

实验报告

开课学期： 2022秋季

课程名称： 计算机体系结构(实验)

实验名称：实验4：层次存储系统分析

学生班级： 6

学生学号： 200110618

学生姓名： 邓皓元

实验与创新实践教育中心印制

2022年11月

# 1. 题目分析

*回顾课上讲解的Cache参数测量方法，谈谈你对其原理的理解。若采用PPT之外的测量方法，介绍其原理并分析其正确性。*

// TODO

1.测量cache容量：

按照一定步长循环访问不同大小的数组，若数组大小小于cache的大小，则第一次访问数组之后整个数组都被存入cache中，之后的访问都将发生cache命中，因此平均访存时间较短；

若数组大小大于cache的大小，必将周期性低发生cache缺失，而cache缺失所需要的时间大大高于cache命中所需要的时间，因此平均访存时间较长。

因此，循环访问大小不断增大的数组，当数组大小大于L1cache大小时，导致的cache缺失增加会导致平均访存时间大大提升，对于L2cache也是同理，因此需要找到访存时间突然增大的两个转折点的数组大小，则这两个转折点之前的数组大小即为L1cache和L2cache的大小。

2.测量cache块大小：

按照不同步长访问一定足够大的数组，若步长大小大于等于cache块大小，则每次访问数组都会发生cache缺失，导致平均访存时间较长；

若步长大小小于cache块大小，则访问数组发生cache缺失较少，平均访存时间较短。

因此，按照不断增大的步长循环访问数组，找到平均访存时间突然开始提升较多的步长大小即为cache块大小。

3.测量cache相联度：

创建一个2倍cache大小的数组，将数组平均分为2n块并依次循环访问其中的奇数块，当2n为组相联度的4倍及以上时，cache将不能存入所有的奇数块，因此不断增大n的大小，按照此规则访问该数组，找到平均访存时间突然开始提升较多时n的值，2n-2即为cache的相联度

4.测量TLBentry数量：

由于访问内存数据都需要访问TLB，而TLB只有有限个entry来记录，因此按照页表的大小作为步长访问足够大的数组，每次按照不断增大页数循环访问该数组，当页数大于TLBentry数量时，发生了TLB缺失导致的平均访存时间增加，因此只需要找到使平均访存时间突然开始增大较多的访问数组页数即为TLBentry数量。

# 2. 设计与实现

*\*注：若做了附加题，除了提交相应代码，还需按照指导书要求，在报告中添加相应的分析、设计过程和运行结果截图和对比等内容，否则不加分。*

## 2.1 方案设计

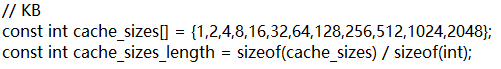
*详细阐述各测量函数的关键实现，建议结合算法流程图等图形化工具来辅助描述。*

// TODO

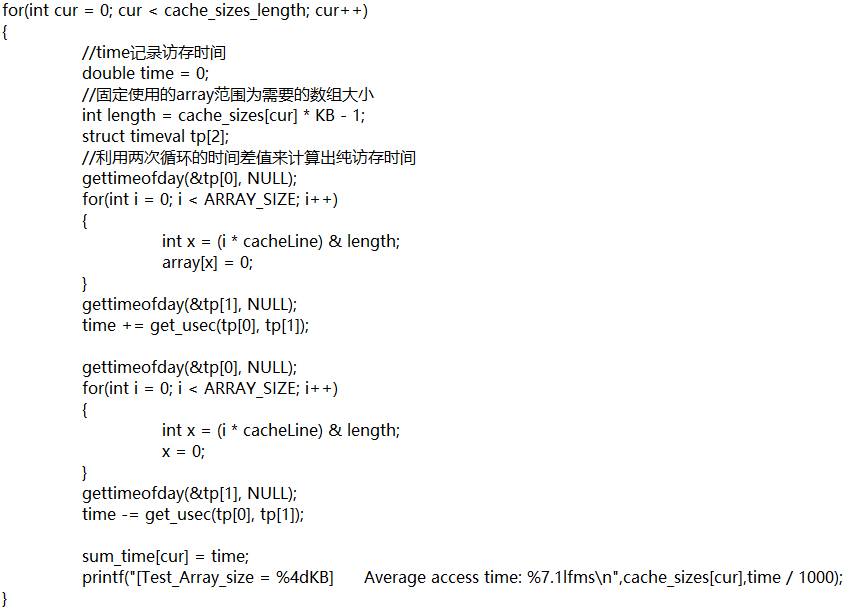
1.cache容量测量：

循环访问不同大小的数组(取用array不同大小部分)ARRAY\_SIZE次，记录对不同大小数组访存相同次数所需要的时间，找到访存时间变化最大的两个转折点，两个转折点之前的数组大小即为对应的L1\_Dcache大小和L2\_cache大小。

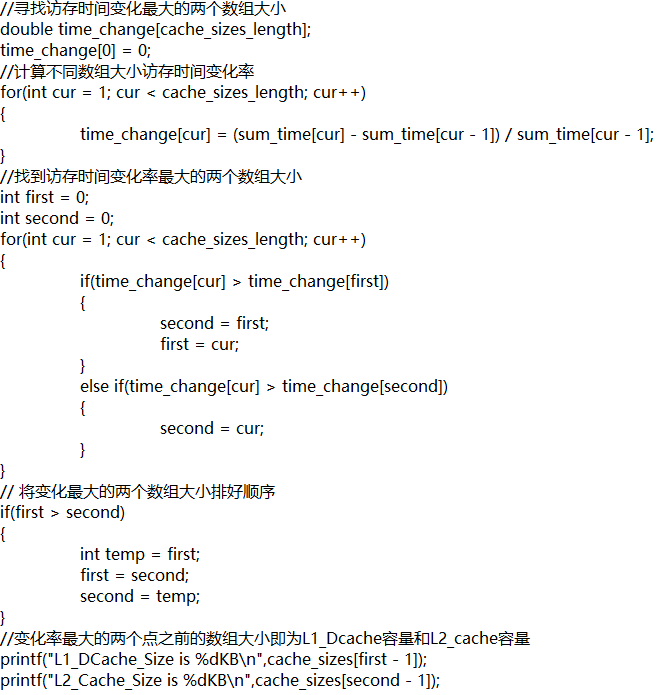
表示数组大小的数组：之后在数值后面\*KB(1024)



关键实现：



利用结果判断L1\_Dcache大小和L2\_cache大小：



2.cache块大小测量：

按照不同步长循环访问相同大小的数组array ARRAY\_SIZE次，记录对不同访问步长访存相同次数所需要的时间，找到访存时间开始发生较大变化的步长大小即为cache的块大小。

测量L1\_Dcache块大小时，需要利用Clear\_L1\_Cache()方法对L1\_Dcache进行清理，且访问的数组为大于L1\_Dcache大小且小于L2\_cache大小的数组

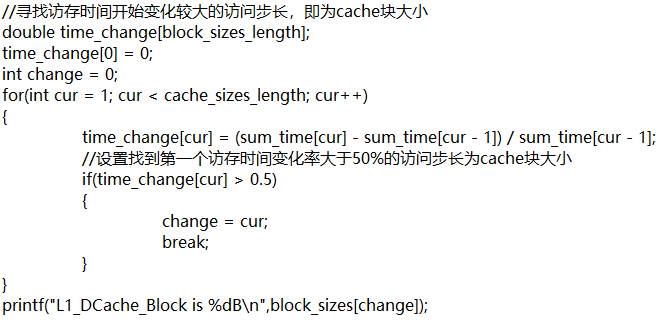
测量L2\_cache块大小时，需要利用Clear\_L2\_Cache()方法对L2\_cache进行清理，且访问的数组为大于L2\_cache大小的数组

表示步长大小的数组：之后在数值后面\*B(1)



关键实现：

利用结果判断L1\_Dcache块大小和L2\_cache块大小：



3.cache相联度测量：

创建一个两倍cache大小的数组，将数组分为逐渐增大的2次幂块数，按照适合步长(cacheLine(64) >> 2的访问步长使实验结果显著)依次访问奇数块，记录不同分块数的访存时间，找到访存时间发生较大变化的n，2n-2即为cache相联度。(每个不同分块数的访存次数相等)

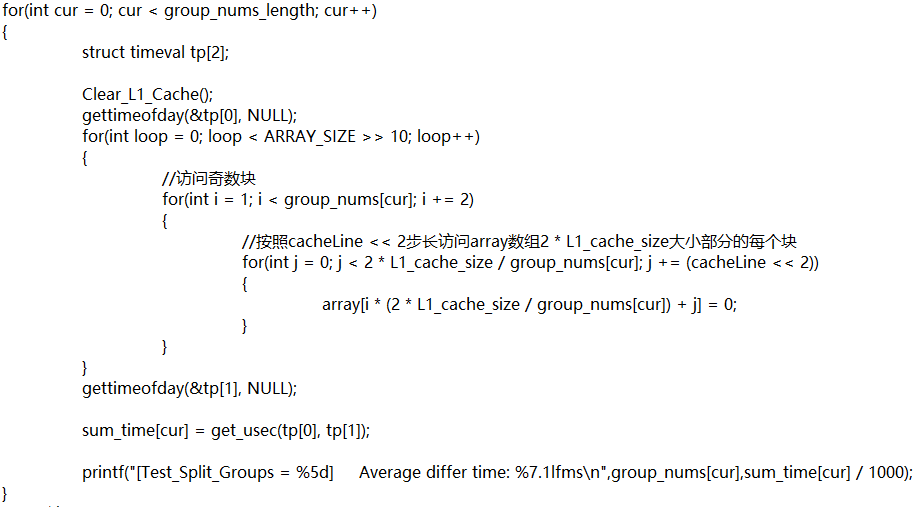
测量L1\_Dcache相联度时，需要使用大小为2 \* L1\_cache\_size的数组(使用array2 \* L1\_cache\_size大小部分)

测量L2\_cache相联度时，需要使用大小为2 \* L2\_cache\_size的数组(使用array2 \* L2\_cache\_size大小部分)

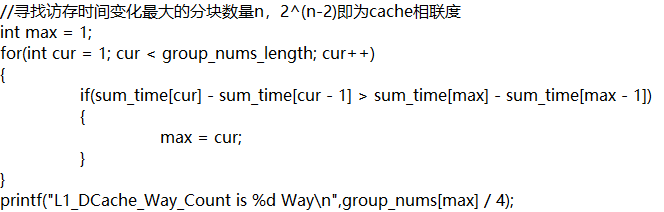
表示分块数量2次幂的数组：



关键实现：



利用结果判断L1\_Dcache相联度和L2\_cache相联度：



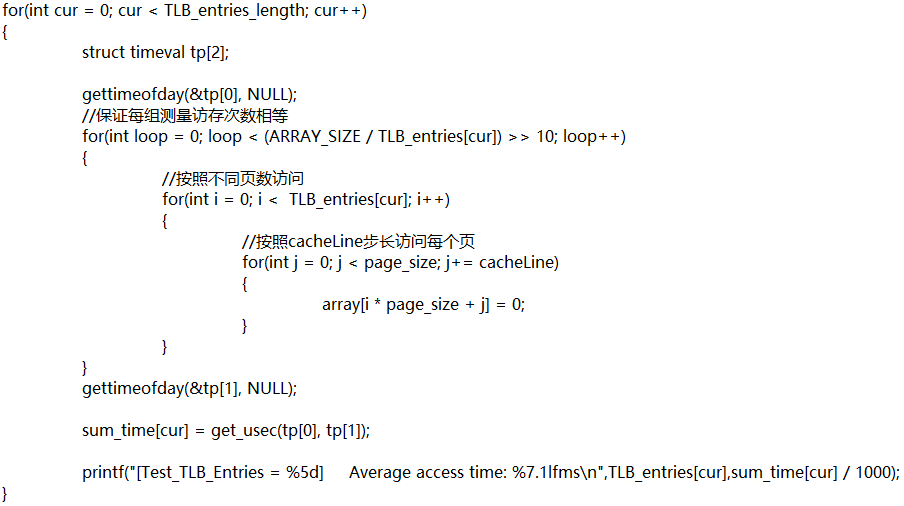
4.TLBentry数量测量：

按照不同页数循环访问每个页，记录对不同访问页数访存相同次数所需要的时间，找到访存时间开始发生较大变化的页数即为TLBentry数量。

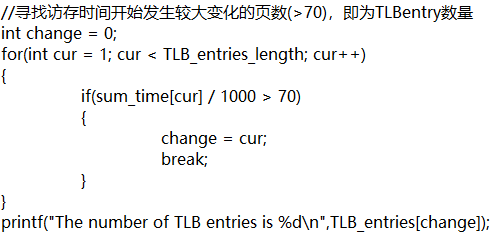
表示TLBentry数量的数组：



关键实现：



利用结果判断TLBentry数量：



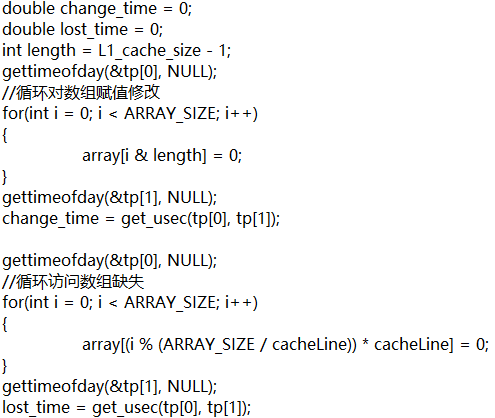
5.测量cache写策略：

记录两种不同情况下访存所需要的时间：

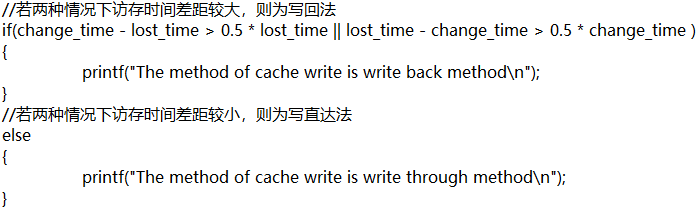
(1)对小于cache大小的数组进行循环赋值操作

(2)对数组访问

关键实现：



判断cache写方法：



6.测量cache替换方法：

已知L1\_Dcache为8路组相联，先依次读取第1组的1-8(0\*64号-7\*64号)数据块将L1\_Dcache的第1组数据块填满，再倒序依次读取第1组的8-1(7\*64号-0\*64号)数据块。

记录倒序访问这8块数据块的时间time，由于全部命中替换因此将此time作为all\_hit\_time

若是L1\_Dcache利用LRU进行数据块替换，则此时第1组数据块替换顺序为8-1

若是L1\_Dcache利用FIFO进行数据块替换，则此时第1组数据块替换顺序为1-8

若是L1\_Dcache利用RAND进行数据块替换，则此时第1组数据块替换顺序随机

再依次读取第1组的9-12(8\*64号-11\*64号)数据块，

记录访问这4块数据块的时间time，由于全部需要替换因此将此time\*2作为all\_replace\_time

若使用LRU替换方法，此时第1组中5-8数据块被替换，1-4号数据块还在第1组中，则最后一步test访存将全部命中，访存时间较短，test\_time接近all\_hit\_time

若使用LRU替换方法，此时第1组中1-4数据块被替换，1-4号数据块不在第1组中，

则最后一步test访存将全部缺失，访存时间较长，test\_time接近all\_replace\_time

若使用LRU替换方法，此时第1组中随机4个数据块被替换，1-4号数据块有1-4块在第1组中，则最后一步test访存将部分命中、部分缺失，test\_time在all\_hit\_time和all\_replace\_time中间左右

最后依次读取第1组的1-4(0\*64号-3\*64号)数据块，记录访问这4块数据块的时间time，将此time\*2作为test\_time

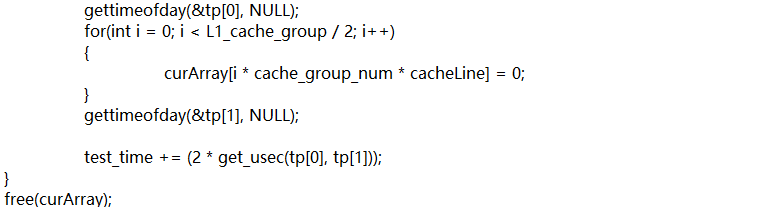
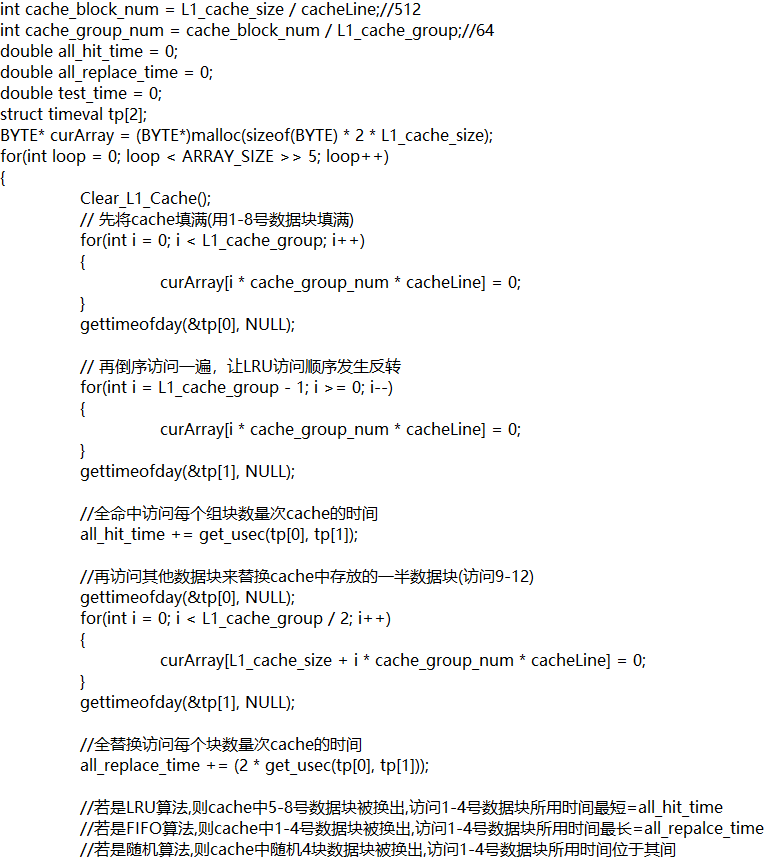
循环多次该操作并将每次不同操作用时求和，且每次操作前利用Clear\_L1\_Cache()函数清理L1\_cache。

最后观察测量结果，若test\_time和all\_hit\_time接近则cache使用LRU方法、

若test\_time和all\_replace\_time接近则cache使用FIFO方法、

若test\_time在all\_hit\_time和all\_replace\_time中间左右，则cache使用RAND方法。

关键实现：



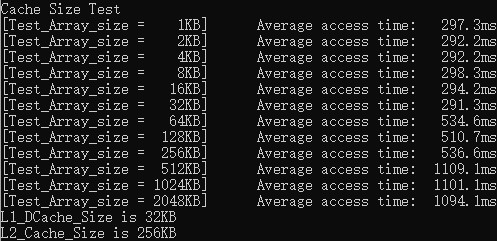
## 2.2 实验结果及分析

*绘制测量各参数时的平均访存时间折线图，并分析实验结果。如果测试结果不明显，试分析原因。*

*如果进行了优化，需给出优化前后的结果对比截图及文字分析（如果优化没有效果，也需分析原因）。*

// TODO

1.cache容量测量：



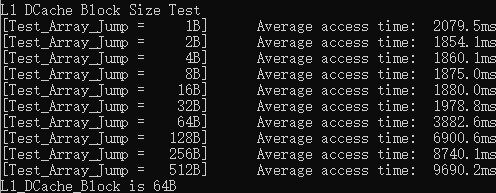
由测量结果可知，访存时间在数组大小为64KB和512KB时大幅突然提升，说明cache的大小为32KB和256KB，L1\_Dcache大小为32KB，L2\_cache大小为256KB



验证正确。

2.测量cache块大小：

L1\_Dcache：

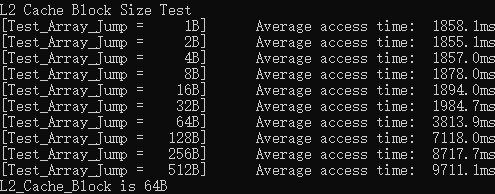


由测量结果可知，访存时间在步长大小为64B时突然大幅提升，说明L1\_Dcache块大小为64B



验证正确。

L2\_cache：



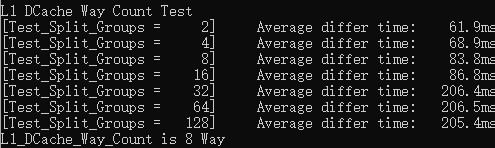
由测量结果可知，访存时间在步长大小为64B时突然大幅提升，说明L2\_cache块大小为64B



验证正确。

3.cache相联度测量：

L1\_Dcache：

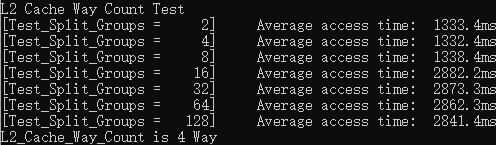


由测量结果可知，访存时间在分组数为32时突然大幅提升，说明L1\_Dcache块相联度为32/4=8



验证正确。

L2\_cache：



由测量结果可知，访存时间在分组数为16时突然大幅提升，说明L2\_cache块相联度为16/4=4

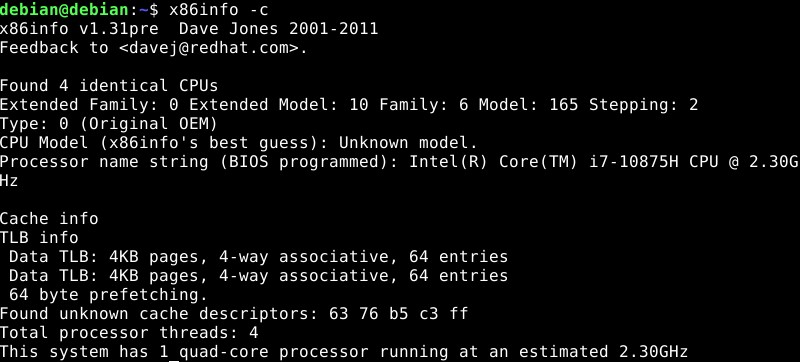


验证正确。

4.TLBentry数量测量：

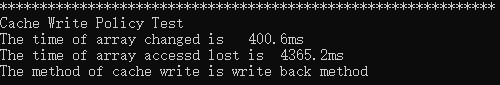


由测量结果可知，访存时间在页数为128时突然大幅提升，说明TLBentry数量为64



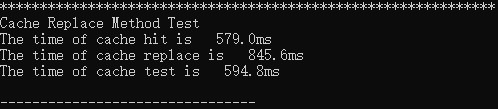
验证正确。

5.测量cache写策略：



由测量结果可知，两种情况访存时间相差巨大，说明cache写方法为写回法。

6.测量cache替换方法：



由测量结果可知，测试时间接近于全命中访存时间，说明cache替换方法为LRU方法或类似LRU方法，检验方法在实验原理中介绍。

# 3. 回答问题

## 3.1 为了进一步提高访存性能，现代处理器往往利用时间局部性和空间局部性，将所访问数据块相邻的数据块预取到Cache中。试分析数据预取将给实验结果带来什么影响？

// TODO

由于cache块大小为64B，可能会导致在测量cache块大小时，访存空间显著提升的步长节点从64B变成128B，在debian虚拟机上运行时，由于数据预取会导致类似的结果，但在dev c++最低级别编译优化运行时能够得到正确的曲线，认为虚拟机会进行数据预取而dev c++最低级别编译优化运行时不进行数据预取

## 3.2 试分析Cache对测量TLB造成何种影响？如何排除Cache的干扰？

// TODO

影响：测量TLB的方法为按照不同页数以页为单位访问数组并记录不同页数的访问时间，首先是访问数组必然会引起cache的时间消耗，由于按照不同页数访问数组有可能导致cache不能完整存放数组，可能导致测量TLB的访存时间较长或者较短，不能得出稳定的测量结果

干扰：首先，应该按照固定步长访问每个页，例如cacheLine的倍数，使得每次访问都是cache缺失，记录前者消耗的时间

之后执行相同次数的cache访问缺失操作(类似的赋值等运算都尽量接近)，记录后者消耗的时间

用前者减去后者，即可得到TLB进行页表替换的时间，排除cache干扰

## 3.3 测量TLB前，是否需要遍历访问一次全部空间？为什么？

// TODO

需要，先遍历一次全部空间可以提前将要测量的内存空间物理页表全部或部分存入TLB，使得接下来的访存时间减去了由于第一次访问，TLB页表项替换的时间，有利于接下来TLB测量结果的稳定。

## 3.4 要写出高效的程序，必须对底层硬件有足够的了解。请谈谈如何利用Cache或TLB提高程序性能。

// TODO

Cache：

1.cache大小：要提升程序的效用，首先当然是尽量使用第1层速度最快的cache来进行数据的访问，提前保证数据的大小范围在第1层cache大小之内，或者在尽量使用第1层cache的情况下保证数据大小不超过第2层cache大小，尽量在第1层cache大小之内进行运算。

2.cache块大小：其次是对于空间的利用，申请占用连续大量内存空间之前，要思考有没有利用到整个连续内存空间，能不能使用较小的多块内存空间来节省空间开销，或者使用占用较小空间的数据类型。

这些都是为了提高第1层cache的利用率和整体cache的命中率来提高程序效能。

TLB：

利用TLB提高程序性能当然是提高TLB命中率，首先是不要过多进行远距离内存调用来减少TLB页替换，其次是从根本上尽量减少太多页的调用。

这些都是为了提高TLB命中率来提高程序效能。

# 4. 总结

*请谈谈学习本课程实验的收获和反思，并提出合理的意见和建议。*

// TODO

在学习了本课程之后，我学习到了许多关于计算机底层的知识，切身感受到了底层的代码运行的艰辛，一点点的不一样都是导致结果千差万别的罪魁祸首，因此也让我对于每一行代码的编写都更加细心，这样才能编写出更优质、更能高效利用计算机每个硬件的优秀代码。

计算机体系结构实验课的实验原理任务完成法大大提高了学生手动独立完成每个实验的发挥能力和编码能力，也让学生在认真贯彻了实验原理的基础上有各种各样不同的完成任务的方法，但这次实验中也存在许多实验原理在理解方面不太容易，一点点对于实验原理的误解都可能导致实验结果不尽人意，为此我认为可以在一些较于晦涩难懂的实验原理部分增加关于实验内容的讲解部分，例如cache组相联的测试办法等等......

总而言之，通过这门实验，我学到了许多有用的知识，明白了细心和钻研才是计算机更久不变的学习秘诀。