第4周　高速缓存概述

第1讲　存储器层次结构概述

第2讲　Cache基本概述

第3讲　Cache映射方式

第4讲　Cache命中率和缺失率

第5讲　Cache的关联度

希望的理想存储器

到目前为止，已经了解到有以下几种存储器：

寄存器，SRAM，DRAM， 硬盘

	Capacity	Latency	Cost
Register	<1KB	1ns	$$$$
SRAM	1MB	2ns	$$$
DRAM	1GB	10ns	$
Hard disk*	1000GB	10ms	¢
Want	100GB	1ns	cheap

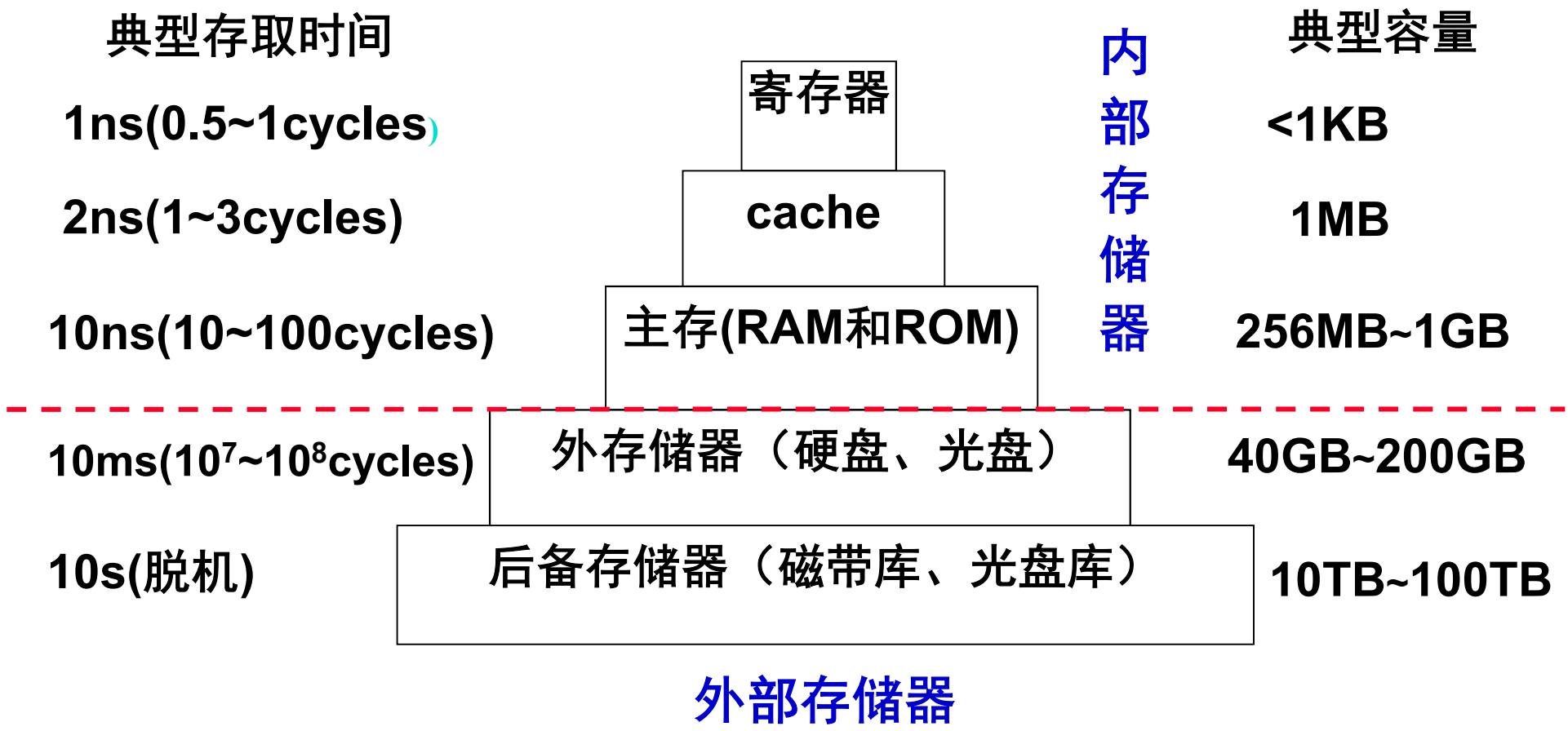
* non-volatile

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！

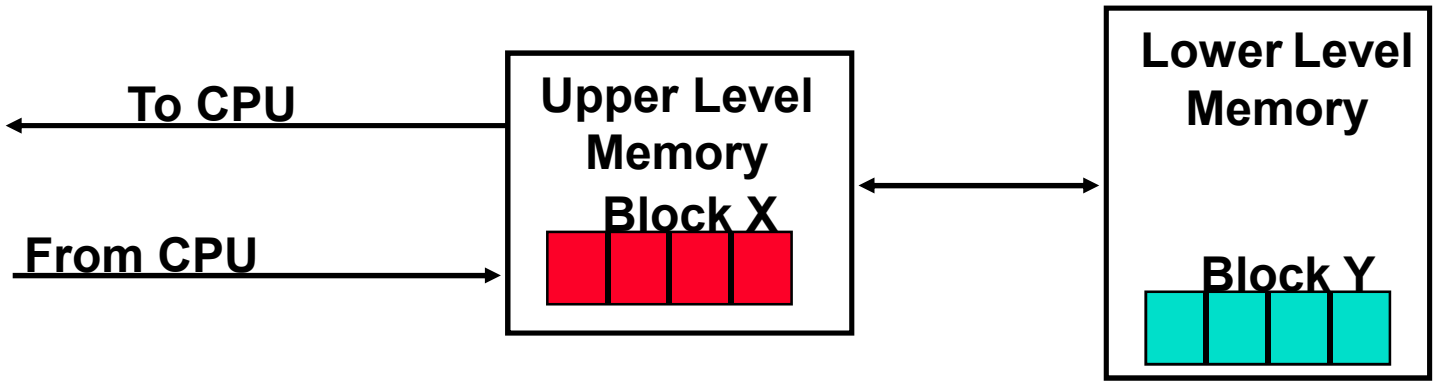
采用分层存储结构来构建计算机的存储体系！

存储器的层次结构



列出的时间和容量会随时间变化，但数量级相对关系不变。

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: 最小传送单位是定长块，互为副本

相当于工厂中设置了多级仓库！

问题：为什么这种层次化结构是有效的？

程序访问局部化特点！
例如，写论文时图书馆借参考书：欲借书附近的书也是欲借书！

- 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：空间局部性、时间局部性
- 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
- 为什么引入Cache会加快访存速度？
  - 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！

程序的局部性原理举例1

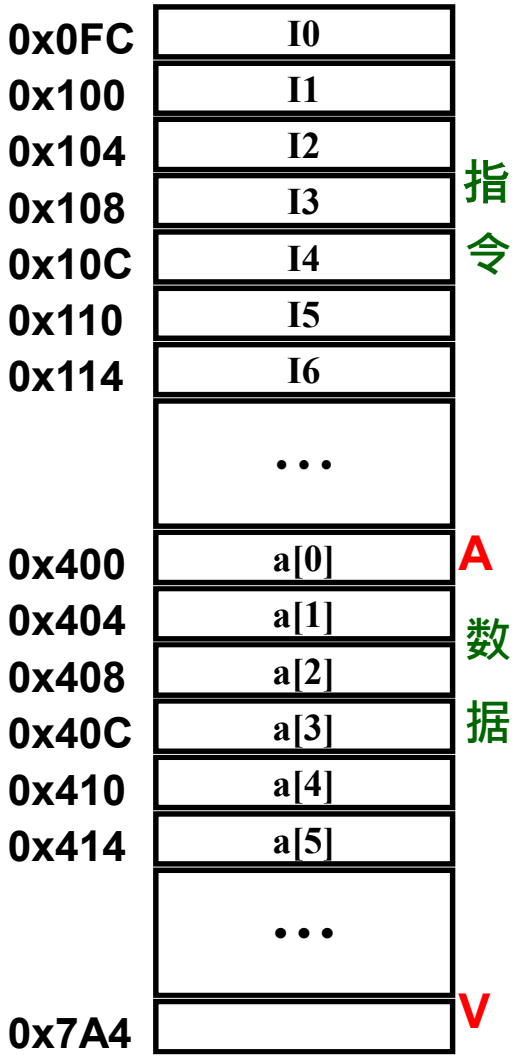
高级语言源程序

```
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）

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，故程序空间局部性好！

数据：只有数组在主存中：

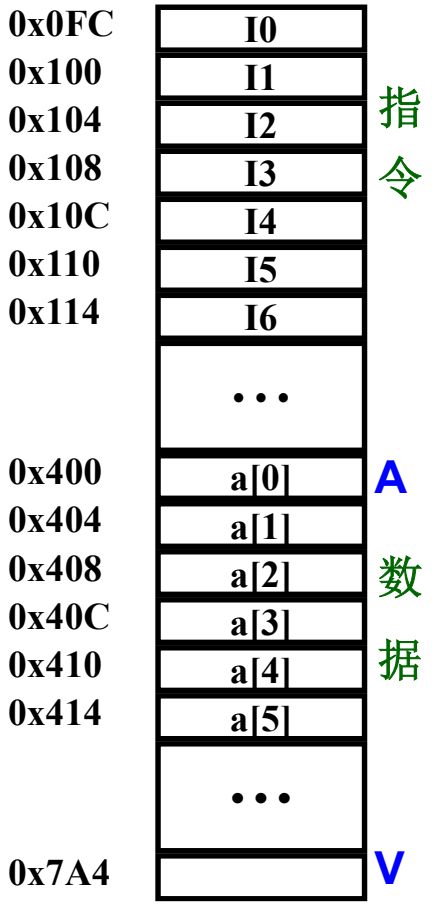
0x400→0x404→0x408

→0x40C→……→0x7A4

数组元素按顺序存放，按顺序访问，故空间局部性好；

每个数组元素都只被访问1次，故没有时间局部性。

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？

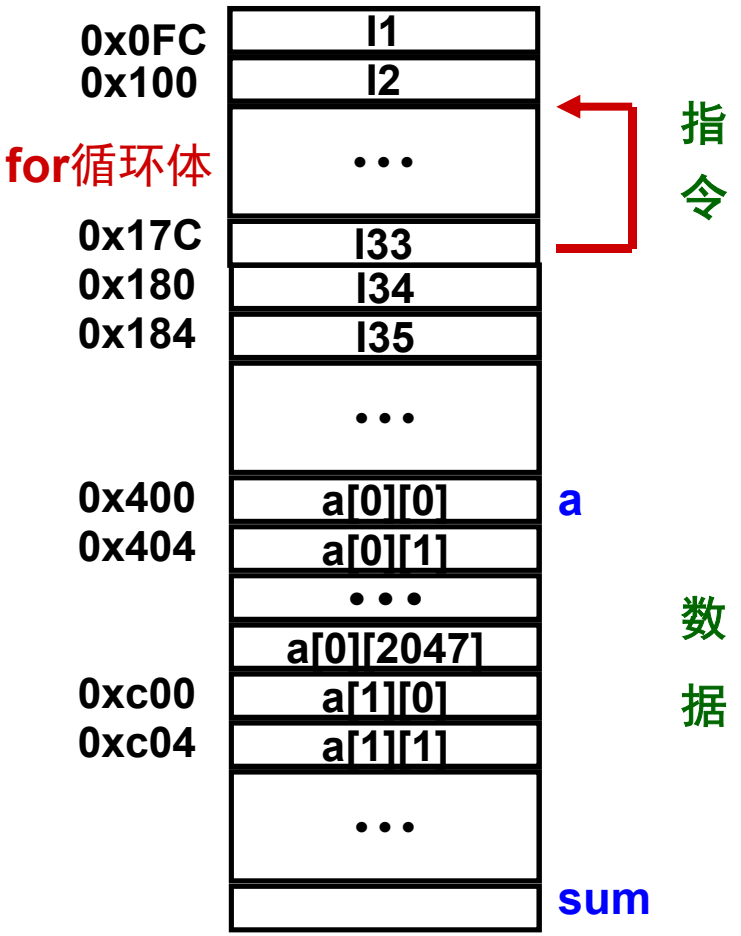
程序段A:

```
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:



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];

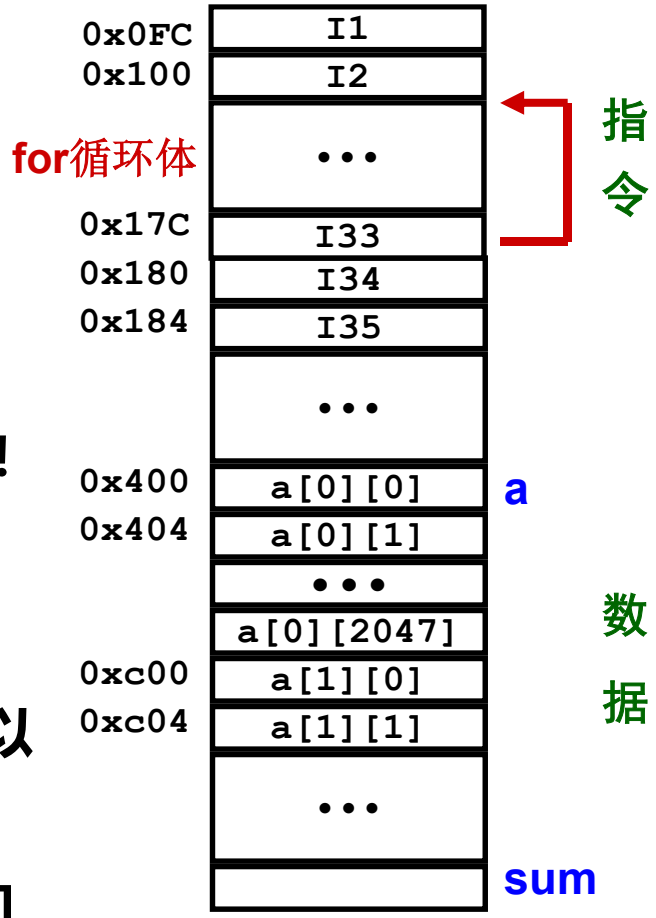
……，与存放顺序一致，故空间局部性好！

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

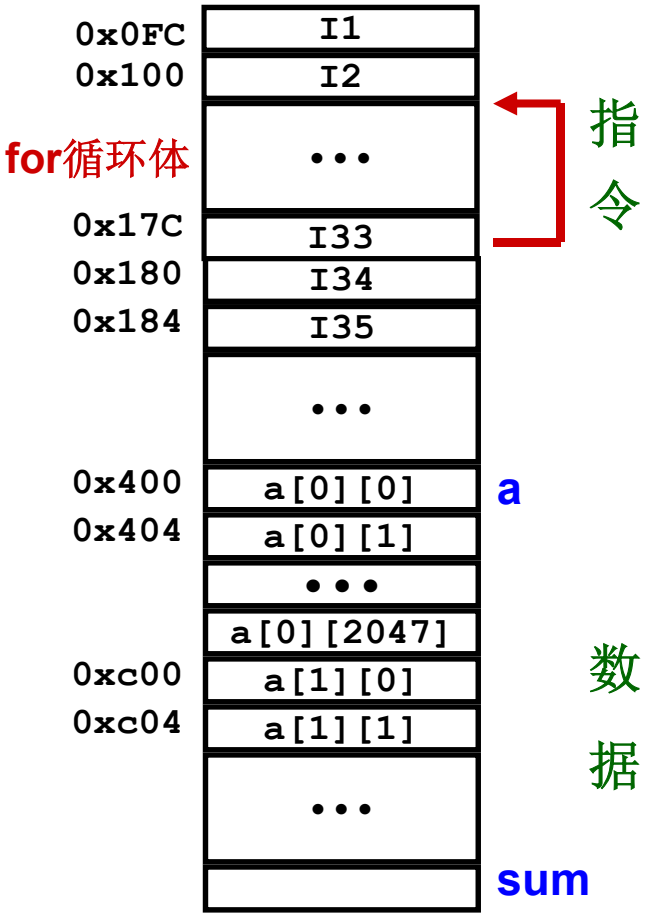
程序段B的时间局部性和空间局部性分析

（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):

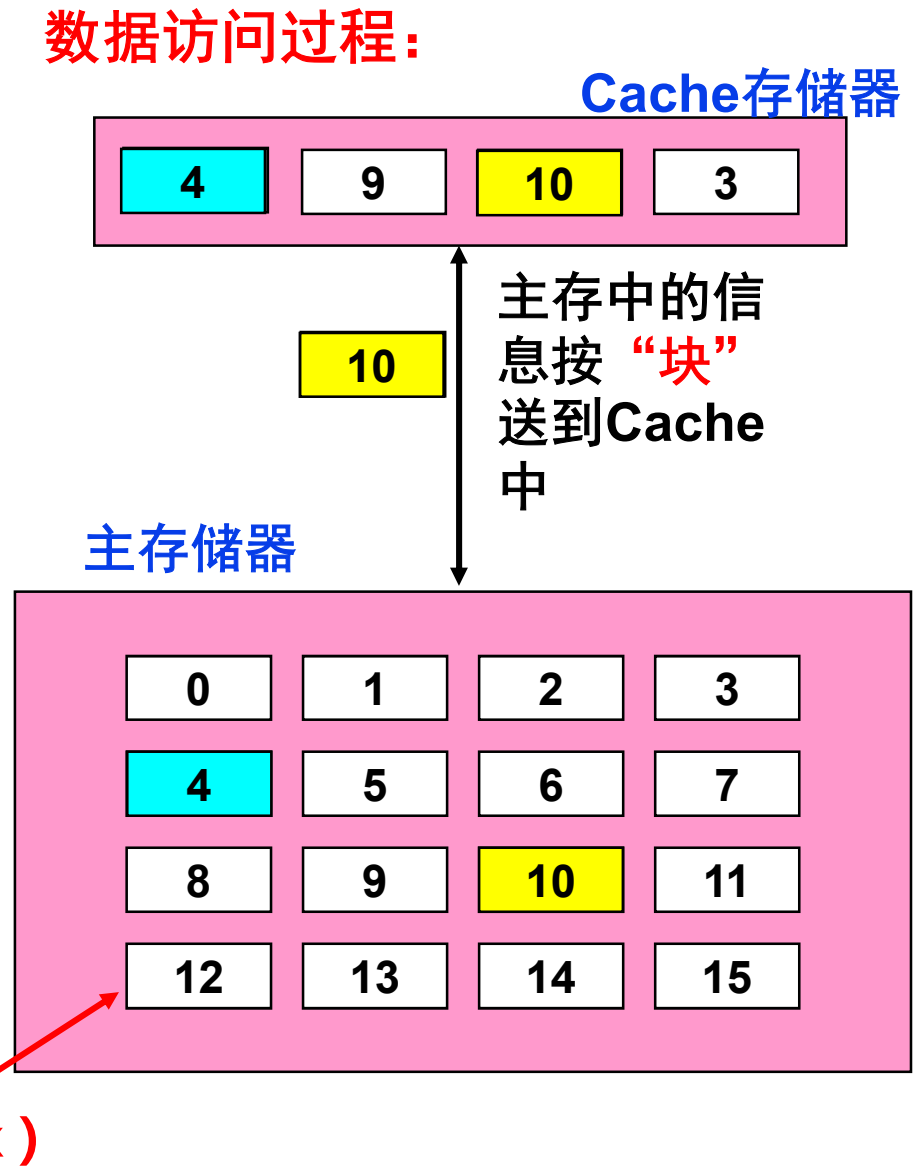
程序A：59,393,288 时钟周期

程序B：1,277,877,876 时钟周期

程序A比程序B快21.5 倍!!

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。



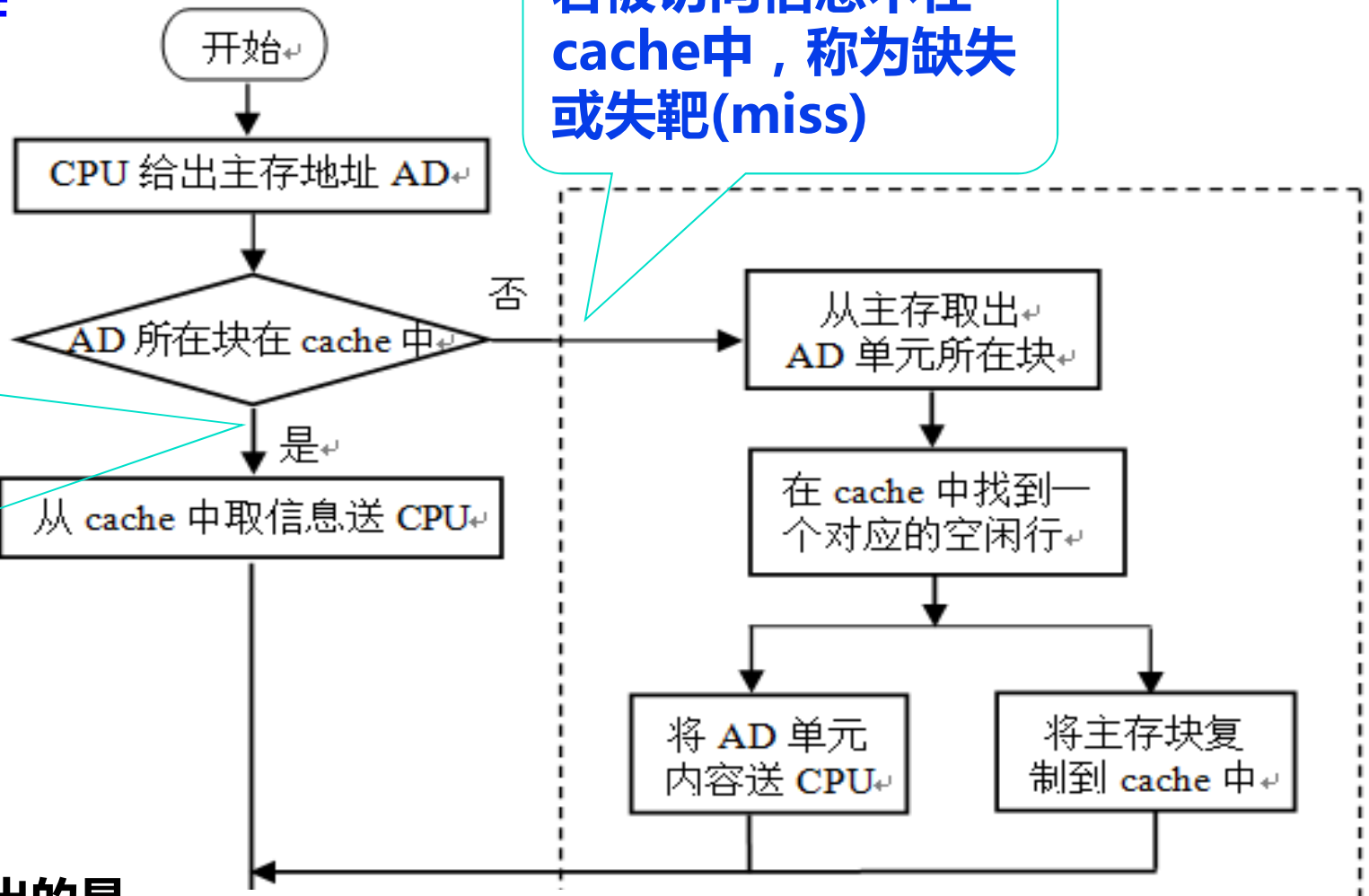
Cache 的操作过程

问题：什么情况下，CPU产生访存要求？

执行指令时！

若被访问信息不在cache中，称为缺失或失靶(miss)

若被访问信息在cache中，称为命中(hit)



指令最初给出的是虚拟地址！

如果将虚拟地址转换为主存地址，后面介绍。

Cache（高速缓存）的实现

问题：要实现Cache机制需要解决哪些问题？

如何分块？

主存块和Cache之间如何映射?

Cache已满时，怎么办？

写数据时怎样保证Cache和MM的一致性？

如何根据主存地址访问到cache中的数据？……

主存被分成若干大小相同的块，称为主存块(Block)，Cache也被分成相同大小的块，称为Cache行（line）或槽（Slot）。

问题：Cache对程序员(编译器)是否透明？为什么？

是透明的，程序员(编译器)在编写/生成高级或低级语言程序时无需了解Cache是否存在或如何设置，感觉不到cache的存在。

但是，对Cache深入了解有助于编写出高效的程序！

Cache映射(Cache Mapping)

- 什么是Cache的映射功能？
  - 把访问的局部主存区域取到Cache中时，该放到Cache的何处？
  - Cache行比主存块少，多个主存块映射到一个Cache行中
- 如何进行映射？
  - 把主存空间划分成大小相等的主存块（Block）
  - Cache中存放一个主存块的对应单位称为槽（Slot）或行（line）

    有书中也称之为块（Block），有书称之为页（page）（不妥！）
  - 将主存块和Cache行按照以下三种方式进行映射
    - 直接(Direct)：每个主存块映射到Cache的固定行
    - 全相联(Full Associate)：每个主存块映射到Cache的任一行
    - 组相联(Set Associate)：每个主存块映射到Cache固定组中任一行

The Simplest Cache: Direct Mapped Cache

° Direct Mapped Cache (直接映射Cache举例)

- 把主存的每一块映射到一个固定的Cache行（槽）
- 也称模映射(Module Mapping)
- 映射关系为：

块（行）都从0开始编号

Cache行号=主存块号 mod Cache行数

举例：4=100 mod 16 （假定Cache共有16行）

(说明：主存第100块应映射到Cache的第4行中。)

◆ 特点：

- 容易实现，命中时间短
- 无需考虑淘汰（替换）问题
- 但不够灵活，Cache存储空间得不到充分利用，命中率低

SKIP

例如，需将主存第0块与第16块同时复制到Cache中时，由于它们都只能复制到Cache第0行，即使Cache其它行空闲，也有一个主存块不能写入Cache。这样就会产生频繁的 Cache装入。

直接映射Cache组织示意图

假定数据在主存和Cache间的传送单位为512B。

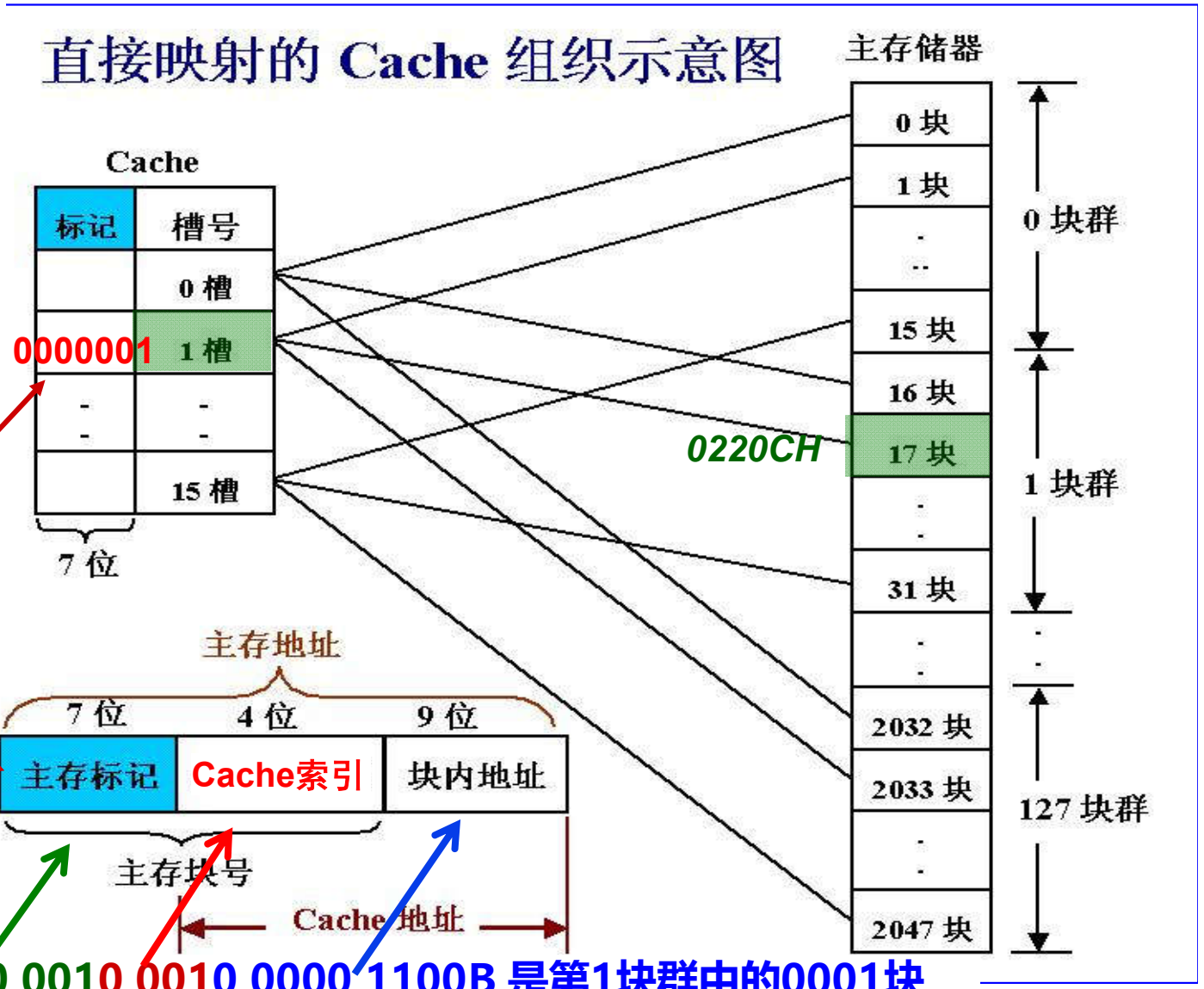
Cache大小：
2^{13}B=8KB=16行 x 512B/ 行

主存大小：
2^{20}B=1024KB=2048块 x 512B/块

Cache标记(tag)指出对应行取自哪个主存块群

指出对应地址位于哪个块群

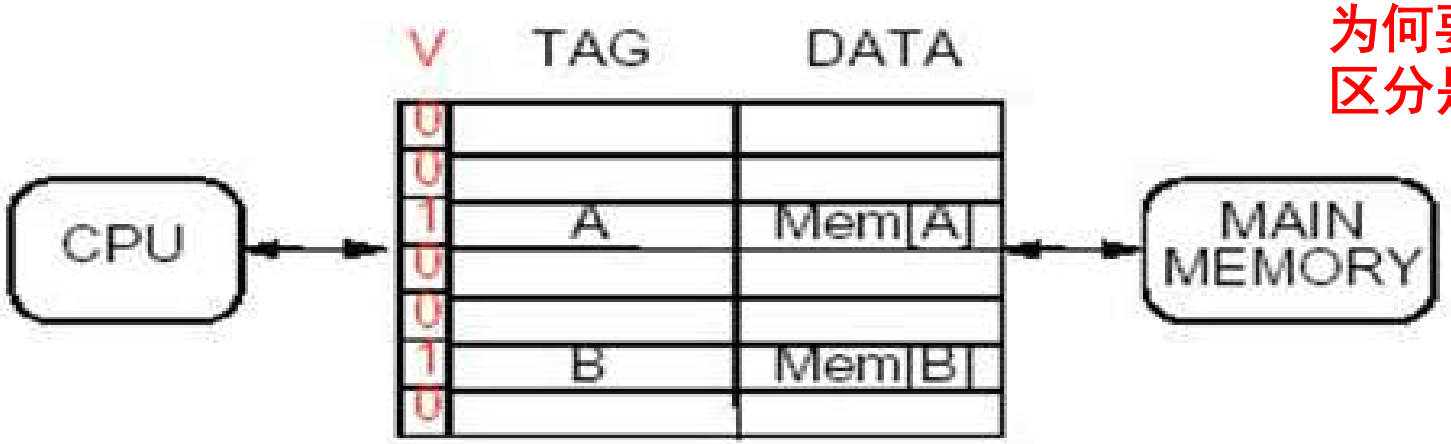
例：如何对0220CH单元进行访问？



0000 0010 0010 0000 1100B 是第1块群中的0001块（即第17块）中第12个单元！

有效位（Valid Bit）

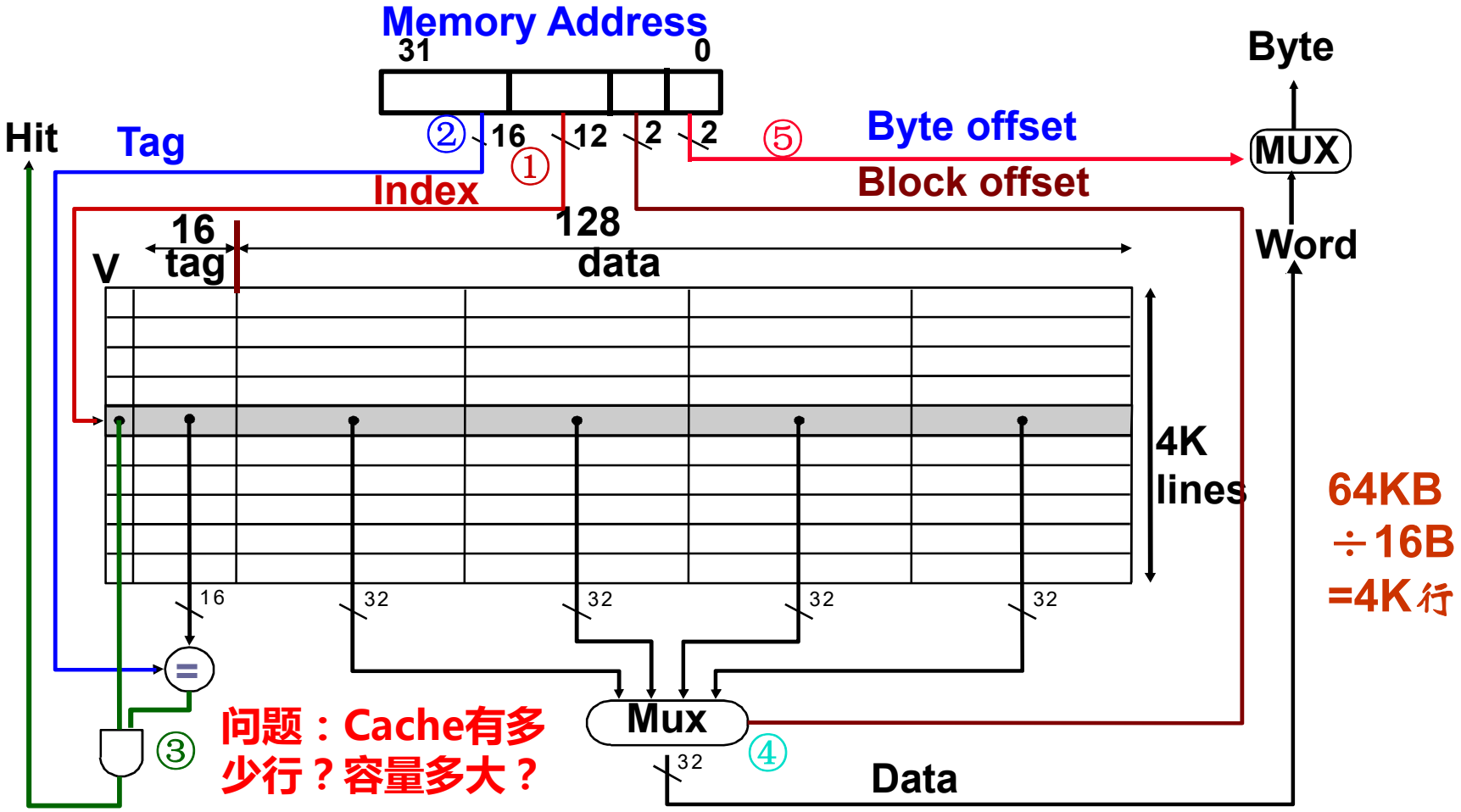
为何要用有效位来区分是否有效？



- V为有效位，为1表示信息有效，为0表示信息无效
- 开机或复位时，使所有行的有效位V=0
- 某行被替换后使其V=1
- 某行装入新块时 使其V=1
- 通过使V=0来冲刷Cache（例如：进程切换时，DMA传送时）
- 通常为操作系统设置“cache冲刷”指令，因此，cache对操作系统程序员不是透明的！

64 KB Direct Mapped Cache with 16B Blocks

主存和Cache之间直接映射，块大小为16B。Cache的数据区容量为64KB，主存地址为32位，按字节编址。要求：说明主存地址如何划分和访存过程。



64KB
÷16B
=4K行

容量 4Kx(1+16)+64Kx8=580Kbits=72.5KB,
数据占64KB / 72.5KB = 88.3%

如何计算Cache的容量？

Consider a cache with 64 Lines and a block size of 16 bytes.
What line number does byte address 1200 map to?

地址1200对应存放在第11行。因为： [1200/16=75] module 64 = 11

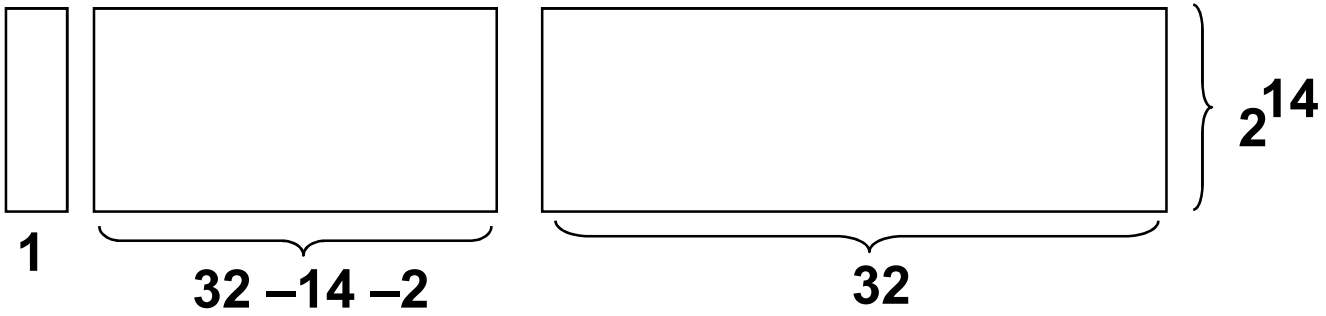
1200 = 1024+128+32+16 = 0…01 001011 0000 B

实现以下cache需要多少位容量？

Cache：直接映射 、16K行数据、块大小为1个字(4B)、32位主存地址

答：Cache的存储布局如下：

Cache共有16K x 4B= 64KB数据



所以，Cache的大小为：$2^{14} \times (32 + (32-14-2)+1) = 2^{14} \times 49 = 784$ Kbits

若块大小为4个字呢？ $2^{14} \times (4 \times 32 + (32-14-2-2)+1) = 2^{14} \times 143 = 2288$ Kbits

若块大小为2^m个字呢？$2^{14} \times (2^m \times 32 + (32-14-2-m)+1)$

BACK

全相联映射Cache组织示意图

假定数据在主存和Cache间的传送单位为512字。

Cache大小：2^{13}字=8K字=16行 x 512字/ 行

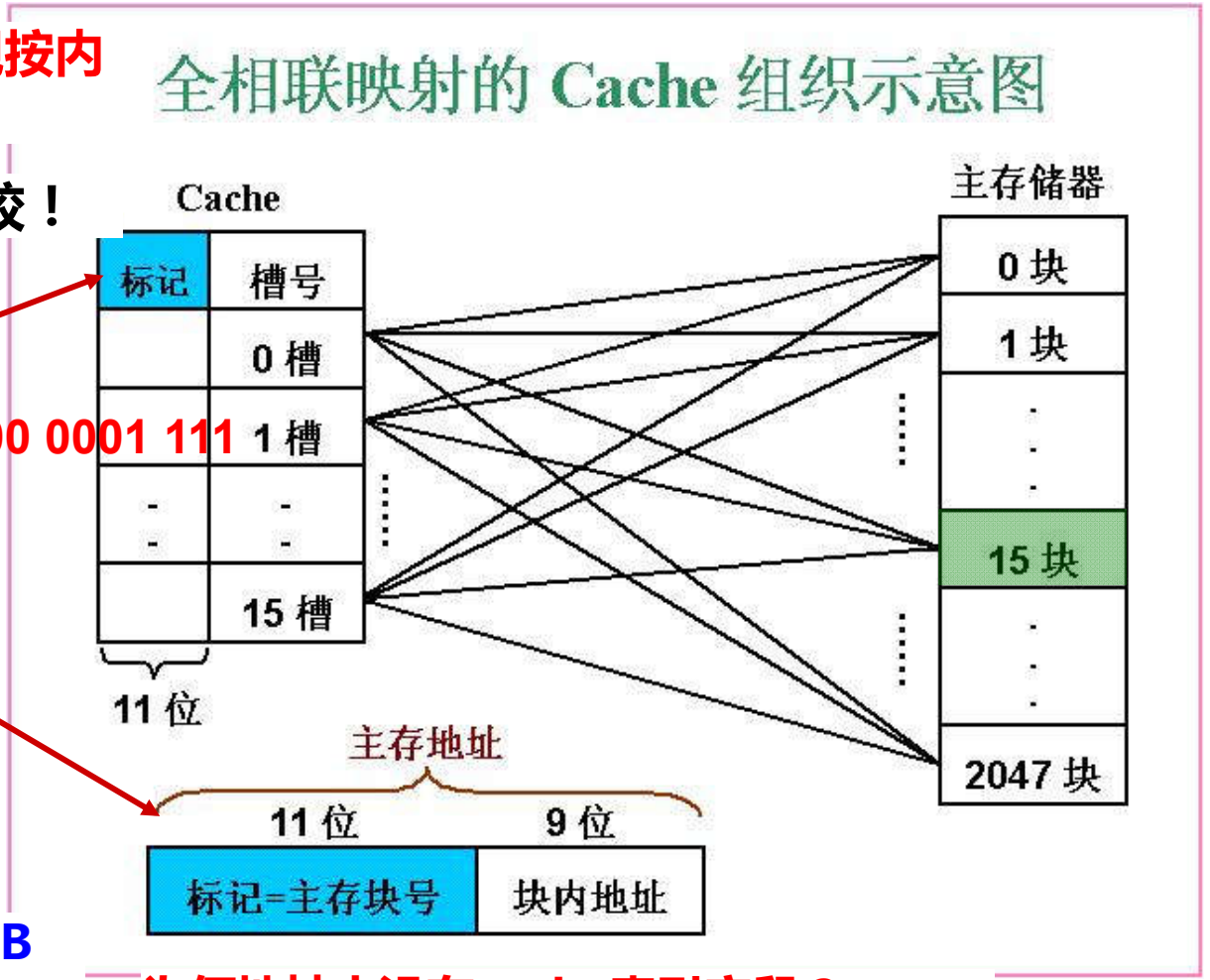
主存大小：2^{20}字=1024K字=2048块 x 512字/ 块

按内容访问，是相联存取方式！

每个主存块可装到Cache任一行中。

如何实现按内容访问？

直接比较！



Cache标记（tag）指出对应行取自哪个主存块

0000 0001 111

主存tag指出对应地址位于哪个主存块

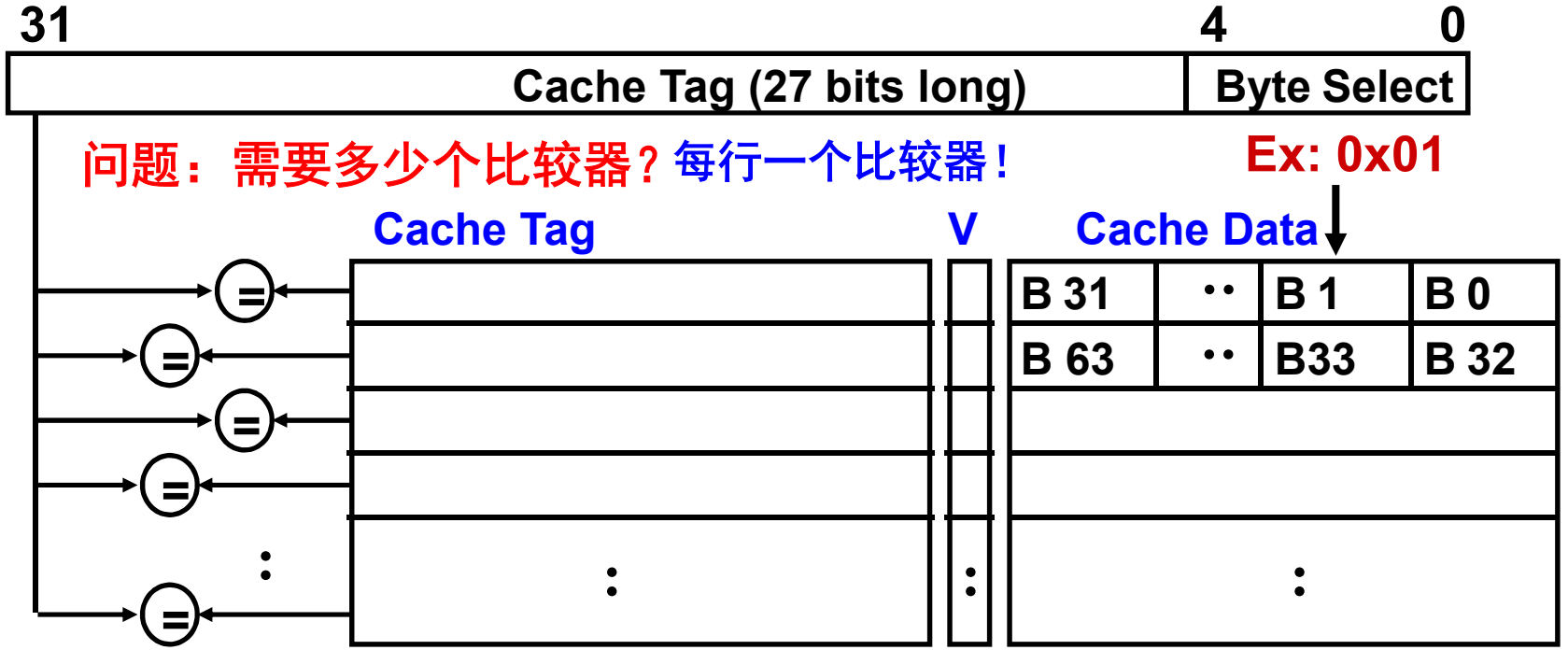
如何对01E0CH单元进行访问？

0000 0001 1110 0000 1100B
是第15块中的第12个单元！

为何地址中没有cache索引字段？
因为可映射到任意一个cache行中！

举例：Fully Associative

- Fully Associative Cache
  - 无需Cache索引，为什么？ 因为同时比较所有Cache项的标志
- By definition: Conflict Miss = 0
  - (没有冲突缺失，因为只要有空闲Cache块，都不会发生冲突)
- Example: 32bits memory address, 32 B blocks. 比较器位数多长？
  - we need N 27-bit comparators



组相联映射（Set Associative）

- 组相联映射结合直接映射和全相联映射的特点
- 将Cache所有行分组，把主存块映射到Cache固定组的任一行中。也即：组间模映射、组内全映射。映射关系为：

  Cache组号=主存块号 mod Cache组数

  举例：假定Cache划分为：8K字=8组x2行/组x512字/行

  4=100 mod 8

  (主存第100块应映射到Cache的第4组的任意行中。)

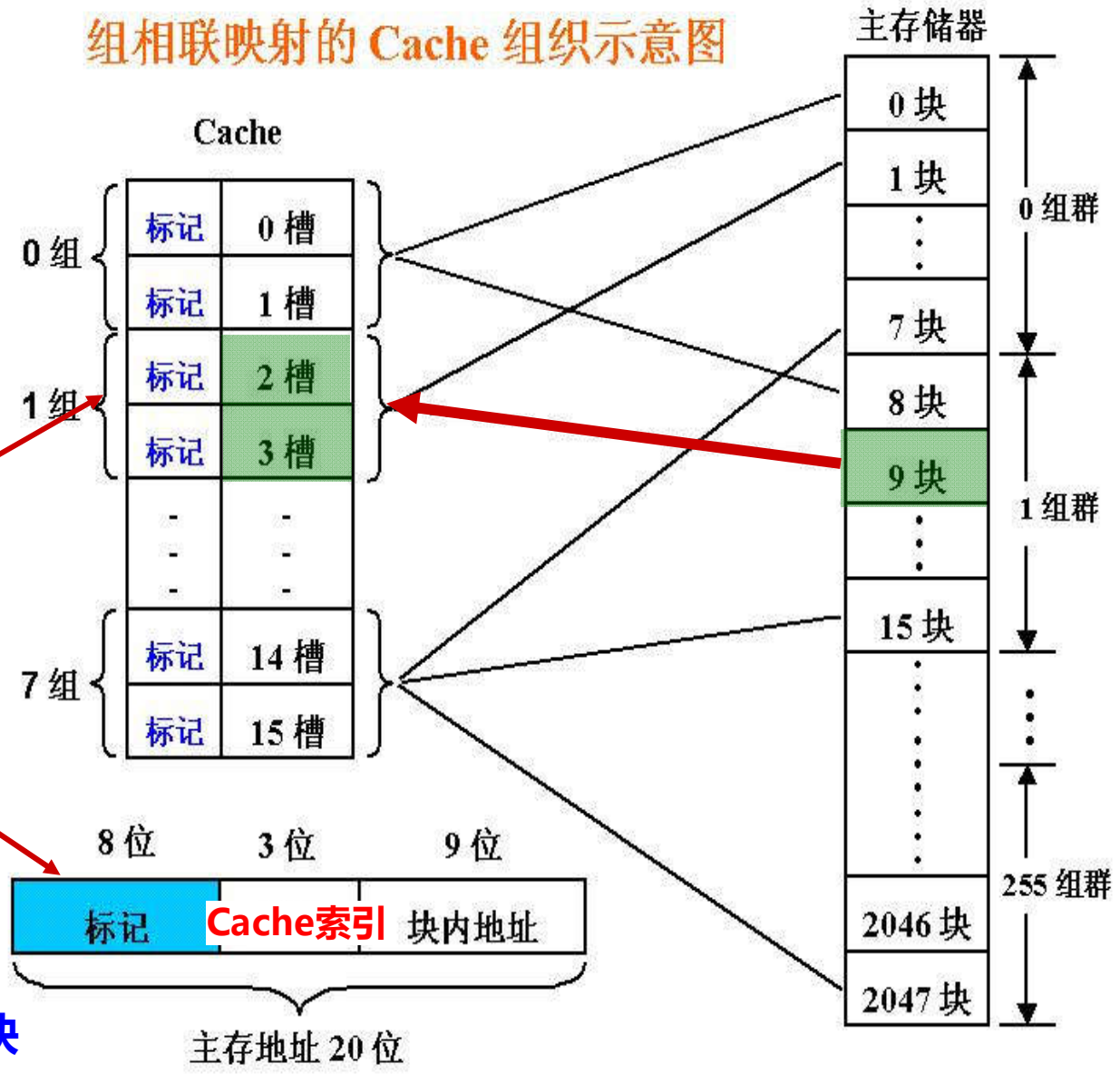
◆ 特点：

- 结合直接映射和全相联映射的优点。当Cache组数为1时，变为相联映射；当每组只有一个槽时，变为直接映射。
- 每组2或4行（称为2-路或4-路组相联）较常用。通常每组4行以上很少用。在较大容量的L2 Cahce和L3 Cahce中使用4-路以上。

假定数据在主存和Cache间的传送单位为512字。

Cache大小：2^{13}字=8K字=16行 x 512字/ 行

主存大小：2^{20}字=1024K字=2048块 x 512字/ 块



指出对应行取自哪个主存组群

指出对应地址位于哪个主存组群中

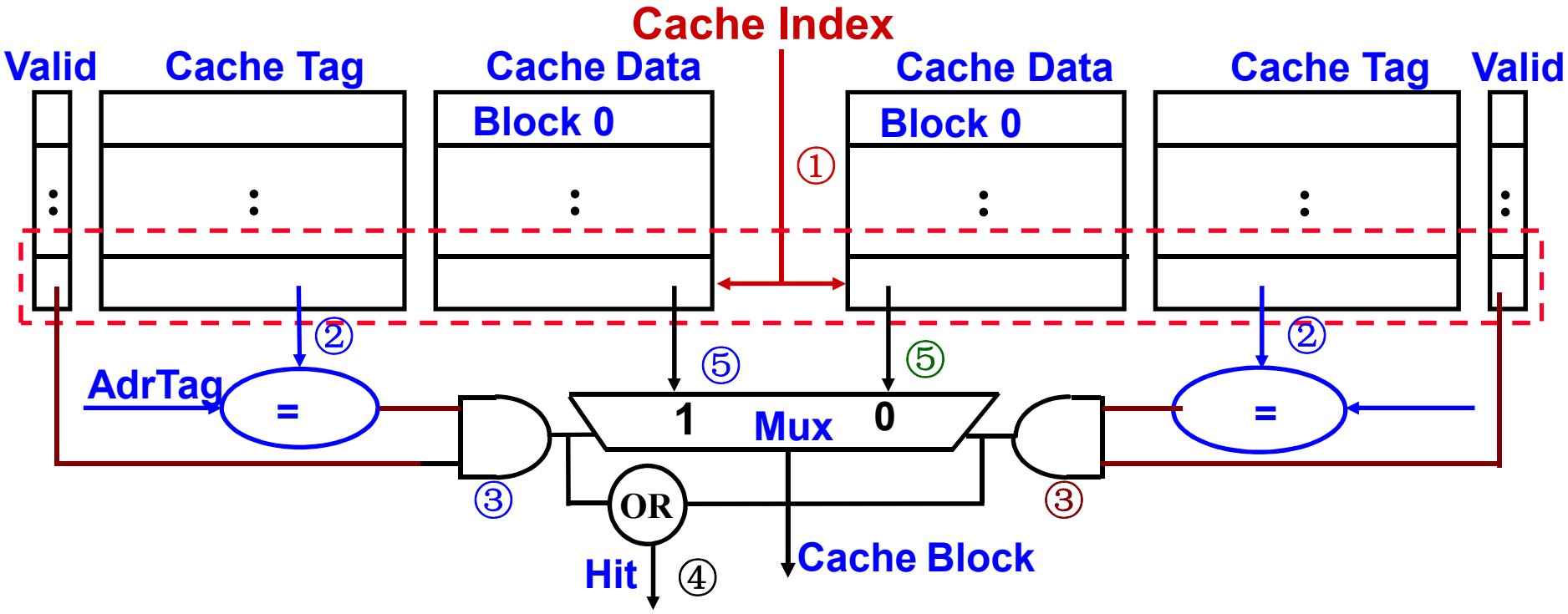
例：如何对0120CH单元进行访问？

0000 0001 0010 0000 1100B是第1组群中的001块（即第9块）中第12个单元。所以，映射到第一组中。

将主存地址标记和对应Cache组中每个Cache标记进行比较！

例1：A Two-way Set Associative Cache

- N-way set associative
  - N 个直接映射的行并行操作
- Example: Two-way set associative cache
  - Cache Index 选择其中的一个Cache行集合（共2行）
  - 对这个集合中的两个Cache行的Tag并行进行比较
  - 根据比较结果确定信息在哪个行，或不在Cache中



命中率、缺失率、缺失损失

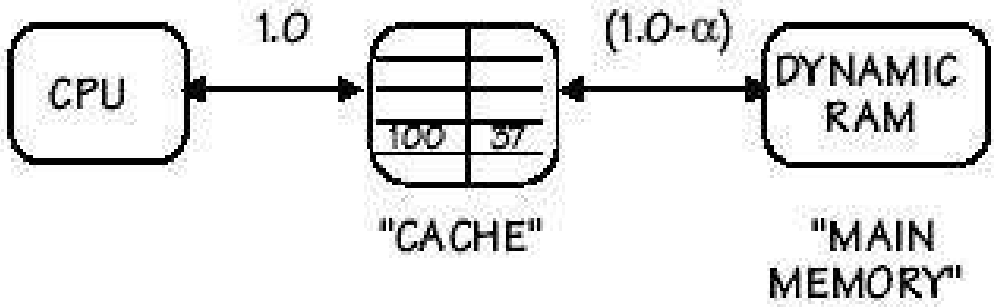
- Hit: 要访问的信息在Cache中
  - Hit Rate(命中率)：在Cache中的概率
  - Hit Time (命中时间) ：在Cache中的访问时间，包括：

    Time to determine hit/miss + Cache access time

    (即： 判断时间 + Cache访问)
- Miss: 要找的信息不在Cache中
  - Miss Rate (缺失率) = 1 - (Hit Rate)
  - Miss Penalty (缺失损失)：访问一个主存块所花时间
- Hit Time << Miss Penalty （Why?）

Average access time(平均访问时间)

Program-Transparent Memory Hierarchy



Cache contains TEMPORARY COPIES of selected main memory locations... eg. Mem[100] = 37

GOALS:

1) Improve the average access time

要提高平均访问速度，必须提高命中率！

α HIT RATIO: Fraction of refs found in CACHE.

(1-α) MISS RATIO: Remaining references.

$$t_{ave} = \alpha t_c + (1-\alpha)(t_c + t_m) = t_c + (1-\alpha)t_m$$

2) Transparency (compatibility, programming ease)

Cache对程序员来说是透明的，以方便编程！

Challenge:
To make the hit ratio as high as possible.

命中率到底应该有多大？

How high of a hit ratio?

Suppose we can easily build an on-chip static memory with a 4 nS access time, but the fastest dynamic memories that we can buy for main memory have an average access time of 40 nS. How high of a hit rate do we need to sustain an average access time of 5 nS?

$$\alpha = 1 - \frac{t_{ave} - t_c}{t_m} = 1 - \frac{5-4}{40} = 97.5\%$$

WOW, a cache really needs to be good?

看看命中率对平均访问时间的影响

设H是命中率，则平均访问时间$T = HT_C + (1 - H)(T_C + T_M)$

$= T_C + (1 - H)T_M$

例1. 若H=0.85, Tc=1ns, T_M=20ns，则T为多少？

答： T = 4ns

例2. 若命中率H提高到0.95，则结果又如何？

答： T = 2ns

例3. 若命中率为0.99呢？

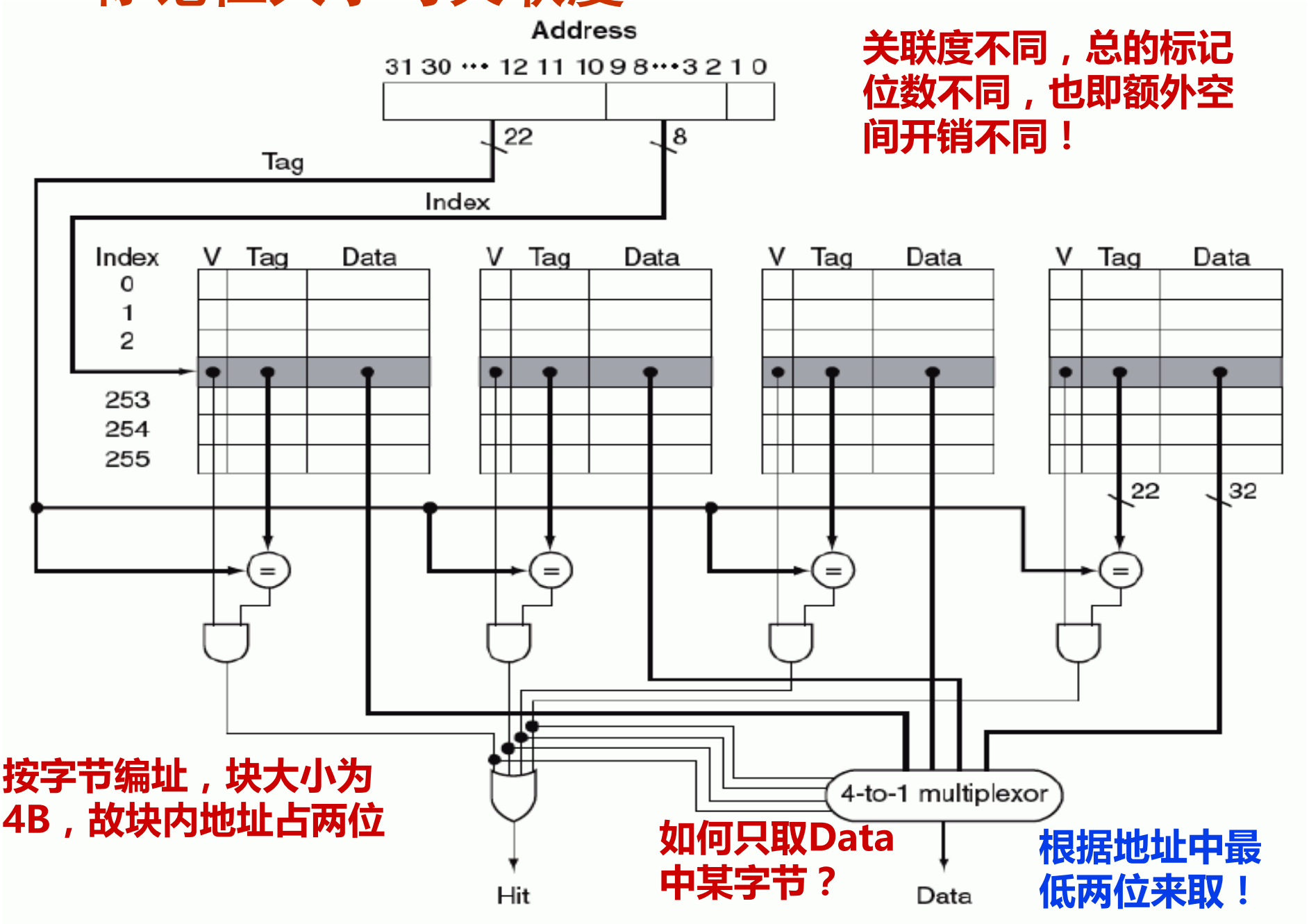
答: T = 1.2ns

访存速度与命中率的关系非常大！

高速缓存的缺失率和关联度

- 三种映射方式
  - 直接映射：唯一映射（只有一个可能的位置）
  - 全相联映射：任意映射（每个位置都可能）
  - N-路组相联映射：N-路映射（有N个可能的位置）
- 什么叫关联度？
  - 一个主存块映射到Cache中时，可能存放的位置个数
    - 直接映射关联度？　关联度最低，为1
    - 全相联映射关联度？　关联度最高，为Cache行数
    - N-路组相联映射关联度？　关联度居中，为N
- 关联度和miss rate有什么关系呢？和命中时间的关系呢？
  - 直观上，你的结论是什么？（ Cache大小和块大小一定时 ）
    - 缺失率：直接映射最高，全相联映射最低
    - 命中时间：直接映射最小，全相联映射最大
  - 用例子来说明

标记位大小与关联度



关联度不同，总的标记位数不同，也即额外空间开销不同！

按字节编址，块大小为4B，故块内地址占两位

如何只取Data中某字节？

根据地址中最低两位来取！

标记位大小与关联度

问题：若主存地址32位，块大小为16字节，Cache总大小为4K行，
问：标记位的总位数是多少？

直接映射方式下：相当于每组1行，共4K组，标志占32-4-12=16位
总位数占 4Kx16=64K位

关联度增加到2倍(2-way)：每组2行，共2K组，标志占32-4-11=17位
总位数占4Kx17=68K位

关联度增加到4倍(4-way)：每组4行，共1K组，标志占32-4-10=18位
总位数占4Kx18=72K位

全相联时：整个为1组，每组4K行，标志占32-4=28位
总位数占4Kx28=112K位

关联度越高，总的标记位数越多，额外空间开销越大！