Lecture 9: Cache I

高速缓冲存储器（Cache）

高速缓冲存储器(Cache)

- 什么是程序访问的局部化特性
- 具有Cache机制的CPU的基本访存过程
- Cache和主存之间的映射方式
  - 直接映射 / 全相联映射 / 组相联映射
- cache容量和块大小的选择
- Cache替换算法
- cache-friendly的程序
- Cache的写策略
  - Write Back 和Write Through
- Cache失靶处理
- Cache性能评估

What we want in a memory

到目前为止，已经了解到有以下几种存储器：

Reg，SRAM，DRAM， Hard Disk，Tape and 光盘等

	Capacity	Latency	Cost
Register	〈1KB	1ns	$$$$
SRAM	1MB	2ns	$$$
DRAM	1GB	10ns	$
Hard disk*	100GB	10ms	¢
Want	100GB	1ns	cheap

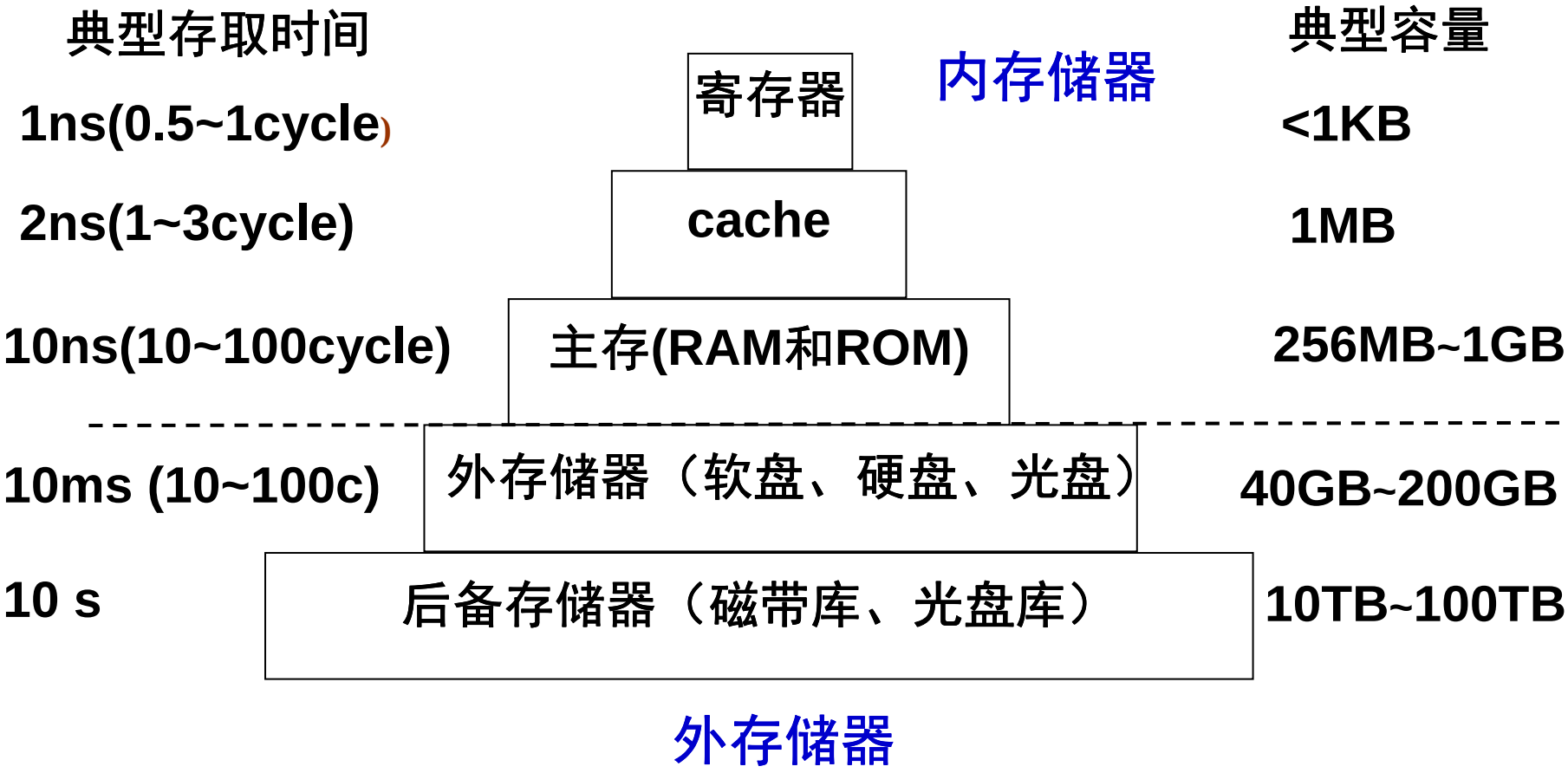
* non-volatile

问题：你认为哪一种最适合做计算机的存储器呢？

单独用某一种存储器，都不能满足我们的需要！

采用分层存储结构来构建计算机的存储体系！

计算机中存储器的层次结构



给出的时间和容量会随时间变化，但数量级相对关系不变。

计算机中存储器的层次结构

- 分析：速度越快，成本越高
- 为提高性能/价格，组成一个层状塔式结构，取长补短，协调工作
- 工作过程：

  1）CPU运行时，需要的操作数大部分来自寄存器

  2）如需要从(向)存储器中取(存) 数据时，先访问cache，如在，取自cache

  3）如操作数不在cache，则访问RAM，如在RAM中，则取自RAM

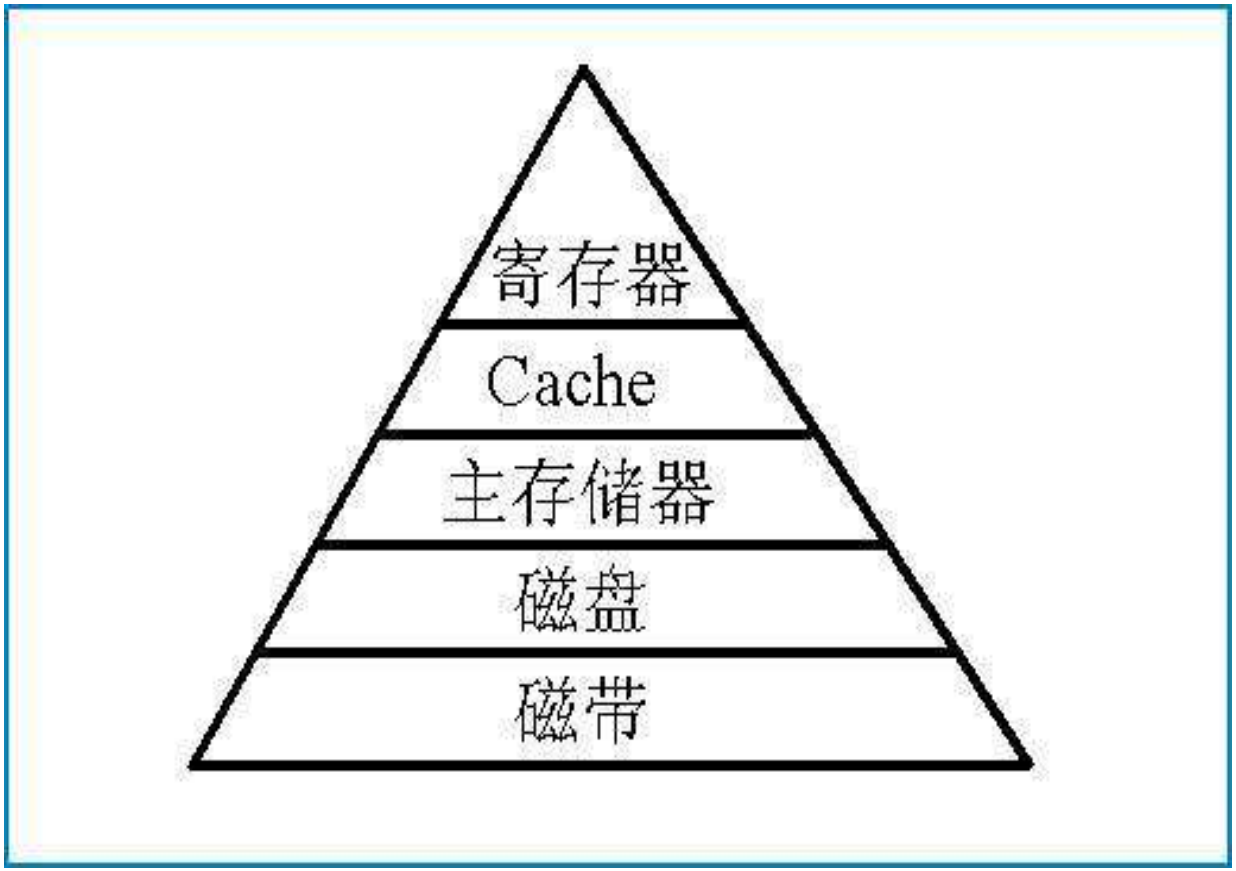
  4）如操作数不在RAM，则访问硬盘，操作数从硬盘中读出→RAM →cache

回顾：传统存储器分级体系结构

五层金字塔形分层系统从上到下的特点：

1. 每位价格降低
2. 容量增大
3 .存取时间增大
4 .访问频度降低

传统结构



Traditional Memory Hierarchy

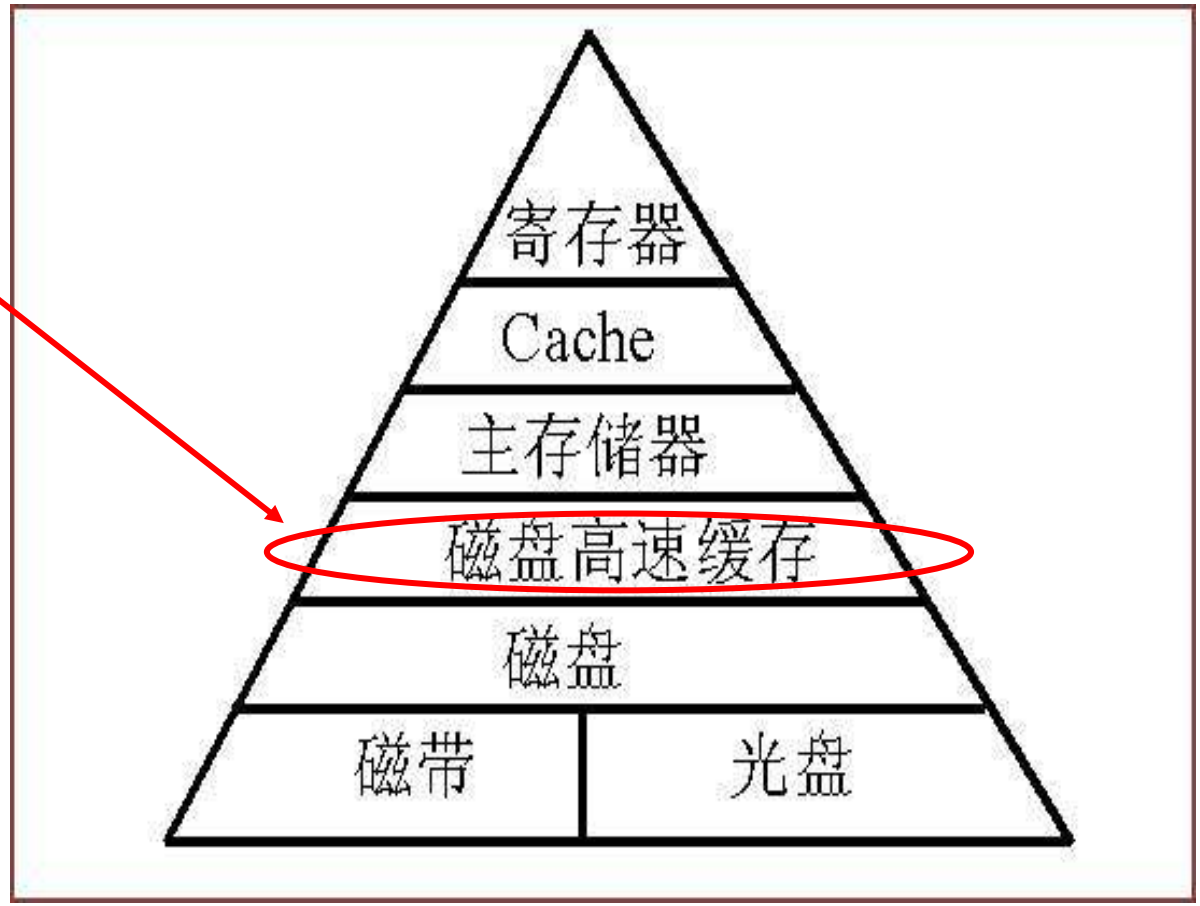
现代存储器分级体系结构

开辟一部分内存区，用作“**Disk Cache**”，用于存放将被送到磁盘上的数据。

引入“**Disk Cache**”的好处：

(1)写盘时按“簇”进行，以避免频繁地小块数据写盘。

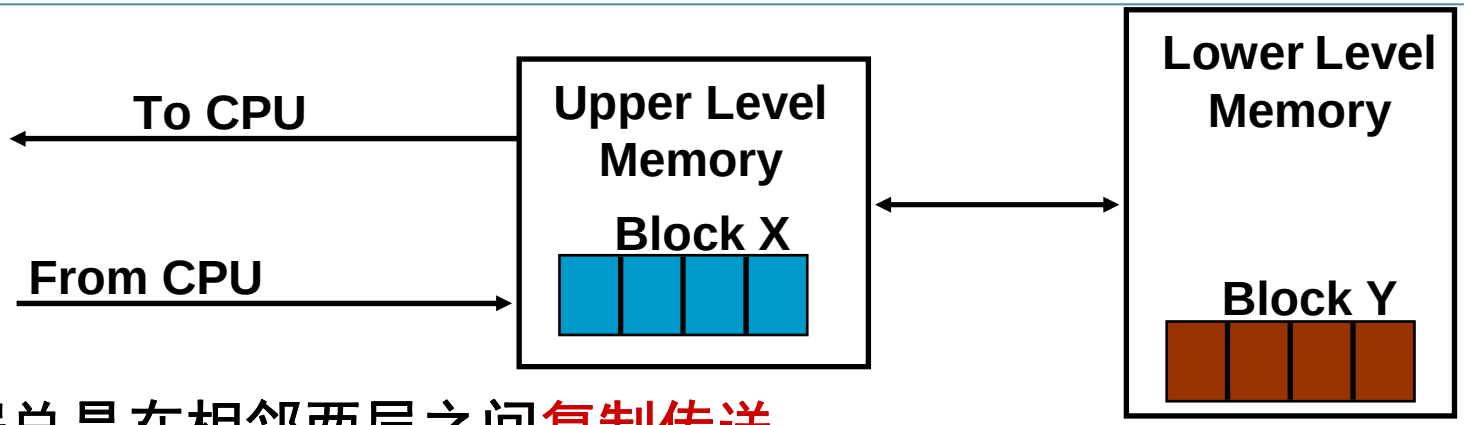
(2)有些中间结果数据在写回盘之前可被快速地再次使用。



Contemporary Memory Hierarchy

（现代结构）

层次化存储器结构（Memory Hierarchy）



- 数据总是在相邻两层之间复制传送
  - Upper Level: 上层更靠CPU
  - Lower Level: 下层更远离CPU
- Block: 最小传送单位是一个定长块，互为副本

问题：为什么这种层次化结构是有效的？

基于“程序访问局部化”特点！

- 时间局部性（Temporal Locality）
  - 含义：刚被访问过的单元很可能不久又被访问
  - 做法：让最近被访问过的信息保留在靠近CPU的存储器中
- 空间局部性 （Spatial Locality）
  - 含义：刚被访问过的单元的邻近单元很可能不久被访问
  - 做法：将刚被访问过的单元的邻近单元调到靠近CPU的存储器中

加快访存速度措施之三：引入Cache

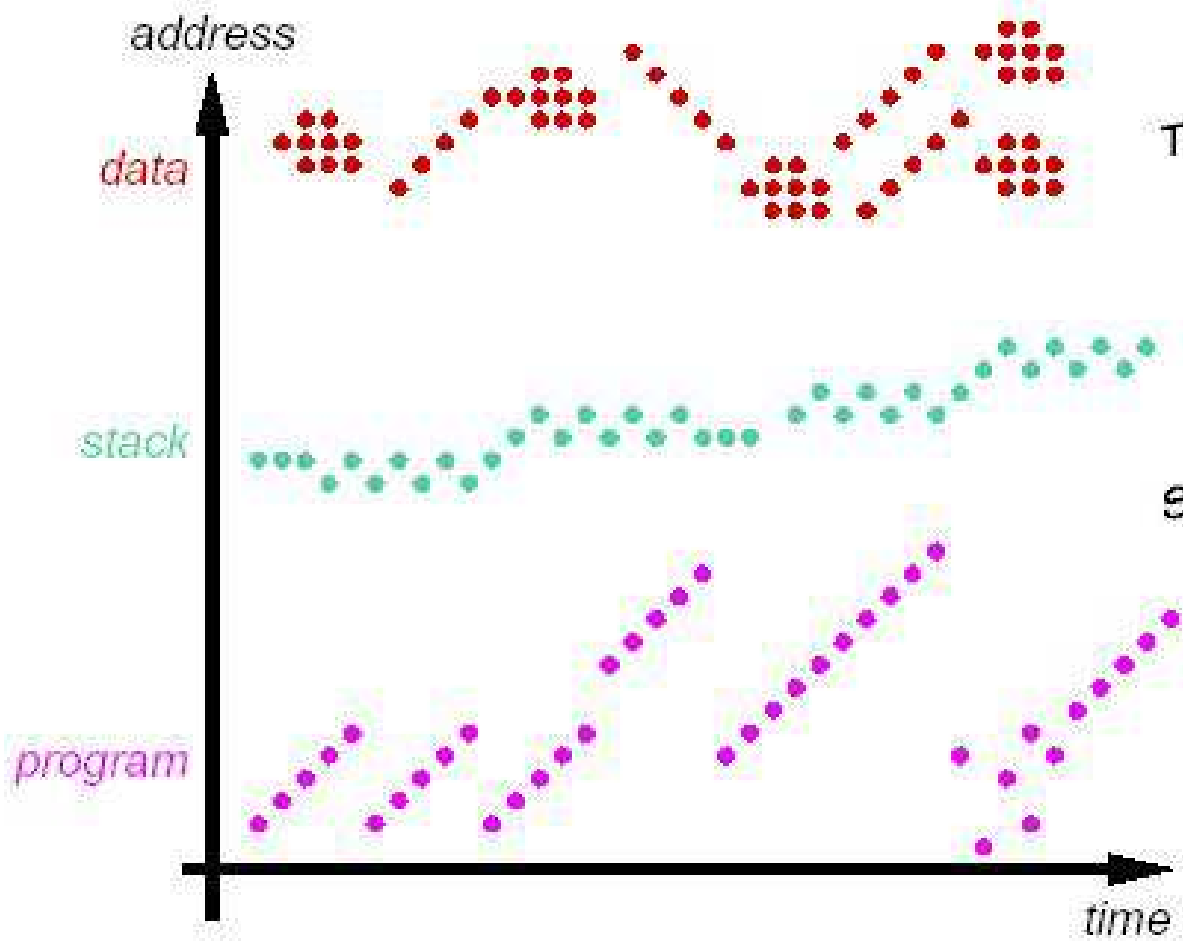
- 大量典型程序的运行情况分析结果表明
  - 在较短时间间隔内，程序产生的地址往往集中在一个很小范围内

  这种现象称为程序访问的局部性
- 程序具有访问局部性特征的原因
  - 指令：指令按序存放，地址连续，循环程序段或子程序段重复执行
  - 数据：连续存放，数组元素重复、按序访问
- 程序访问局部性分为空间局部性和时间局部性
- 基于程序访问的局部性使访存要求能快速得到响应
  - 在CPU和主存之间设置一个快速小容量的存储器，其中总是存放最活跃（被频繁访问）的程序块和数据，由于程序访问的局部性特征，大多数情况下，CPU能直接从这个高速缓存中取得指令和数据，而不必访问主存。

这个高速缓存就是位于主存和CPU之间的Cache！

SKIP

Typical Memory Reference Patterns



TEMPORAL LOCALITY –
If an item is referenced, it will tend to be referenced again soon

SPATIAL LOCALITY –
If an item is referenced, nearby items will tend to be referenced soon.

下面用一个例子来说明！

程序的局部性原理举例1

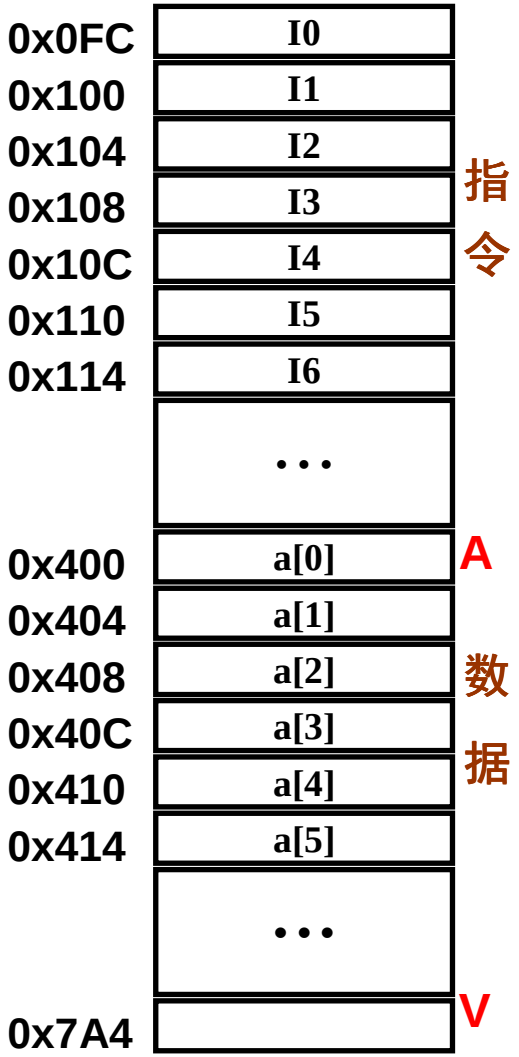
高级语言源程序

```
sum = 0;
for (i = 0; i < n; i++)
      sum += a[i];
*v = sum;
```

对应的汇编语言程序

```
I0:            sum  <-- 0
I1:            ap  <-- A   A是数组a的起始地址
I2:            i   <-- 0
I3:            if (i >= n) goto done
I4:   loop:    t   <-- (ap) 数组元素a[i]的值
I5:            sum <-- sum + t   累计在sum中
I6:            ap  <-- ap + 4   计算下个数组元素地址
I7:            i   <-- i + 1
I8:            if (i < n) goto loop
I9:   done:    V  <-- sum   累计结果保存至地址v
```

主存的布局:



每条指令4个字节；每个数组元素4字节

指令和数组元素在内存中均连续存放

sum, ap ,i, t 均为通用寄存器；A，V为内存地址

程序的局部性原理举例1

```
sum = 0;
for (i = 0; i < n; i++)
      sum += a[i];
*v = sum;
```

问题：指令和数据的时间局部性和空间局部性各自体现在哪里？

指令：　0x0FC（I0）

→0x108（I3）

→0x10C（I4）

→0x11C（I8）　循环n次

→0x120（I9）

数据：只有数组在主存中：

0x400→0x404→0x408

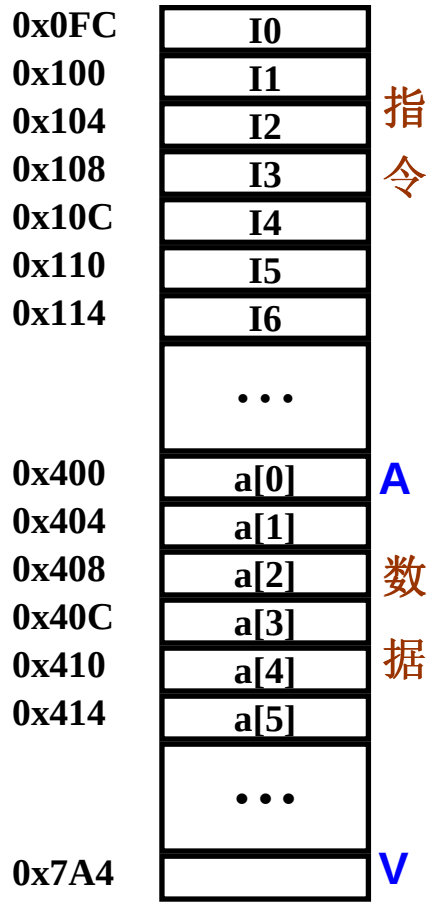
→0x40C→……→0x7A4

若n足够大，则在一段时间内一直在局部区域内执行指令，故循环内指令的时间局部性好；按顺序执行，故程序的空间局部性好！

数组元素按顺序存放，也按顺序访问，所以，空间局部性好；

每个数组元素都被访问1次，所以没有时间局部性。

主存的布局：



程序的局部性原理举例2

以下哪个对数组A引用的空间局部性更好？时间局部性呢？变量sum的空间局部性和时间局部性如何？对于指令来说，for循环体的空间局部性和时间局部性如何？

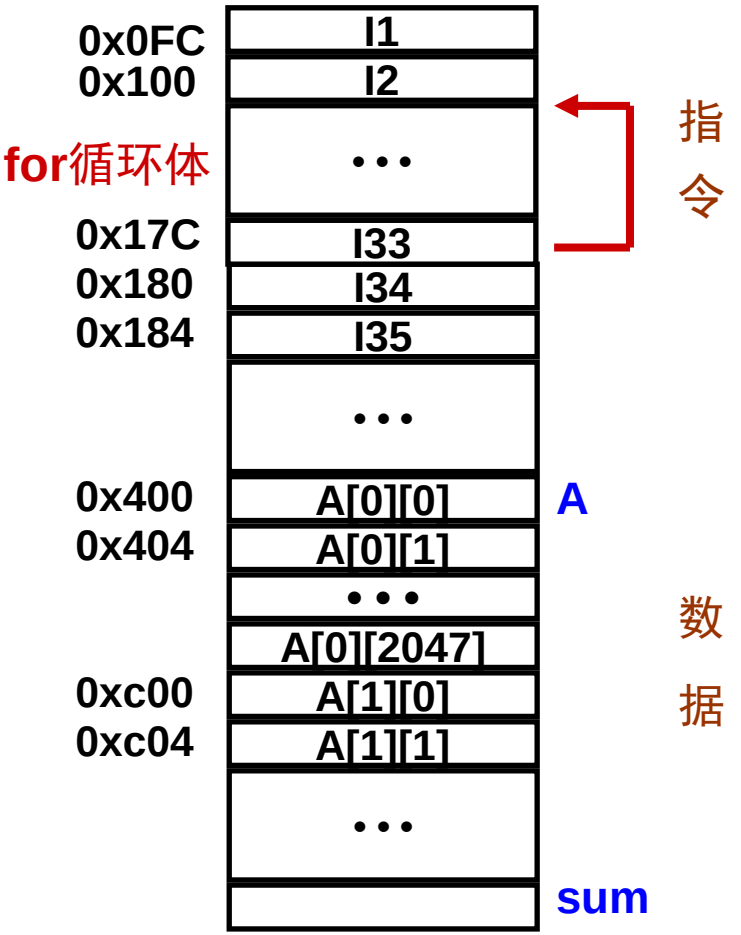
程序段A:

```
int sumarrayrows(int A[M][N])
{
  int i, j, sum=0;
     for  (i=0; i<M, i++)
         for (j=0; j<N, j++)  sum+=A[i][j];
     return sum;
}
```

程序段B:

```
int sumarraycols(int A[M][N])
{
   int i, j, sum=0;
       for  (j=0; j<N, j++)
           for (i=0; i<M, i++)  sum+=A[i][j];
        return sum;
}
```

M=N=2048时主存的布局:



假定数组在存储器中按行优先顺序存放

程序的局部性原理举例2

程序段A的时间局部性和空间局部性分析

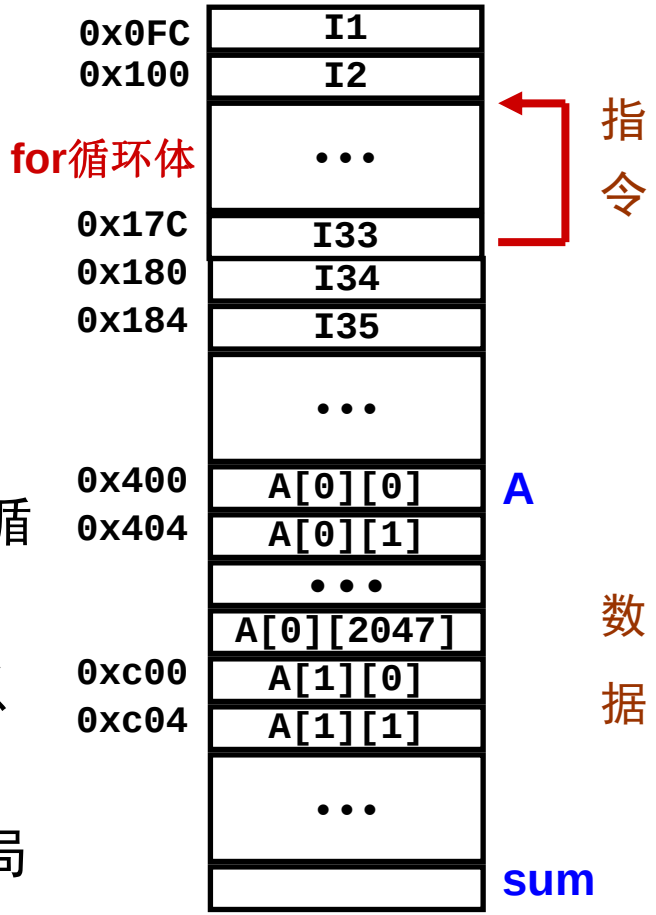
（1）数组A：访问顺序为A[0][0], A[0][1] ,……, A[0][2047]; A[1][0], A[1][1],…… ,A[1][2047]; ……，与存放顺序一致，故空间局部性好！

因为每个A[i][j]只被访问一次，故时间局部性差！

（2）变量sum：单个变量不考虑空间局部性；每次循环都要访问sum，所以其时间局部性较好！

（3） for循环体：循环体内指令按序连续存放，所以空间局部性好！

循环体被连续重复执行2048x2048次，所以时间局部性好！



实际上 优化的编译器使循环中的sum分配在寄存器中，最后才写回存储器！

程序的局部性原理举例2

程序段B的时间局部性和空间局部性分析

（1）数组A：访问顺序为A[0][0], A[1][0] ,……，A[2047][0]; A[0][1], A[1][1],…… ,A[2047][1];……，与存放顺序不一致，每次跳过2048个单元，若交换单位小于2KB，则没有空间局部性！

（时间局部性差，同程序A）

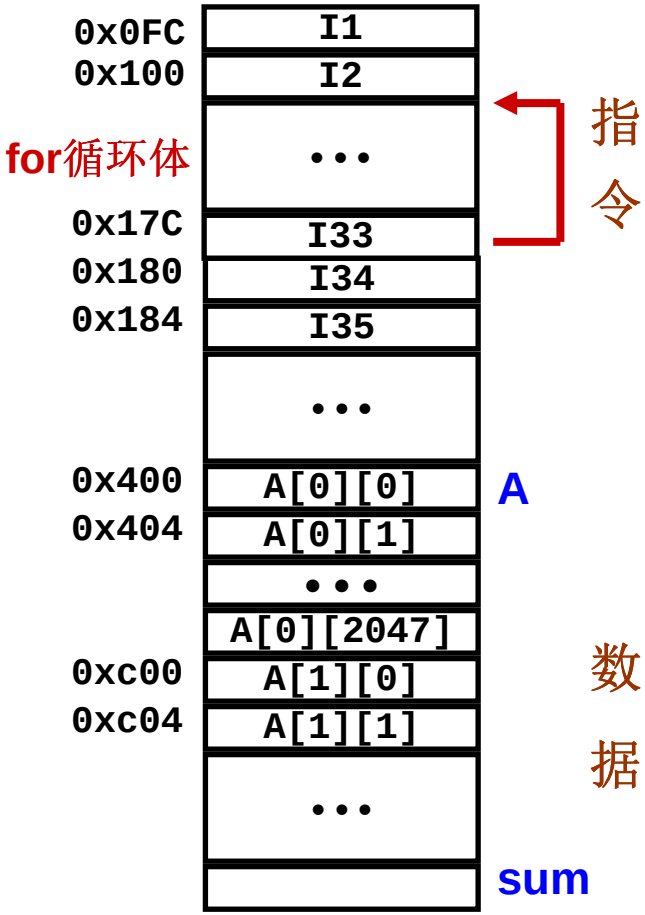
（2）变量sum：（同程序A ）

（3） for循环体：（同程序A）

实际运行结果(2GHz Intel Pentium 4):

程序A：59,393,288 时钟周期

程序B：1,277,877,876 时钟周期



程序A比程序B快 21.5 倍!!

BACK

Cache(高速缓存)是什么样的？

- Cache是一种小容量高速缓冲存储器，它由SRAM组成。
- Cache直接制作在CPU芯片内，速度几乎与CPU一样快。
- 程序运行时，CPU使用的一部分数据/指令会预先成批拷贝在Cache中，Cache的内容是主存储器中部分内容的映象。
- 当CPU需要从内存读(写)数据或指令时，先检查Cache，若有，就直接从Cache中读取，而不用访问主存储器。