1. **实验目的**

The purpose of this project is to measure the size of the data cache。

1. **实验内容**

volatile long array[N];

register long sum = 0;

for (int t = 0; t < M; t++) /\* many times\*/

for (int i=0; i < N; i = i + Stride)

sum += array[i]; /\* touch an item in the array \*/

Changing the stride parameter in the above code allows you to test the properties of cache

1. **实验环境**

Vc2010

1. **实验结果**

结果：当Stride小于16时，所花时间几乎一致；当Stride大于16时，运行时间随Stride加倍而减半。

分析：这是由于CPU会整块访问内存，每次访问大小即CPU行大小，此时该数据周边数据也被存入缓存，所以当Stride小于16时，虽然我们读取不同数据，但相隔16个数据之间都访问了同一个缓存块里面的数据，实际访问内存次数为N/16，因此时间几乎一致。所以行大小为：16\*sizeof(long)=16\*4=64。

数组所占字节大小

每元素平均访问时间（ns）

结果：当数组所占字节大小在32KB以下时，每元素平均访问时间一致；接下来有小幅增加，直至数组超过4MB时，平均访问时间又开始增长。

分析：当数组所占字节大小不超过一级缓存大小时，平均每元素的访问时间都是一级缓存的访问时间；当数组大小介于一级缓存和二级缓存之间时，访问会在一级、二级缓存之间进行，时间有所增幅；当数组大小超过一级和二级大小之和时，访问会扩展到三级缓存甚至内存，因此时间又增幅了。因此一级缓存大小应为：32KB；二级缓存大小应为4M。

1. **附录**

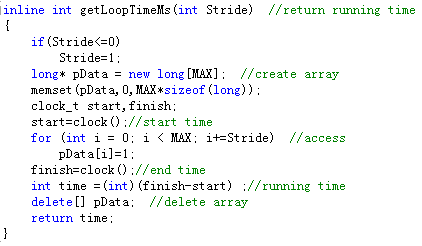
**(一)、问题分析：**

如何利用对数组的访问实现缓存的测试？

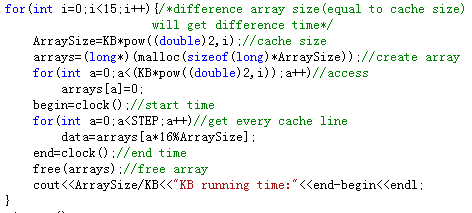
程序在运行时，数组是在内存中。当我们访问数组时，缓存会以缓存行大小来将数据周围的数读入缓存，以便下次访问。当控制其他时间因素时，内存和缓存的访问时间不同，会时数组访问产生一些时间上的特点。以此可求得缓存行大小。

当测完缓存行大小以后，我们可以在缓存中一行一行的访问数据。当数组很大超过缓存时，我们的访问会在一级、二级、三级缓存中依次进行，而他们的访问时间是不同的。因此我们可以通过调节数组大小来测试缓存行的平均访问时间，此时间会在缓存位置变化时发生变化。以此可求得缓存大小。

**（二）。方案设计：**

**1、缓存行大小测试：**

如图，我们通过调节Stride的大小来改变对数组的访问。当Stride小于（行大小/sizeof(long)）时，不同的Stride访问的数据多少差异很大，但因为缓存的特点，我们可以在缓存中找到范围内所需的数据，所以可以发现他们所花时间几乎一致，约为（每次访问内存的时间）\*（数组大小/每次缓存读取个数）。当Stride超过（行大小/sizeof(long)）后，时间会产生突变，因为此后缓存中并没有我们所需的数据，所以我们会访问内存，且因为Stride过大，访问内存的次数将变小，所需时间也将减少。由此突变我们可以得到行大小的值：行大小=突变点\*sizeof(long)。

**2、缓存大小测试：**

如图，我们可以利用外循环中的i值来改变数组array的分配空间，然后通过对数组所有元素的赋值来将之读入缓存。接着我们将在缓存中一行一行访问，由此得到访问STEP次数后的时间。通过改变数组大小，我们将发现当数组大小超过一级缓存大小以后，访问时间将会有所增大，再次超过二级缓存大小后，访问时间又将增大。这是由于各级缓存的访问时间是不同的，因此我们可以根据变化的时间点对应的数组大小确定缓存大小。

**（三）、调试心得：**

1、尽力减少除了访问内存以外影响时间的因素：

（1）、（测试行大小时）当Stride为1、2时，加减法及多个循环等因素会极大影响程序运行时间，此时测出的时间误差较大，并不作为参考；

（2）、（测试行大小时）不该在同一程序中多次访问同一数组，因为此时数组数据已被缓存，所测时间并不准确，不同Stride当新建数组去访问；

（3）、数据类型为long有助于减少运算时间的影响；

（4）、数据访问时，若对数据进行了修改，整行的缓存将失效，因此data=array[i]与array[i]=0所得出的结果是不同的；

2、行大小测试时根据Stride的不同，实际上时间并不一定一直平稳或递减，因为当缓存块的数据使用完后，缓存块中的最后一个数据与下一个要使用的数据之间可能有一些数据未被装入缓存，这将使某些Stride的内存访问次数变少，时间也相对减少。相对的有些与之临近的Stride会把所有数据装入缓存, 内存访问次数较多，时间也较多（比如此次测试中Stride=12比Stride=16所花时间少1ms）。

3、测试缓存大小时，要先将数组读入缓存，然后对数组的访问要每隔[行大小/sizeof(long)]，如此遍历缓存区域的每一个块。之后测试访问相同次数的数组数据，不同数组大小所需的时间将会在缓存满额时发生变化，此时数组所占字节大小即缓存大小。