

存储器层次结构

Part1：概述

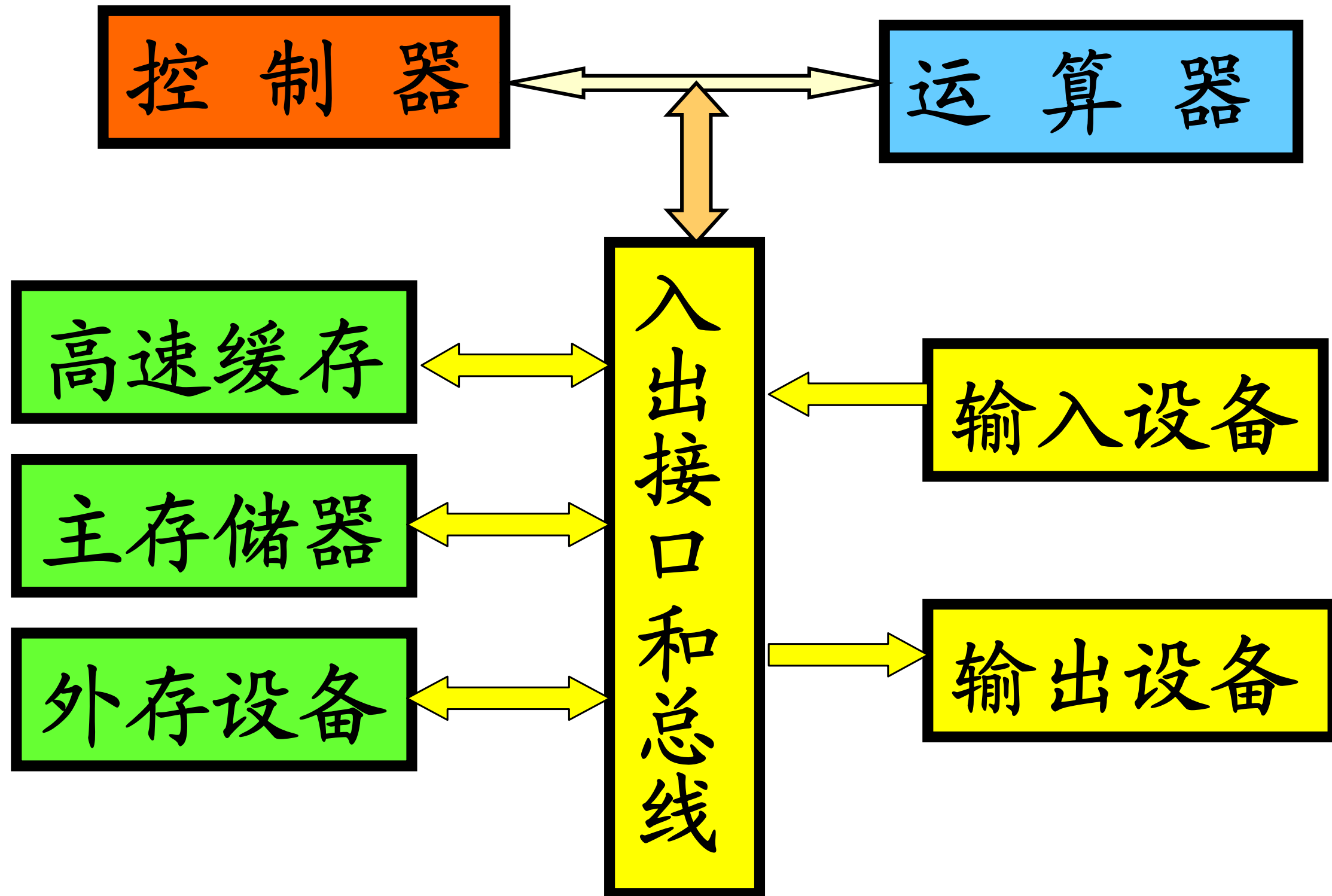
存储器层次结构

- 层次存储器系统概述及动态存储器
- 静态存储器及高速缓冲存储器
- 高速缓冲存储器的组成与运行原理
- 虚拟存储器的运行原理
- 磁表面存储设备的存储原理与组成
- 光盘的存储原理与组成

提纲

- 层次存储器系统概述
 - 存储器系统功能
 - 存储器系统的设计目标
 - 需要解决的问题
 - 解决方案
 - 静态和动态存储器

计算机硬件系统



存储器的作用

- 计算机中用来存放**程序和数据**的部件,是Von Neumann结构计算机的重要组成部分
- 存储程序使计算机走向通用
- 程序和数据的特点
 - 源程序、汇编程序、机器语言程序
 - 各种类型的数据
 - 共同点:二进制数据

对存储介质的基本要求

- 能够有两个稳定状态来表示二进制中的“0”和“1”
- 容易识别
- 两个状态能方便地进行转换
- 几种常用的存储介质
 - 磁颗粒、电平、电容、光

存储器分类（按工作性质、存取方式）

■ 随机存取存储器Random Access Memory (RAM)

- 每个单元读写时间一样，且与各单元所在位置无关。如：内存。
- （注：原意主要强调地址译码时间相同。现在的DRAM芯片采用行缓冲，因而可能因为位置不同而使访问时间有所差别。）

■ 顺序存取存储器Sequential Access Memory (SAM)

- 数据按顺序从存储载体的始端读出或写入，因而存取时间的长短与信息所在位置有关。例如：磁带。

■ 直接存取存储器Direct Access Memory(DAM)

- 直接定位到要读写的数据块，在读写某个数据块时按顺序进行。例如：磁盘。

■ 相联存储器Associate Memory/Content Addressed Memory (CAM)

- 按内容检索到存储位置进行读写。例如：快表。

存储器分类

■ 按存储介质

- 读写存储器 (Read / Write Memory): 可读可写
- 只读存储器 (Read Only Memory): 只能读不能写

■ 按信息的可更改性分类

- 半导体存储器: 双极型, 静态MOS型, 动态MOS型
- 磁表面存储器: 磁盘 (Disk)、磁带 (Tape)
- 光存储器: CD, CD-ROM, DVD

存储器分类

■ 按断电后信息的可保存性分类

■ 非易失（不挥发）性存储器(Nonvolatile Memory)

- 信息可一直保留，不需电源维持。
- （如：ROM、磁表面存储器、光存储器等）

■ 易失（挥发）性存储器(Volatile Memory)

- 电源关闭时信息自动丢失。（如：RAM、Cache等）

存储器分类（按功能/容量/速度/所在位置）

■ 寄存器(Register)

- 封装在CPU内，用于存放当前正在执行的指令和使用的数据
- 用触发器实现，速度快，容量小（几十个）

■ 高速缓存(Cache)

- 位于CPU内部或附近，用来存放当前要执行的局部程序段和数据
- 用SRAM实现，速度可与CPU匹配，容量小（几MB）

■ 内存存储器MM（主存储器Main (Primary) Memory）

- 位于CPU之外，用来存放已被启动的程序及所用的数据
- 用DRAM实现，速度较快，容量较大（几GB）

■ 外存储器AM (辅助存储器Auxiliary / Secondary Storage)

- 位于主机之外，用来存放暂不运行的程序、数据或存档文件
- 用磁表面或光存储器实现，容量大而速度慢

现代计算机存储器

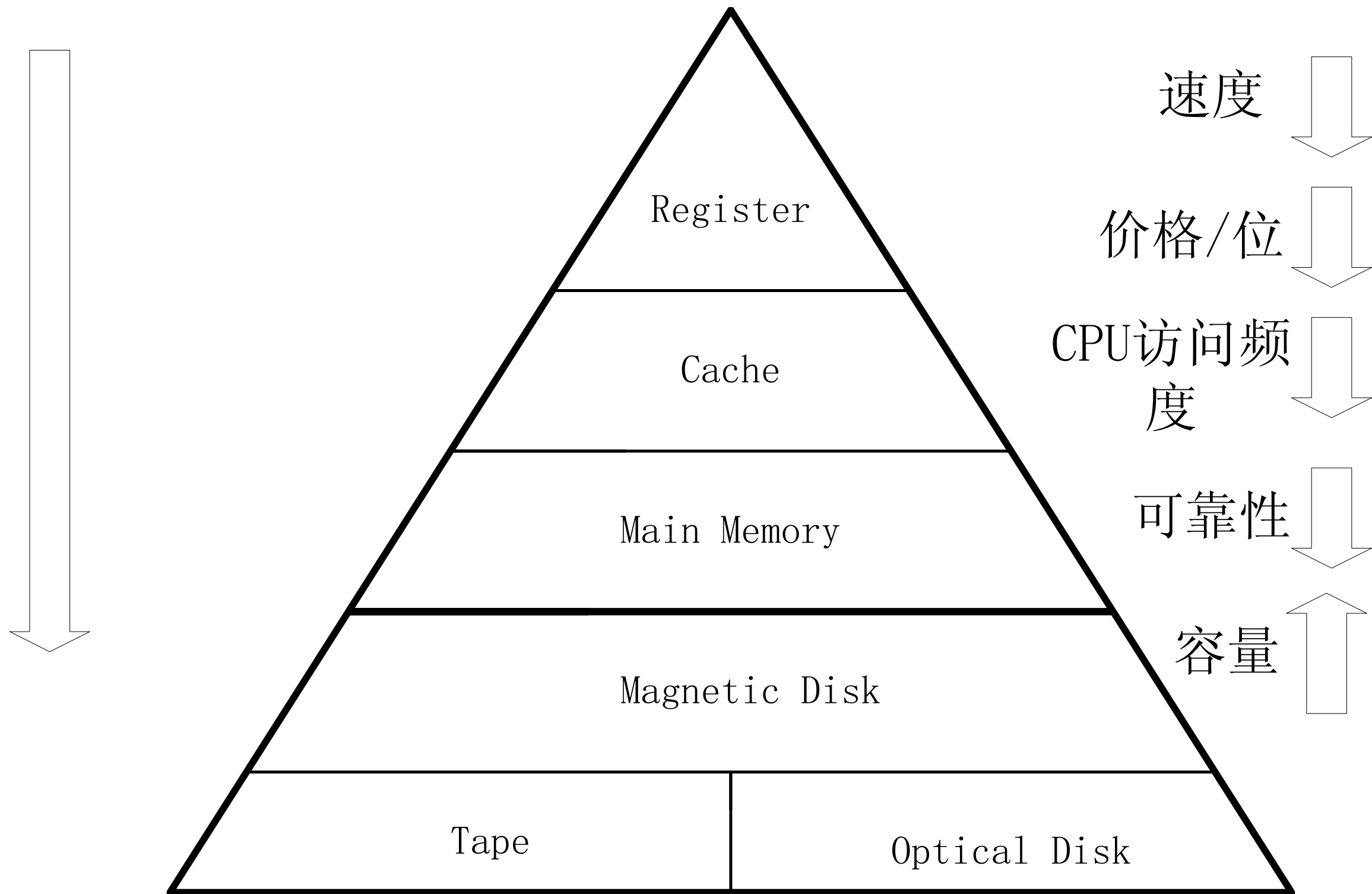
■ 主存储器

- 寄存器 Register
- 高速缓存 Cache
- 主存储器 Main Memory

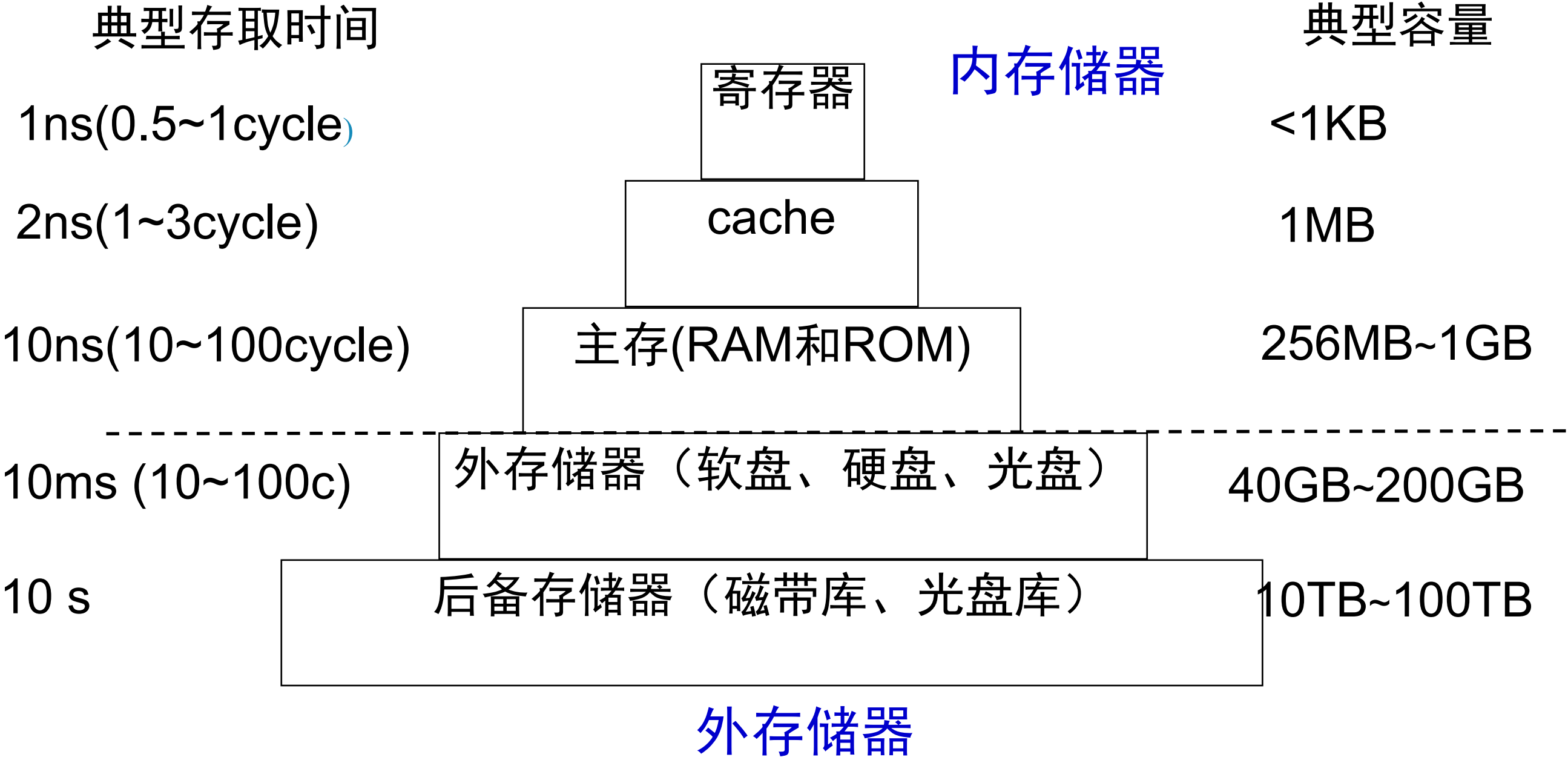
■ 辅助存储器

- 磁盘 Disk
- 磁带 Tape
- 光盘 Compact Disc

不同类型存储器比较



现代计算机的层次存储系统

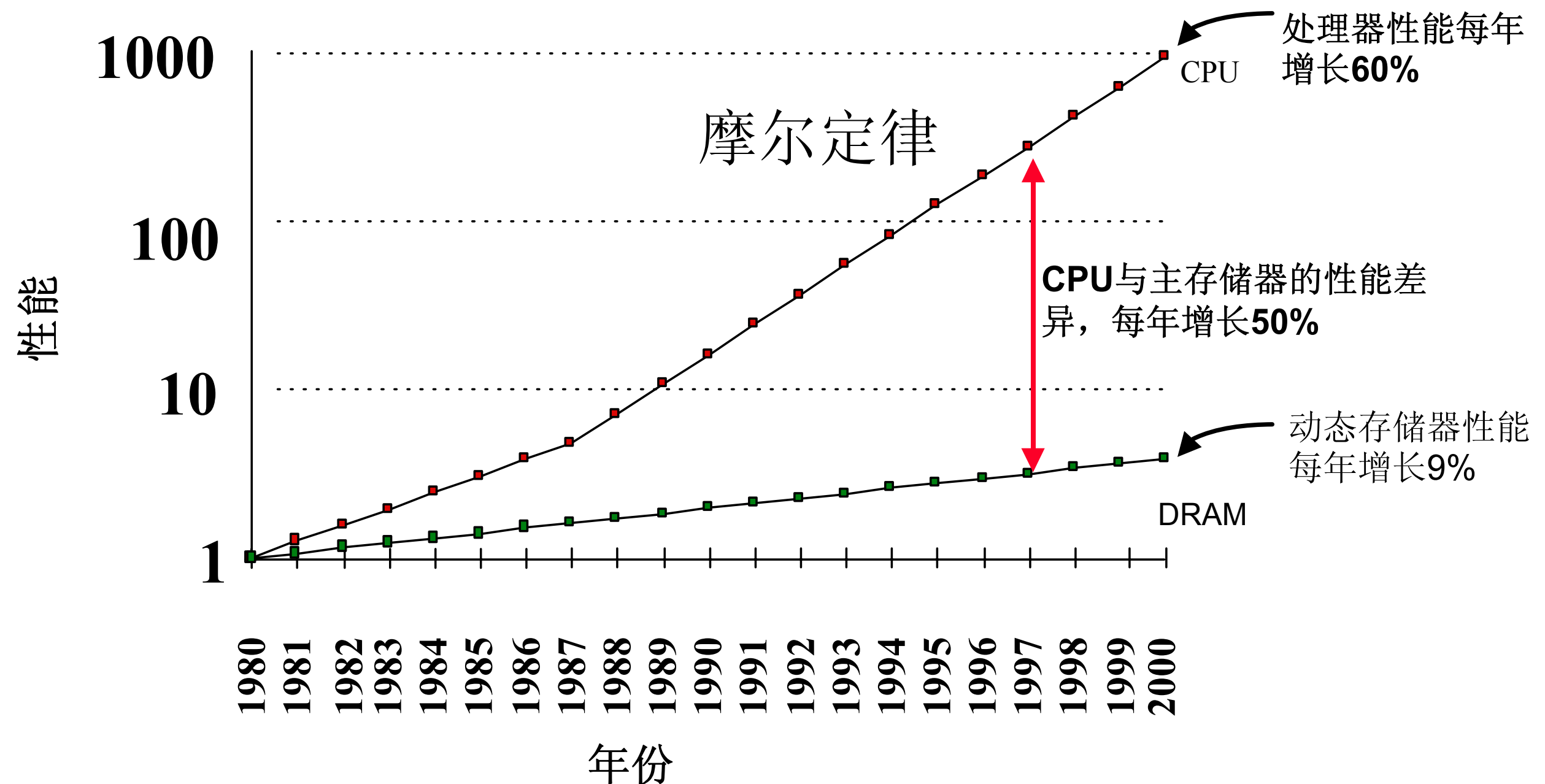


存储器设计目标

- 尽可能快的存取速度
 - 应能基本满足CPU对数据的访问要求
- 尽可能大的存储空间
 - 可以满足程序对存储空间的要求
- 尽可能低的单位成本(价格/位)
 - 用户能够承受的范围内
- 较高的可靠性

Moore定律

- 每一美元所能买到的电脑性能，将每隔18个月翻两倍以上



微电子技术的发展趋势

容量

逻辑电路: 2倍/ 3 年

DRAM: 4倍/ 3 年

磁盘: 4倍/ 3 年

速度

2倍/ 3 年

2倍/ 10 年

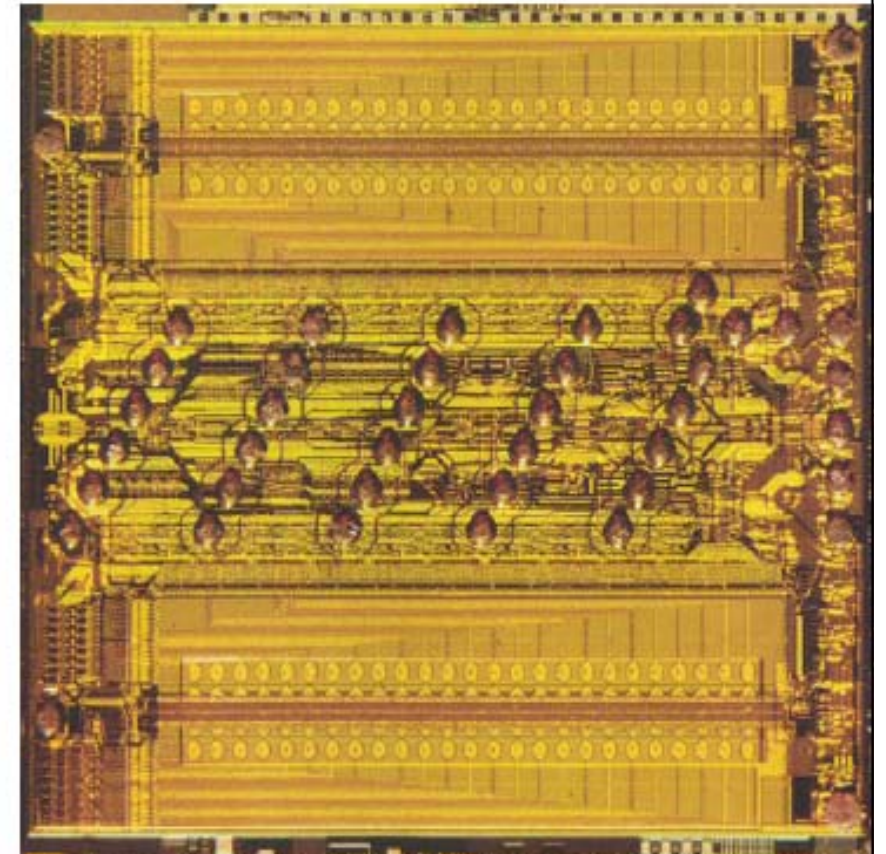
2倍/ 10 年

DRAM		
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1! **2:1!**

DRAM的发展

年代	容量	价格 (\$/MB)	总访问时间 (新行/列)	列访问时间 (现访问行)
1980	64 Kbit	1500	250 ns	150 ns
1983	256 Kbit	500	185 ns	100 ns
1985	1 Mbit	200	135 ns	40 ns
1989	4 Mbit	50	110 ns	40 ns
1992	16 Mbit	15	90 ns	30 ns
1996	64 Mbit	10	60 ns	20 ns



存储器对性能的影响

- 假定某台计算机的处理器工作在:
 - 主频 = 1GHz (机器周期为1 ns)
 - CPI = 1.1
 - 50% 算逻指令, 30% 存取指令, 20% 控制指令
- 再假定其中10% 的存取指令会缺失,需要50个周期的延迟。(当前主存的典型值)
- $CPI = \text{理想CPI} + \text{每条指令的平均延迟}$
 $= 1.1 + (0.30 \times 0.10 \times 50)$
 $= 1.1 \text{ cycle} + 1.5 \text{ cycle} = 2.6 \text{ CPI!}$

存储器对性能的影响

- 假定某台计算机的处理器工作在:
 - 主频 = 1GHz (机器周期为1 ns)
 - CPI = 1.1
 - 50% 算逻指令, 30% 存取指令, 20% 控制指令
- 再假定其中10% 的存取指令会缺失,需要50个周期的延迟。(当前主存的典型值)
- $CPI = \text{理想CPI} + \text{每条指令的平均延迟}$
 $= 1.1 + (0.30 \times 0.10 \times 50)$
 $= 1.1 \text{ cycle} + 1.5 \text{ cycle} = 2.6 \text{ CPI!}$
- 也就是说,处理器58 %的时间花在等待存储器给出数据上面!
- 每1%的指令缺失将给CPI附加0.5个周期!

存储器设计目标

■ 目标

- 大容量、高速度、低成本、高可靠性

■ 目前现实

- 大容量存储器速度慢
- 快速存储器容量小

■ 如何实现我们的目标呢？

存储器设计目标

■ 目标

- 大容量、高速度、低成本、高可靠性

■ 目前现实

- 大容量存储器速度慢
- 快速存储器容量小

■ 如何实现我们的目标呢？

- 层次存储器系统
- 采用并行技术

层次存储系统

■ 高速度

- 静态存储器速度高
- 设置较小容量的高速缓冲存储器

■ 大容量

- 动态存储器价格适中,速度适中
- 可作为主存储器

■ 低成本

- 磁盘存储器价格低廉
- 作为辅助存储器,暂存CPU访问频率不高的数据 和程序
- 作为虚拟存储器的载体

程序运行的局部性原理

程序运行时的局部性原理表现在：

在一小段时间内，最近被访问过的程序和数据很可能再次被访问

在空间上 这些被访问的程序和数据往往集中在一小片存储区

在访问顺序上，指令顺序执行比转移执行的可能性大(大约 5:1)

程序运行的局部性原理

程序运行时的局部性原理表现在：

在一小段时间内，最近被访问过的程序和数据很可能再次被访问

在空间上 这些被访问的程序和数据
往往集中在一小片存储区

在访问顺序上， 指令顺序执行比转移执行的
可能性大 (大约 5:1)

合理地把程序和数据分配在不同存储介质中

程序运行的局部性原理

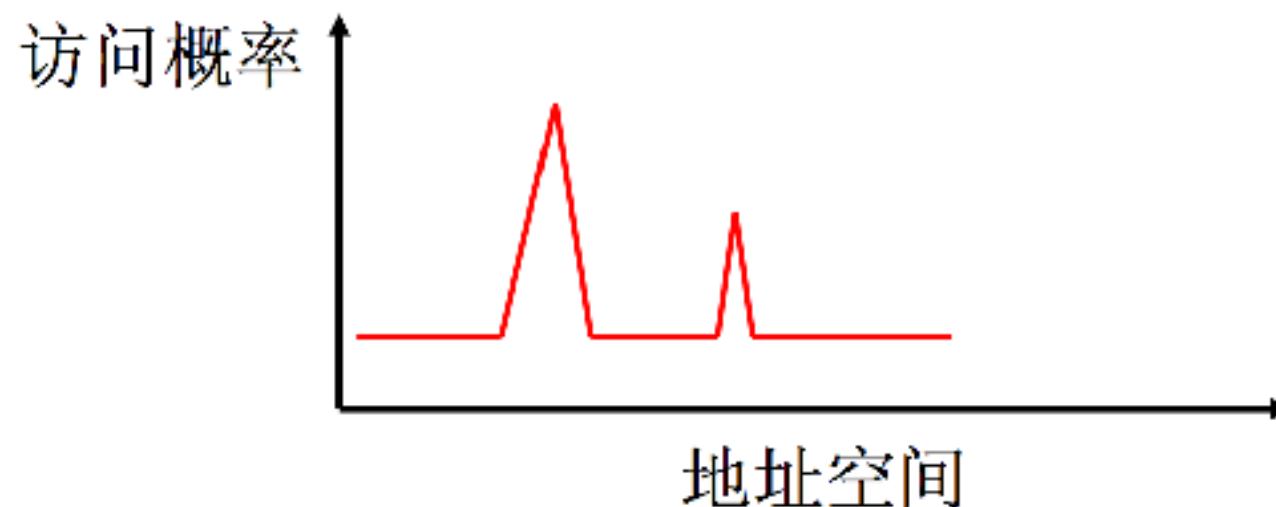
```
1 int sumvec(int v[N])
2 {
3     int i, sum = 0;
4
5     for (i = 0; i < N; i++)
6         sum += v[i];
7     return sum;
8 }
```

(a)

Address	0	4	8	12	16	20	24	28
Contents	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Access order	1	2	3	4	5	6	7	8

(b)

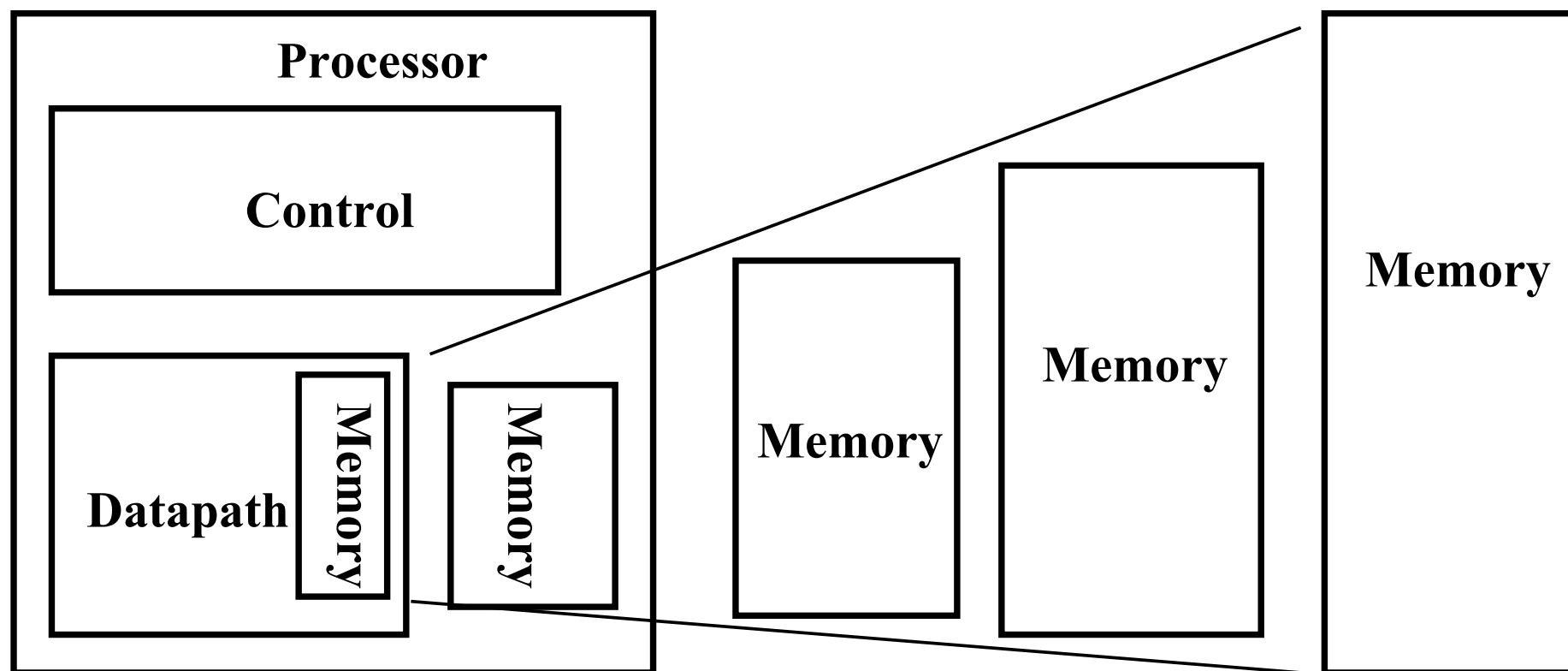
- 循环中的 `sum` 有良好的时间局部性。因为在for循环结束之前，每次执行循环体都有对 `sum` 的访问。
- 对于循环体中的 `v` 则有良好的空间局部性。向量 `v`在实际中多数情况也按顺序存储，每次访问 `v[i]`总是在 `v[i-1]` 的下一个位置。



现代计算机的层次存储系统

■ 利用程序的局部性原理:

- 以最低廉的价格提供尽可能大的存储空间
- 以最快速的技术实现高速存储访问



Speed: Fastest
Size: Smallest
Cost: Highest

Slowest
Biggest
Lowest

层次之间应该满足的原则

■(1).一致性原则:

- 处在不同层次存储器中的同一个信息应保持相同的值。

■(2).包含性原则:

- 处在内层的信息一定被包含在其外层的存储器中, 反之则不成立
- 即内层存储器中的全部信息, 是其相邻外层存储器中一部分信息的复制品。

高速缓冲存储器Cache

■ 定义

- 设置于主存和CPU之间的存储器,用高速的静态存储器实现,缓存了CPU频繁访问的信息。

■ 特点

- 高速:与CPU的运行速度基本匹配
- 透明:完全硬件管理,对程序员透明

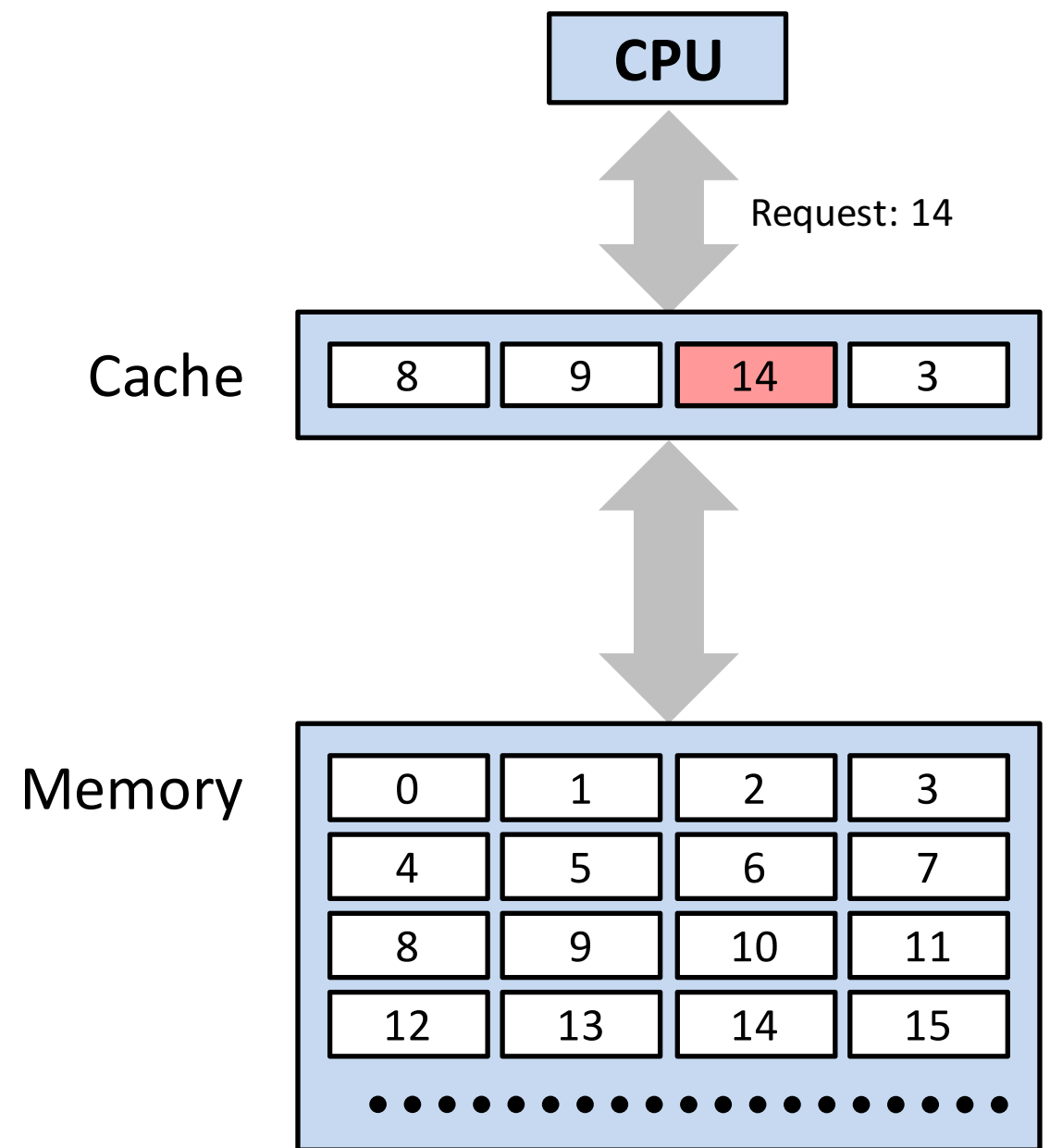
高速缓冲存储器Cache

■功能

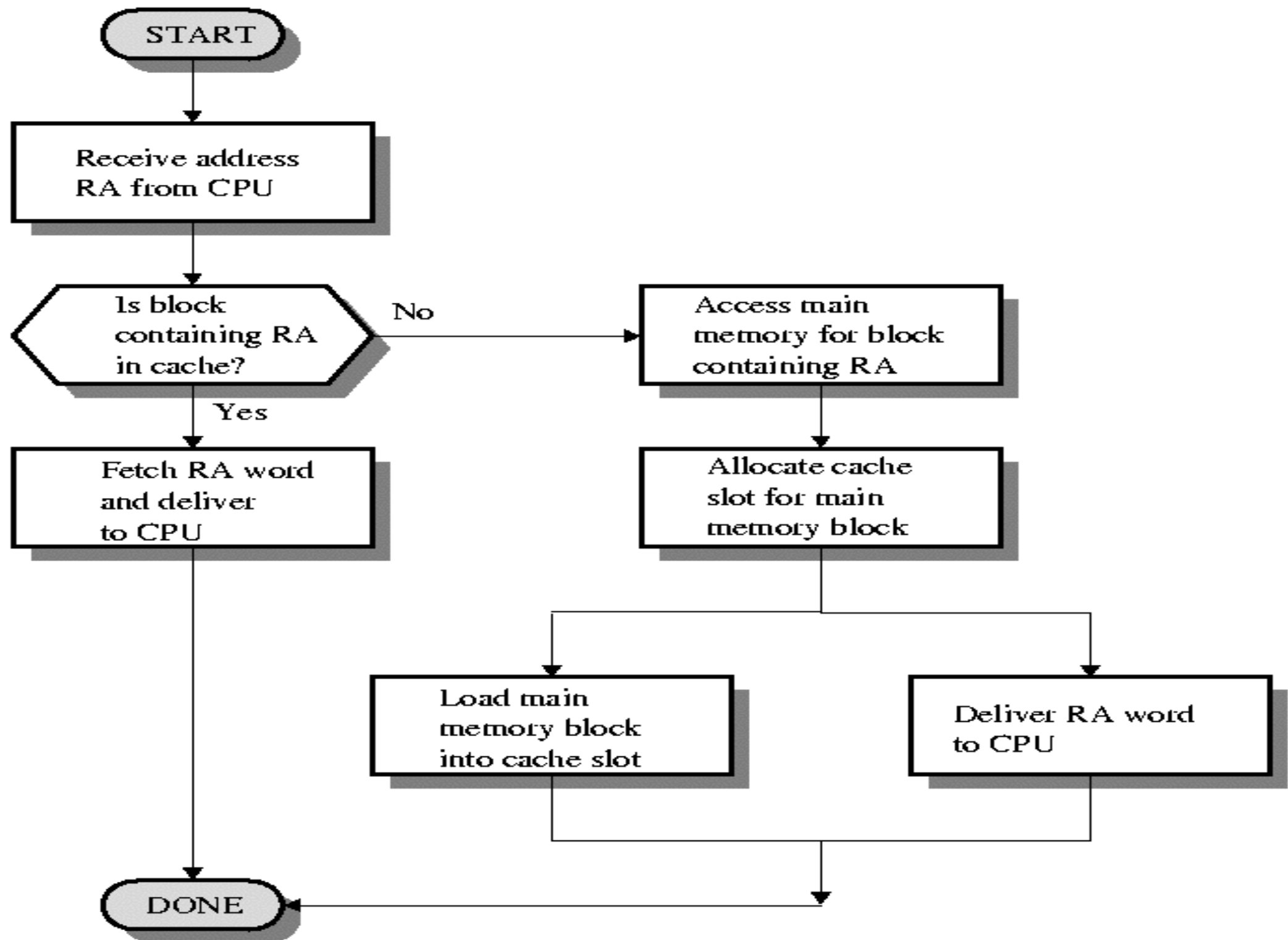
- 解决CPU和主存之间的速度不匹配问题
- 一般采用高速的SRAM构成
- CPU和主存之间的速度差别很大采用两级或多级Cache系统
- 早期的一级Cache在CPU内，二级在主板上
- 现在的CPU内带L1 Cache、L2 Cache、L3 Cache
- (Intel 酷睿i7 930，一级缓存：256KB；二级缓存：1MB；三级缓存8MB)

Cache的基本工作原理

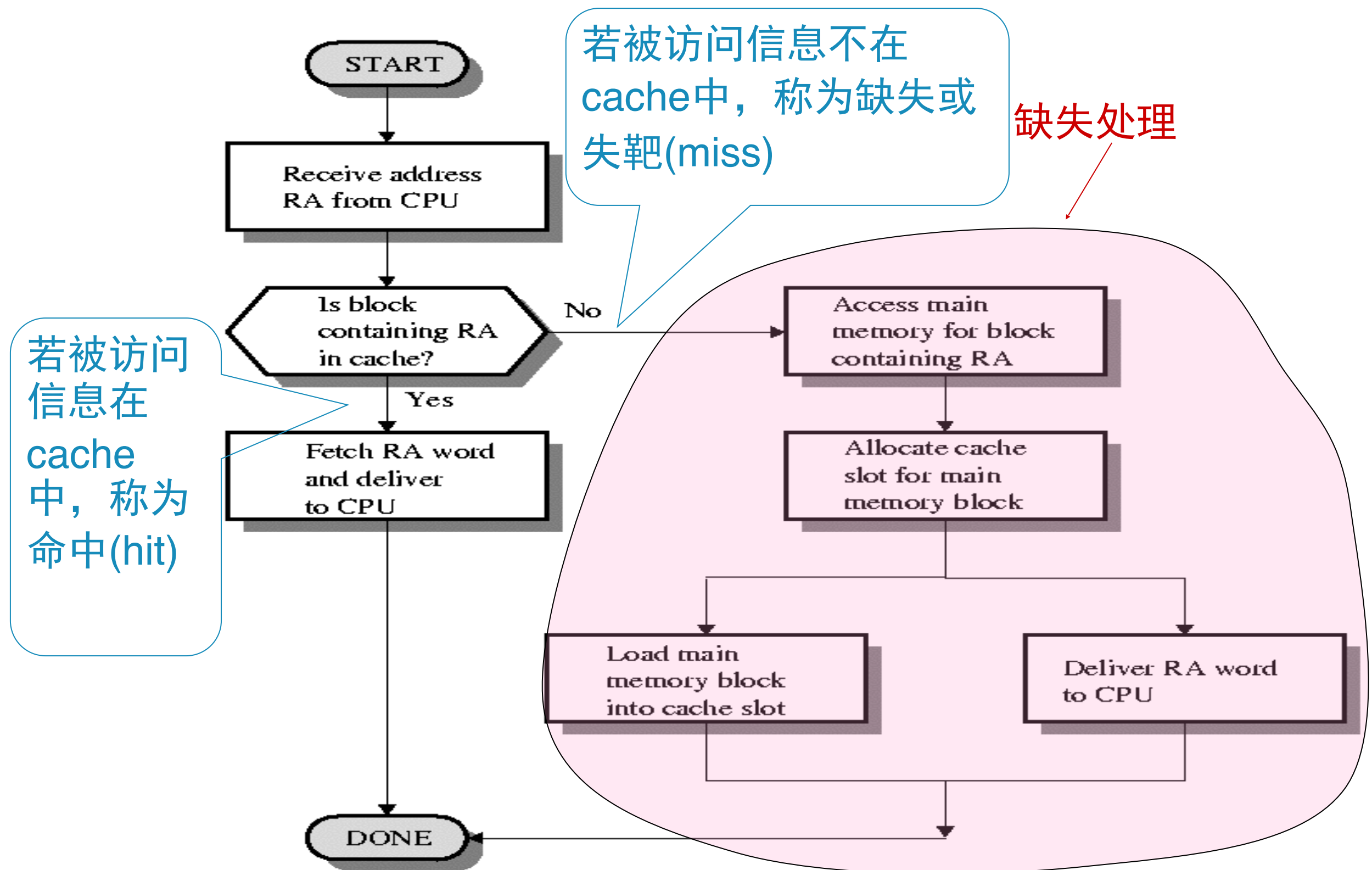
- CPU与Cache之间的数据传送是
以字为单位
- 主存与Cache之间的数据传送是
以块为单位
- CPU读主存时，便把地址同时送给Cache和主存，Cache控制逻辑依据地址判断此字是否在Cache中，若在此字立即传送给CPU，否则，则用主存读周期把此字从主存读出送到CPU，与此同时，把含有这个字的整个数据块从主存读出送到cache中。



Cache的基本操作

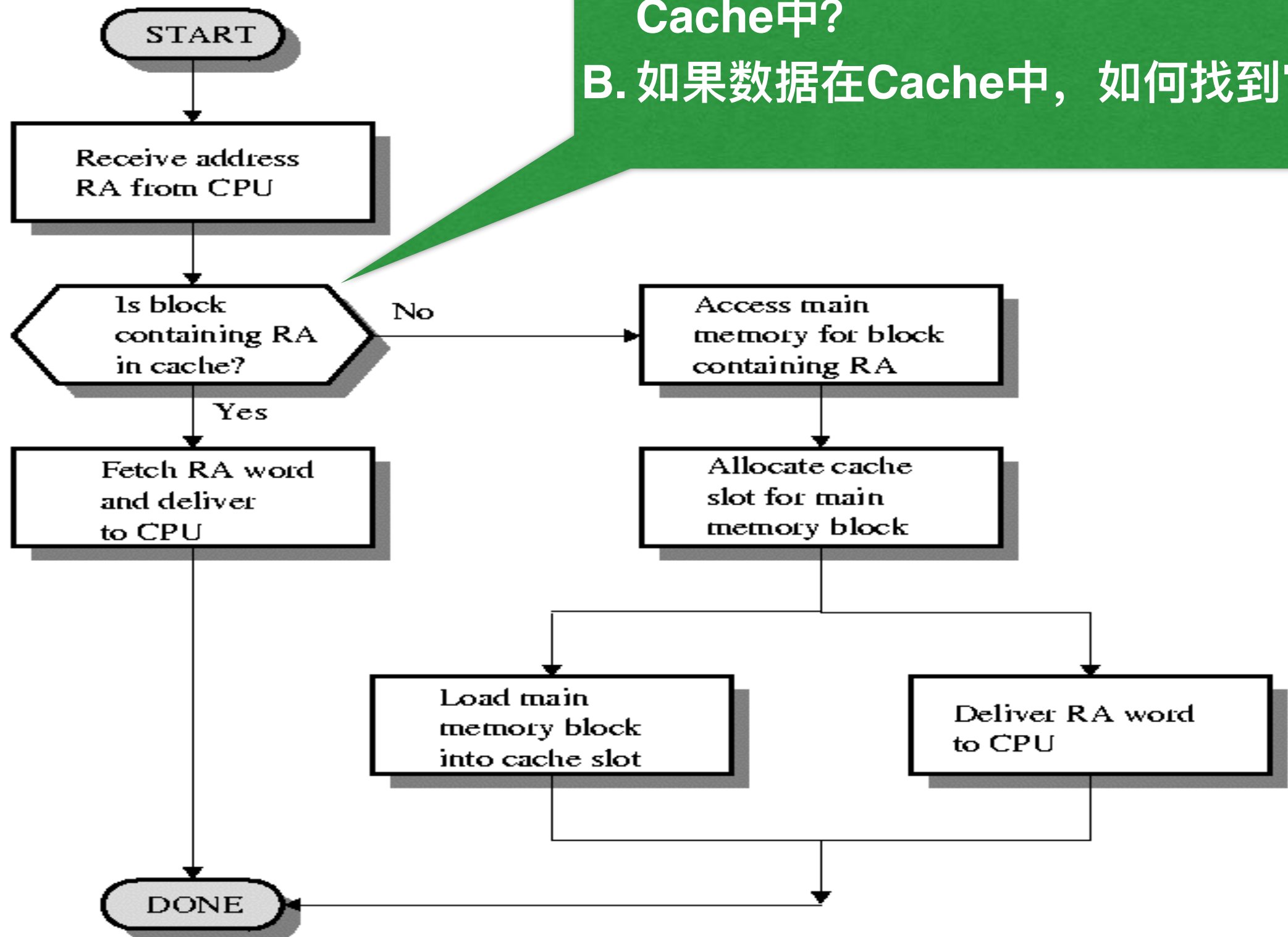


Cache的基本操作



关键问题

- A. 如何根据主存地址判断数据项是否在Cache中?
- B. 如果数据在Cache中, 如何找到它?



使用Cache需要解决的问题

■ 地址之间的映射关系:

- 如何从主存地址得到Cache地址?

■ 数据之间一致性:

- Cache中的内容是否已经是主存对应地址的内容?

■ Cache内容装入和替换策略

- 如何提高Cache的命中率?

基本参数

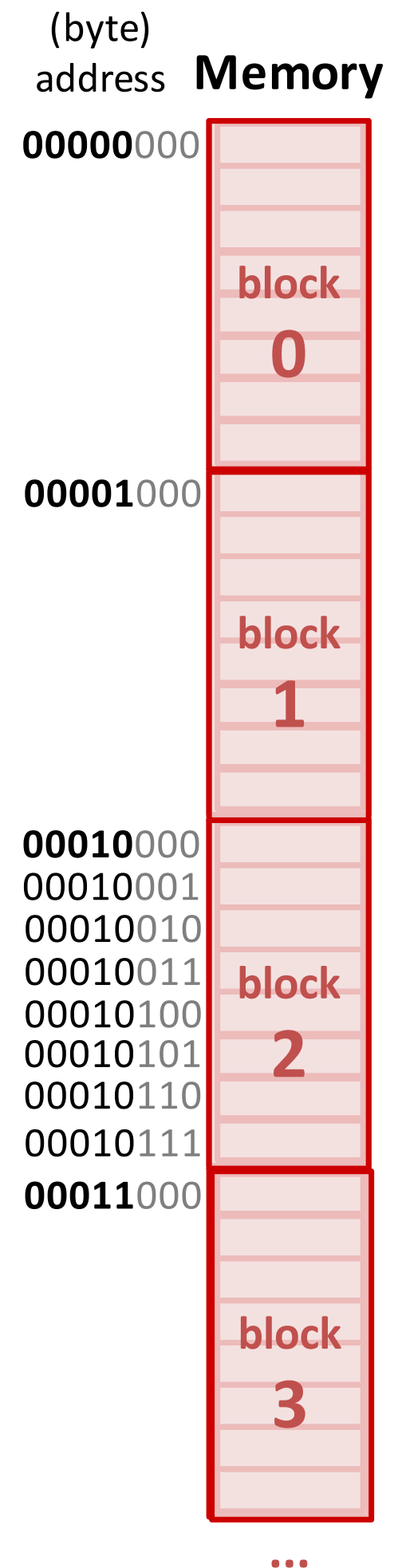
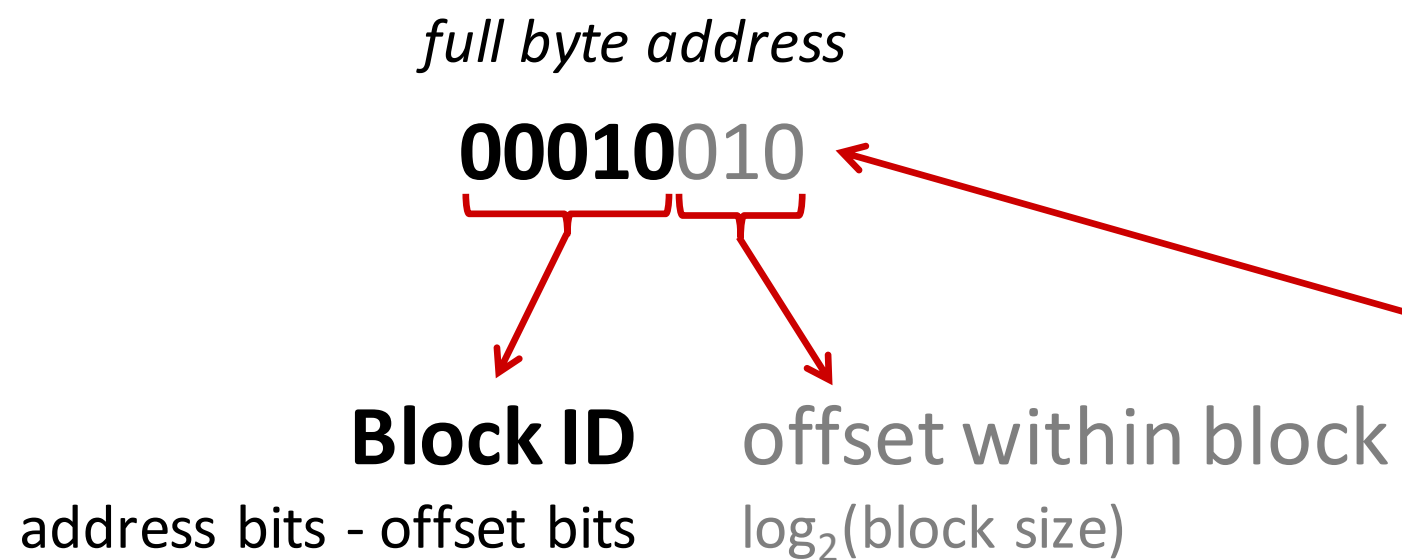
- 块(Block):数据交换的最小单位
- 命中(Hit):在较高层次中发现要访问的内容
 - 命中率(Hit Rate):命中次数/访问次数
 - 命中时间:访问在较高层次中数据的时间
- 失效(Miss):需要在较低层次中访问块
 - 失效率(Miss Rate):1-命中率
 - 失效损失(Miss Penalty):替换较高层次数据块的时间+将该块交付给处理器的时间
- 命中时间 \ll 失效损失

参数的典型值

- 块大小:4~128Byte
- 命中时间:1~4周期
- 失效损失:
 - 访问时间:6~10个周期
 - 传输时间:2~22个周期
- 命中率:80%~99%
- Cache容量:1KB~256KB

数据块

Example: block size = 8

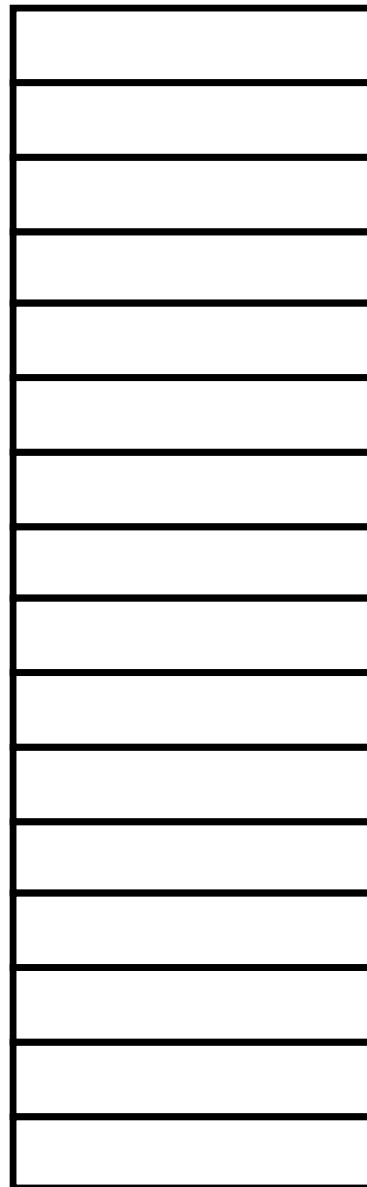


Cache映射

Memory

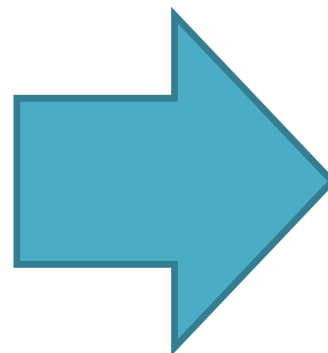
Block ID

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111



Mapping:

$\text{index}(\text{Block ID}) = ???$



Cache

Index

00
01
10
11



$S = \# \text{ slots} = 4$

Small, fixed number of block slots.

Large, fixed number of block slots.

Cache映射

- 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

Cache映射

■ 什么是Cache的映射功能？

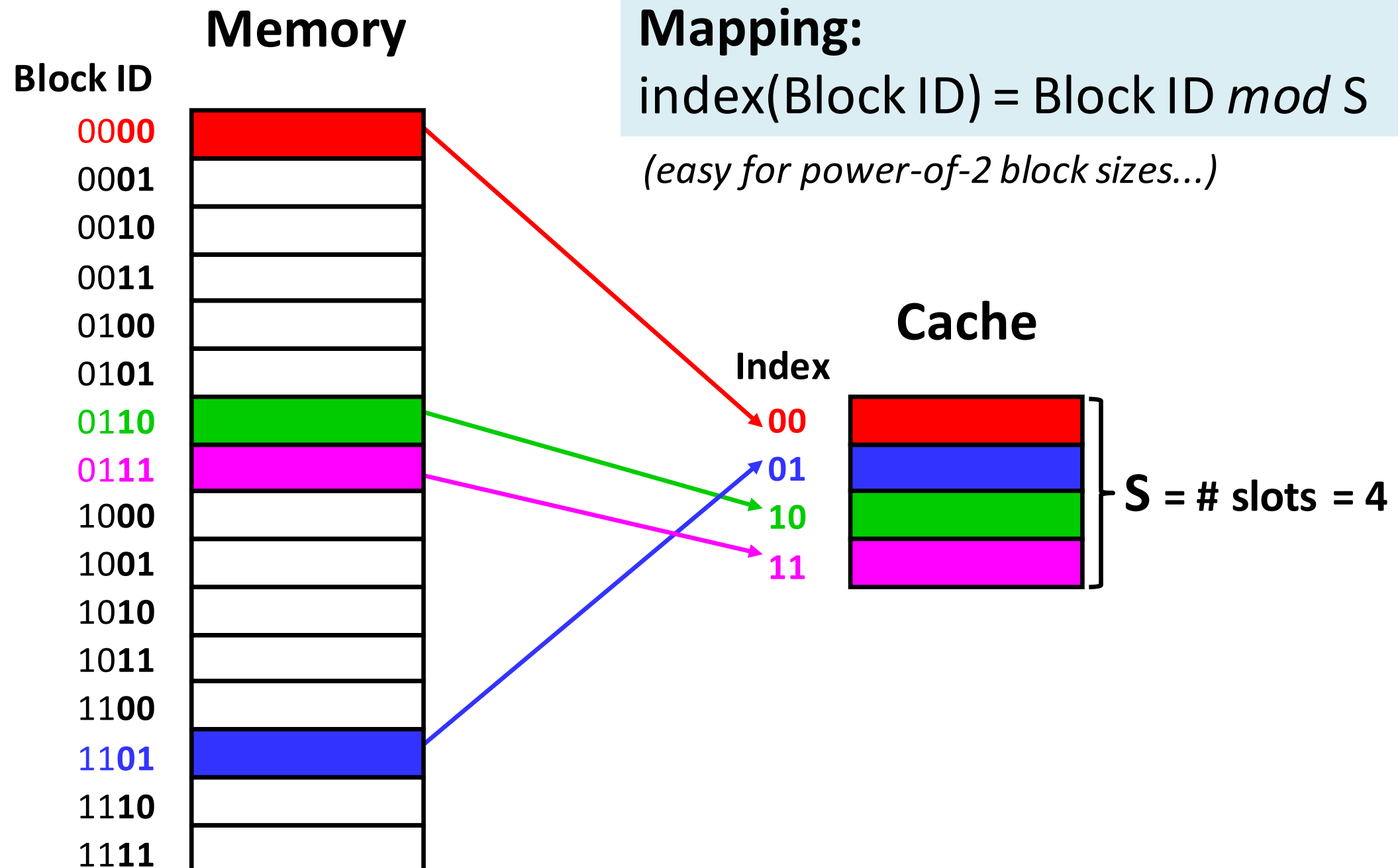
- 把访问的局部主存区域取到Cache中时，该放到Cache的何处？
- Cache行比主存块少，多个主存块映射到一个Cache行中

■ 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) 或块 (Block)
- 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的固定行中
 - 全相联(Full Associate)：每个主存块映射到Cache的任意行中
 - 组相联(Set Associate)：每个主存块映射到Cache的固定组中的任意一行中

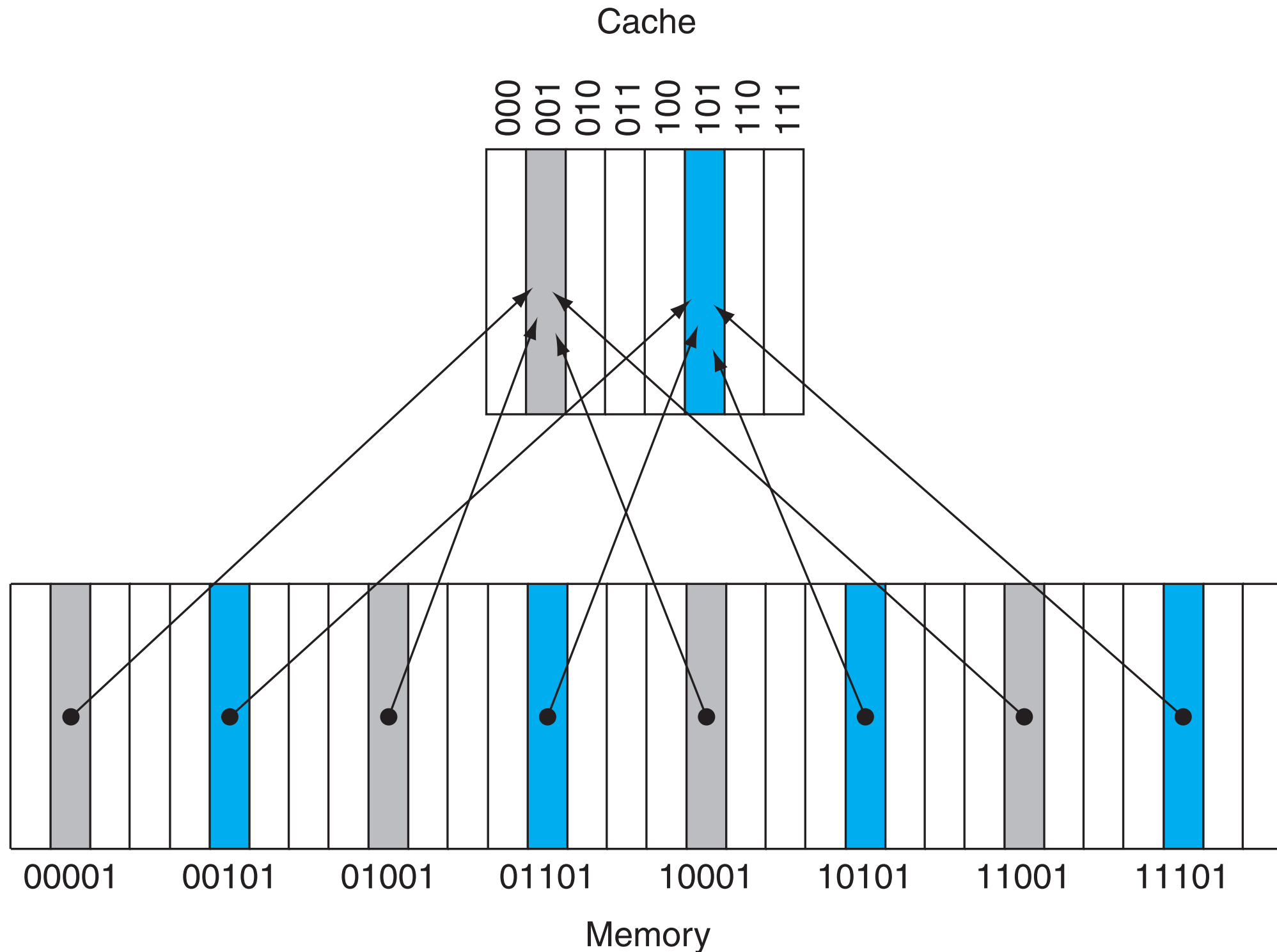
直接映射方式

- 一个主存块只能拷贝到cache的一个特定行位置上去



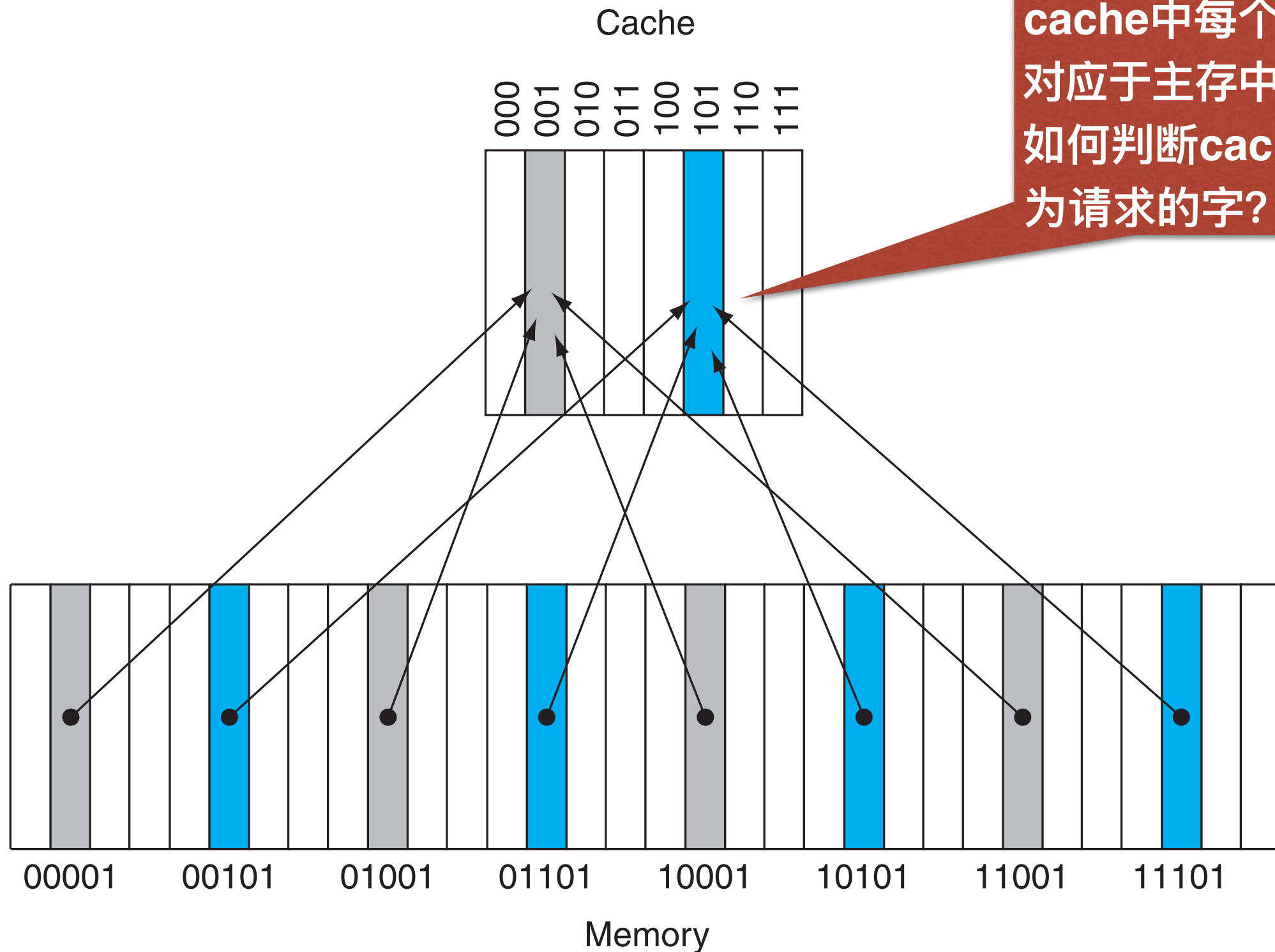
直接映射方式

- 主存地址为0-31被映射到cache中，cache有8块，1字/块



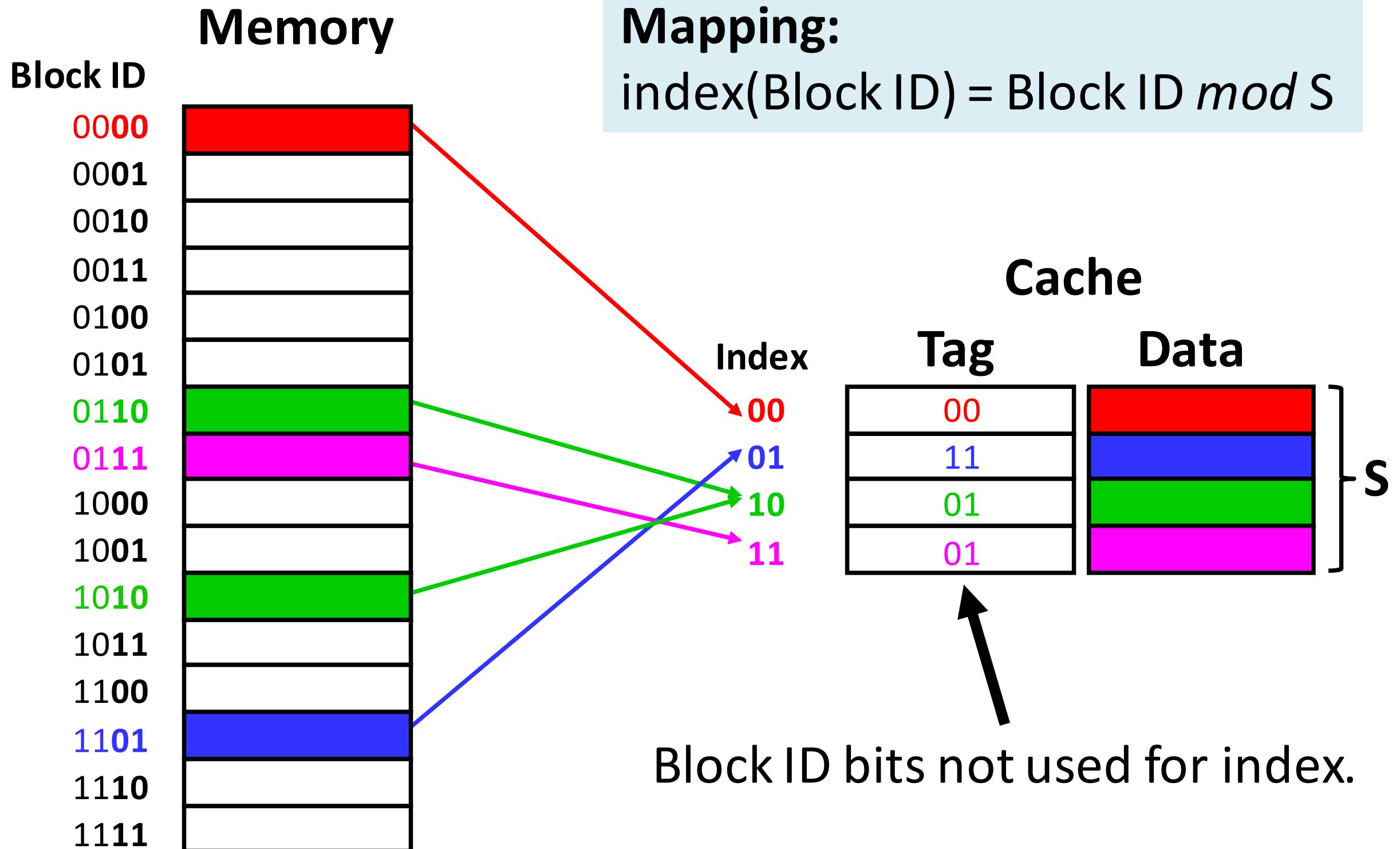
直接映射方式

- 主存地址为0-31被映射到cache中，cache有8块，1字/块



cache中每个位置可能对应于主存中多个地址，如何判断cache中是否为请求的字？

直接映射方式



直接映射方式

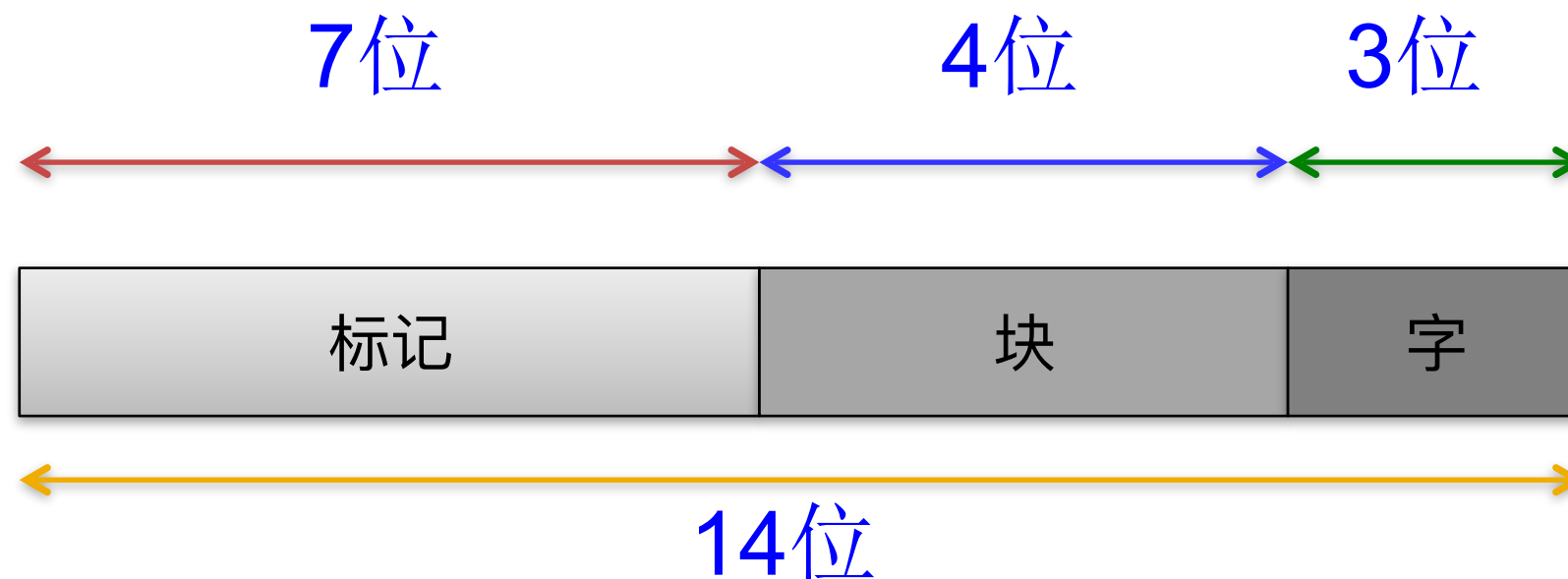
- 存储器 2^{14} 字组成
- Cache有16行
- 每个块、行由8个字组成
- 存储器字地址位数？
- 存储器共划分为多少块？

直接映射方式

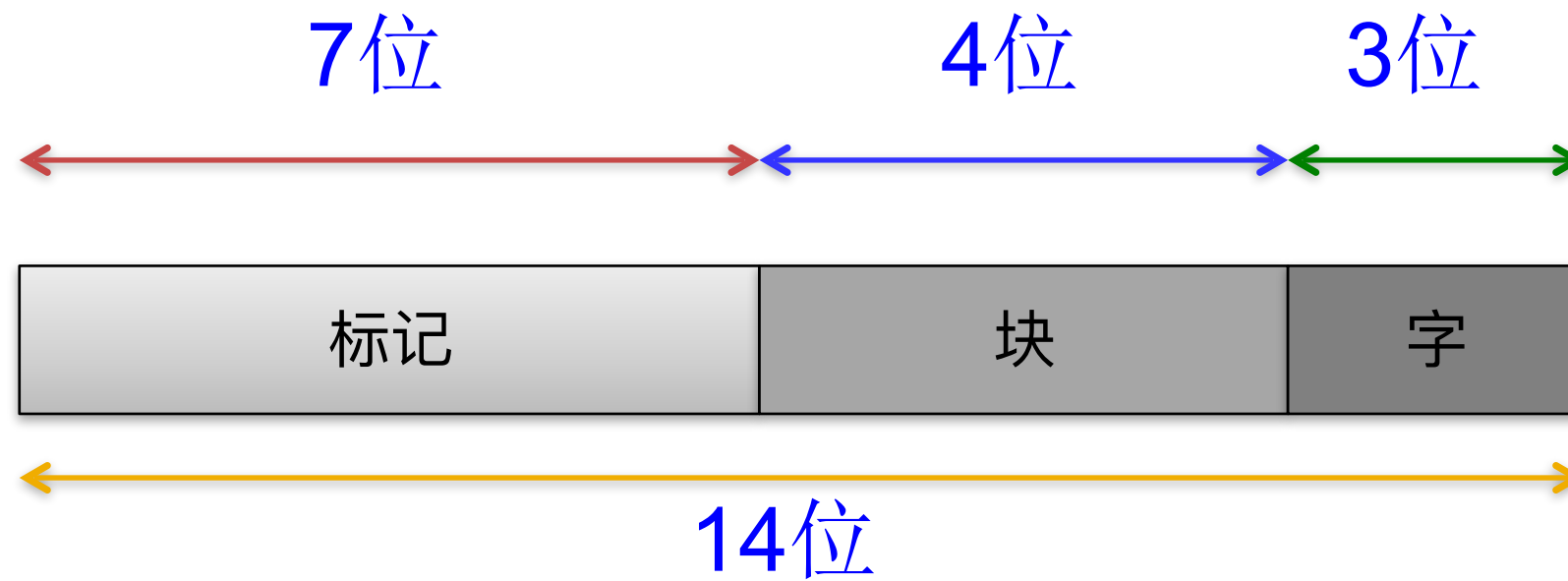
- 存储器 2^{14} 字组成
- Cache有16行
- 每个块、行由8个字组成
- 存储器字地址位数？
 - 14位
- 存储器共划分为多少块？
 - $2^{14}/2^3=2^{11}$

直接映射方式

- 存储器 2^{14} 字组成
- Cache有16行
- 每个块、行由8个字组成
- 存储器字地址位数?
 - 14位
- 存储器共划分为多少块?
 - $2^{14}/2^3=2^{11}$

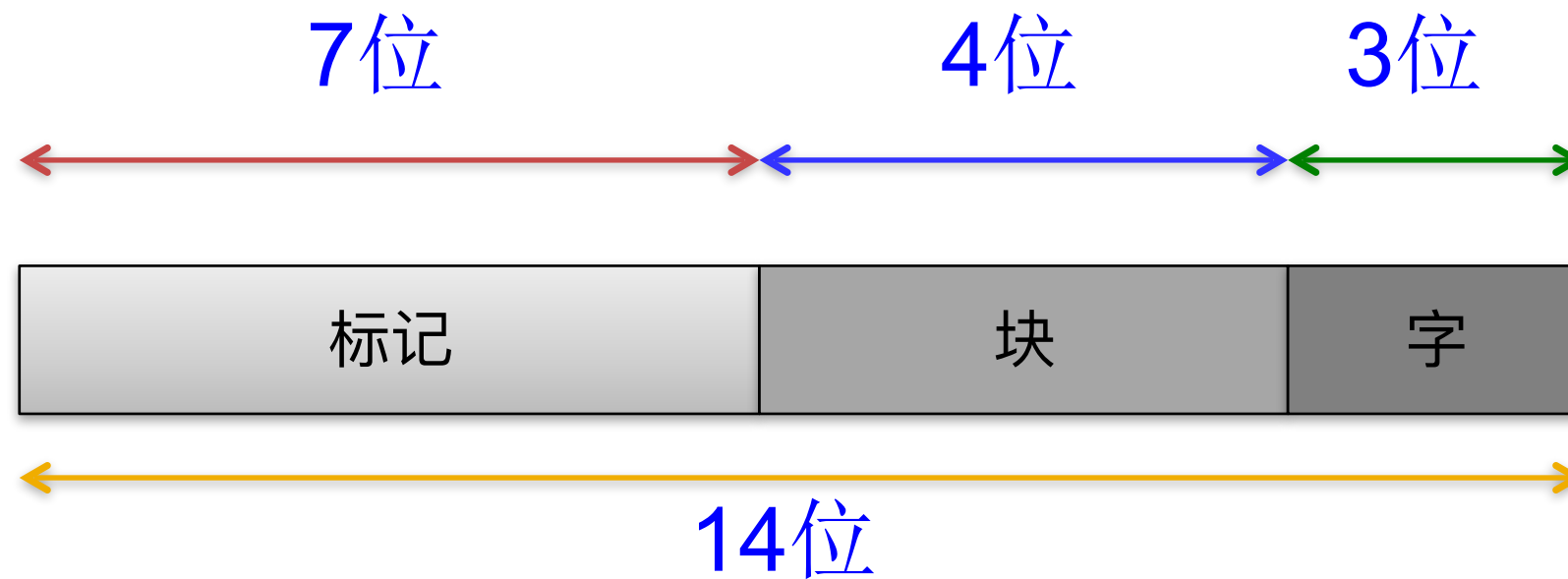


直接映射方式



- 主存地址：
 - d=11011001100011
- 在主存中的块号：
- 映射到cache中的行号：

直接映射方式



■ 主存地址：

- $d=11011001100011$

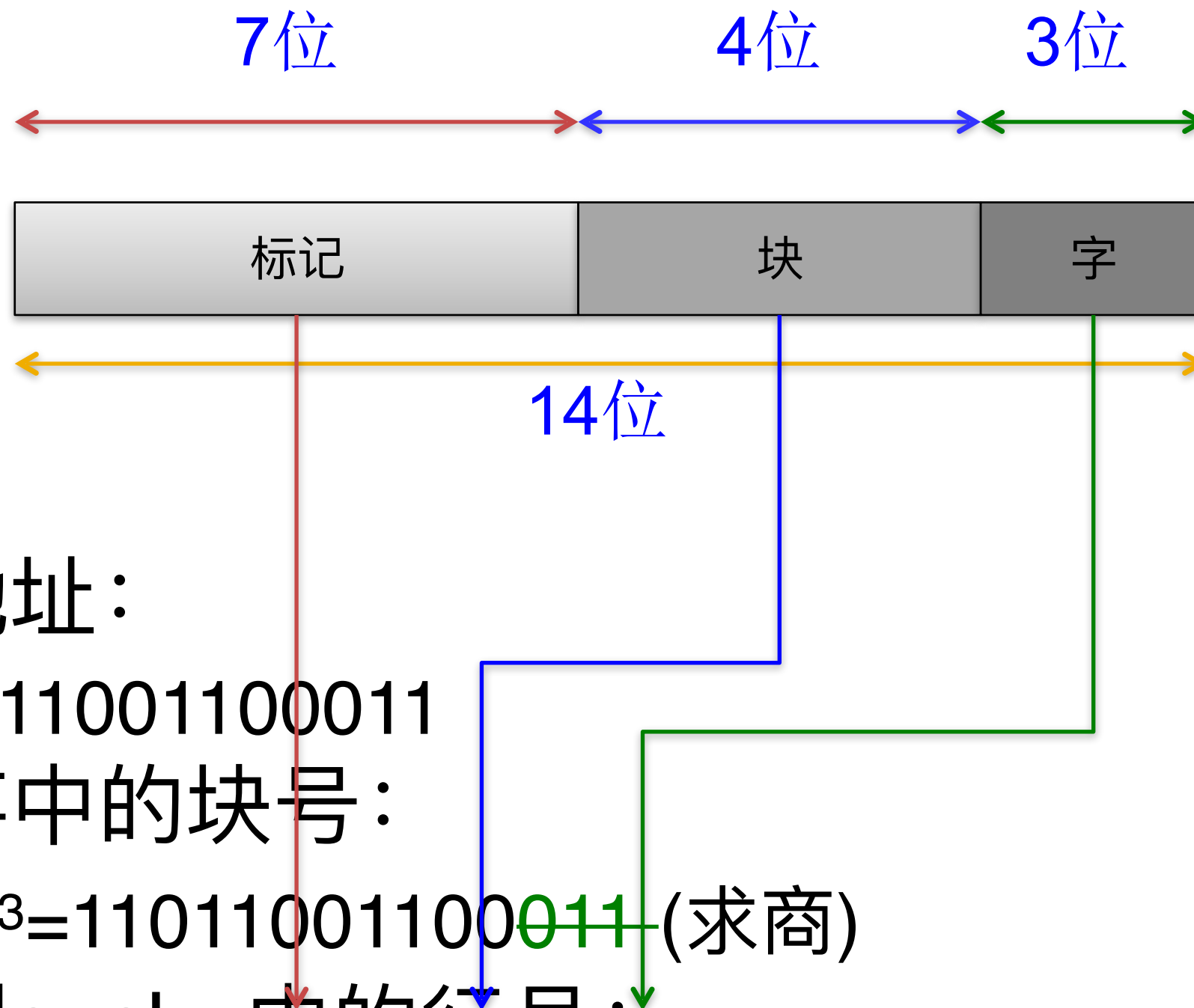
■ 在主存中的块号：

- $j=d \setminus 2^3=11011001100\cancel{011}$ (求商)

■ 映射到cache中的行号：

- $i=j \% 2^4 = \underline{1101100}1100\cancel{011}$ (取余)

直接映射方式



■ 主存地址：

- $d=11011001100011$

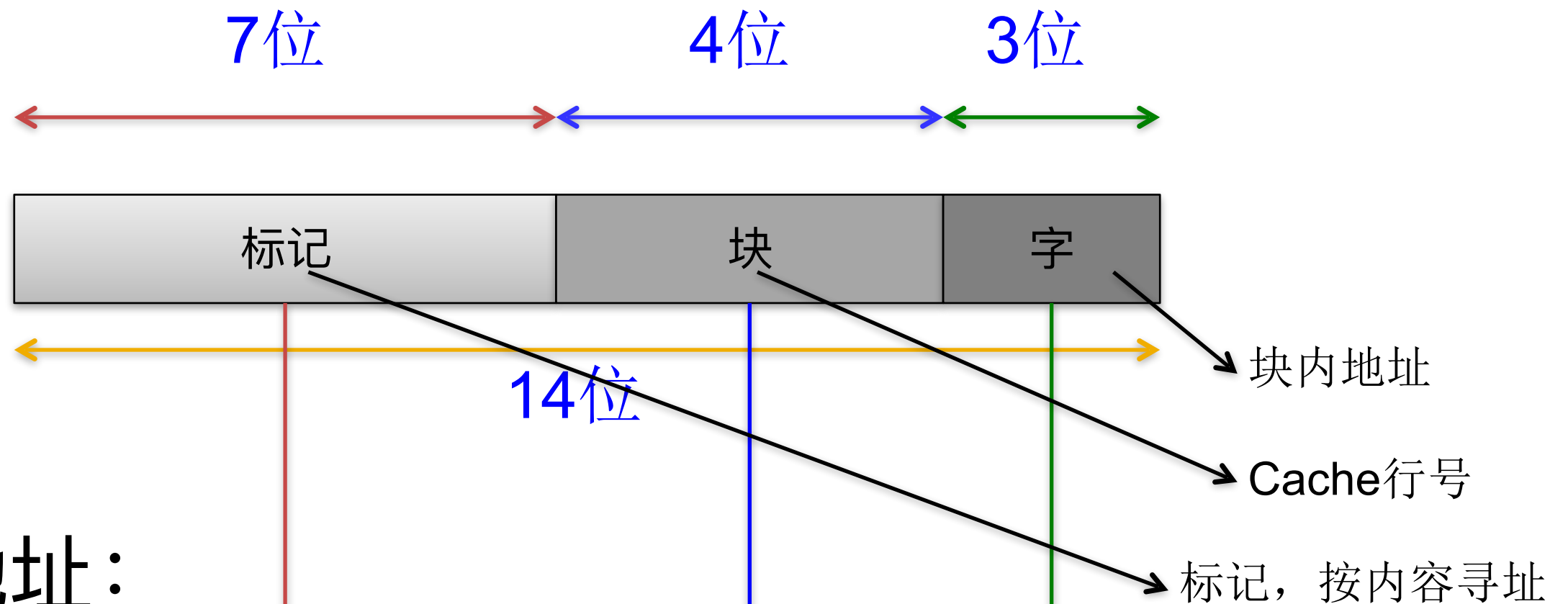
■ 在主存中的块号：

- $j=d \div 2^3=110110011000$ ~~11~~ (求商)

■ 映射到cache中的行号：

- $i=j \% 2^4 = \underline{1101100}11000$ ~~11~~ (取余)

直接映射方式



■ 主存地址：

- $d=11011001100011$

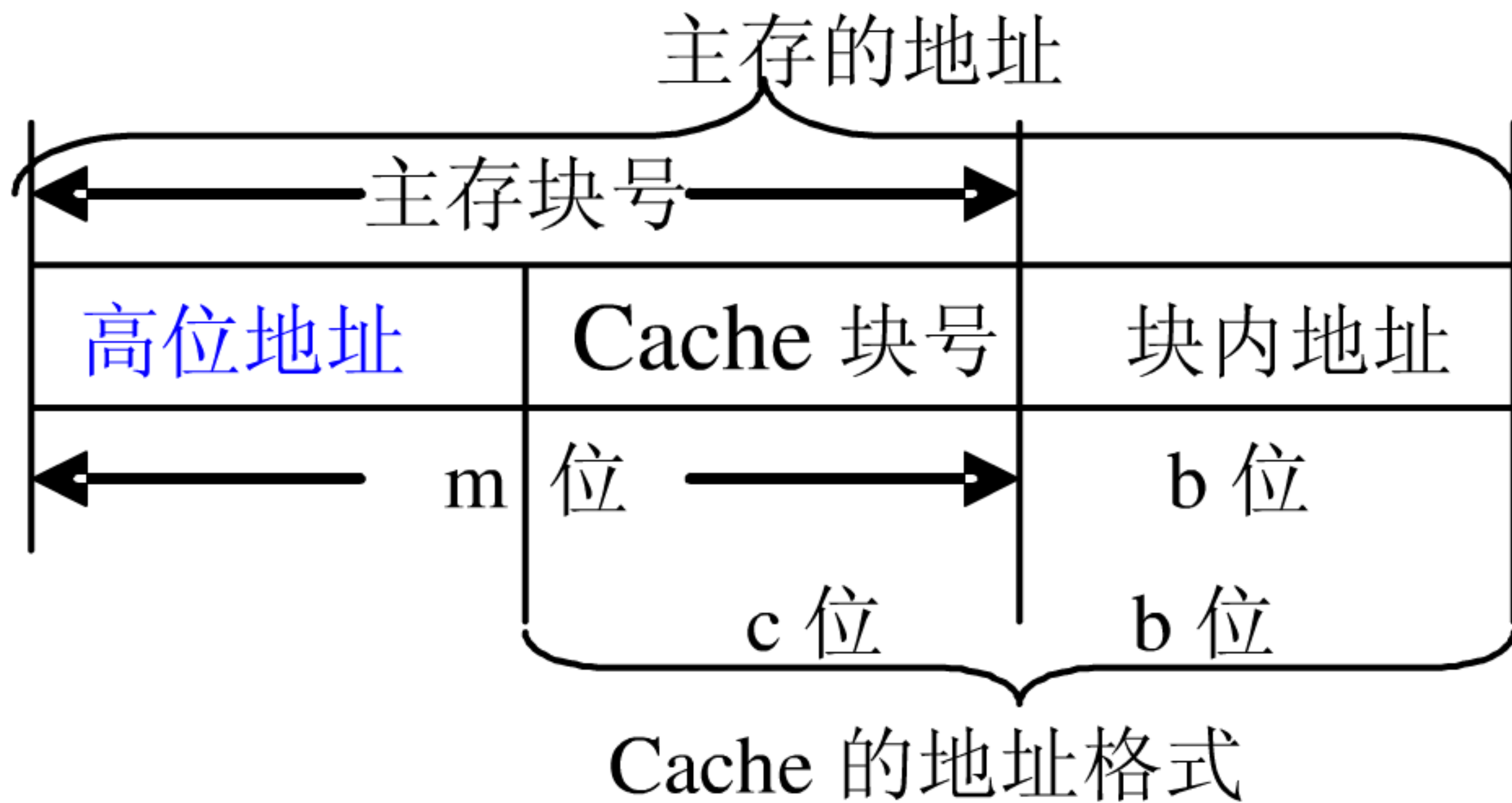
■ 在主存中的块号：

- $j=d \div 2^3=11011001100011$ (求商)

■ 映射到cache中的行号：

- $i=j \% 2^4 = \underline{1101100}1100011$ (取余)

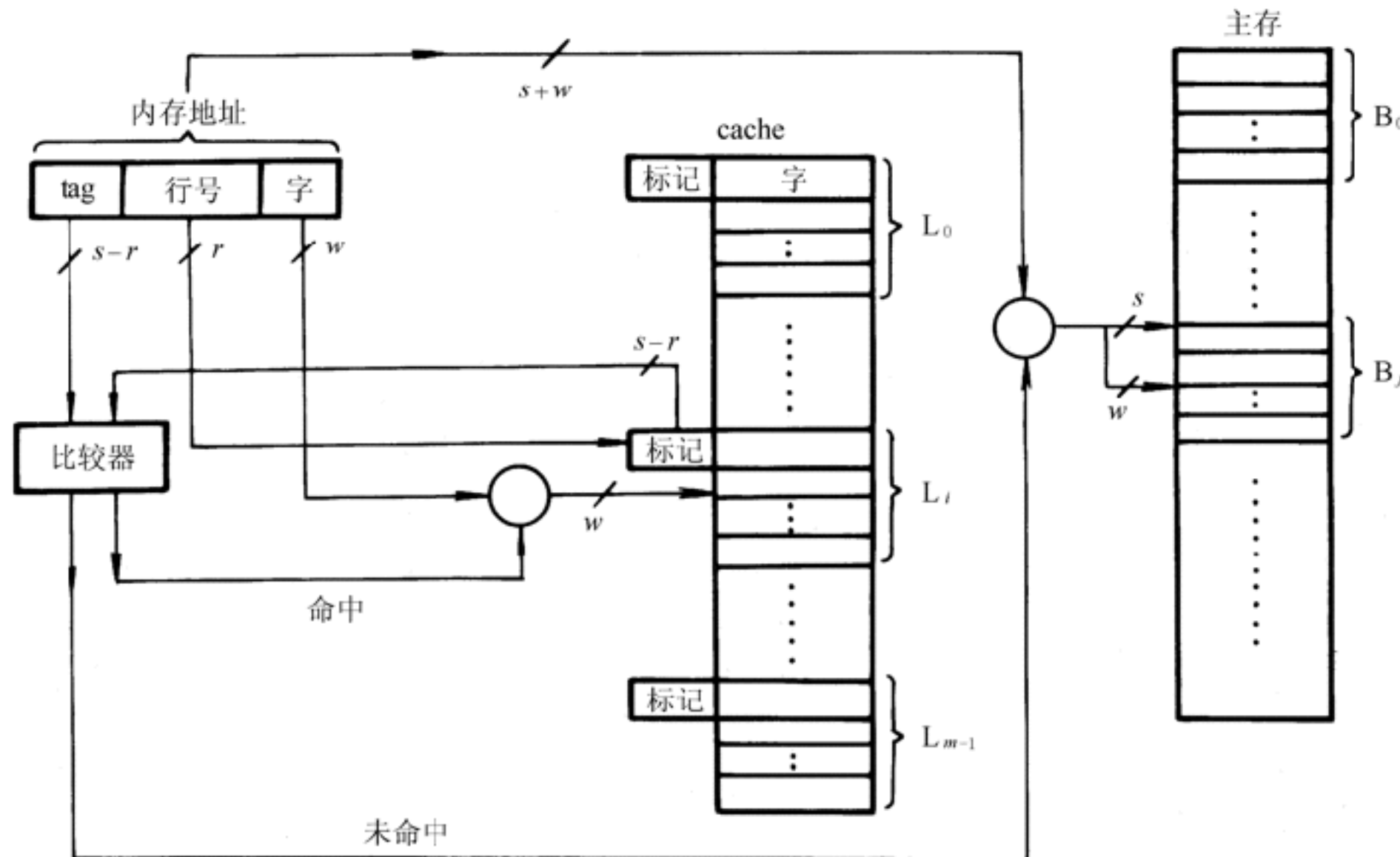
直接映射方式



直接映射方式

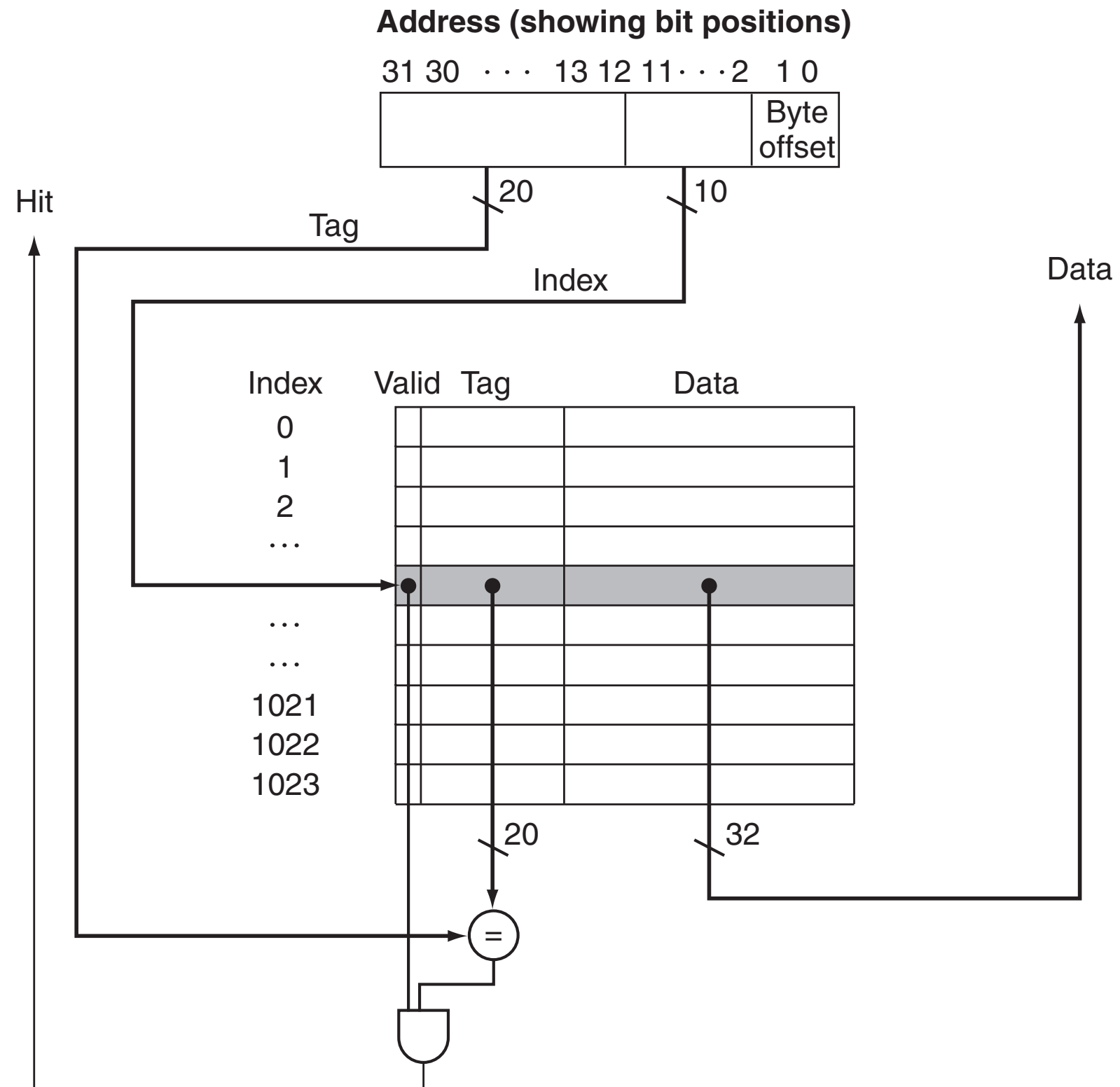
■ 映射过程

- 首先利用内存地址中的行号选择cache相应行；
- 把行标记与CPU访问地址中tag进行比较，相同表示命中，访问Cache；
- 如果没有命中，访问内存，并将相应块写入Cache

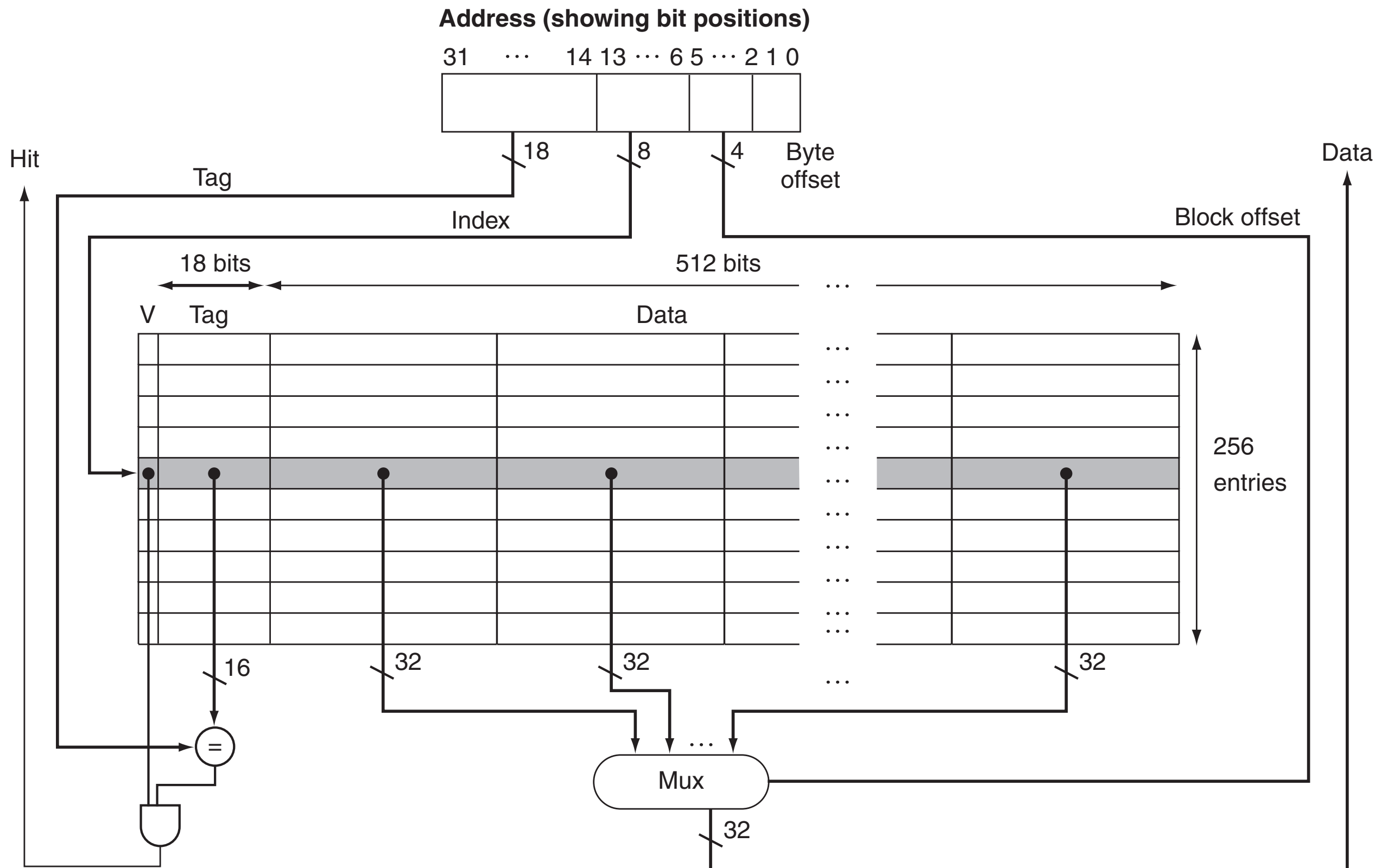


(b) 直接映射cache的检索过程

直接映射Cache硬件实现



直接映射Cache硬件实现



例

■ 假定数据在主存和Cache间的传送单位为512字。

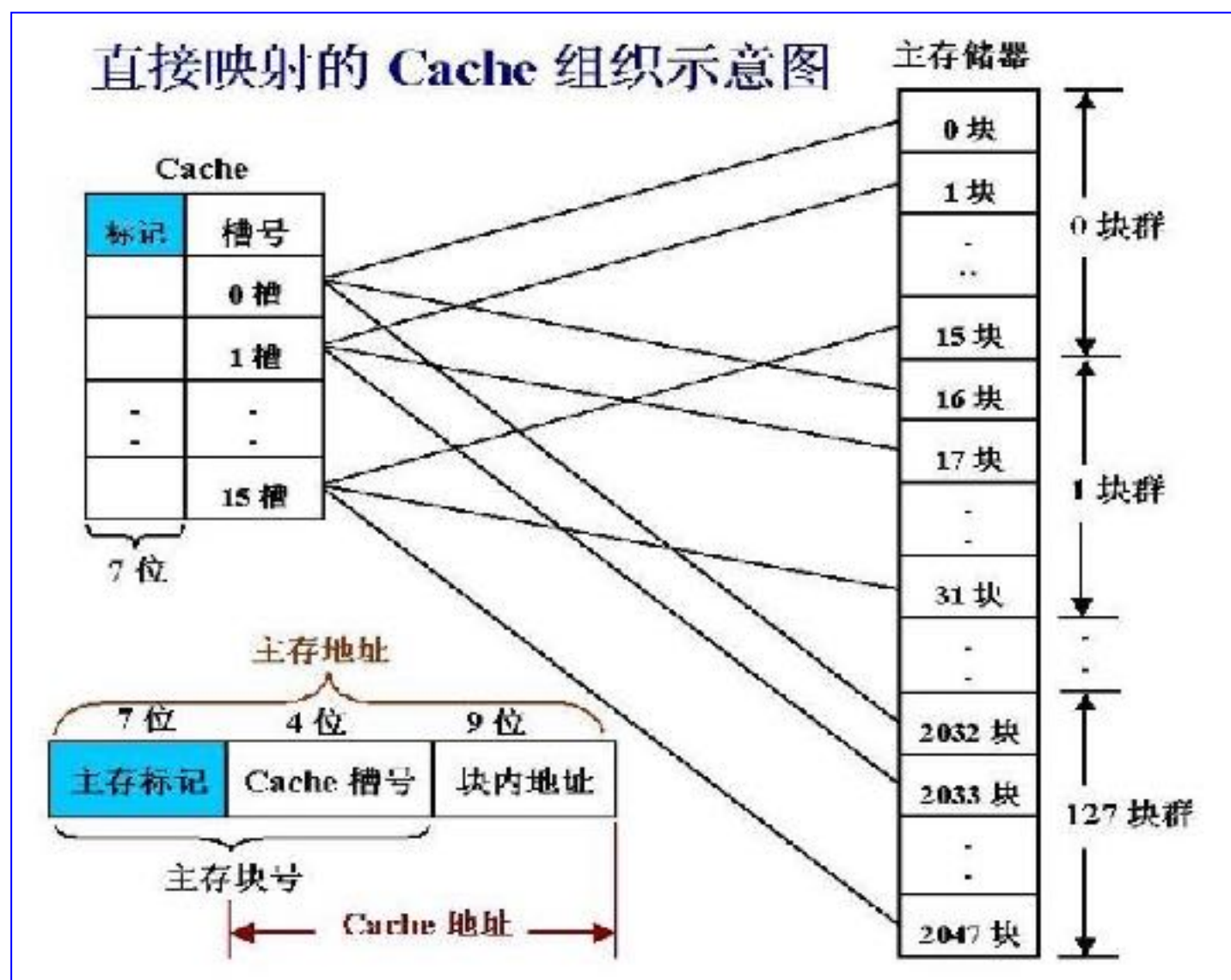
■ Cache大小：

■ 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

如何对0220CH单元进行访问？



例

■ 假定数据在主存和Cache间的传送单位为512字。

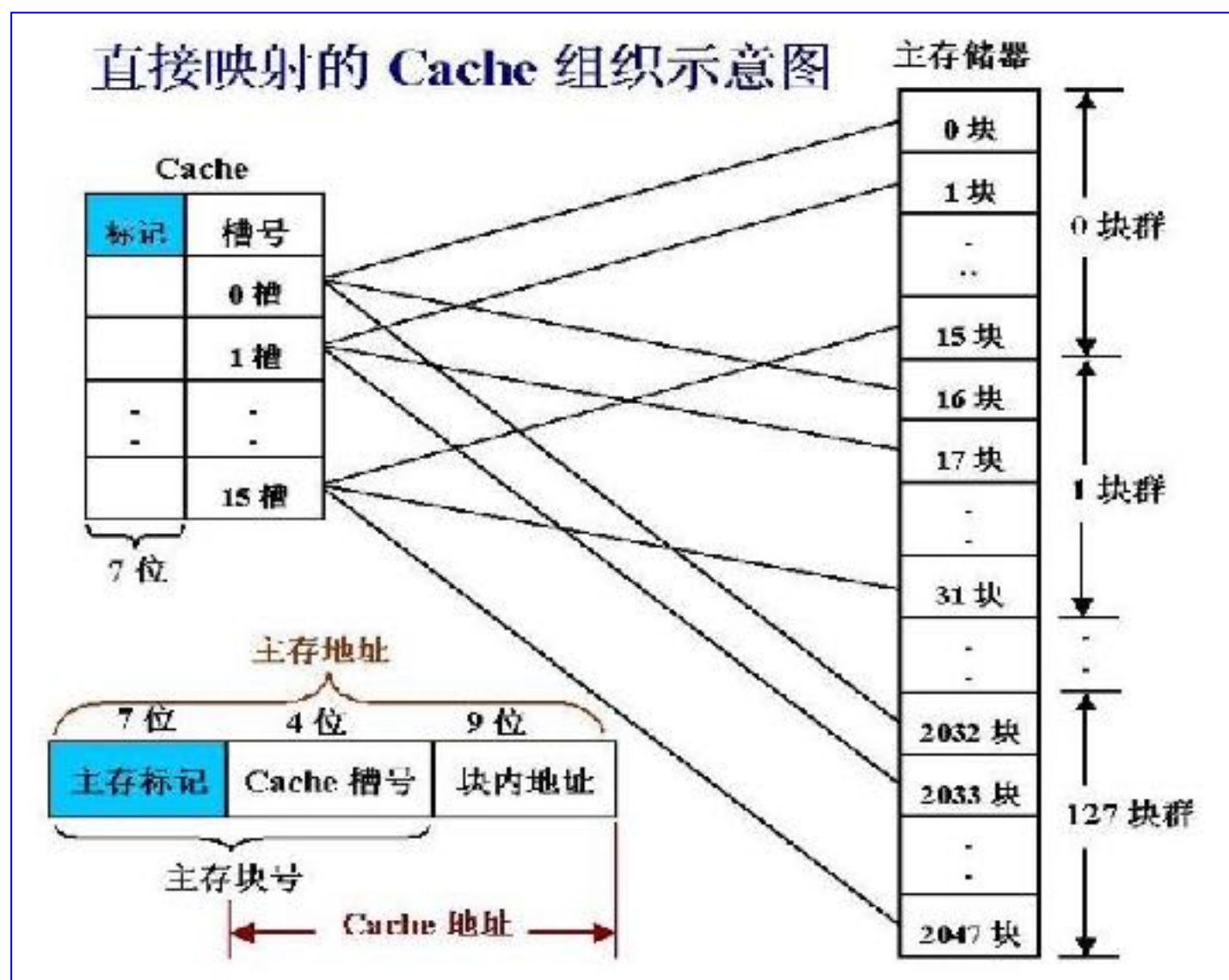
■ Cache大小：

■ 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

如何对0220CH单元进行访问？



0000 0010 0010 0000 1100B

例

■ 假定数据在主存和Cache间的传送单位为512字。

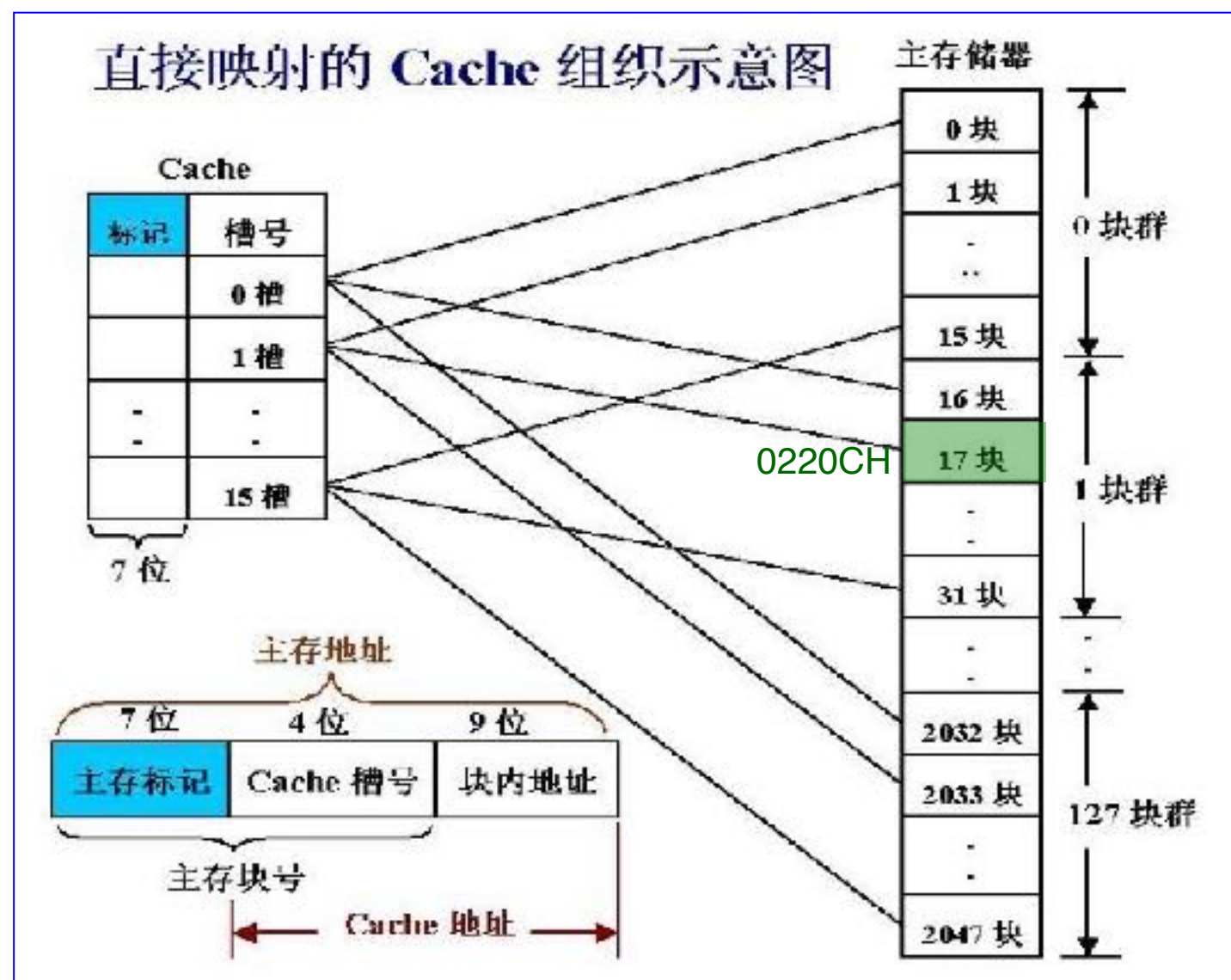
■ Cache大小：

■ 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

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！

例

■ 假定数据在主存和Cache间的传送单位为512字。

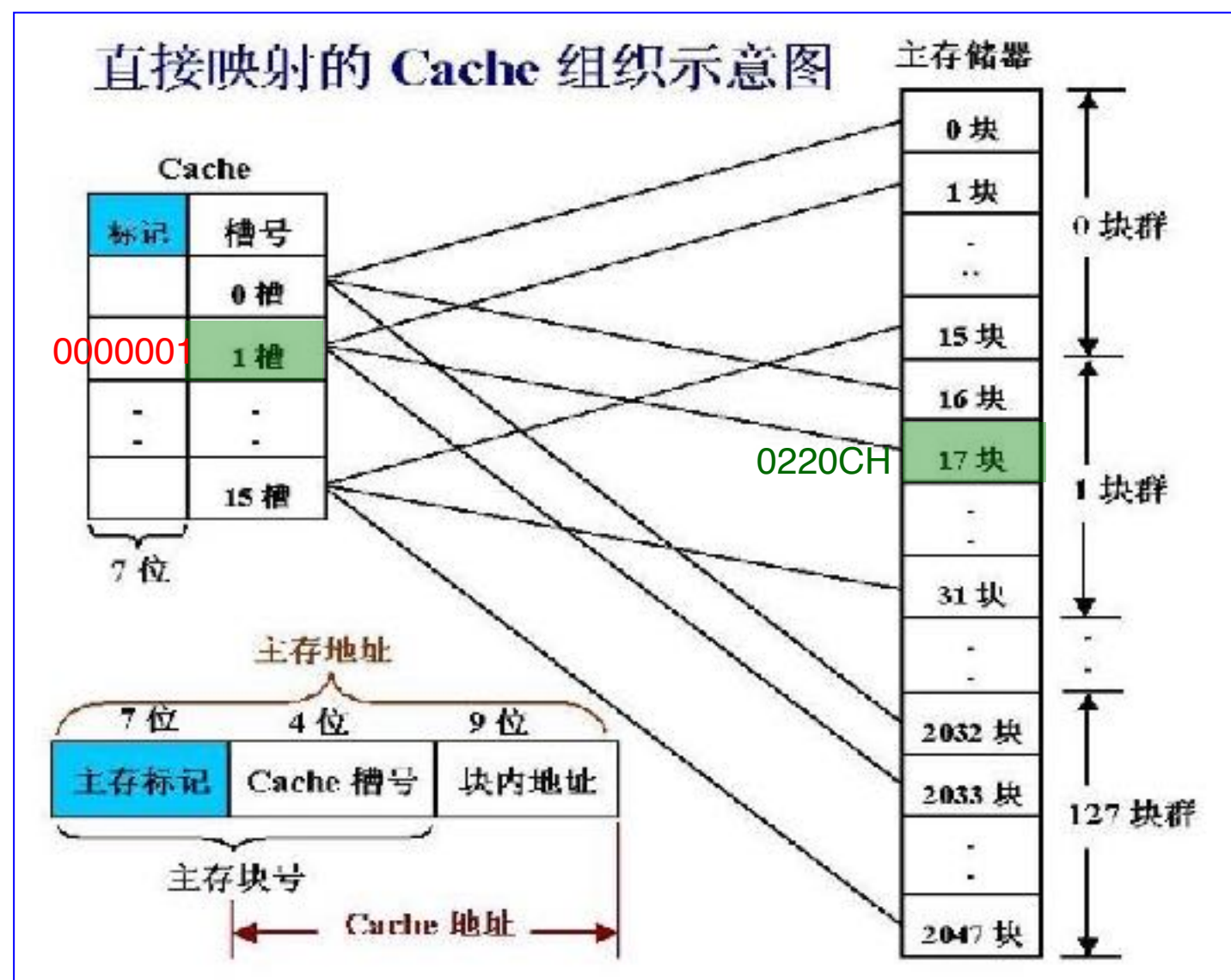
■ Cache大小：

■ 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

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！

直接映射方式特点

- 主存的字块只可以和固定的Cache 字块对应,方式直接,利用率低。
- 标志位较短,比较电路的成本低。如果主存空间有 2^m 块,Cache中 字块有 2^c 块,则标志位只要有 $m-c$ 位。且仅需要比较一次。
- 应用场合
 - 适合大容量cache, 更多的行数可减小冲突的机会

cache中的位数

- cache不仅存储数据，而且存储标记位，因此cache所需的总位数是cache大小、地址位数的函数
- 32位字节地址
- 直接映射cache
- cache大小为 2^n 块
- 块大小为 2^m 字
- 标记域大小为 $32-(m+n+2)$
- cache总位数为 $2^n \times (\text{块大小} + \text{标记域大小} + \text{有效位大小})$

cache中的位数

- 思考题：直接映射cache，16KB的数据，块大小为4字，地址为32位，那么该cache总共需要的位数是？

cache中的位数

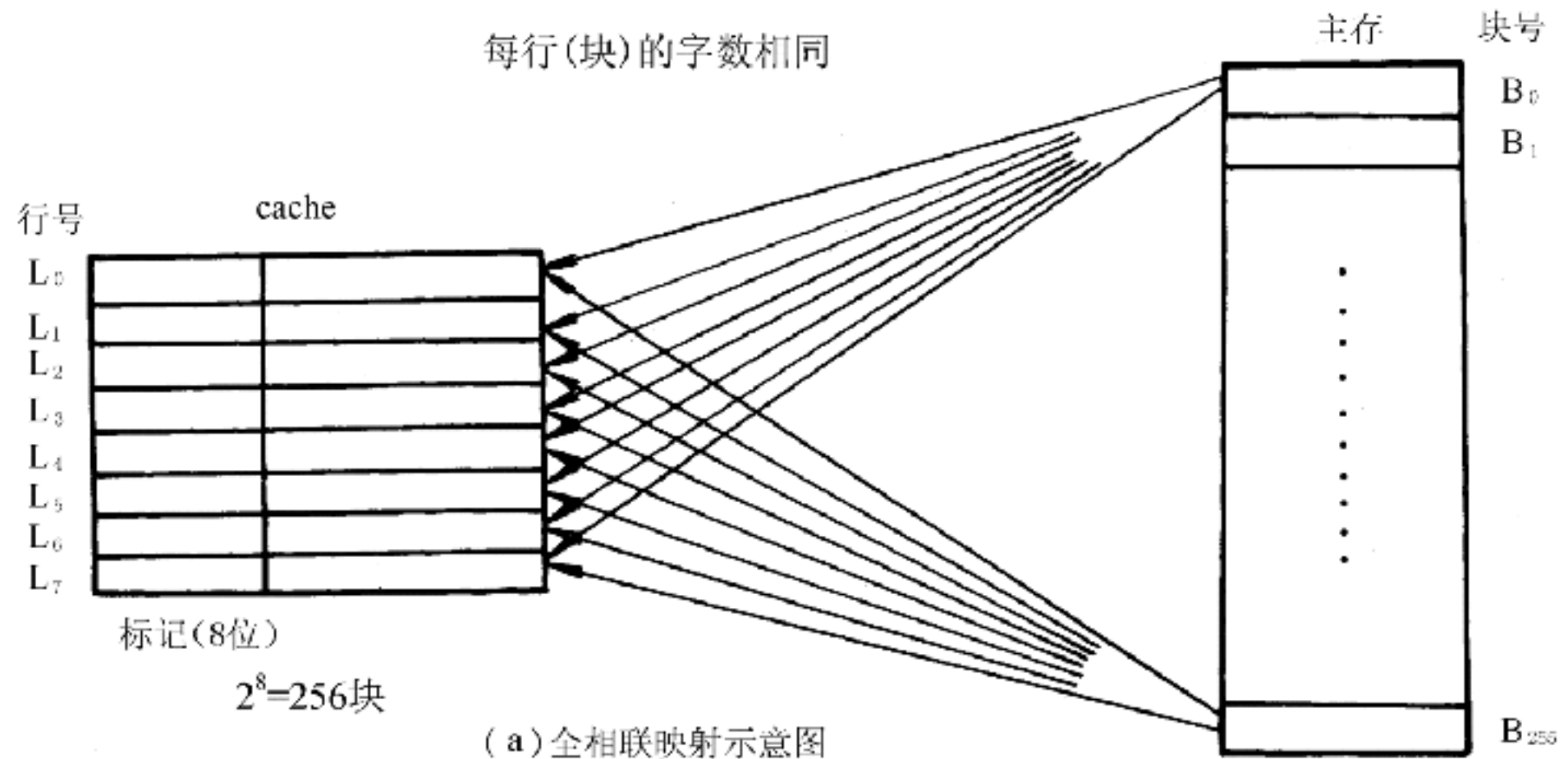
- 思考题：直接映射cache，16KB的数据，块大小为4字，地址为32位，那么该cache总共需要的位数是？
- $2^n \times (\text{块大小} + \text{标记域大小} + \text{有效位大小})$

cache中的位数

- 思考题：直接映射cache，16KB的数据，块大小为4字，地址为32位，那么该cache总共需要的位数是？
- $2^n \times (\text{块大小} + \text{标记域大小} + \text{有效位大小})$
- $2^{10} \times (4 \times 32 + (32 - 10 - 2 - 2) + 1) = 2^{10} \times 147$

全相联映射方式

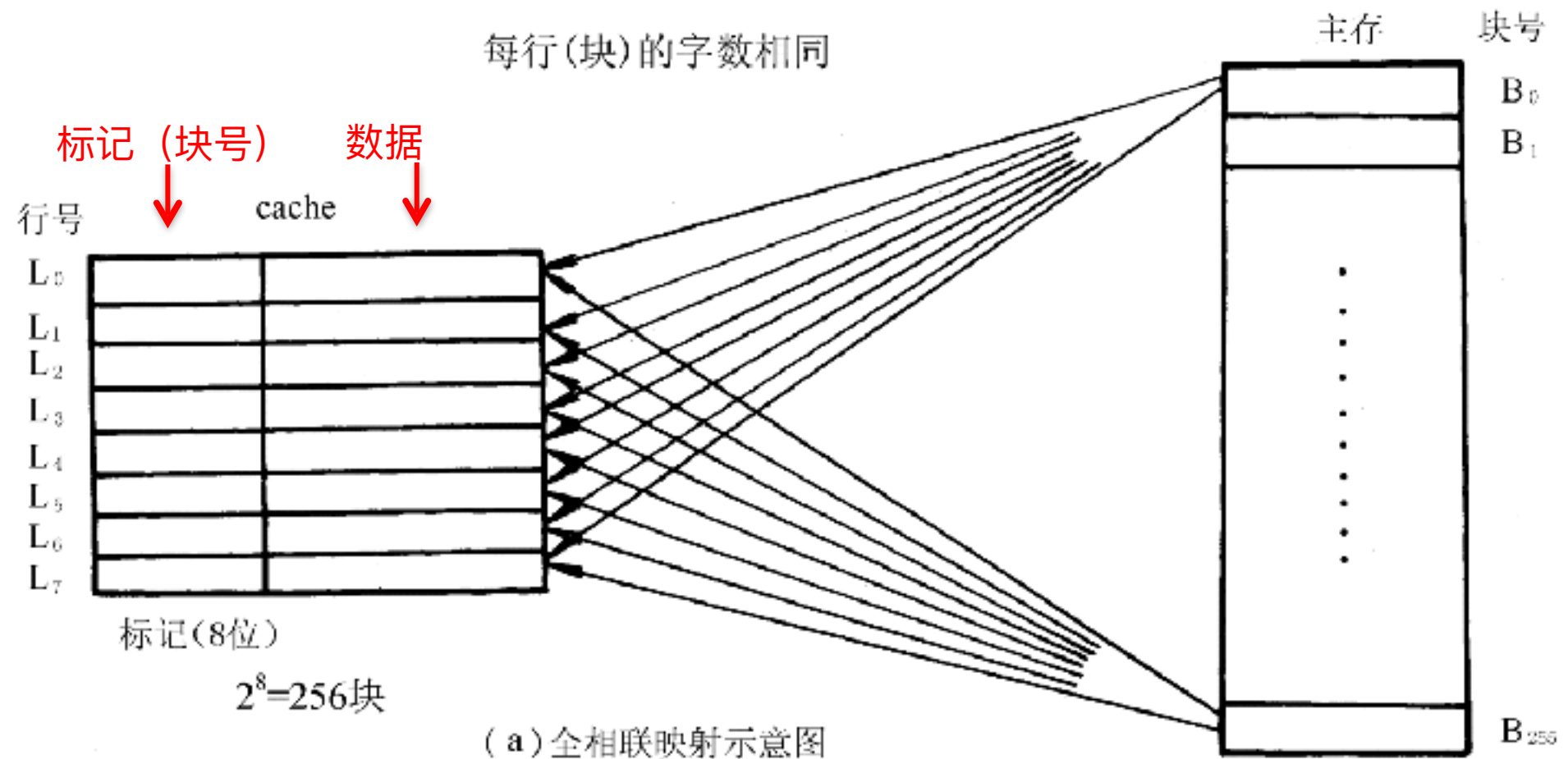
全相联映射方式基本思想：一个主存块可以装入cache任意一行



- Cache中的数据块大小称为行， L_i 表示
- 主存中的数据块大小称为块， B_j 表示
- 行与块等长，每个块由连续的字组成
- Cache的一行存储了一个块的地址（块号）和块的内容（字）

全相联映射方式

全相联映射方式基本思想：一个主存块可以装入cache任意一行



主存地址

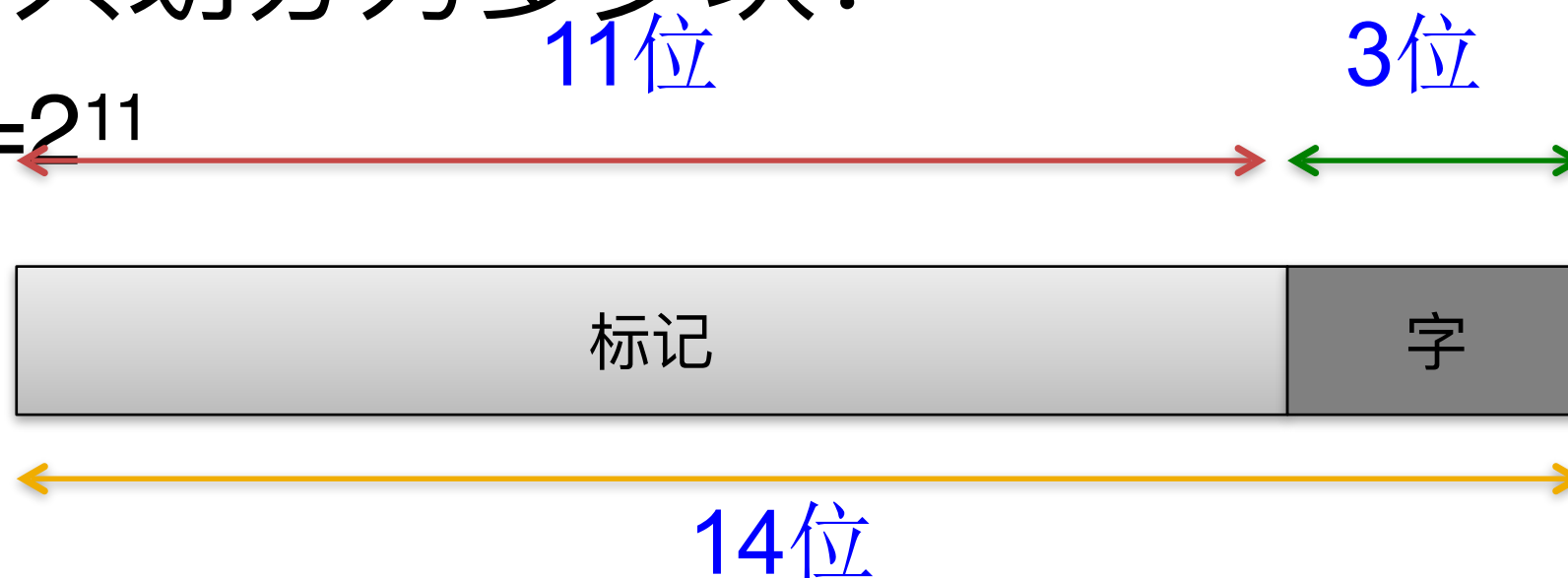
标记

块内地址

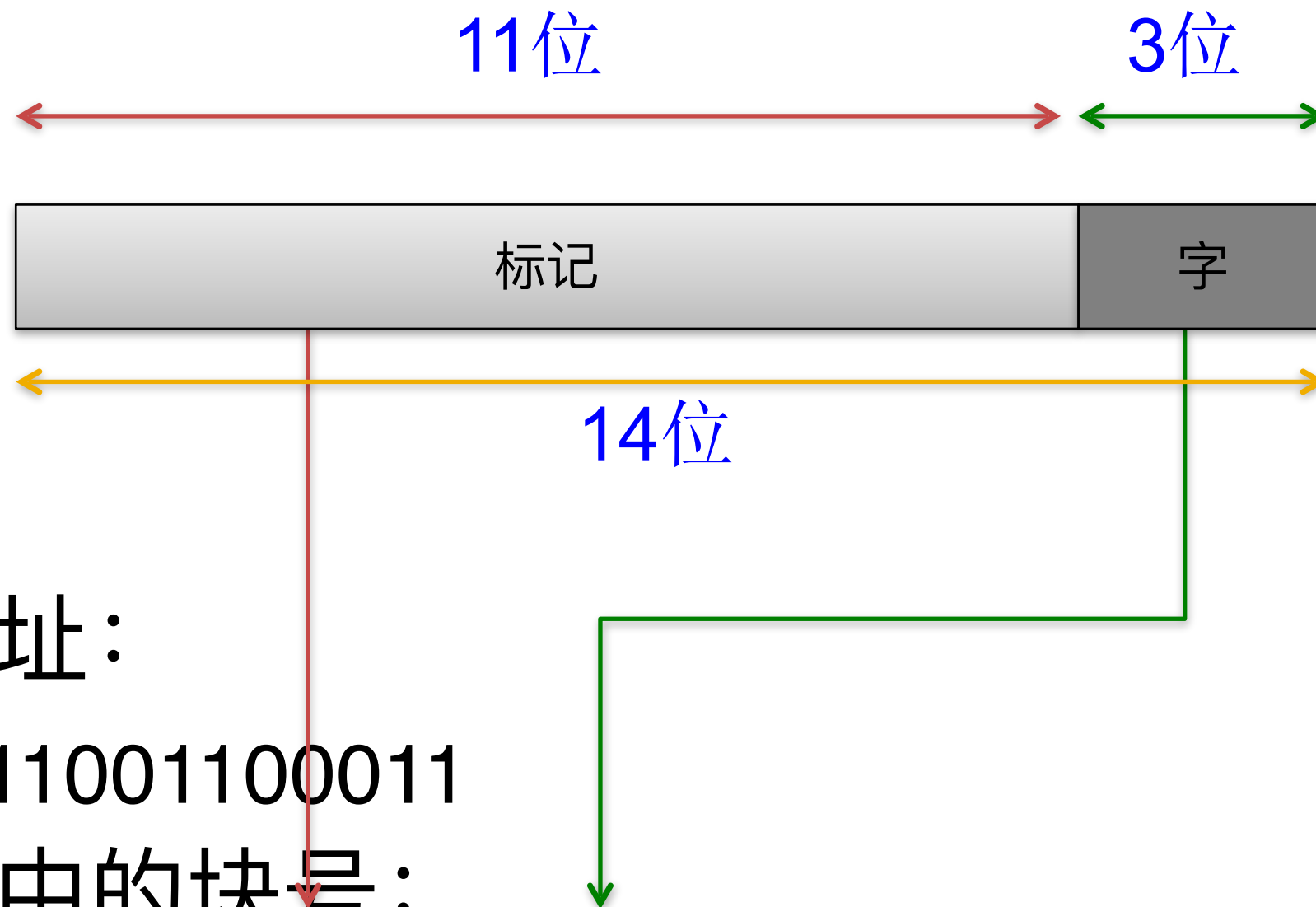
- Cache中的数据块大小称为行， L_i 表示
- 主存中的数据块大小称为块， B_j 表示
- 行与块等长，每个块由连续的字组成
- Cache的一行存储了一个块的地址（块号）和块的内容（字）

全相联映射方式

- 存储器 2^{14} 字组成
- Cache有16行
- 每个块、行由8个字组成
- 存储器地址位数?
 - 14位
- 存储器共划分为多少块?
 - $2^{14}/2^3=2^{11}$



全相联映射方式



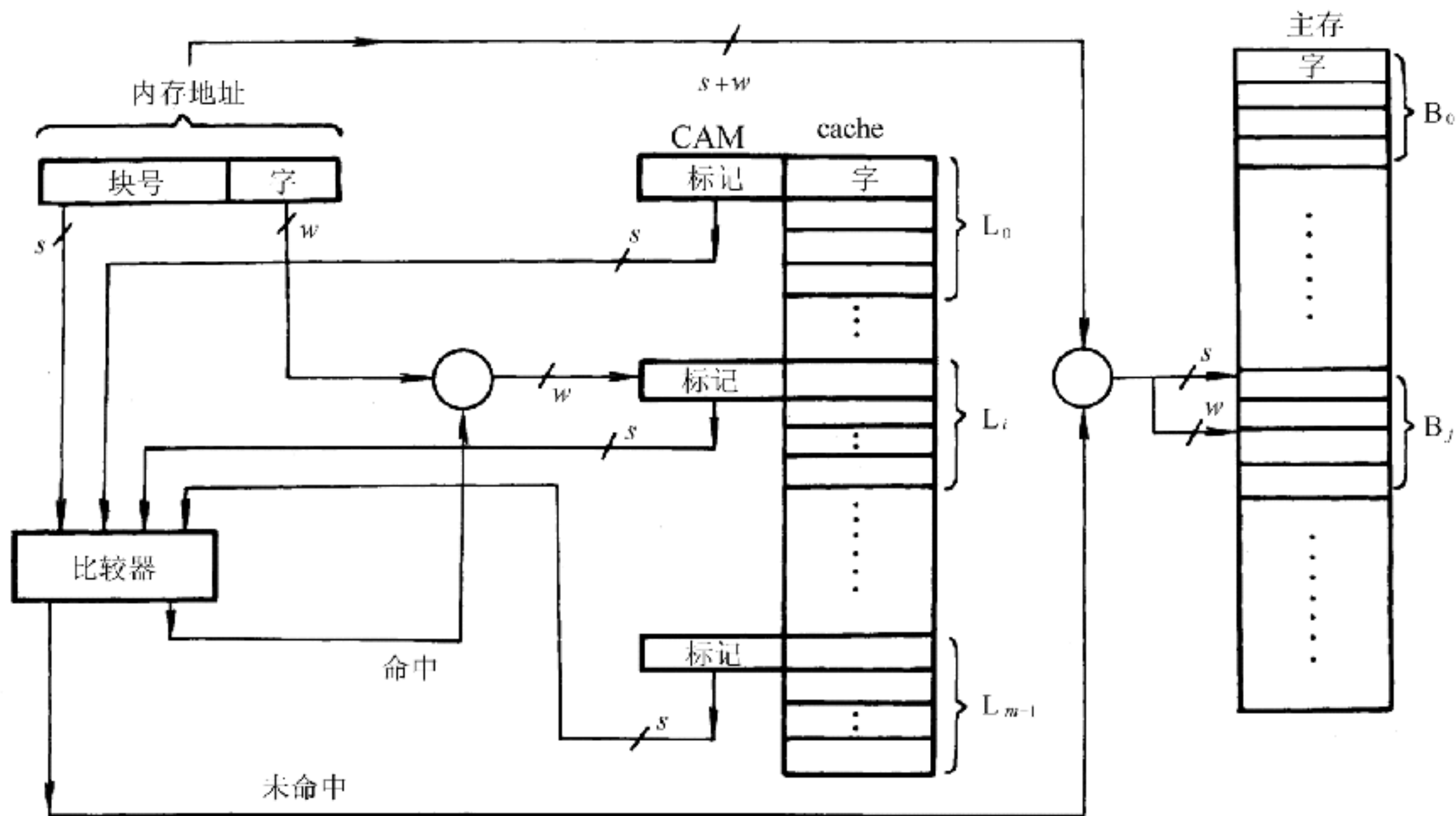
■ 主存地址：

- $d=11011001100011$

■ 在主存中的块号：

- $j=d \setminus 2^3=11011001100011$ (求商)

全相联映射方式



(b) 全相联cache的检索过程

CPU给出访问地址后，也将地址分为两部分（块号和字），比较电路块号与Cache表中的标记进行比较，相同表示命中，访问相应单元；如果没有命中访问内存，CPU直接访问内存，并将被访问内存的相对应块写入Cache。

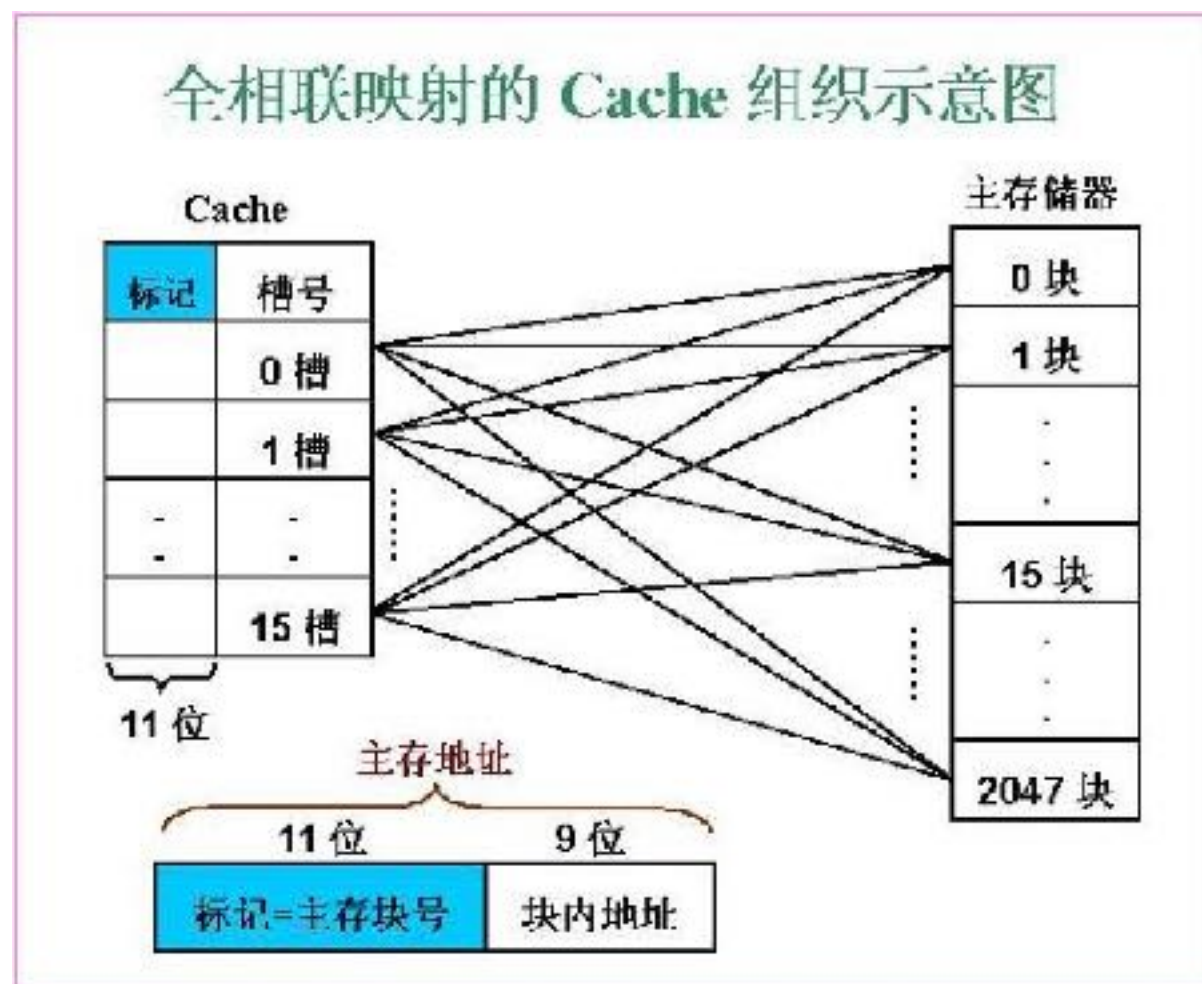
例

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

如何对01E0CH单元进行访问？



例

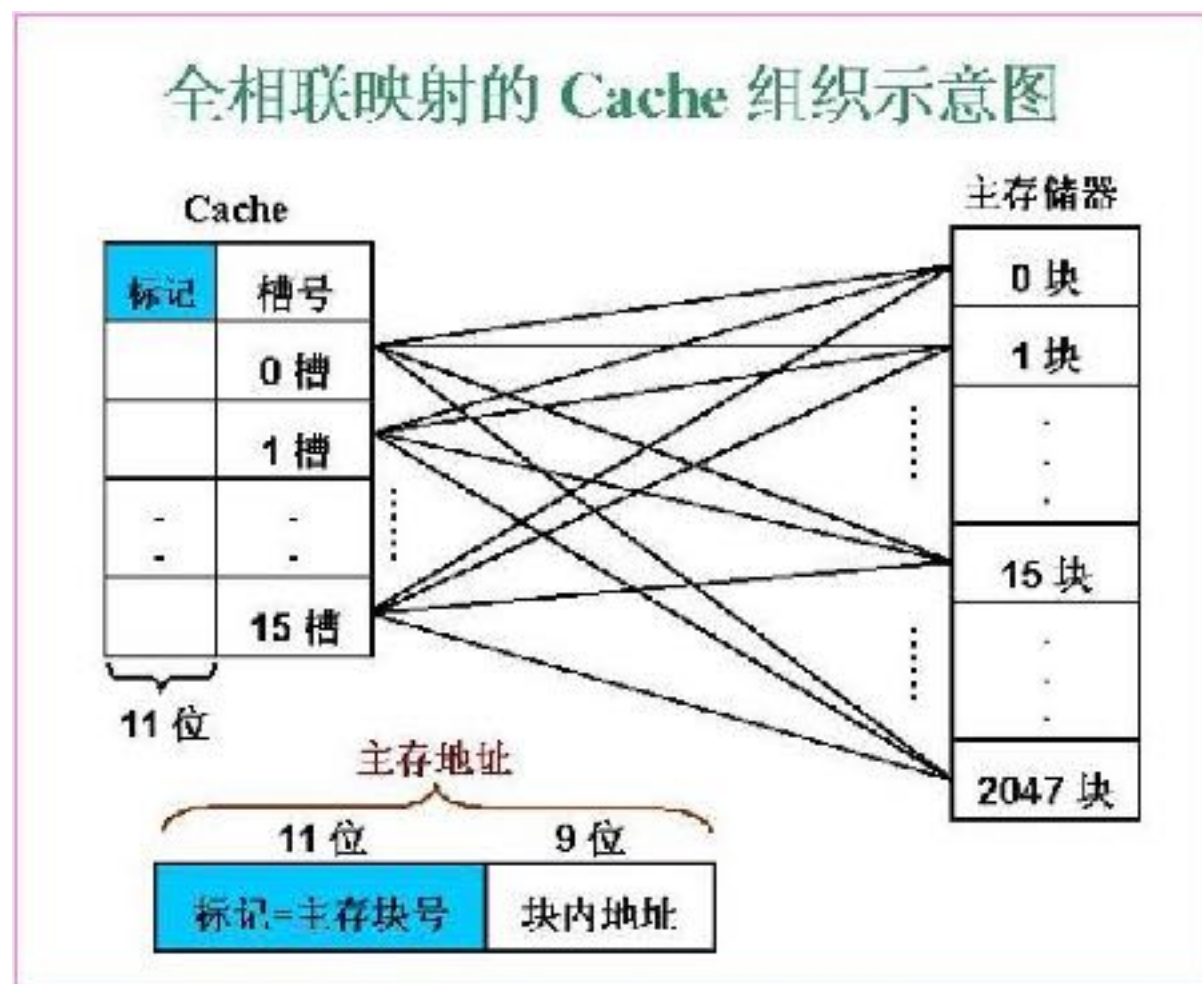
假定数据在主存和Cache间的传送单位为512字。

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

如何对01E0CH单元进行访问?

0000 0001 1110 0000 1100B



例

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

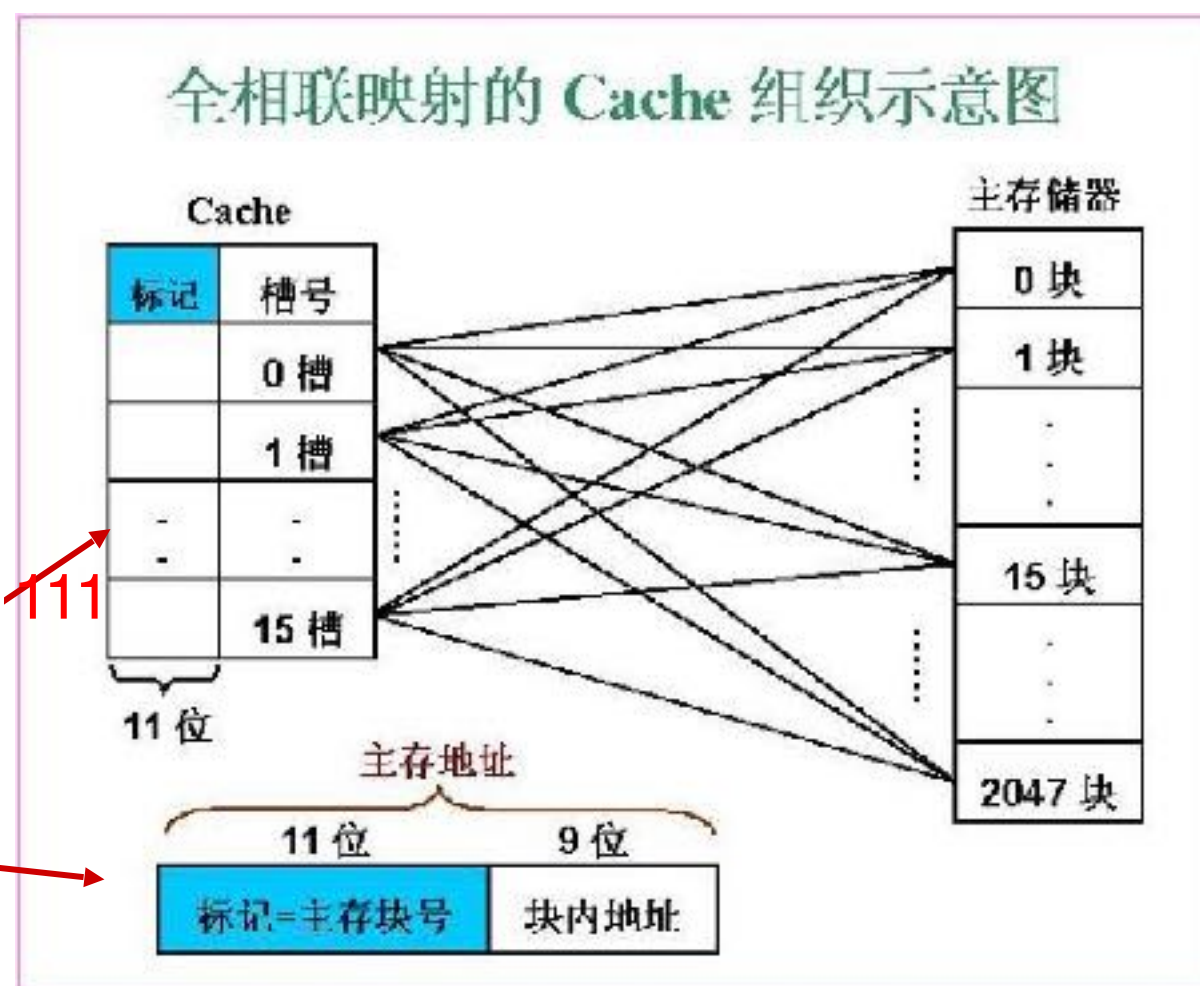
主存大小: 2^{20} 字=1024K字=2048块 x 512字/块

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!

可映射到任意cache行中



全相联的映射方式

■ 特点：

- 优点：冲突概率小，Cache的利用率高。
- 缺点：标志位较长,比较电路的成本太高。如: n 位的主存地址,块内地址为 b 位,Cache有 m 块,则需要有 m 个比较电路,标志位需 $n-b$ 位。

■ 应用场合：

- 适用于小容量的Cache

组相联映射方式

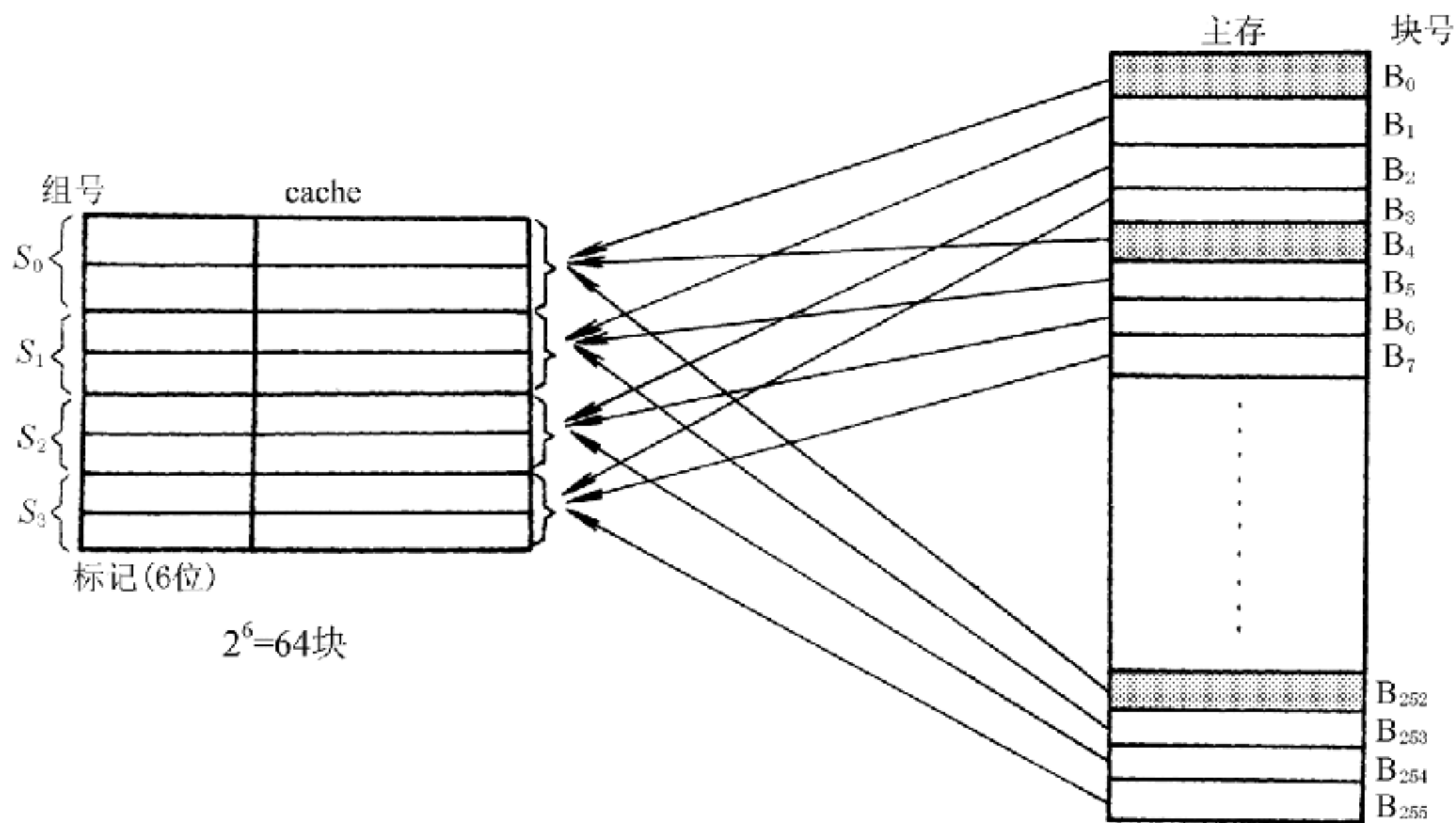
■ 全相联映射和直接映射的组合

- 全相联映射：灵活性好，命中率高
- 直接映射：硬件简单，成本低
- 组相连映射：前两种方案的折衷方案

■ 映射方法

- 将cache分组，组间采用直接映射方式，组内采用全相联的映射方式
- Cache分 u 组，组内容量 v 行；映射方法：
 - 组号 $q = j \bmod u$
 - 主存第 j 块内容拷贝到Cache的 q 组中的某行
- 地址变换
 - 设主存地址 x ，看是不是在cache中，先 $y = x \bmod u$ ，则在 y 组中一次查找

组相联映射方式

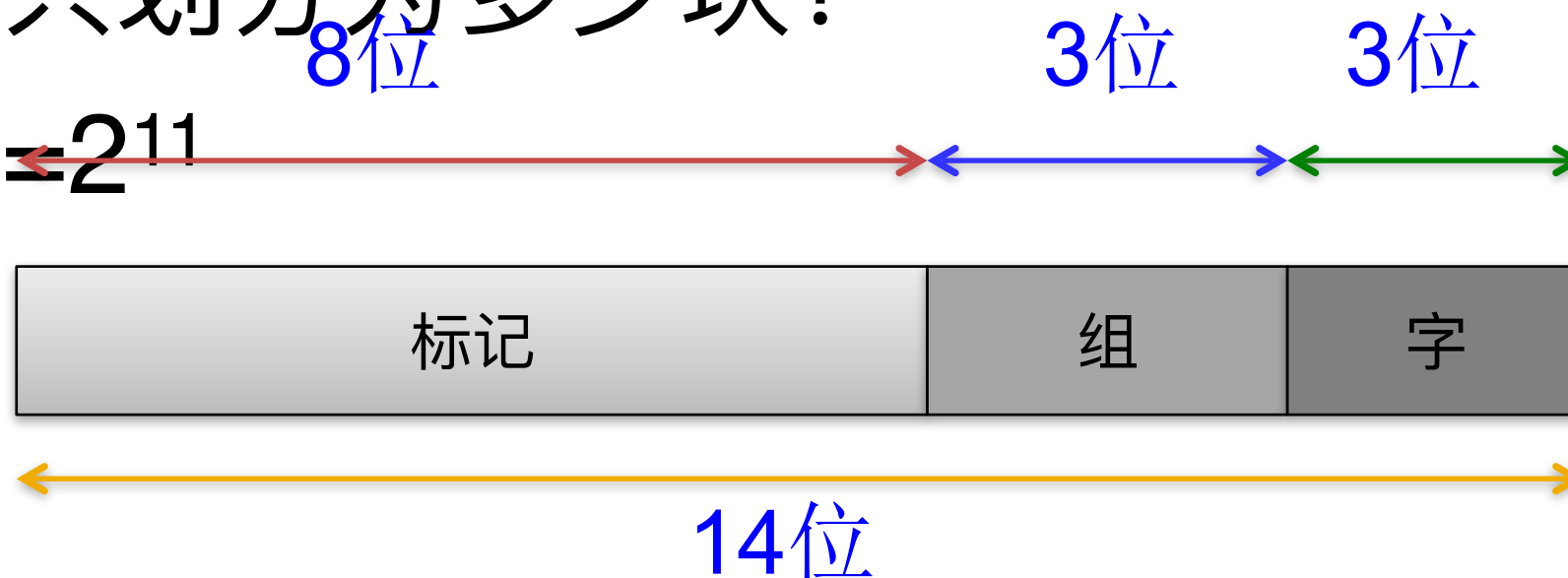


(a) 组相联映射示意图(4组)

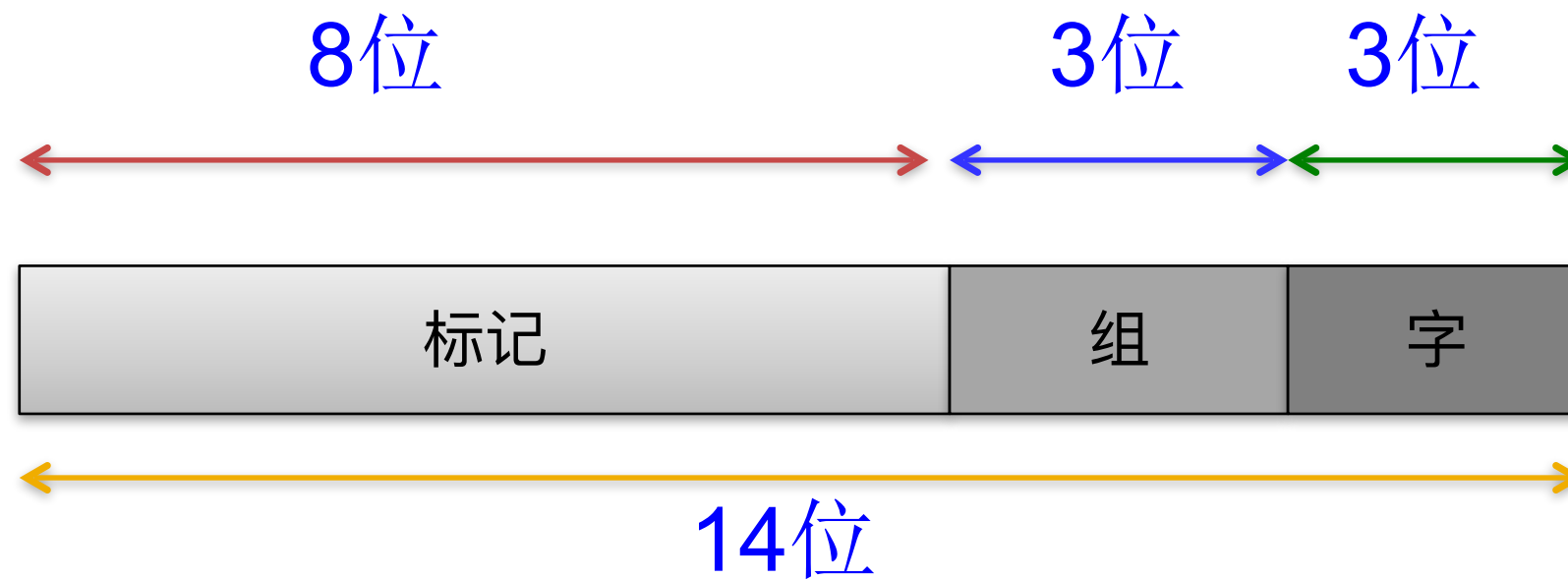
组相联映射方式

- 存储器 2^{14} 字组成
- Cache有16行，分为8组
- 每个块、行由8个字组成
- 存储器地址位数？
 - 14位
- 存储器共划分为多少块？

- $2^{14}/2^3 = 2^{11}$



组相联映射方式



■ 主存地址：

- $d=11011001100011$

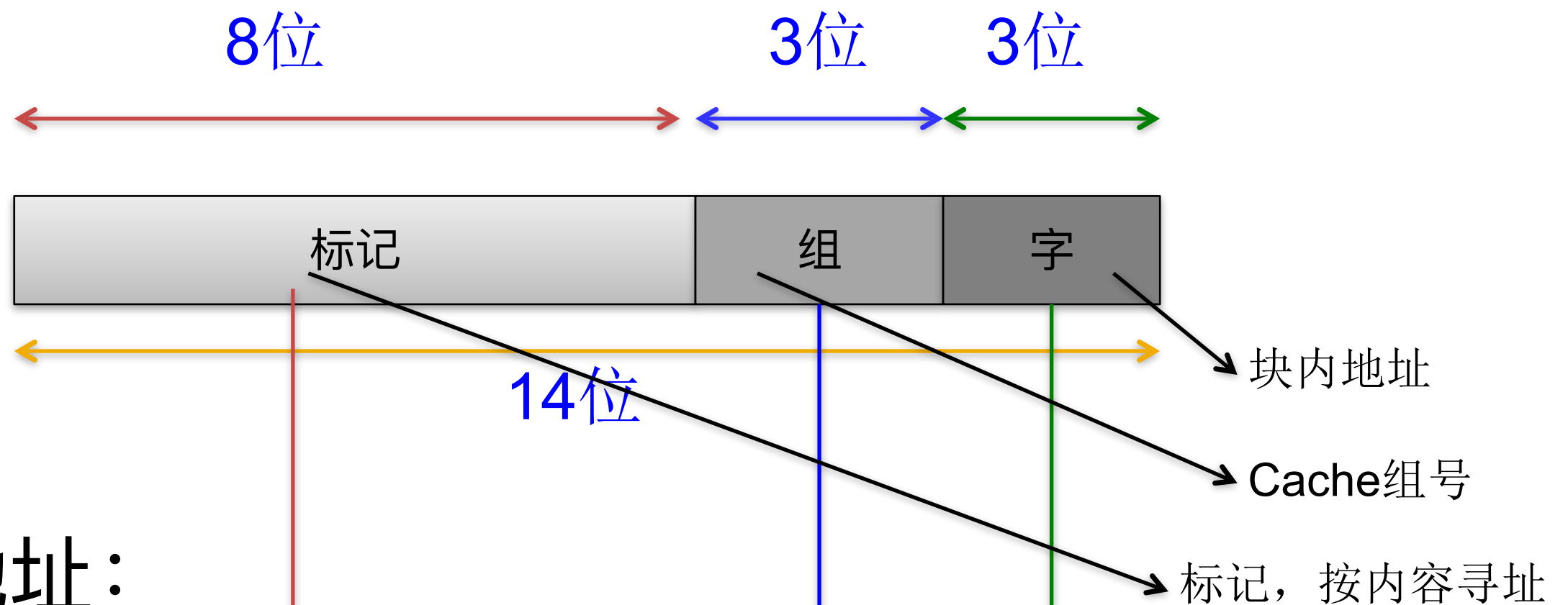
■ 在主存中的块号：

- $j=d \setminus 2^3=11011001100\cancel{011}$ (求商)

■ 映射到cache中的组号：

- $i=j \% 2^3 = \underline{11011001}100\cancel{011}$ (取余)

组相联映射方式



■ 主存地址:

- $d = 11011001100011$

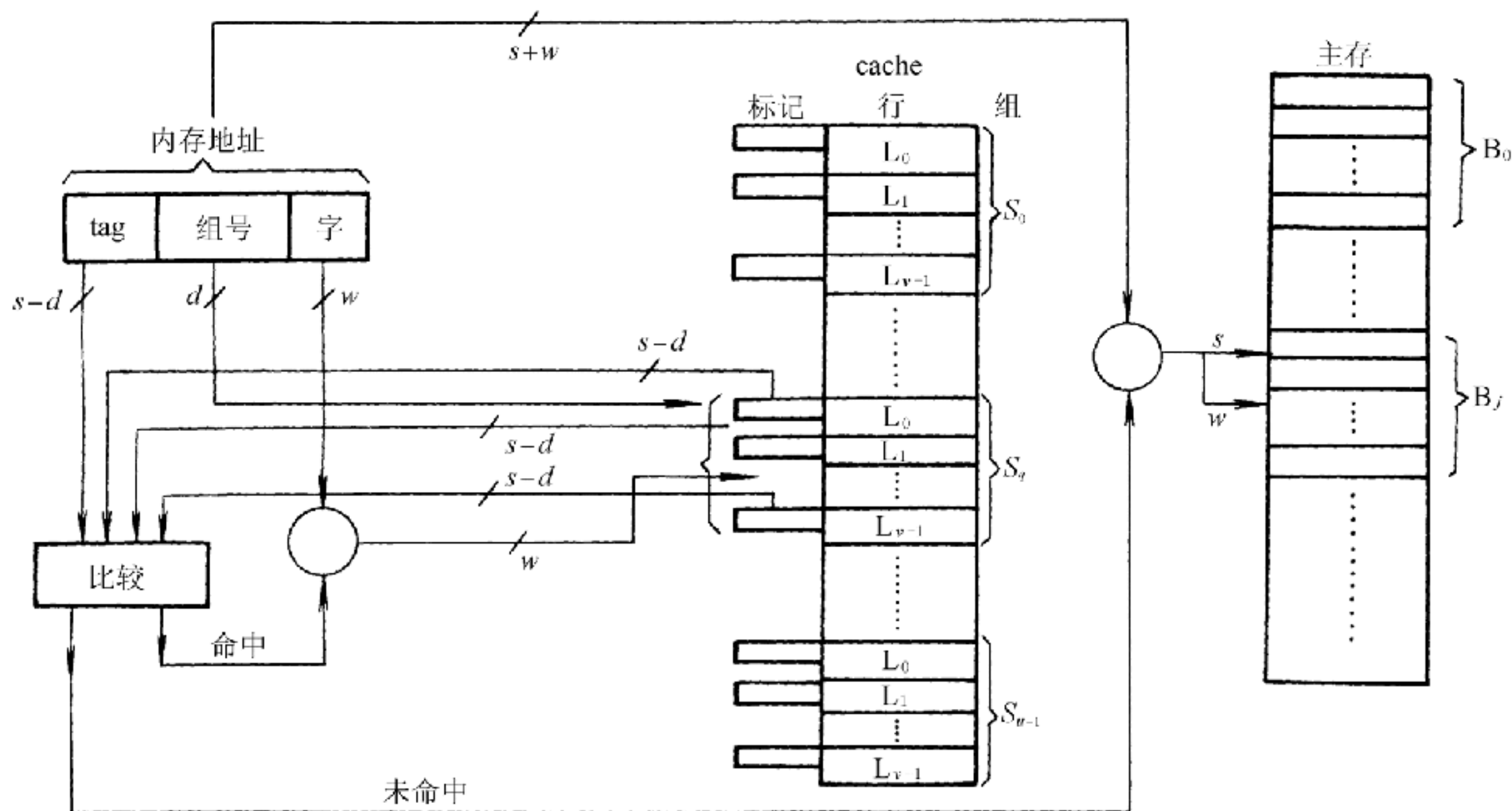
■ 在主存中的块号:

- $j = d \setminus 2^3 = 11011001100011$ (求商)

■ 映射到cache中的组号:

- $i = j \% 2^3 = \underline{11011001}100011$ (取余)

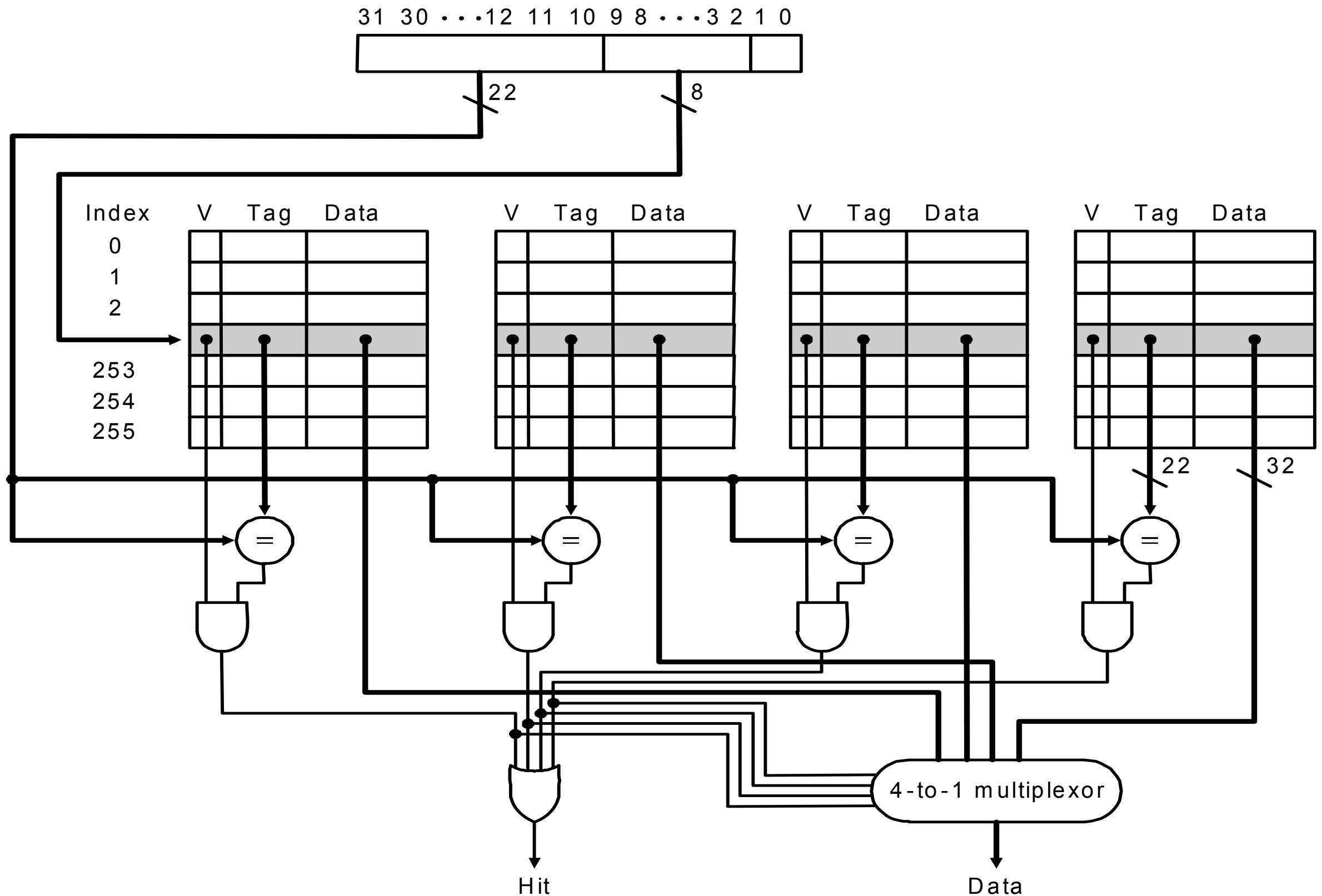
组相联映射方式



(b) 组相联cache的检索过程

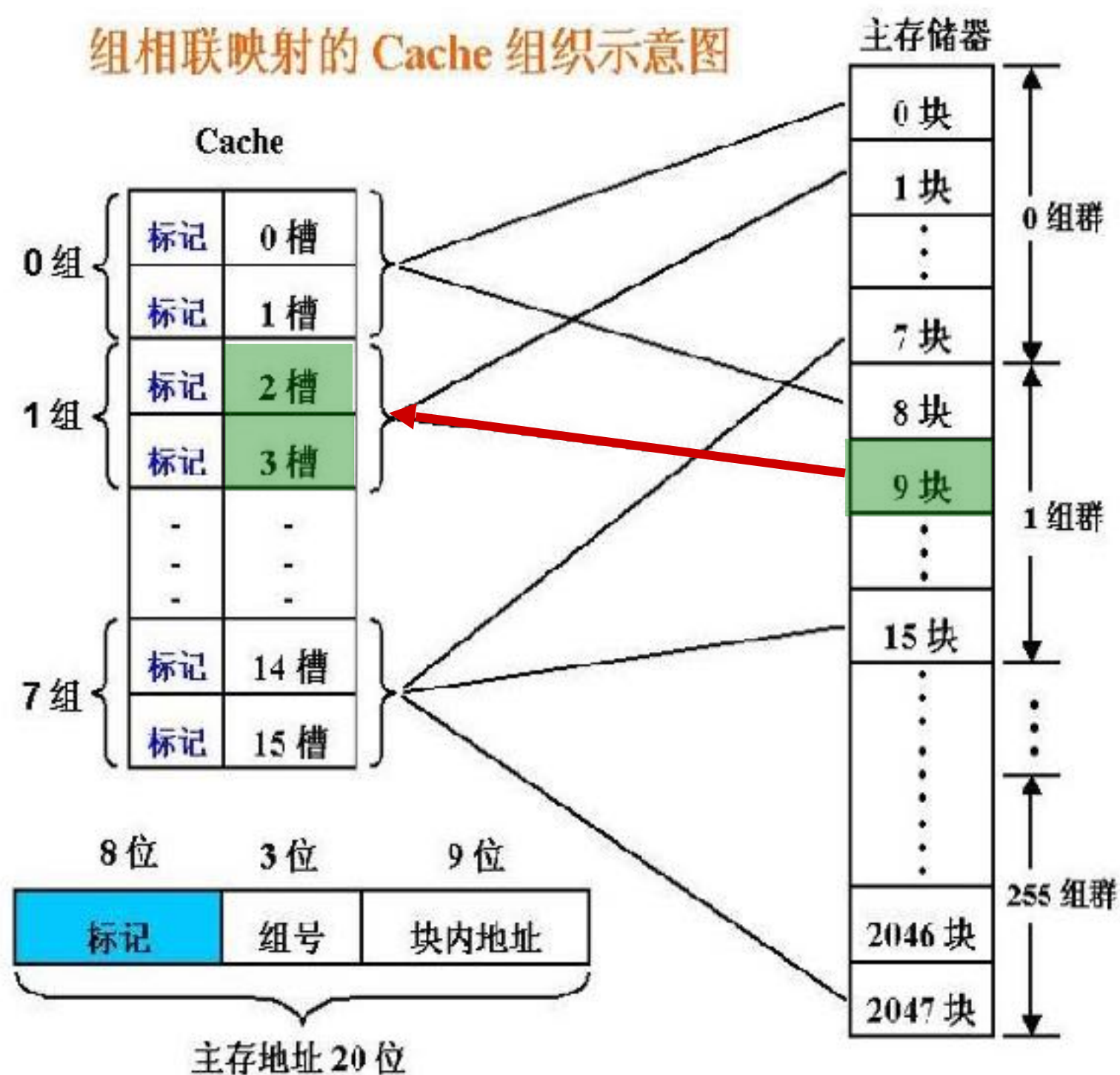
- 首先，用内存地址的块号域的低d位找到cache的对应组；
- 然后，将块号域高s-d位与该组v行中所有标记域进行比较；
- 若有标记相符，则命中，以w位字域部分检索此行具体字；
- 如果未命中，则按内存地址访问主存

组相联映射Cache硬件实现



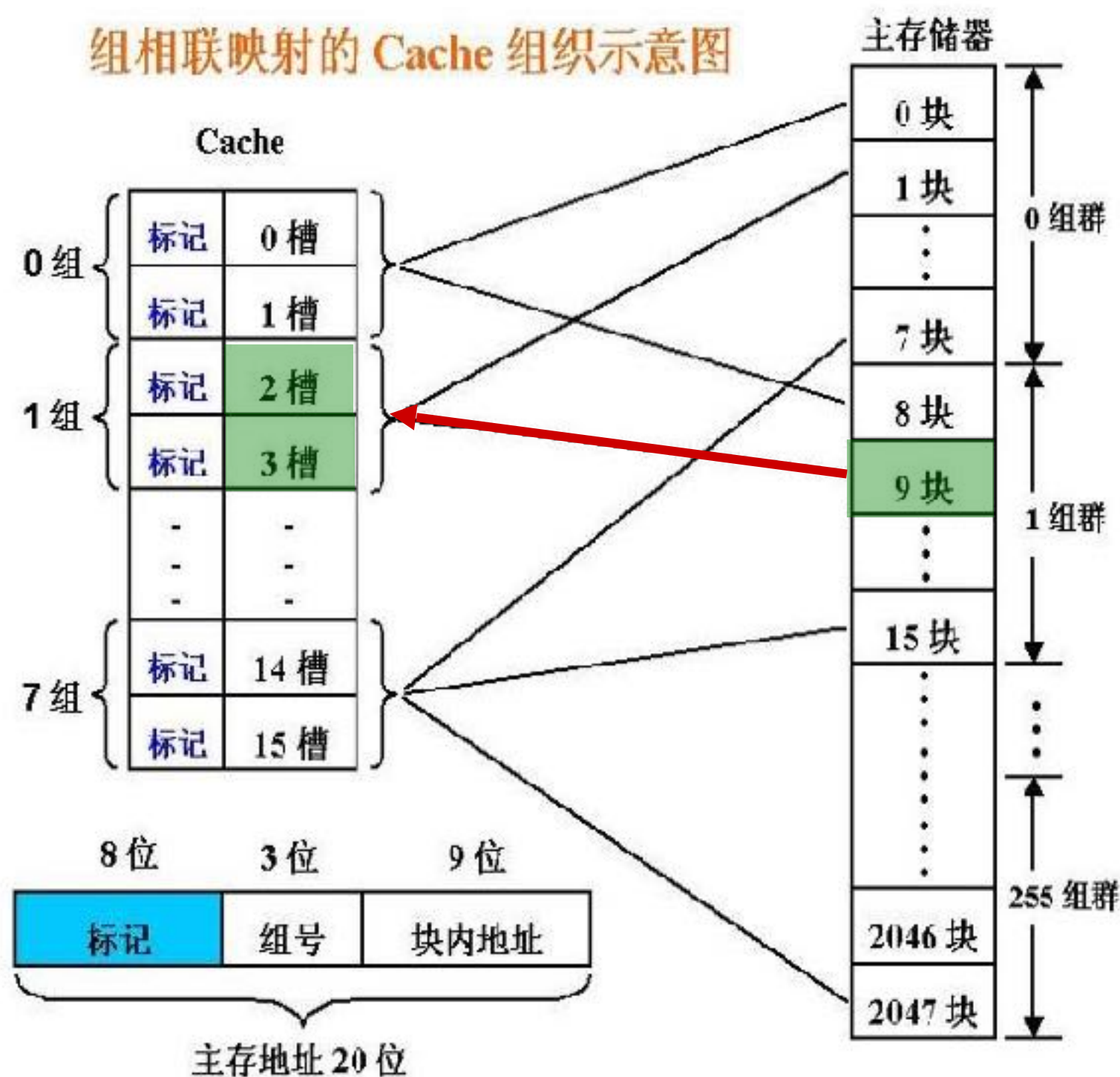
例

- 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
- 如何对0120CH单元进行访问？



例

- 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
- 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。

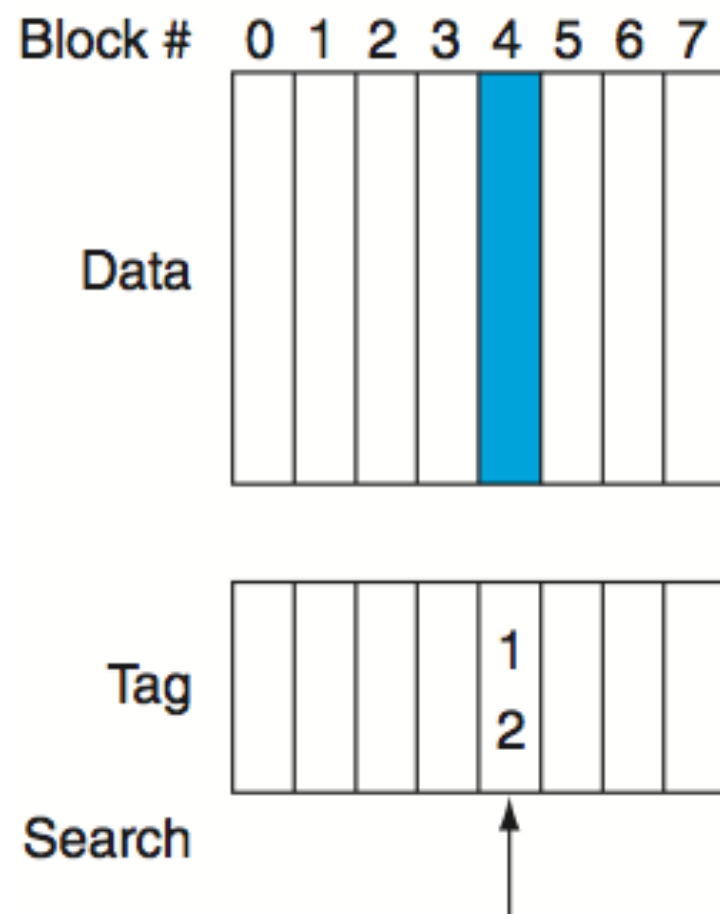
组相联映射方式特点

- 比全相联容易实现，冲突低
- $v=1$ ，则为直接相联映射方式
- $u=1$ ，则为全相联映射方式
- v 的取值一般比较小，一般是2的幂，称之为 v 路组相联cache.

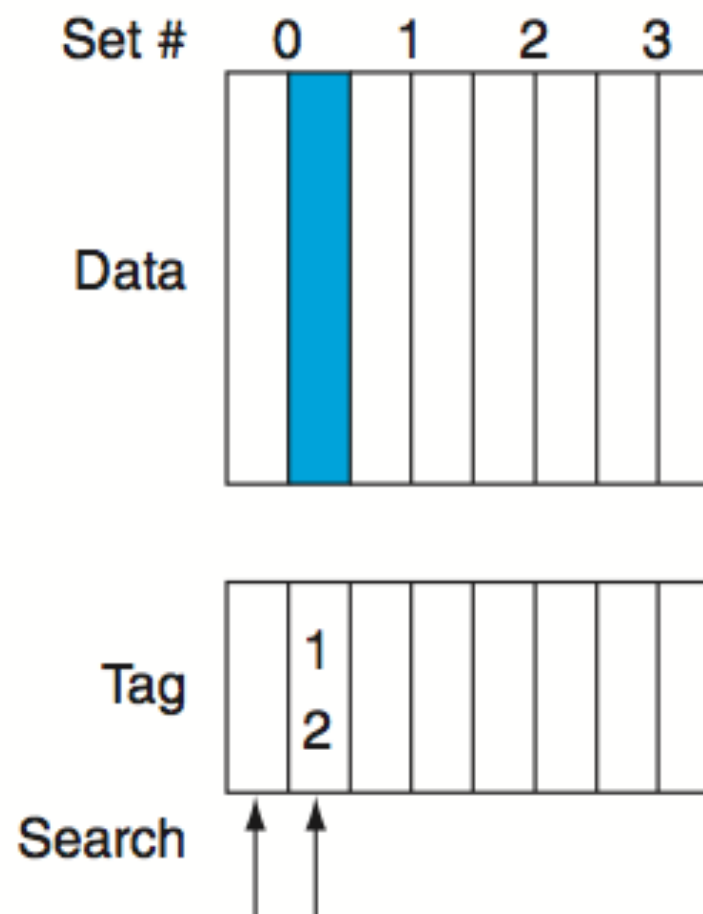
三种映射方式比较

主存块号为12的块在cache中的位置

Direct mapped



Set associative



Fully associative



三种映射方式比较

■ 直接映射

- 主存中的一块只能映射到Cache中唯一的一个位置 定位时,不需要判断,只需替换

■ 全相联映射

- 主存中的一块可以映射到Cache中任何一个位置

■ N路组相联映射

- 主存中的一块可以选择映射到Cache中N个位置

■ 全相联映射和N路组相联映射的失效处理

- 从主存中取出新块
- 为了腾出Cache空间,需要替换出一个Cache块
- 不唯一,则需要判断应替出哪块

8块cache配置成不同映射方式

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

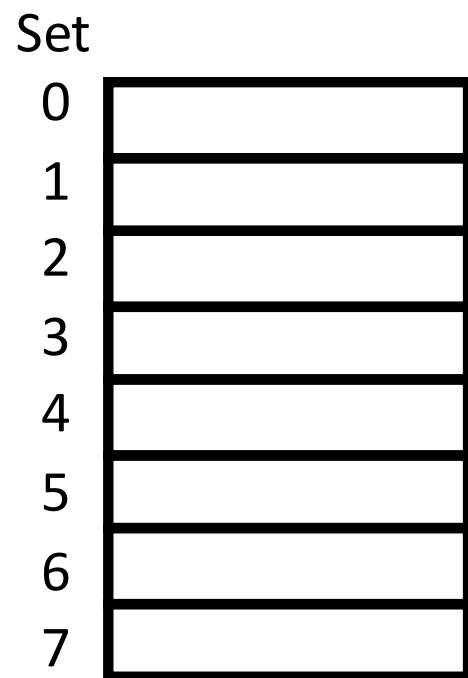
Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

8块cache配置成不同映射方式

1-way

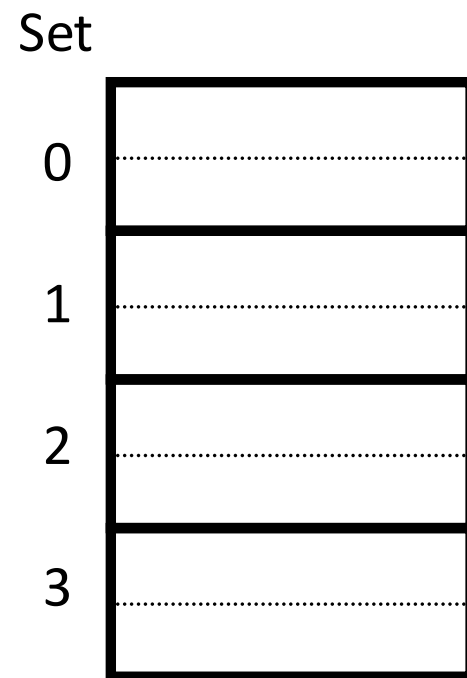
8 sets,
1 block each



direct mapped

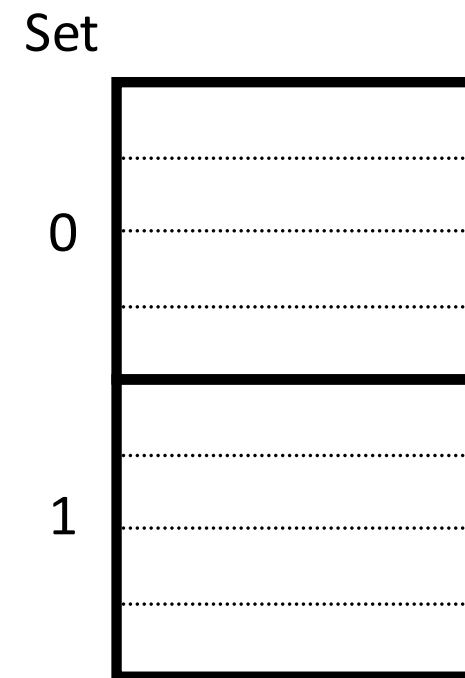
2-way

4 sets,
2 blocks each



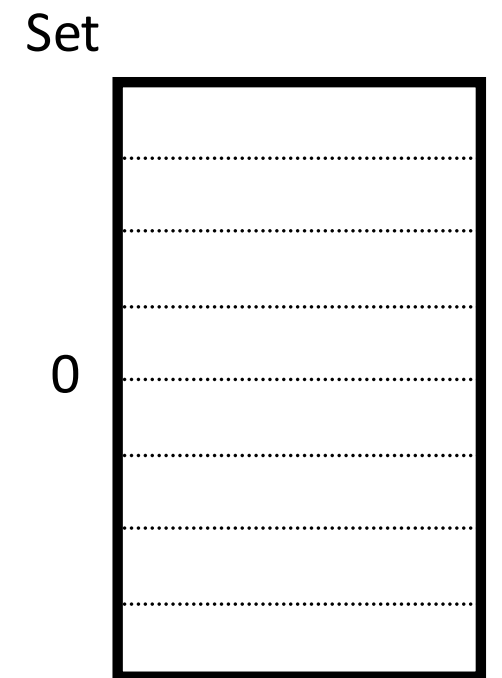
4-way

2 sets,
4 blocks each



8-way

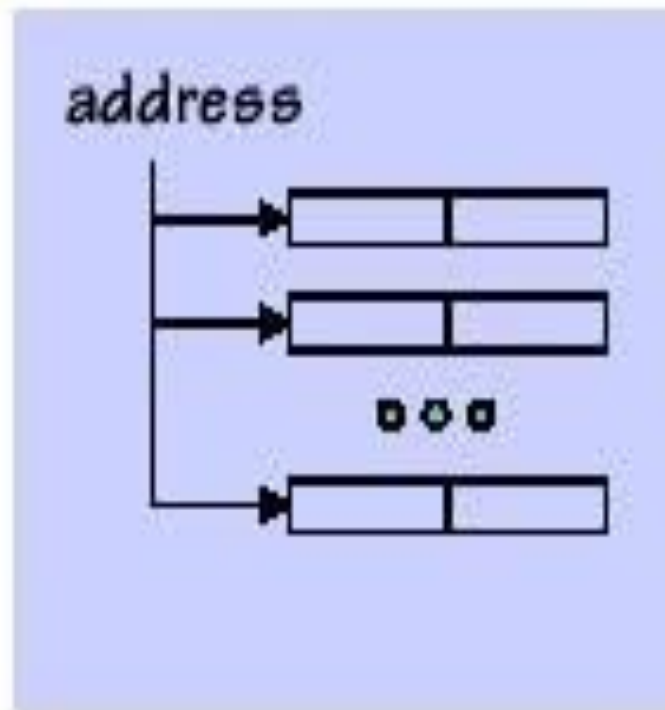
1 set,
8 blocks



fully associative

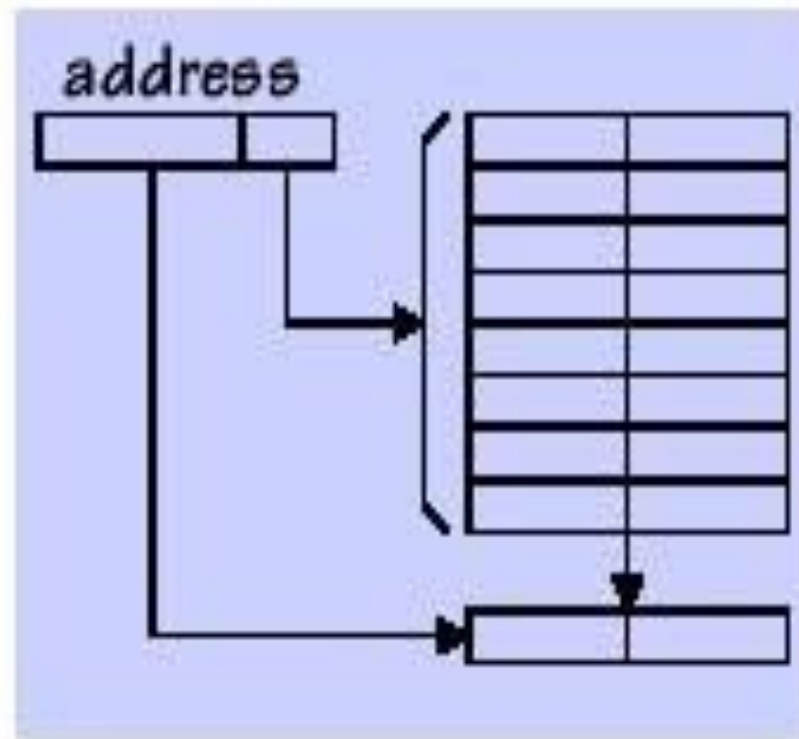
三种映射方式比较

Fully associative



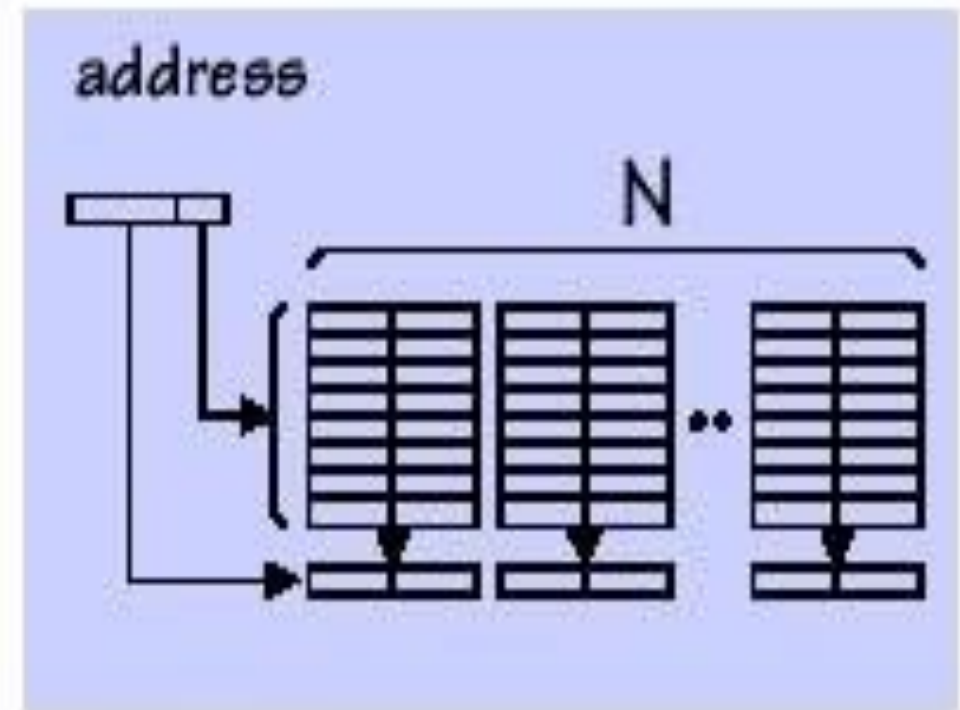
- compare addr with each tag simultaneously
- location A can be stored in any cache line

Direct-mapped



- compare addr with only one tag
- location A can be stored in exactly one cache line

N-way set associative



- compare addr with N tags simultaneously
- location A can be stored in exactly one set, but in any of the N cache lines belonging to that set

思考题

- 某计算机的 Cache 共有 16 行,采用 2 路组相连映射方式。每个主存块大小为 32 字节,按字节编址。主存 129 号单元所在主存块应转入到 Cache 组号是()
- A. 0
- B. 2
- C. 4
- D. 6

思考题

- 某计算机的 Cache 共有 16 行,采用 2 路组相连映射方式。每个主存块大小为 32 字节,按字节编址。主存 129 号单元所在主存块应转入到 Cache 组号是()
- A. 0
- B. 2
- C. 4
- D. 6