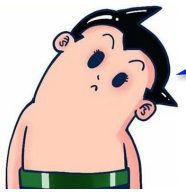
5.3 Cache高速缓存

刘 芳 副教授
国防科学技术大学计算机学院

5.3.1 程序访问局部性

计算机需要什么样的存储器？

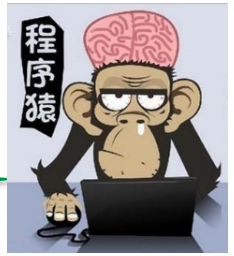


已经学过的存储器：SRAM、DRAM、ROM
还要学习的存储器：Hard Disk、Optical Disk、Flash Memory

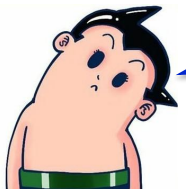
	Capacity	Latency	Cost
Register	<1KB	1ns	$$$$
SRAM	1MB	2ns	$$$
DRAM	1GB	10ns	$
Hard disk*	100GB	10ms	¢
Want	100GB	1ns	cheap

* non-volatile

哪一种是最适合做计算机的存储器呢？



计算机需要什么样的存储器？



已经学过的存储器：SRAM、DRAM、ROM
还要学习的存储器：Hard Disk、Optical Disk、Flash Memory

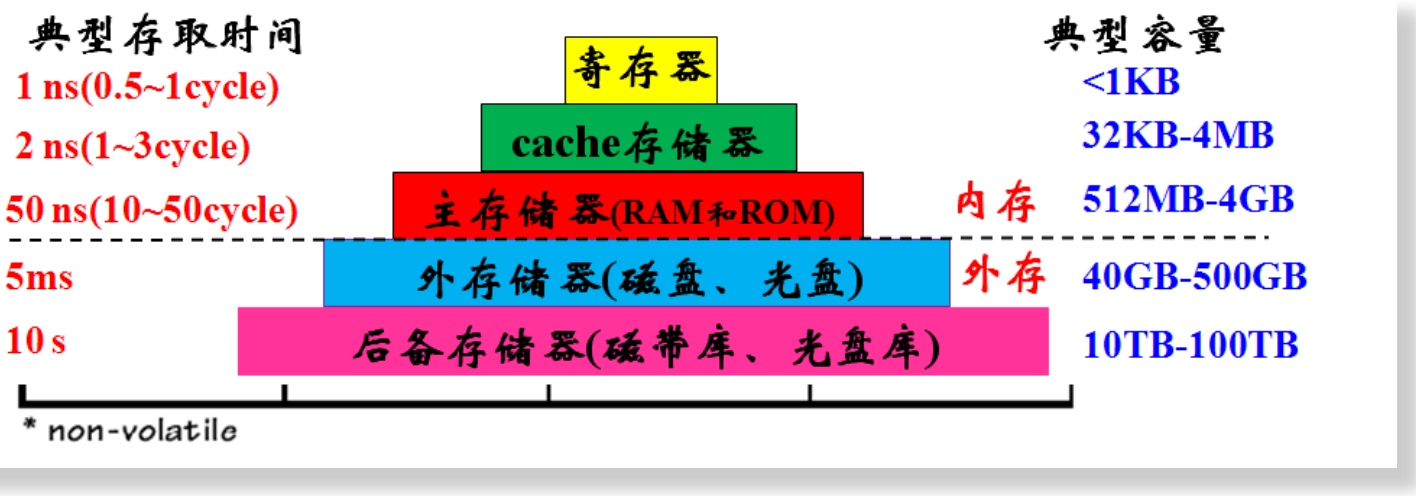
单独用某一种存储器不能满足我们的需要！

结合各种存储器的特点，采用层次式存储器结构来构建计算机的存储系统！

哪一种是最适合做计算机的存储器呢？

5.3.1 程序访问局部性

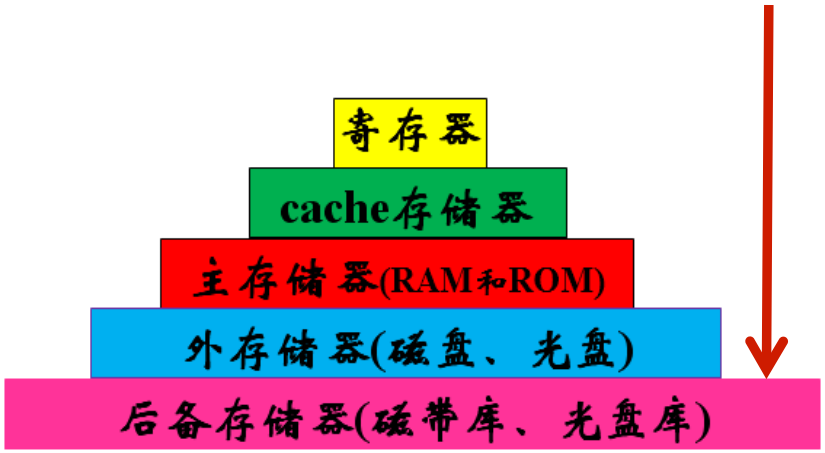
速度越快，容量越小、成本越高



计算机存储层次结构

为提高性能/价格，将各存储器组成一个金字塔式的层次结构，取长补短协调工作

计算机存储层次结构



工作过程

- CPU运行时，需要的操作数大部分来自寄存器
- 需从(向)存储器中取(存) 数时，先访问cache；若在，取自cache
- 若不在cache，则访问RAM；若在，则取自RAM
- 若不在RAM，则访问磁盘，操作数从磁盘中读出→RAM →cache

计算机存储层次结构

离处理器较近的数据是较远的那些层次的子集，最底层存放着所有数据。离处理器越远的存储器访问时间越长

数据总是在相邻两层之间复制传送

- Cache：更靠近CPU

  Smaller, faster, and uses more expensive technology

- 主存： 更远离CPU

  Bigger, slower, and uses less expensive technology

计算机存储层次结构

离处理器较近的数据是较远的那些层次的子集，最底层存放着所有数据。离处理器越远的存储器访问时间越长

数据总是在相邻两层之间复制传

- Cache：更靠近CPU
  Smaller, faste
- 主存： 更远离CPU
  Bigger, slower, and uses less expensive technology

- 主存是一个“不争气的孩子”，不像人们期望的那样越来越快，而是越变越胖
- CPU访问主存的相对延时进一步扩大Latency Improvement < Bandwidth Improvement

计算机存储层次结构

块(Block)是一个定长块，是两层存储器之间存储信息的最小单元。Cache是主存一部分的副本

为什么存储层次化结构是有效的？

“程序访问局部性”特点！

5.3.1 程序访问局部性

什么是程序访问局部性？

分析大量典型程序的运行情况，结果表明：

在较短时间间隔内，程序产生的地址/所访问数据往往集中在存储器的一个很小范围内

为什么会具有局部性？

指令：按序存放，地址连续。如循环程序段等重复执行

数据：连续存放。如数组元素重复、按序访问

程序访问局部性分为

时间局部性(Temporal Locality) ：刚被访问过的存储单元很可能不久又被访问

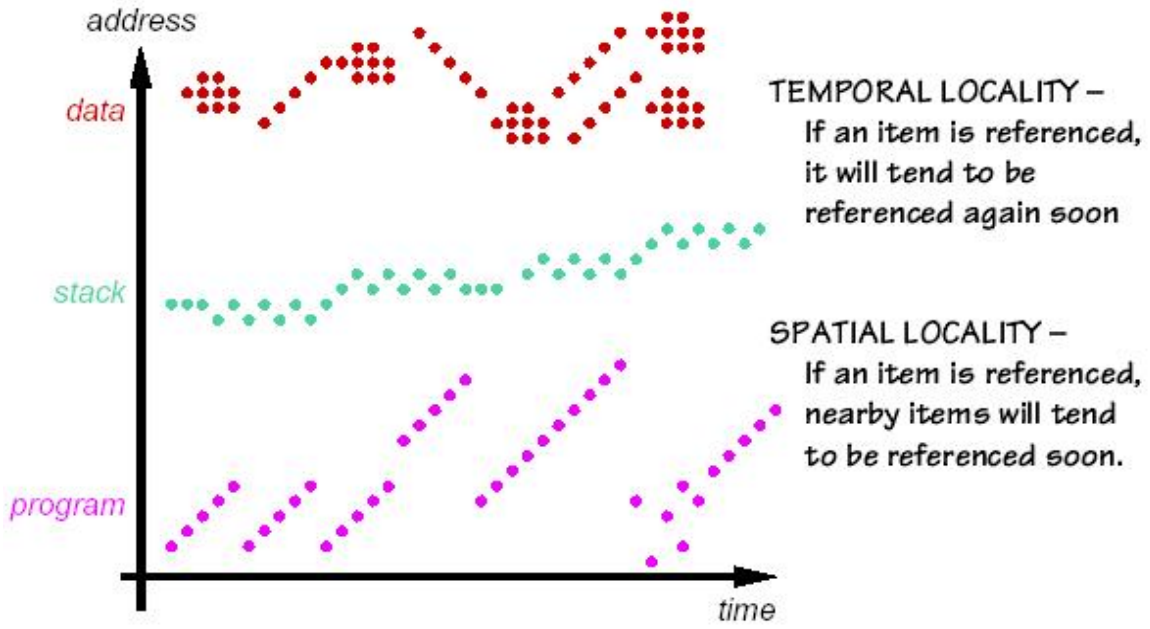
让最近被访问过的信息保留在靠近CPU的存储器中

空间局部性 (Spatial Locality) ：刚被访问过的存储单元的邻近单元很可能不久被访问

将刚被访问过的存储单元的邻近单元调到靠近CPU的存储器中

5.3.1 程序访问局部性

Typical Memory Reference Patterns



5.3.1 程序访问局部性（Ⅱ）

程序局部性原理举例1

高级语言源程序

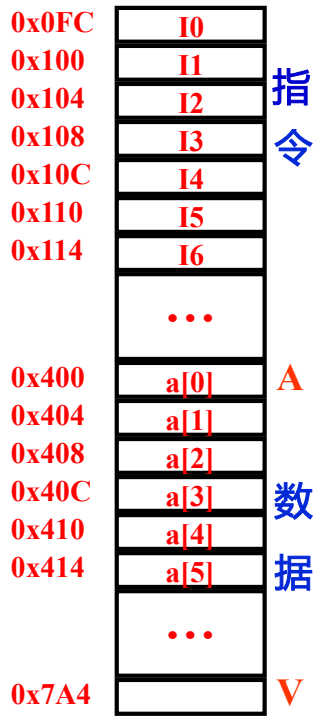
```
sum = 0;
for (i = 0; i < n; i++)
        sum += a[i];
*v = sum;
```

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   计算下个元素地址
I7:          i ← i + 1
I8:          if (i < n) goto loop
I9:   done:  V ← sum   结果保存至地址V
```

- sum, ap ,i, t 已存放在通用寄存器；A,V为内存地址
- 指令：4字节；数组元素：4字节
- 指令和数组元素在主存中均连续存放

主存的布局



问题：指令和数据的时间局部性和空间局部性？

计算机需要什么样的存储器？

问题：指令和数据的时间局部性和空间局部性？

指令：

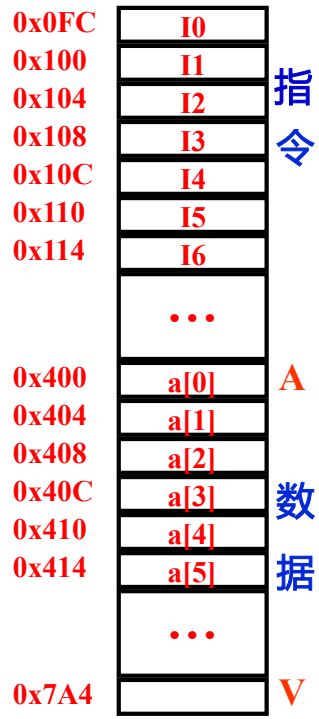
0x0FC(I0)→ ... →0x108(I3)→0x10C(I4)→…→0x11C(I8) →0x120(I9)

N次

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

```
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   计算下个元素地址
I7:          i ← i + 1
I8:          if (i < n) goto loop
I9:   done:  V ← sum   结果保存至地址V
```

主存的布局



计算机需要什么样的存储器？

问题：指令和数据的时间局部性和空间局部性？

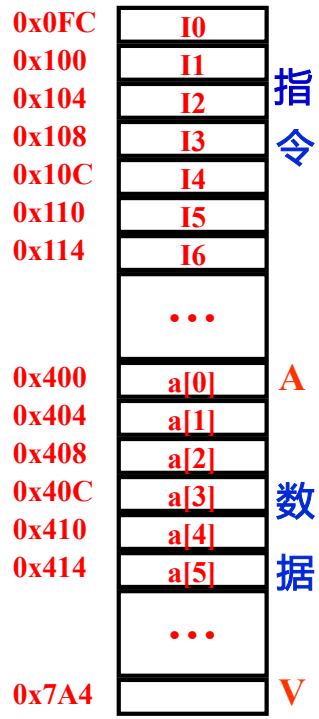
数据：仅数组在主存中

0x400→0x404→0x408→0x40C→……→0x7A4

- 数组元素按顺序存放，也按顺序访问，所以，空间局部性好；
- 每个数组元素只被访问1次，所以没有时间局部性

```
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   计算下个元素地址
I7:          i ← i + 1
I8:          if (i < n) goto loop
I9:   done:  V ← sum   结果保存至地址V
```

主存的布局



程序局部性原理举例2

程序A和B中：

- 对数组A引用的局部性？
- 变量sum的局部性如何？
- 对于指令，for循环体的局部性如何？

程序段A：

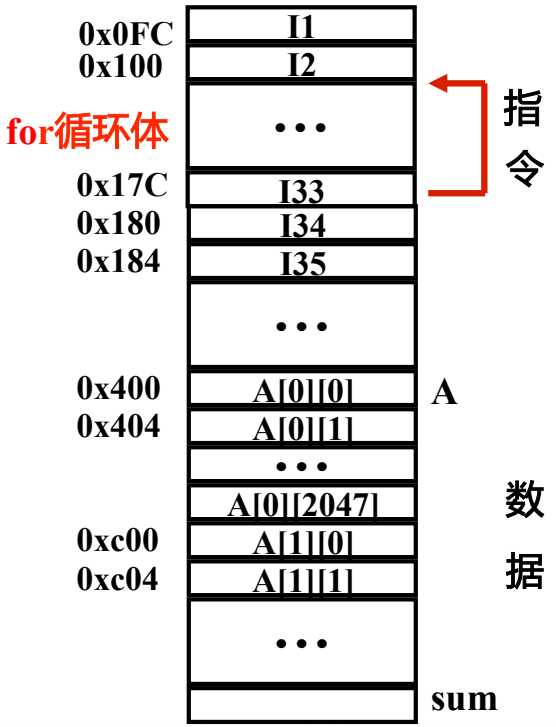
```
int sumarrayrows(int A[M][N])
{
  int i, j, sum=0;
     for  (i=0; i<M, i++)
          for (j=0; j<N, j++)
            sum+=A[i][j];
     return sum;
}
```

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;
}
```

假定数组在存储器中按行优先顺序存放

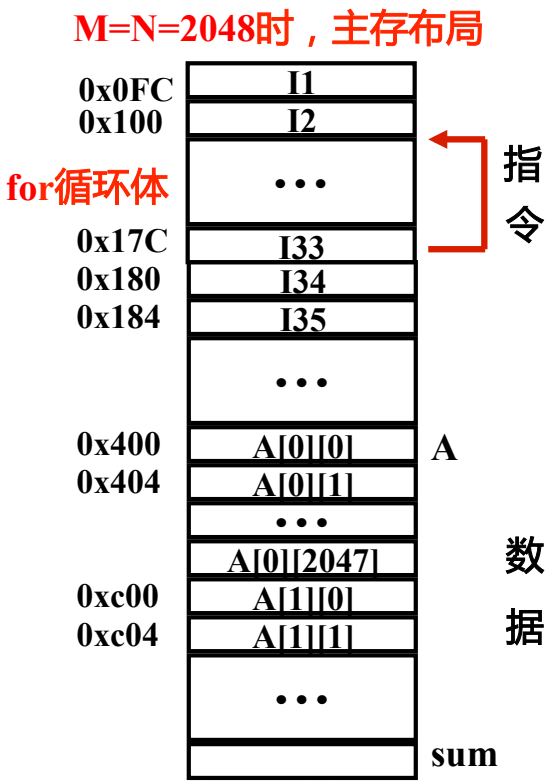
M=N=2048时，主存布局



计算机需要什么样的存储器？

```
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;
}
```

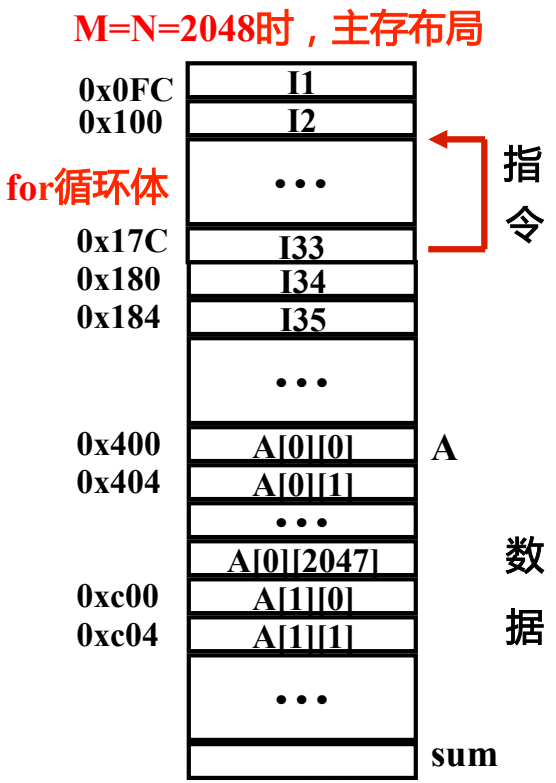
- 数组A：访问顺序为A[0][0], A[0][1] ,…, A[0][2047]; A[1][0], A[1][1],…,A[1][2047]，与存放顺序一致，空间局部性好！而每个A[i][j]只被访问一次，故时间局部性差
- 变量sum：单个变量不考虑空间局部性；每次循环都访问sum，故其时间局部性好！
- for循环体：循环体内指令按序连续存放，空间局部性好！循环体被连续重复执行2048×2048次，故时间局部性好！



计算机需要什么样的存储器？

```
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;
}
```

- 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CPU与Cache数据交换单位小于2KB，则没有空间局部性！时间局部性差(同程序A)
- 变量sum：(同程序A)
- for循环体：(同程序A)



计算机需要什么样的存储器？

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;
}
```

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B快21.5 倍!!

在评估机器性能时，存储器系统性能是一个重要指标。利用存储器的层次结构能够提高性能，意味着过去程序员可以把存储器看作是一个平面的随机访问存储设备，而现在必须理解存储层次结构如何工作才能使系统达到良好的性能。