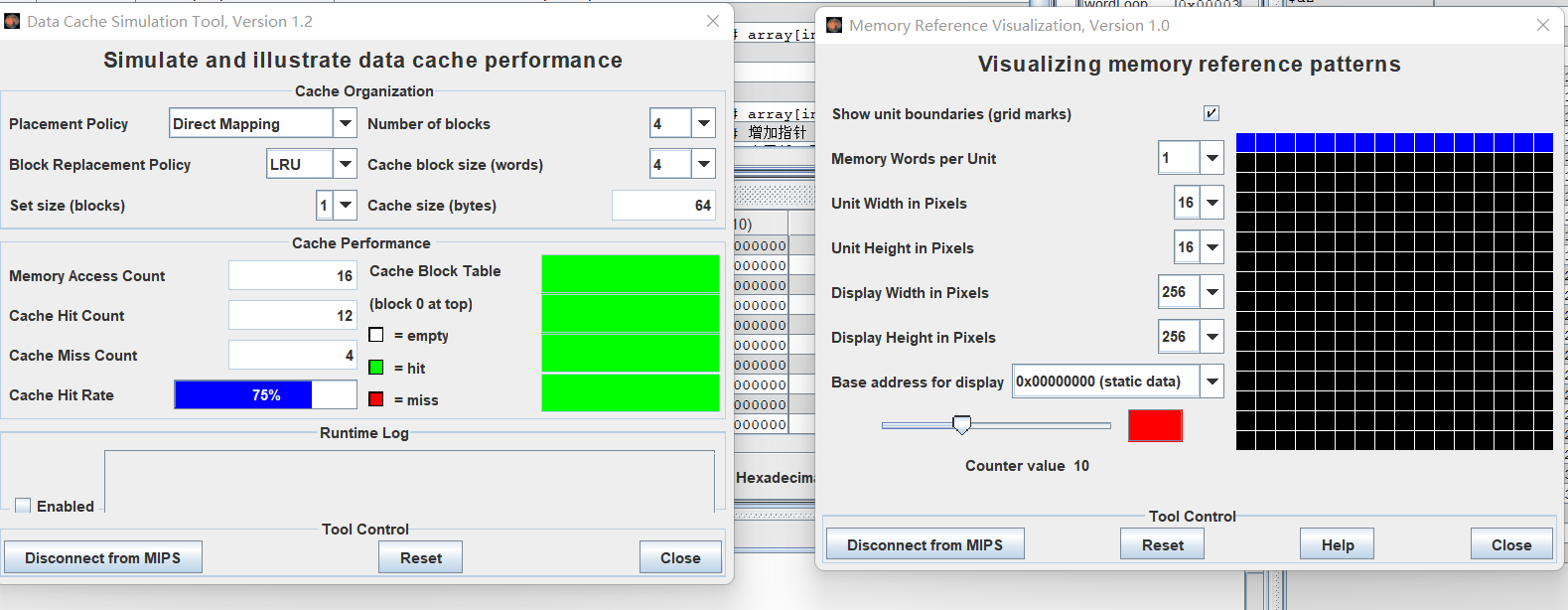


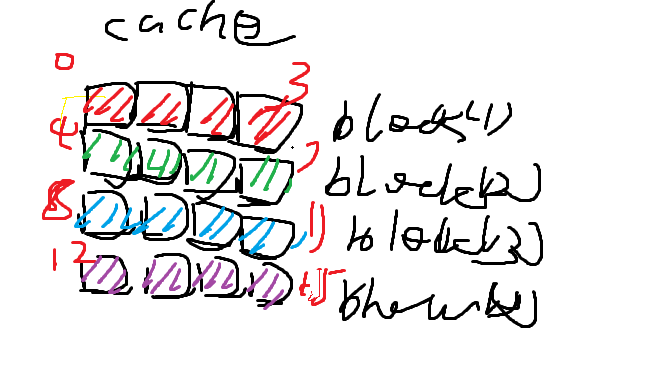
**情况二：**



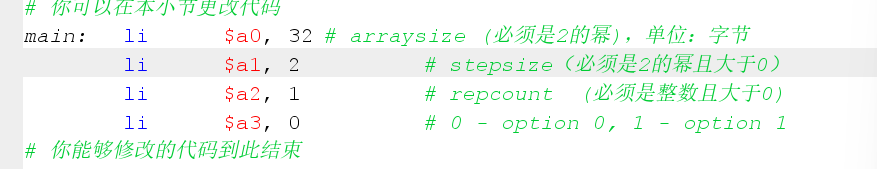
把数组扩大到64

与情况一相似，但这次扩大数组使Cache用到了所有的blocks。





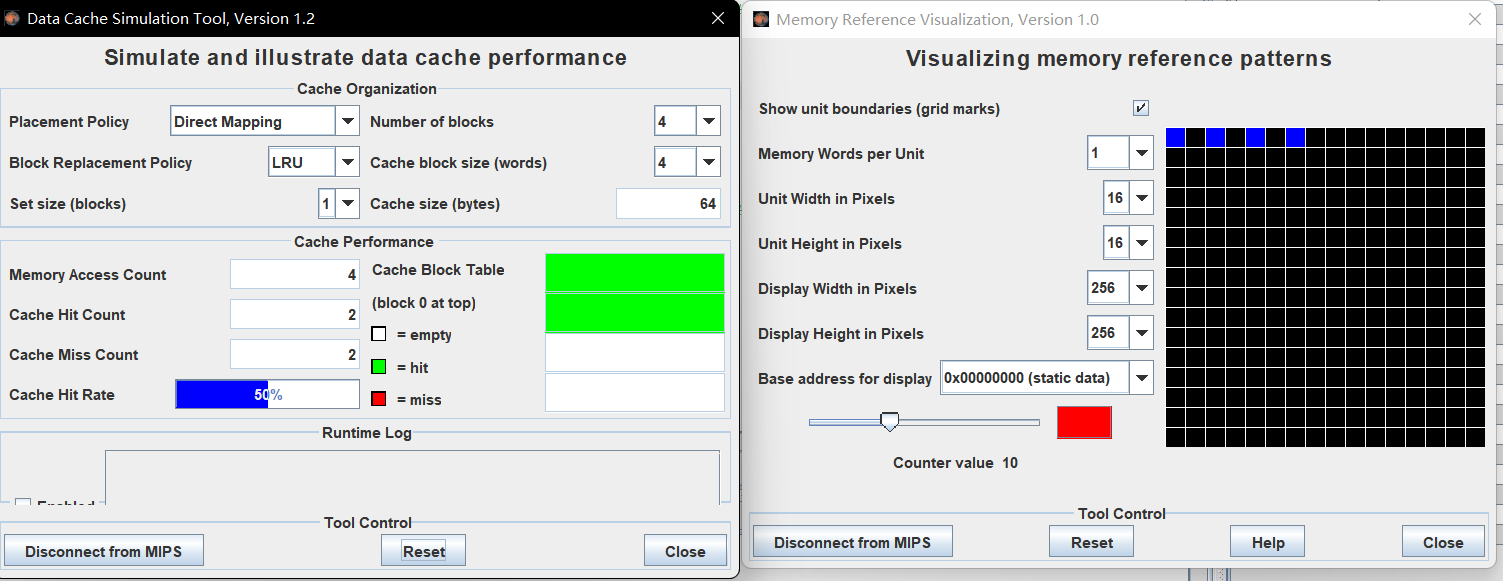
**情况三：**

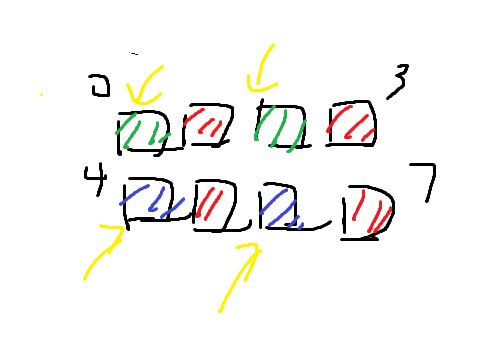


把每次存储的字节数变为2。

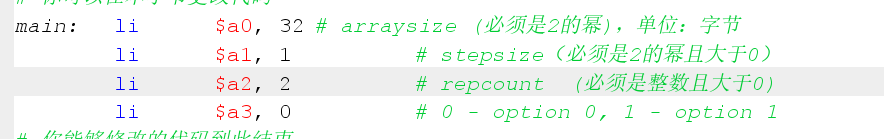
这样设置导致了每个Block 只有2个数据。

有一边的数据为空，而在访问中每个Block只会被访问两次，一次为miss,一次为Hit导致rate下降。



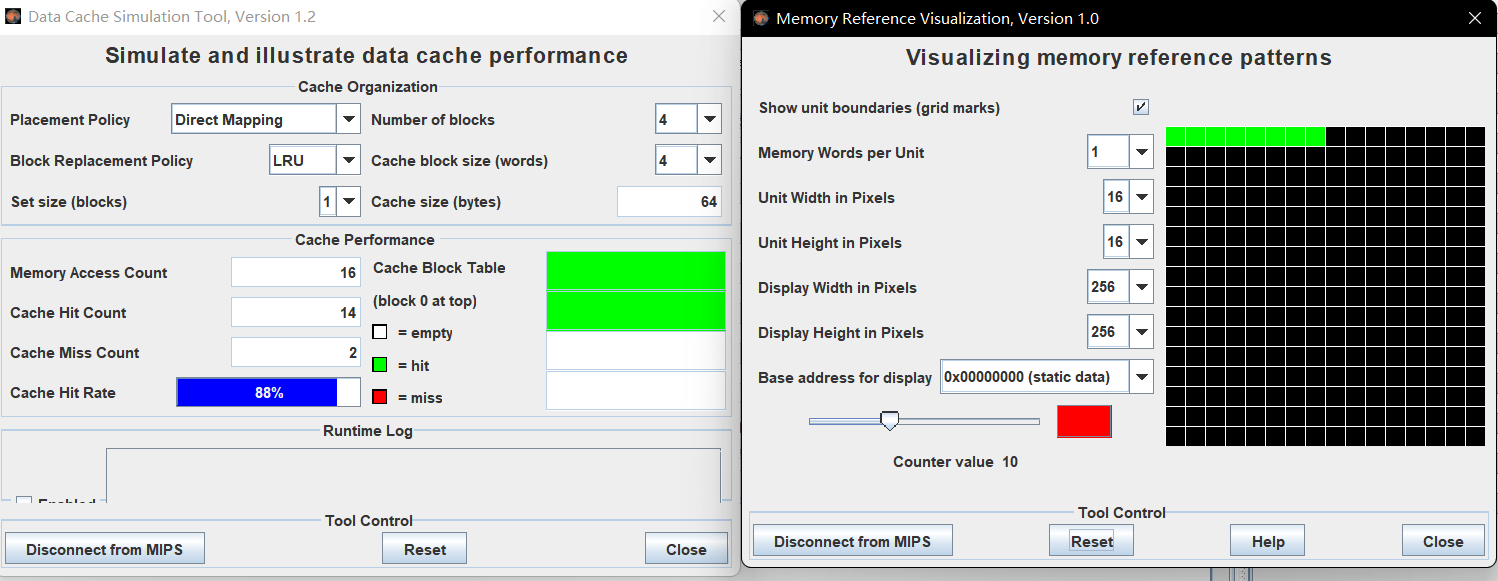


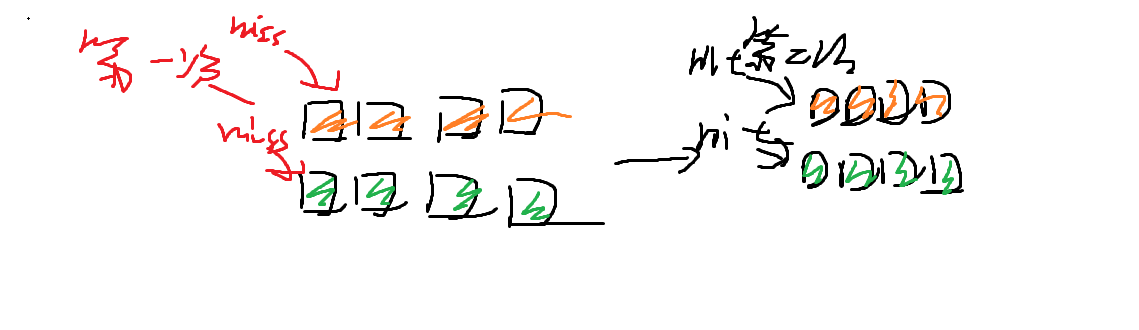
**情况四：**



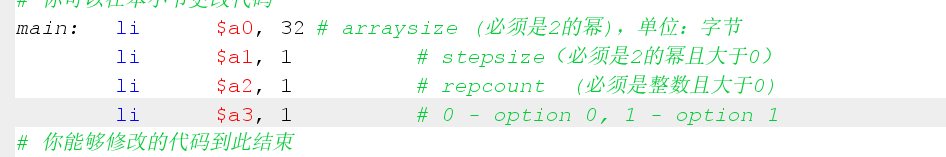
外循环设置成2次。

在第一次循环中，已经把字节存储到Block中，所有第一次中 Hit为6 ，miss为2，rate75%， 第二次访问中因为数据已经都在Block中,所有全部Hit 。综合rate为88%。



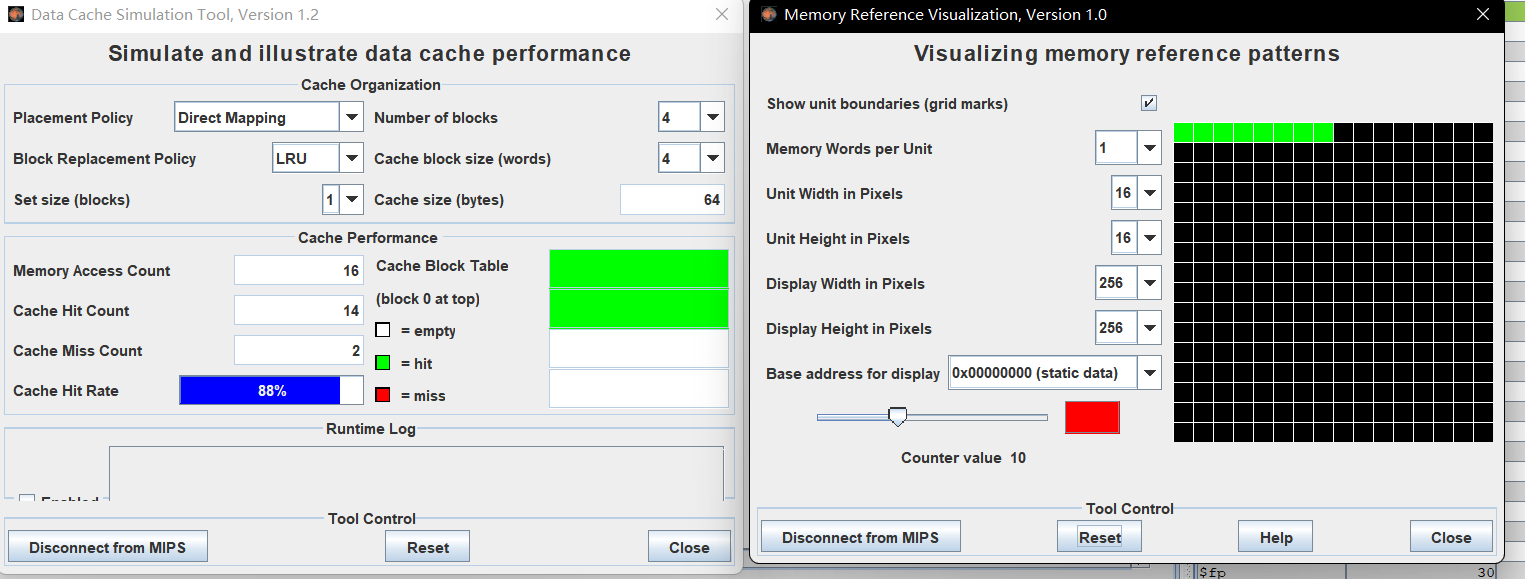


**情况五：**



设置存储方式为模式1，即先读入数据，加一后在写入数组中。（访问两次）。

所有和情况四相似，总共访问了16次，其中一开始读取的两次为miss,其他都为Hit。所有综合rate为88%。



# 五 实验总结

1. 通过实验A,我先复习了加速比和阿姆达尔定律的概念。学会通过开启OpenMp实现多线程，减少程序的运行时间，提升了效率。
2. 掌握了加速比的分析方法。在之中一开始忘记了阿姆达尔定律的计算方法，百度后解决。
3. Cache实验中，掌握了Cache的分析法，学到了控制CacheRate的方法，使得不友好性也可以变友好。