

计算机组成原理

(第四讲-3)



厦门大学信息学院软件工程系 曾文华

2021年4月2日



第4章 存储器

共87页

4.1 概述

4.2 主存储器

4.3 高速缓冲存储器

4.4 辅助存储器



4.3 高速缓冲存储器

一、概述

二、**Cache** – 主存的地址映射

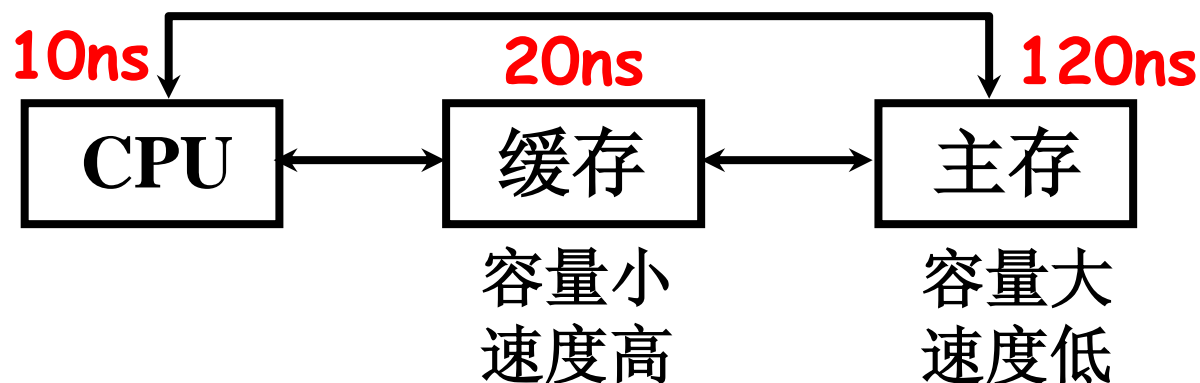
三、替换算法

一、概述

1. 问题的提出

避免 CPU “空等” 现象

CPU 和主存（DRAM）的速度差异

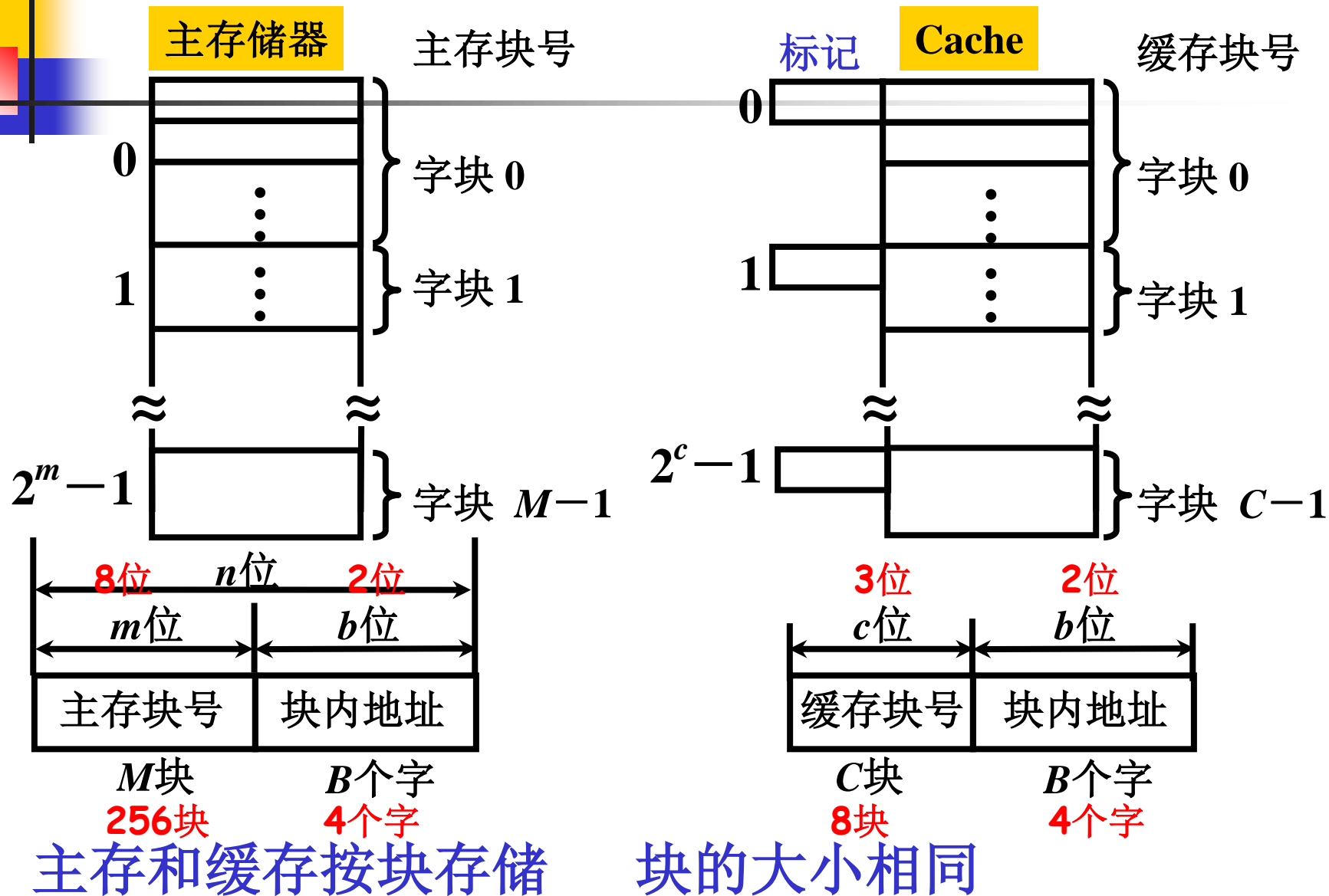


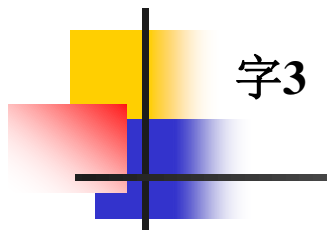
利用程序访问的局部性原理

20/80现象

2. Cache 的工作原理

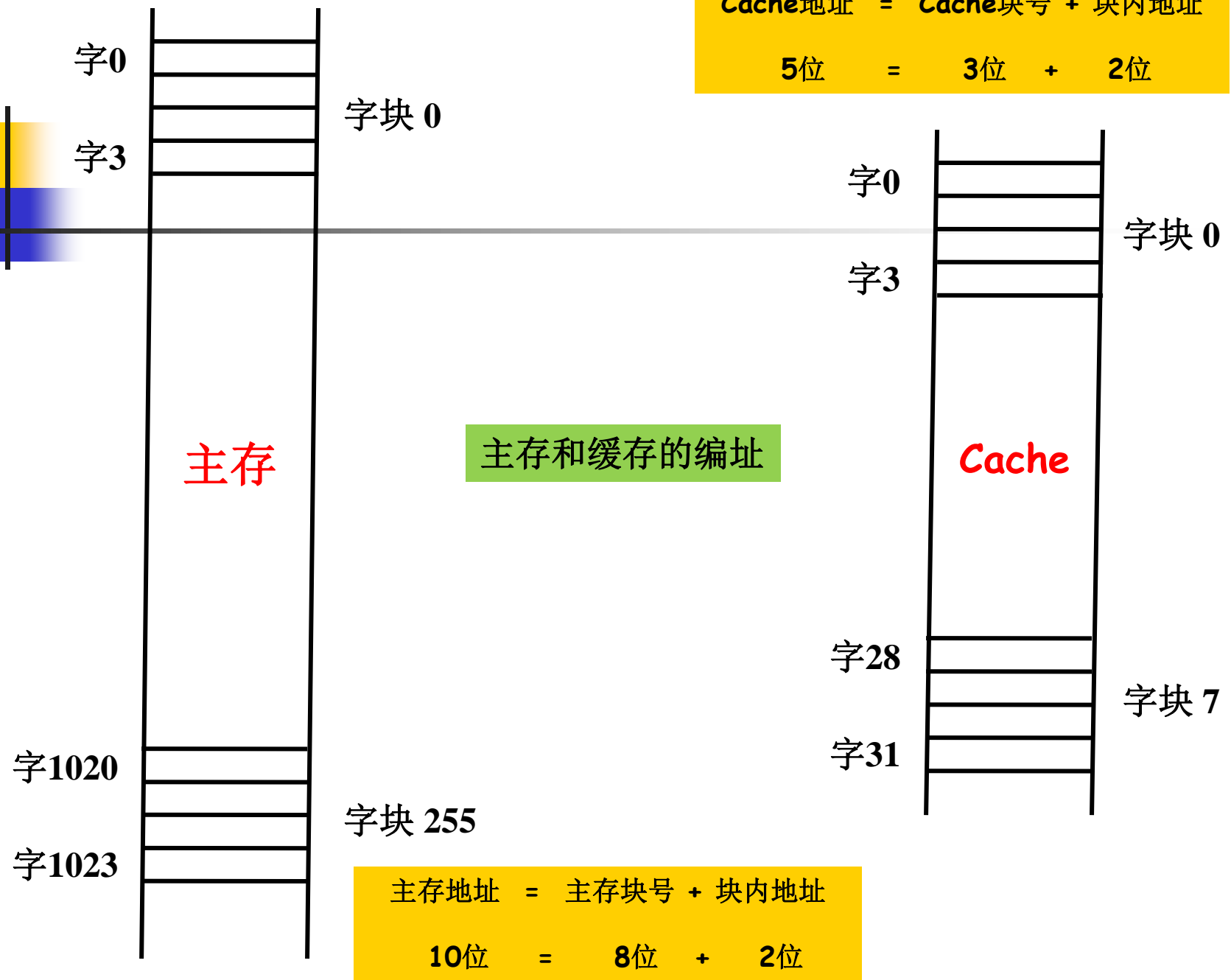
(1) 主存和缓存的编址





Cache地址 = Cache块号 + 块内地址

5位 = 3位 + 2位



(2) 命中与未命中

缓存共有 C 块

主存共有 M 块 $M \gg C$

“标记”记录了缓存
的这个块与主存的那
一个块对应

命中：主存块 已调入 缓存

主存块与缓存块 建立 了对应关系

用 标记记录 与某缓存块建立了对应关系的 主存块号

未命中：主存块 未调入 缓存

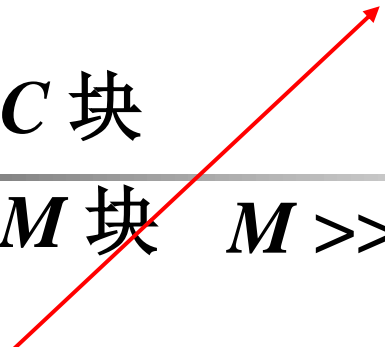
主存块与缓存块 未建立 对应关系

(2) 命中与未命中

缓存共有 C 块

主存共有 M 块 $M \gg C$

“标记”记录了缓存
的这个块与主存的那
一个块对应



所谓命中，即CPU要访问的主存的内容在Cache中

所谓不命中，即CPU要访问的主存的内容不在Cache中，这时CPU要直接访问主存

(3) Cache 的命中率

B=4~8

CPU 欲访问的信息在 Cache 中的 **比率**

命中率与 Cache 的 **容量** 与 **块长** 有关

一般每块可取 4 ~ 8 个字

块长取一个存取周期内从主存调出的信息长度

CRAY_1 16体交叉 块长取 **16** 个存储字

IBM 370/168 4体交叉 块长取 **4** 个存储字

(64位 × 4 = 256位)

(3) Cache 的命中率

B=4~8

CPU 欲访问的信息在 Cache 中的 **比率**

命中率 = 访问**Cache**的次数 / (访问
Cache的次数 + 访问主存的次数)

IBM 5/0/100

4 个交叉

块长取

4 个字节于

(64位 × 4 = 256位)

(4) Cache –主存系统的效率

效率 e 与 命中率 有关，命中率 $h = N_c / (N_c + N_m)$

访问 **Cache** 的
总命中次数

$$e = \frac{\text{访问 Cache 的时间}}{\text{平均访问时间}} \times 100\%$$

访问主存的
总命中次数

设 Cache 命中率为 h ，访问 Cache 的时间为 t_c ，
访问 主存 的时间为 t_m

$$t_m \gg t_c$$

$$\text{则 } e = \frac{t_c}{h \times t_c + (1-h) \times t_m} \times 100\%$$

如果命中率 $h=1$ ，则效率 $e=100\%$

(4) Cache –主存系统的效率

访问Cache的
总命中次数

- 命中率 h =访问Cache的次数 / （访问Cache的次数+访问主存的次数）
- 效率 e =（访问Cache的时间/平均访问时间）X 100%
- 平均访问时间 $t_a=ht_c+(1-h)t_m$
- 访问Cache的时间 t_c
- 访问主存的时间 t_m

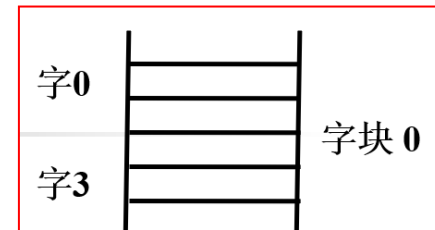
- 
- **例4.7**：假设**CPU**执行某段程序时，共访问**Cache**命中**2,000**次，访问主存**50**次。已知**Cache**的存取周期为**50ns**，主存的存取周期为**200ns**。求**Cache-主存**系统的命中率、效率和平均访问时间。

■ **解**：

- 命中率 $h=2,000/(2,000+50)=0.97$
- 效率 $e=t_c/(ht_c+(1-h)t_m)$, $t_m=4t_c$, $e=91.7\%$
- 平均访问时间 $t_a=ht_c+(1-h)t_m=54.5ns$

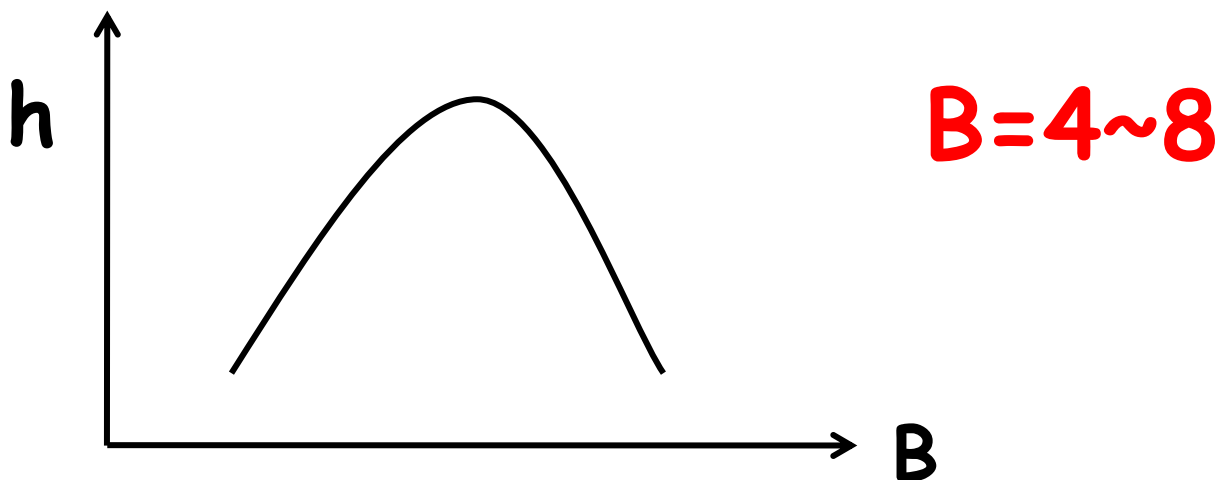
接近于**Cache**的存取周期**50ns**

块长：1个Cache字块包括多少个字



■ 块长**B**与命中率**h**的关系：

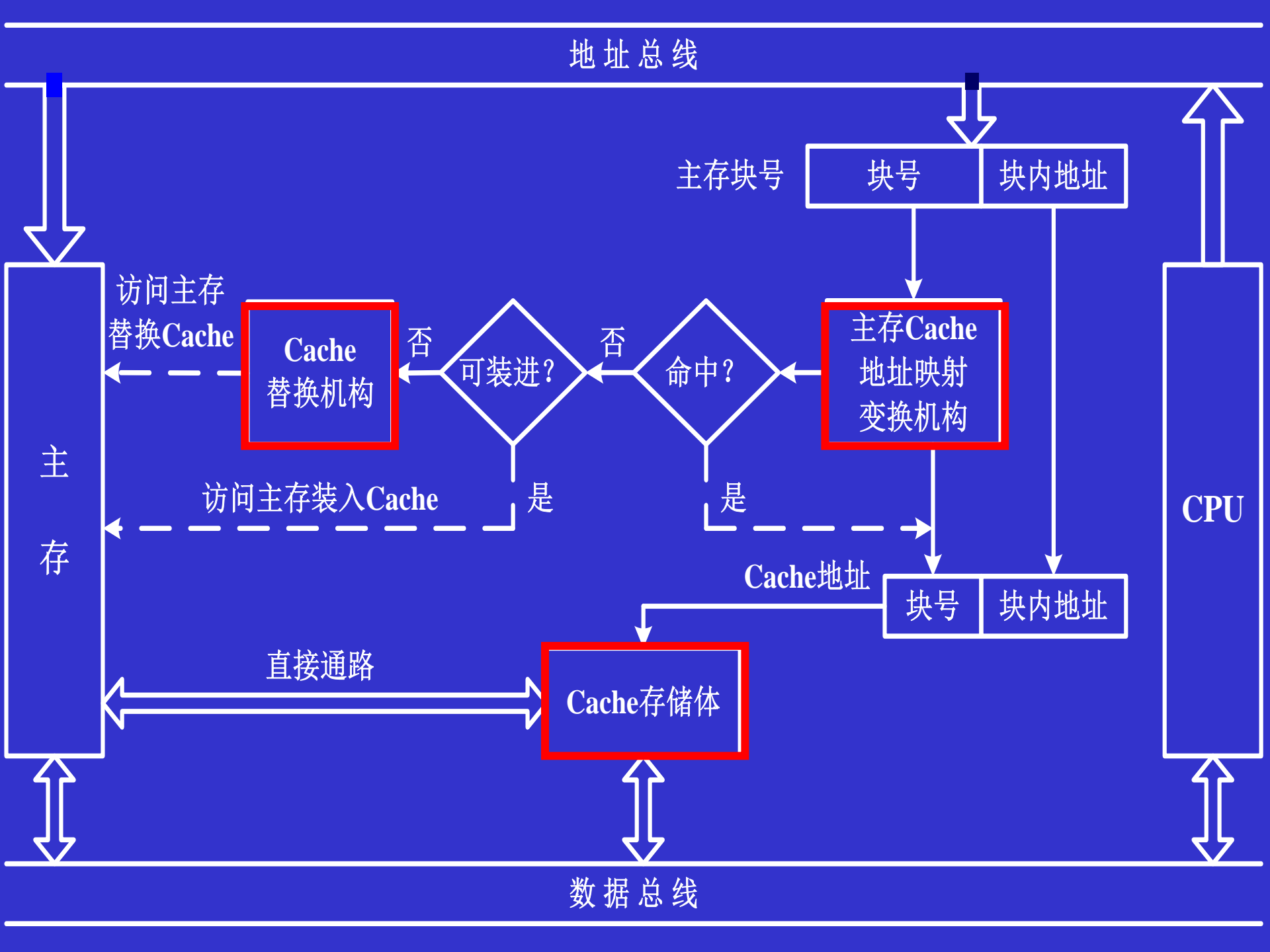
- 块长较小时，增加块长可以提供命中率
- 但是当块长一定大时，再增加块长反而会使命中率下降





3. Cache 的基本结构

- **Cache**主要由**Cache**存储体、地址映射变换机构、**Cache**替换机构等模块组成：
 - **Cache存储体**：以块（如**1块=4个字**）为单位与主存交换信息
 - **地址映射变换机构**：将**CPU**送来的主存地址转换为**Cache**地址
 - **Cache替换机构**：当**Cache**内容已满，无法接受来自主存块的信息时，就由**Cache**内的替换机构按一定的替换算法来确定应从**Cache**内移出哪个块返回主存，而把新的主存块调入**Cache**

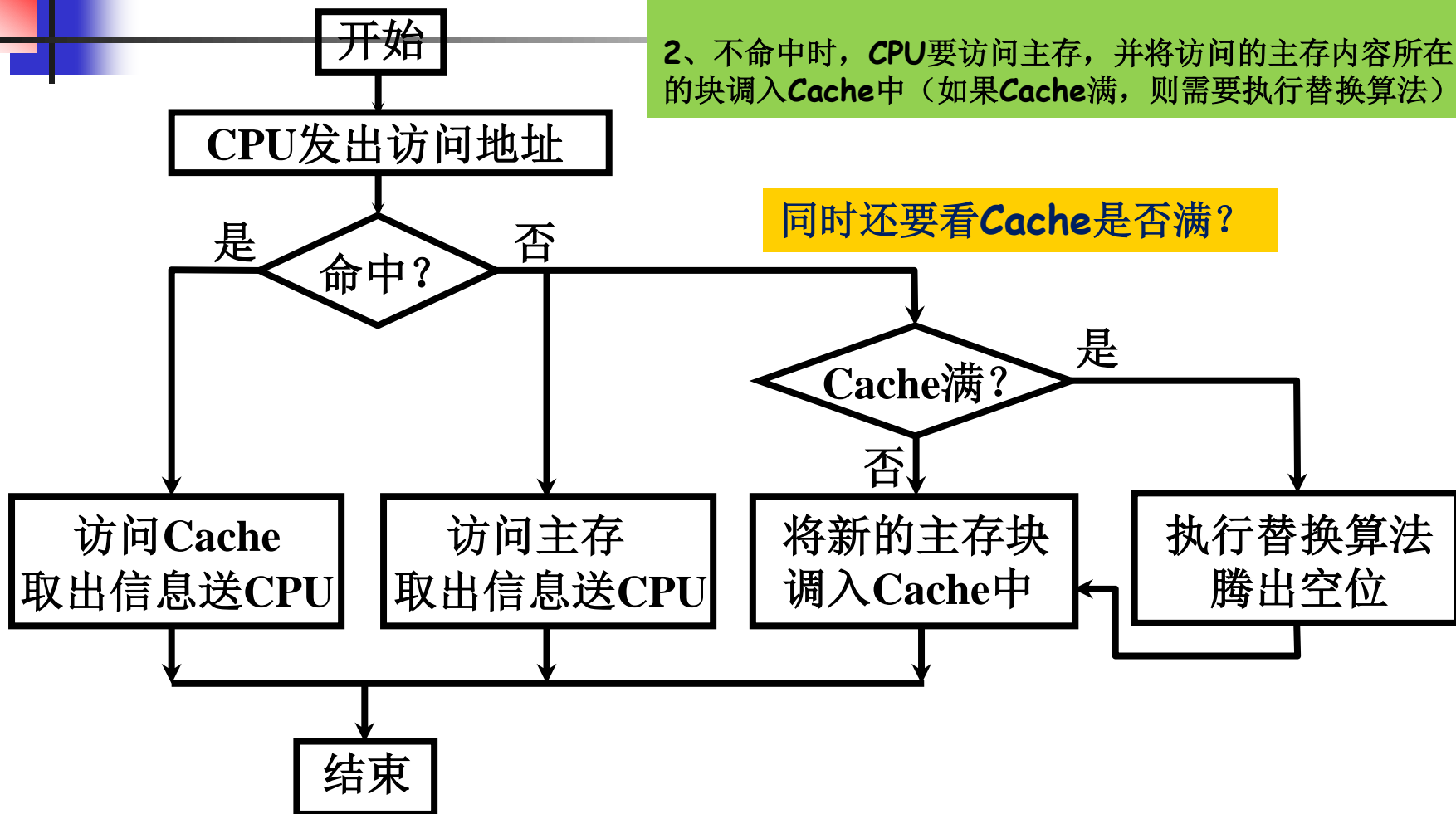


Cache 的 读/写 操作

读

1、CPU发出的访问地址是主存的地址（如果命中，则CPU访问Cache，并需要进行地址变换）

2、不命中时，CPU要访问主存，并将访问的主存内容所在的块调入Cache中（如果Cache满，则需要执行替换算法）



Cache 的 读/写 操作 写

要保证Cache 和主存的一致性

存直达法

(Store-through)

- 写直达法 (Write-through)

写操作时数据既写入Cache又写入主存

写操作时间就是访问主存的时间，读操作时不涉及对主存的写操作，更新策略比较容易实现

- 写回法 (Write-back)

拷回法

(Copy-back)

写操作时只把数据写入 Cache 而不写入主存
当 Cache 数据被替换出去时才写回主存

写操作时间就是访问 Cache 的时间，

读操作 Cache 失效发生数据替换时，

被替换的块需写回主存，增加了 Cache 的复杂性

4. Cache 的改进

(1) 增加 Cache 的级数(单一缓存和两级缓存)

片载（片内）Cache

片外 Cache

(2) 统一缓存和分立缓存（分开）

指令 Cache 数据 Cache

与主存结构有关

与指令执行的控制方式有关 是否流水

Pentium 8K 指令 Cache 8K 数据 Cache

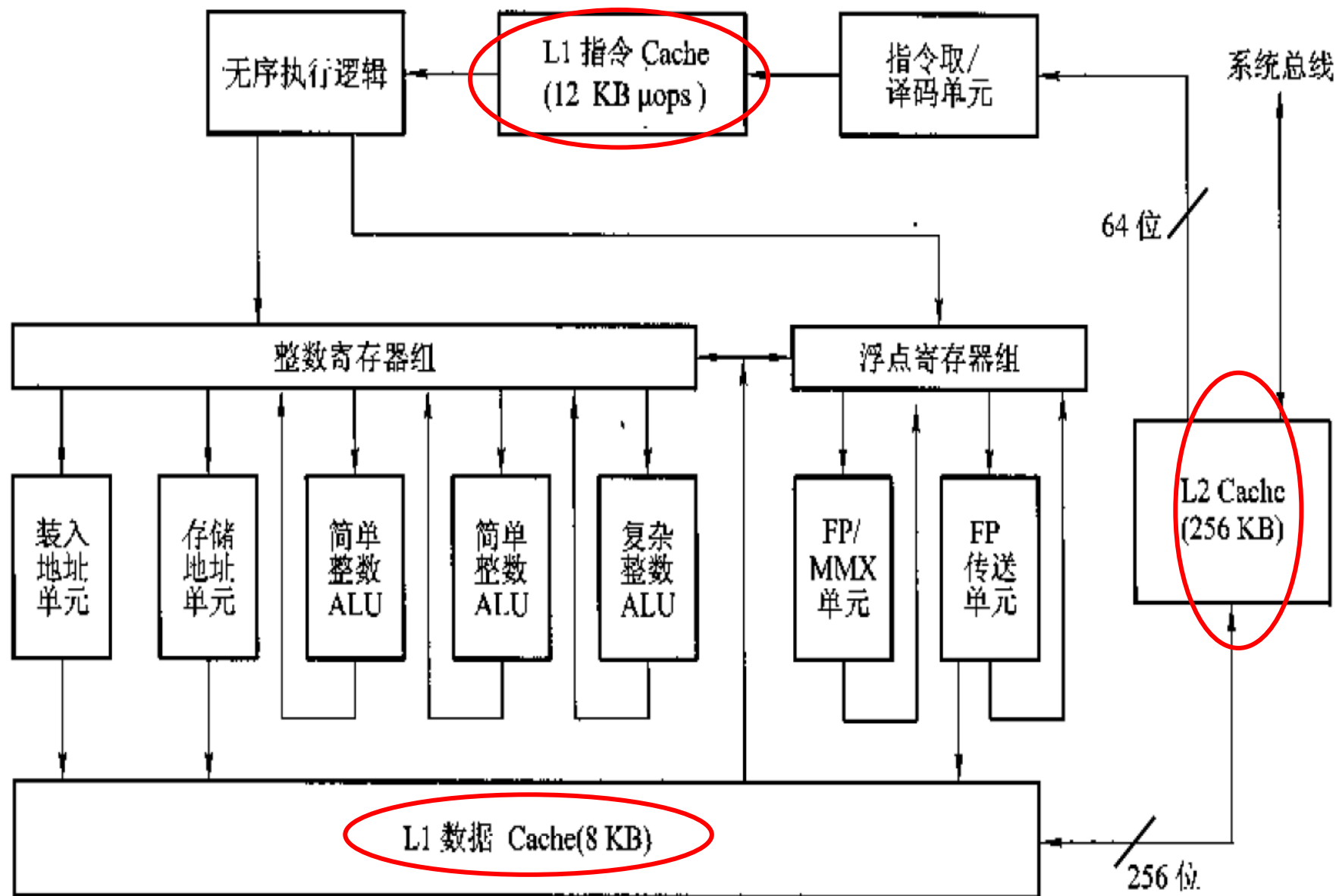
PowerPC620 32K 指令 Cache 32K 数据 Cache

- 
- **Pentium 4处理器的Cache:** 共有两级3个**Cache**; 其中, 一级**Cache**分**L1指令Cache**和**L1数据Cache**, 另外还有一个二级**L2 Cache**

图4.52

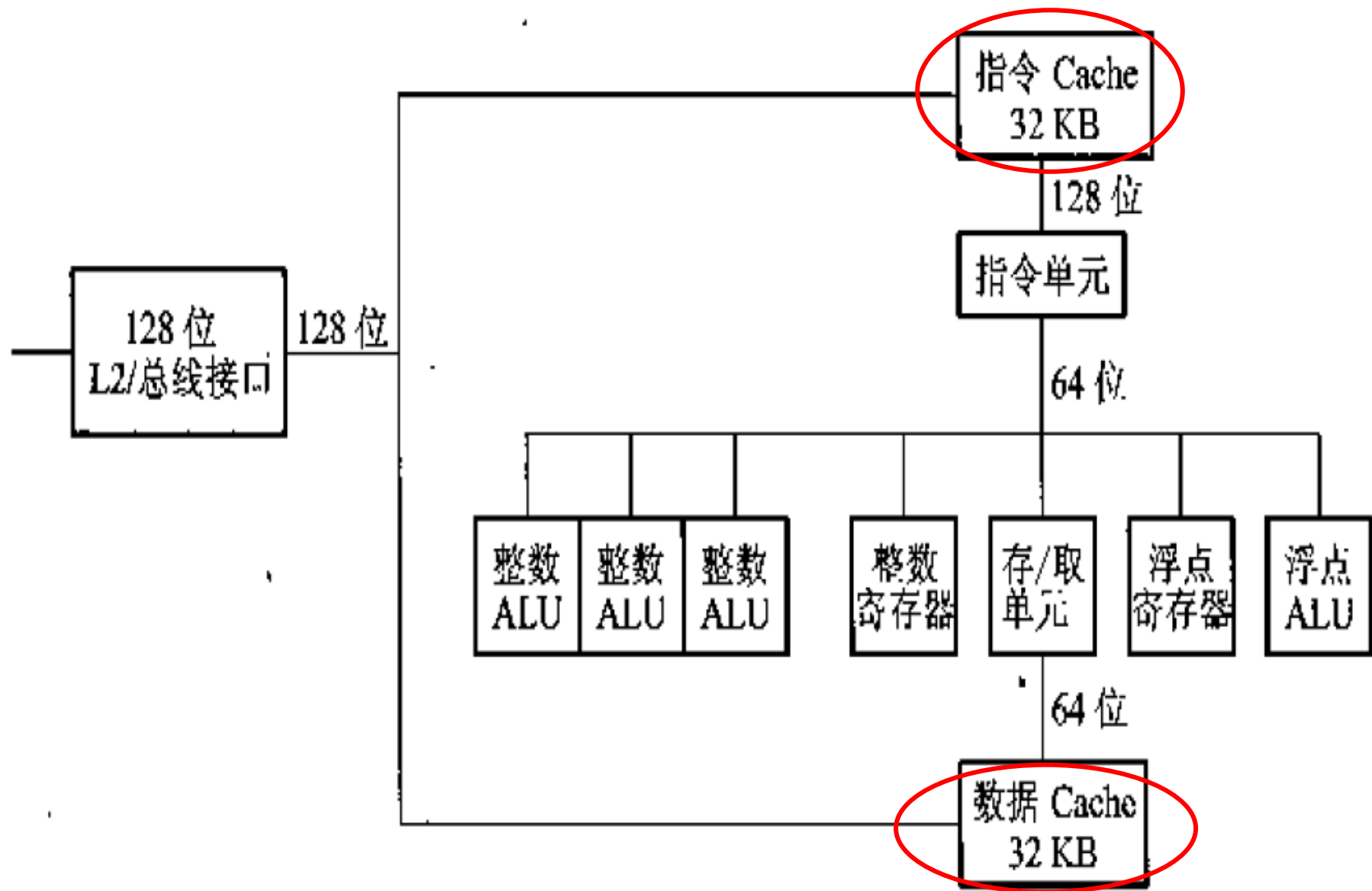
- **PowerPC 620处理器的Cache:** **指令Cache**和**数据Cache**

图4.53



两级3个Cache

图 4.52 Pentium 4 处理器框图



2个Cache

图 4.53 PowerPC 620 处理器框图



二、Cache – 主存的地址映射

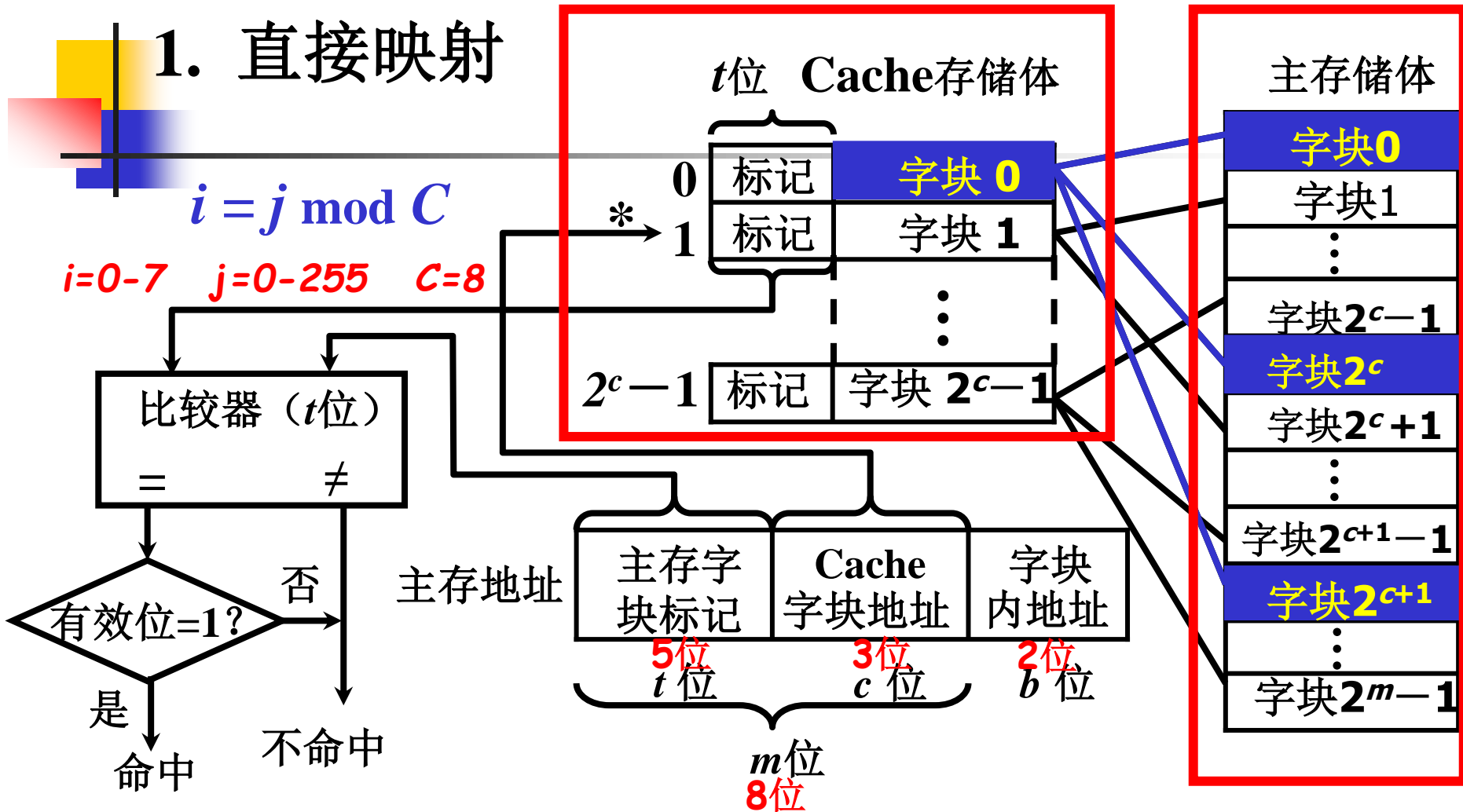
- 地址映射：由主存地址映射到Cache地址称为地址映射
- 直接映射（固定的映射关系）
- 全相联映射（灵活性大的映射关系）
- 组相联映射（直接映射与全相联映射的折中）

二、Cache – 主存的地址映射（续）

1. 直接映射

$$i = j \bmod C$$

$$i=0-7 \quad j=0-255 \quad C=8$$



每个缓存块 i 可以和 若干 ($32=2^5$) 个 主存块 对应

每个主存块 j 只能和 一个 缓存块 对应

Cache地址 = Cache块号 (Cache字块地址) + 块内地址
5位 = 3位 + 2位

字0

字3

字28

字31

字块 0

字块 7

字块 248

字块 255

字0

字3

字28

字31

字块 0

字块 7

主存

直接映射

Cache

字992

字999

字1020

字1023

主存地址 = 主存块号 + 块内地址 = 主存字块标志 + Cache字块地址 + 块内地址
10位 = 8位 + 2位 = 5位 + 3位 + 2位

$$\begin{aligned} \text{Cache地址} &= \text{Cache块号 (Cache字块地址)} + \text{块内地址} \\ 5\text{位} &= 3\text{位} + 2\text{位} \end{aligned}$$

字0

字3

字28

字31

字块 0

字块 7

字块 248

字块 255

字0

字3

字28

字31

字块 0

字块 7

主存

直接映射

Cache

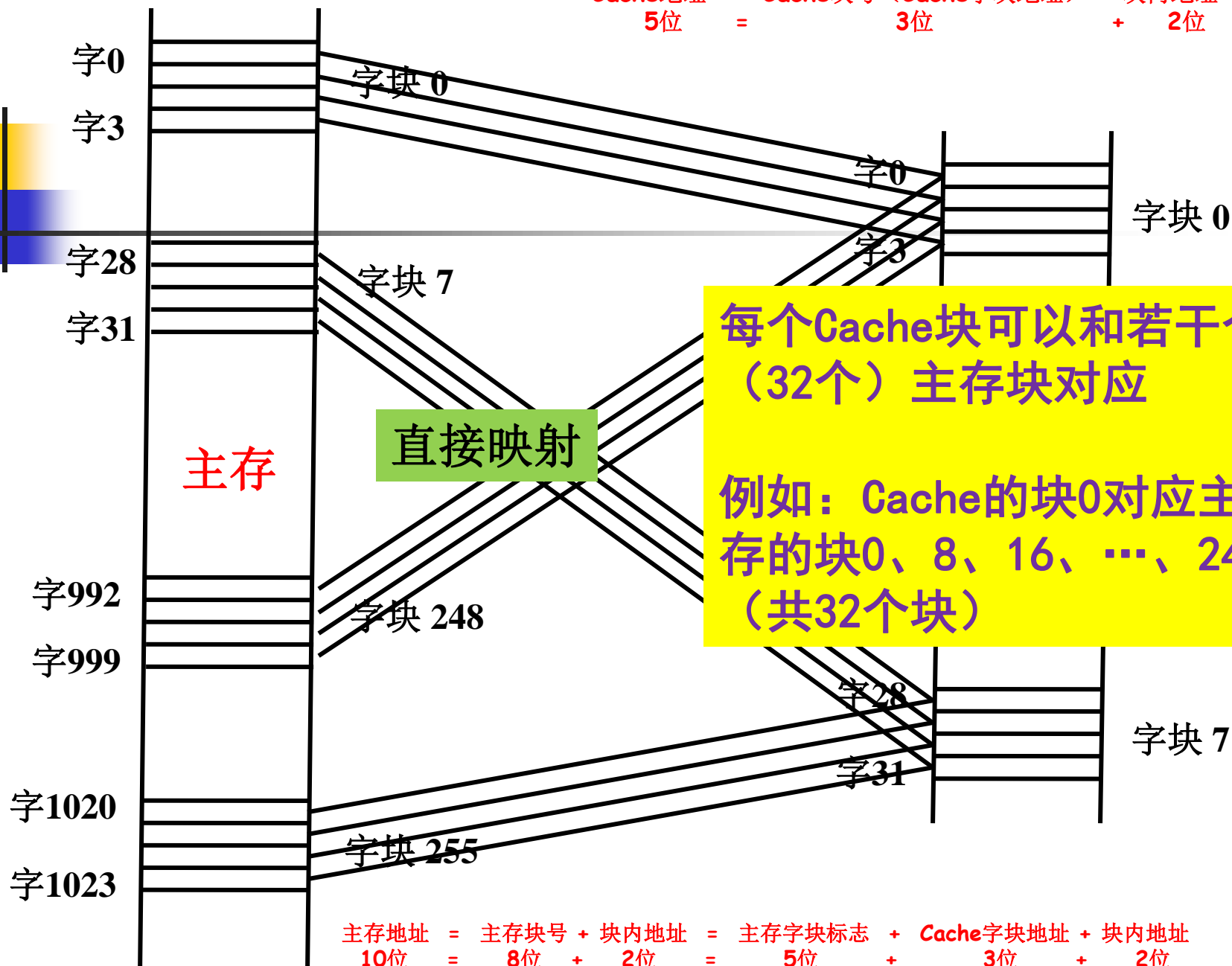
每个主存块只能和一个
Cache块对应

字1020

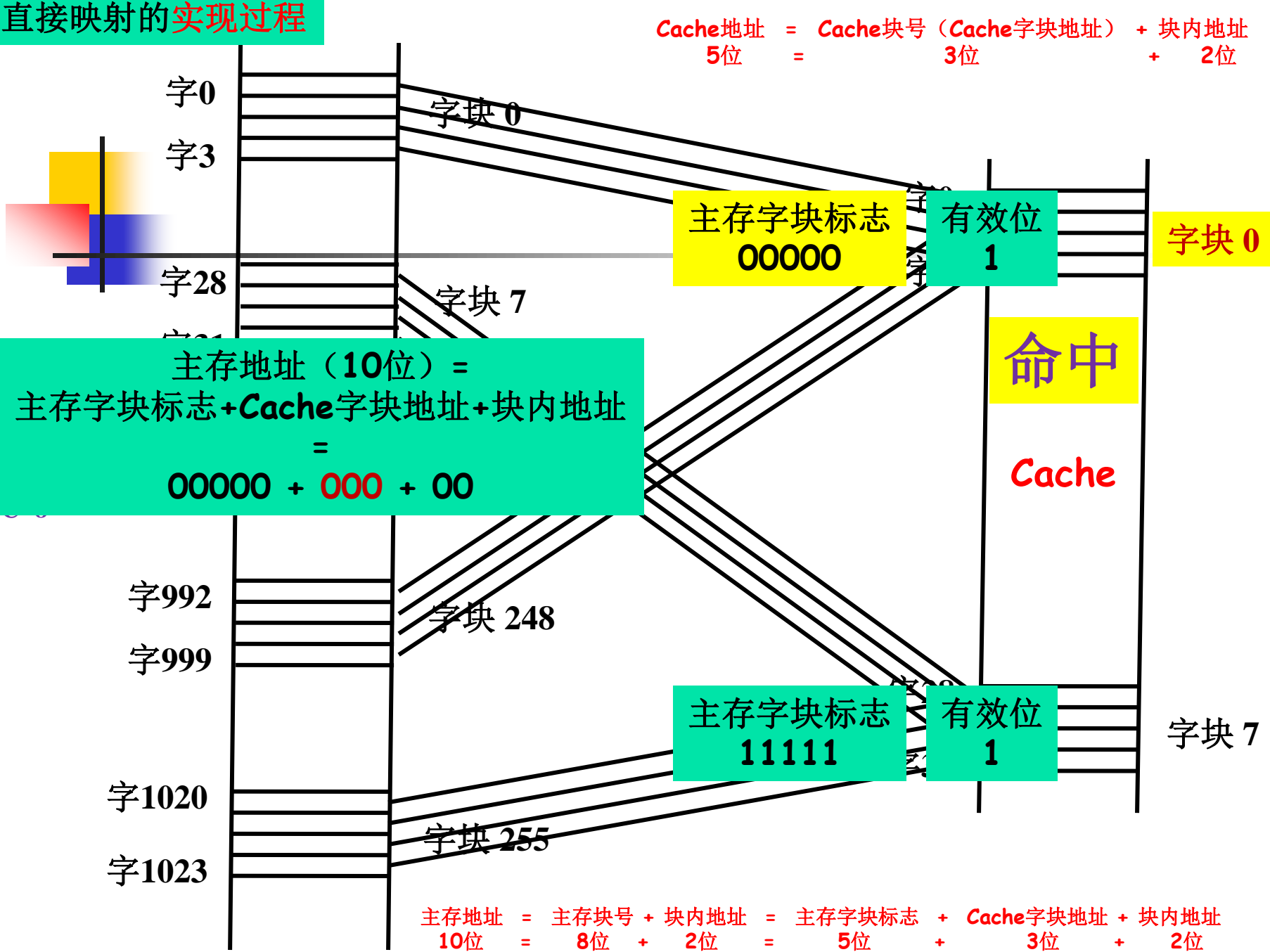
字1023

$$\begin{aligned} \text{主存地址} &= \text{主存块号} + \text{块内地址} = \text{主存字块标志} + \text{Cache字块地址} + \text{块内地址} \\ 10\text{位} &= 8\text{位} + 2\text{位} = 5\text{位} + 3\text{位} + 2\text{位} \end{aligned}$$

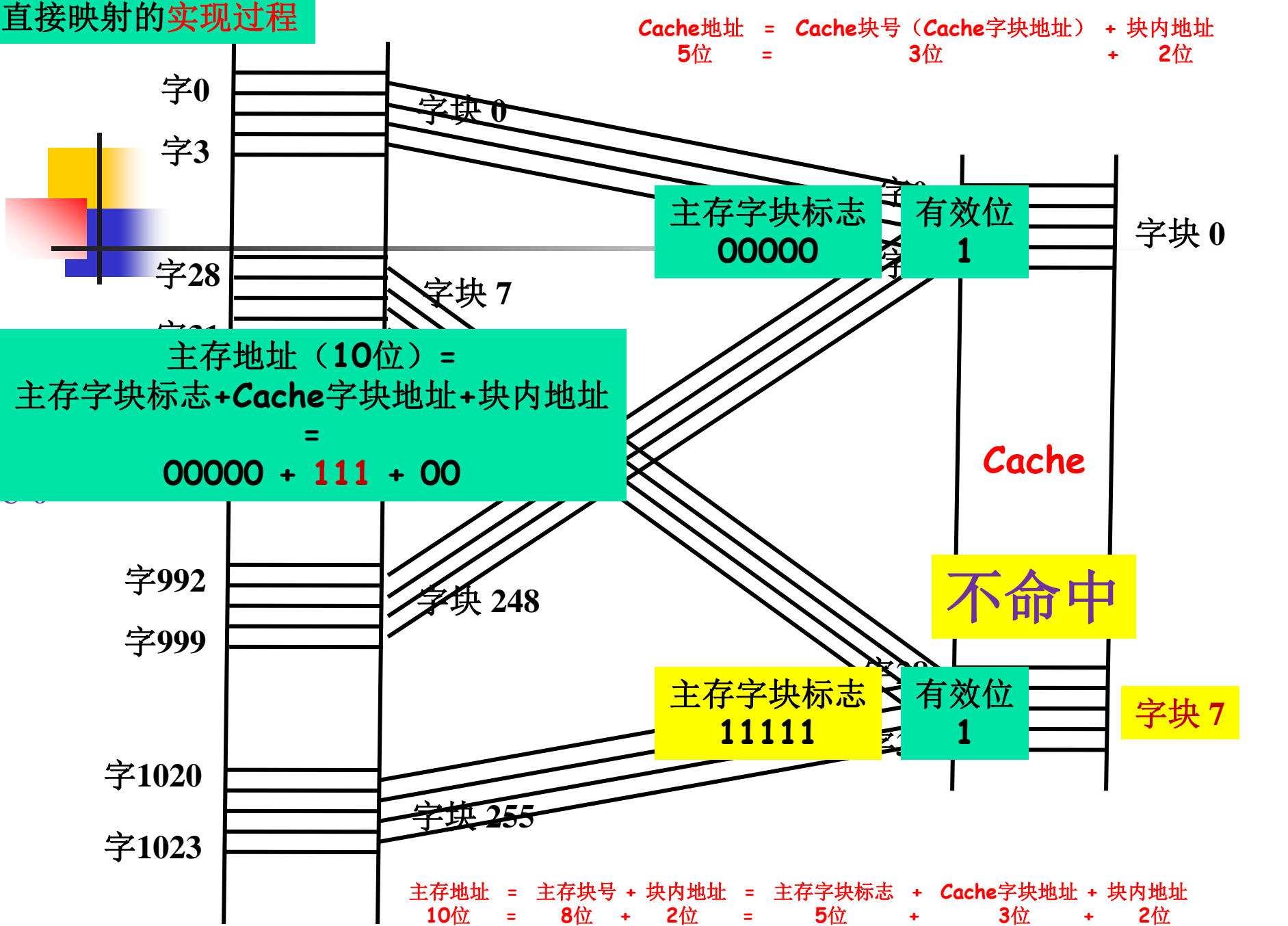
$$\begin{aligned} \text{Cache地址} &= \text{Cache块号 (Cache字块地址)} + \text{块内地址} \\ 5\text{位} &= 3\text{位} + 2\text{位} \end{aligned}$$



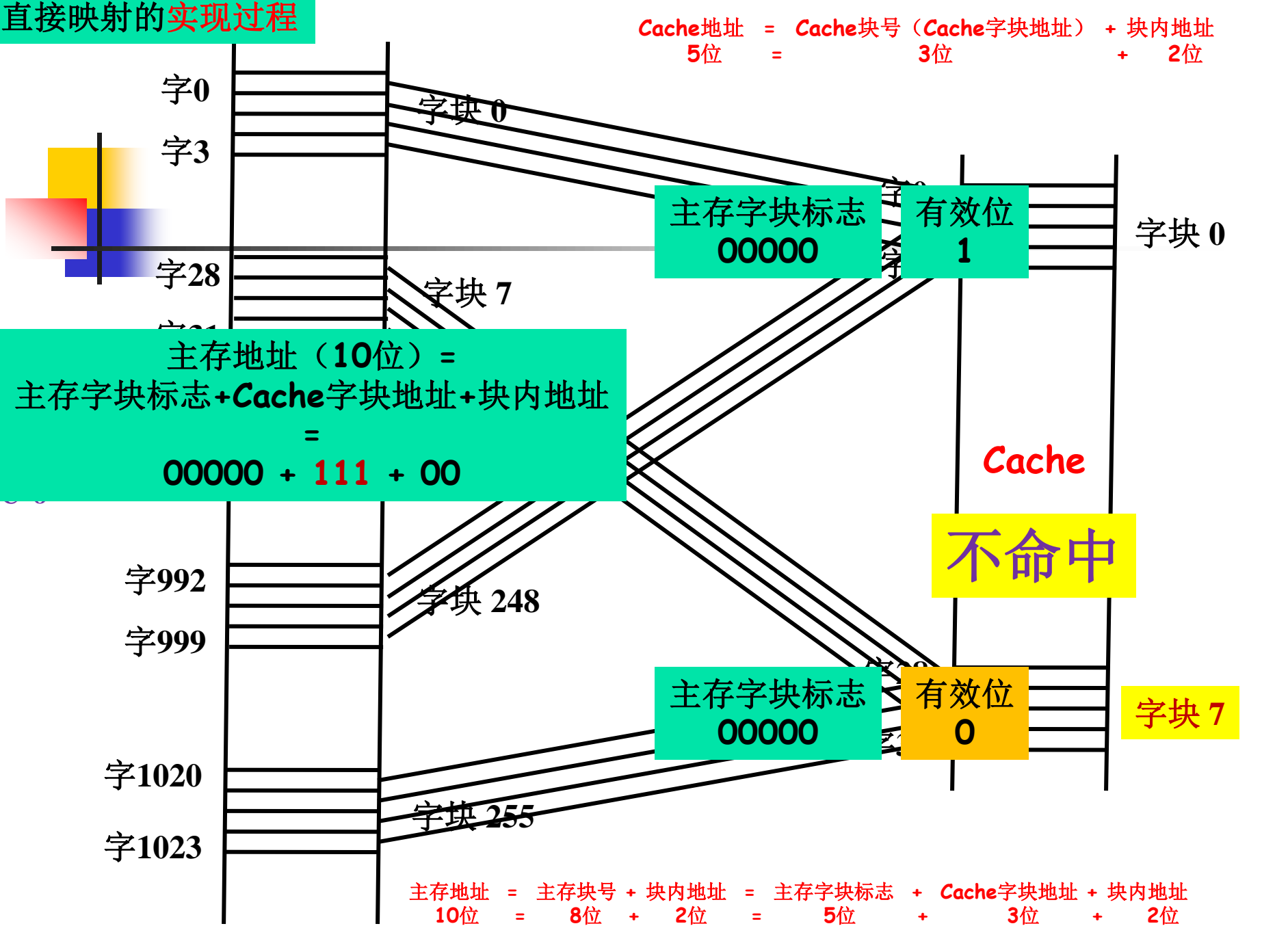
直接映射的实现过程



直接映射的实现过程



直接映射的实现过程



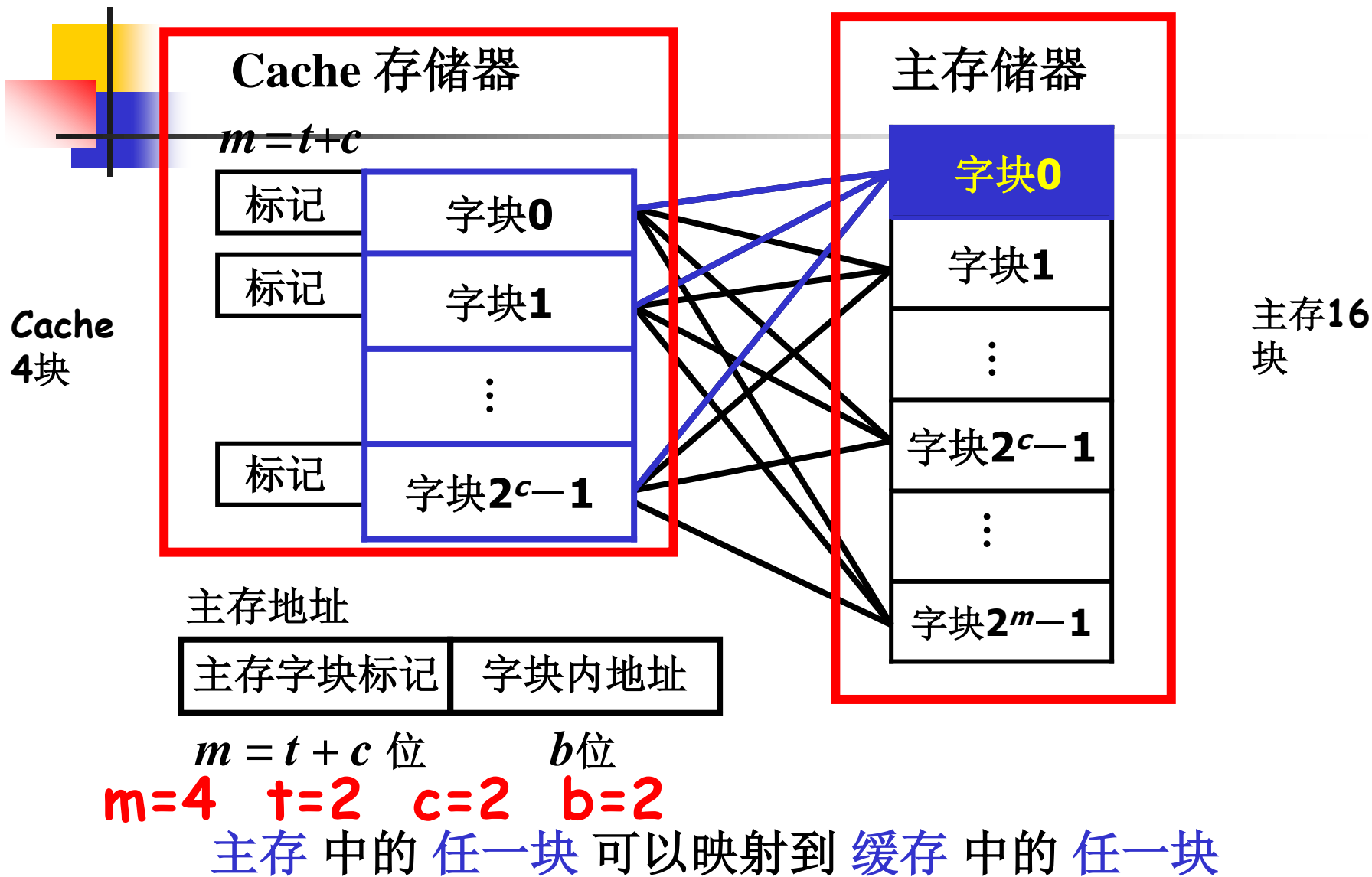


直接映射

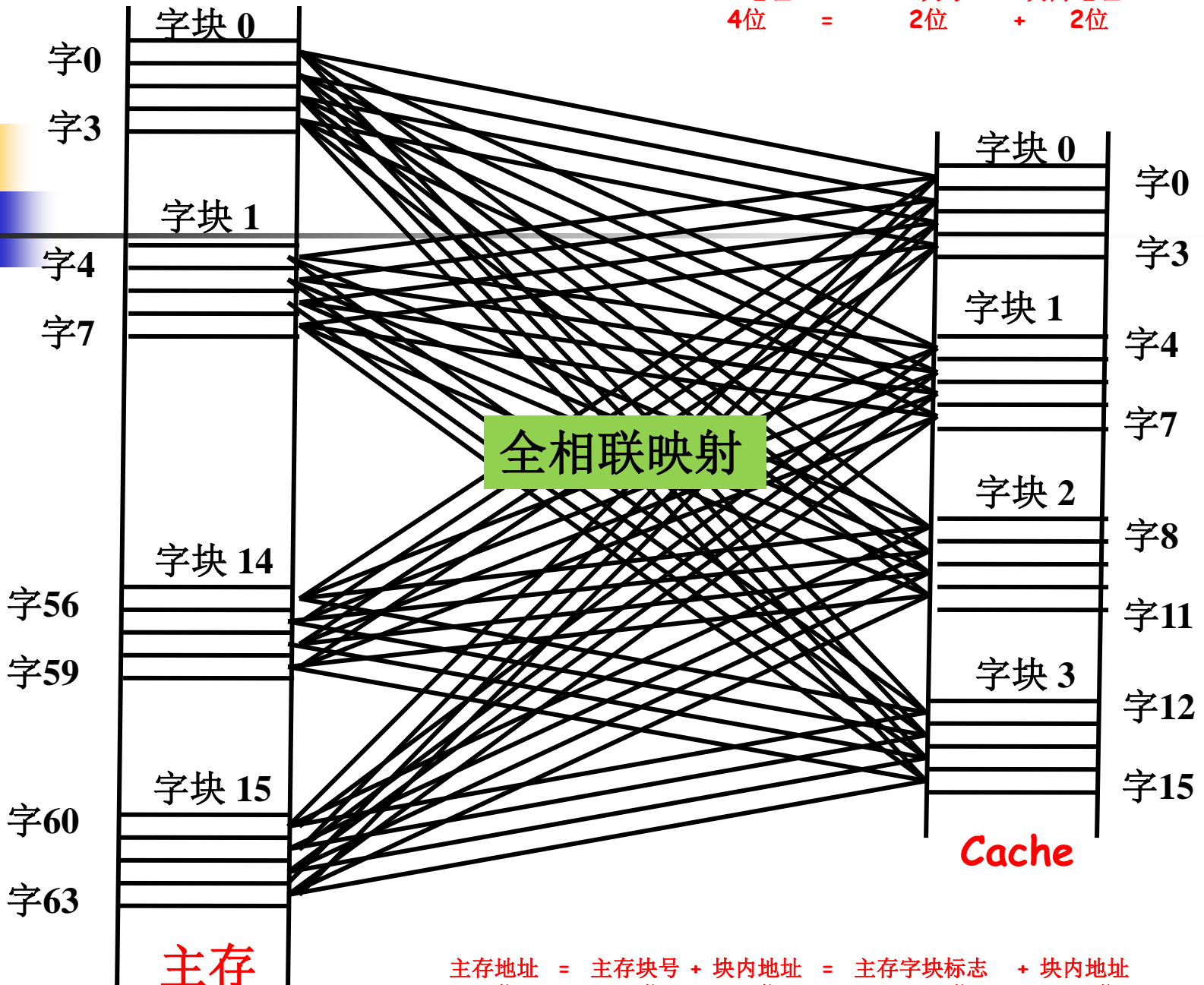
- 直接映射方式的**缺点**是不够灵活，因为每个主存块只能固定地对应某个缓存块，即使缓存内还空着许多位置也不能占用，使缓存的存储空间得不到充分利用
- 此外，如果程序恰好要重复访问对应同一缓存位置的不同主存块，就要不停地进行替换，从而降低命中率（比如访问主存的第**0、8、16.....**块，则都要访问**Cache**的第**0**块）

因为主存的第**0、8、16.....**块，都对应**Cache**的第**0**块

2. 全相联映射

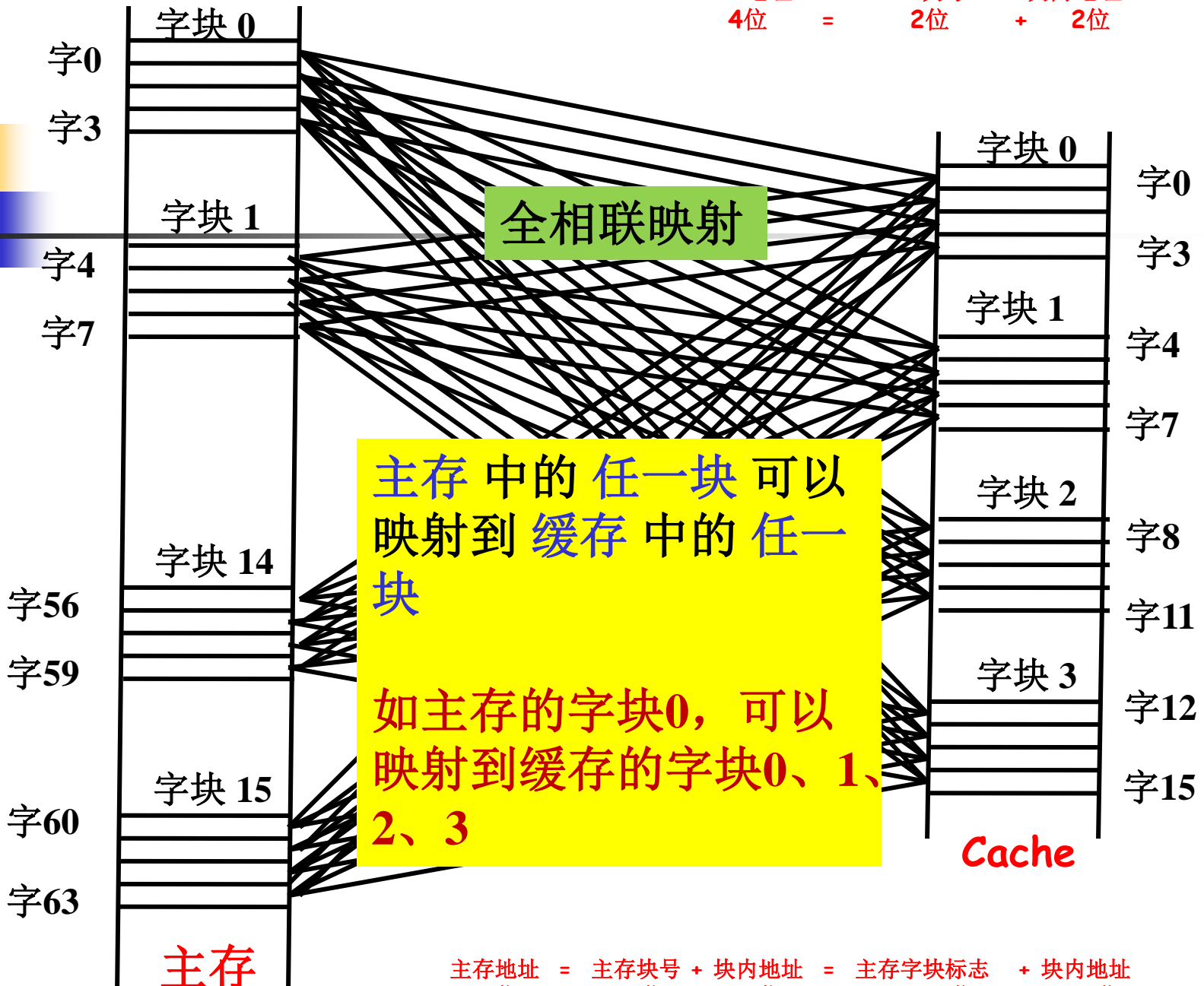


$$\begin{aligned} \text{Cache地址} &= \text{Cache块号} + \text{块内地址} \\ 4\text{位} &= 2\text{位} + 2\text{位} \end{aligned}$$

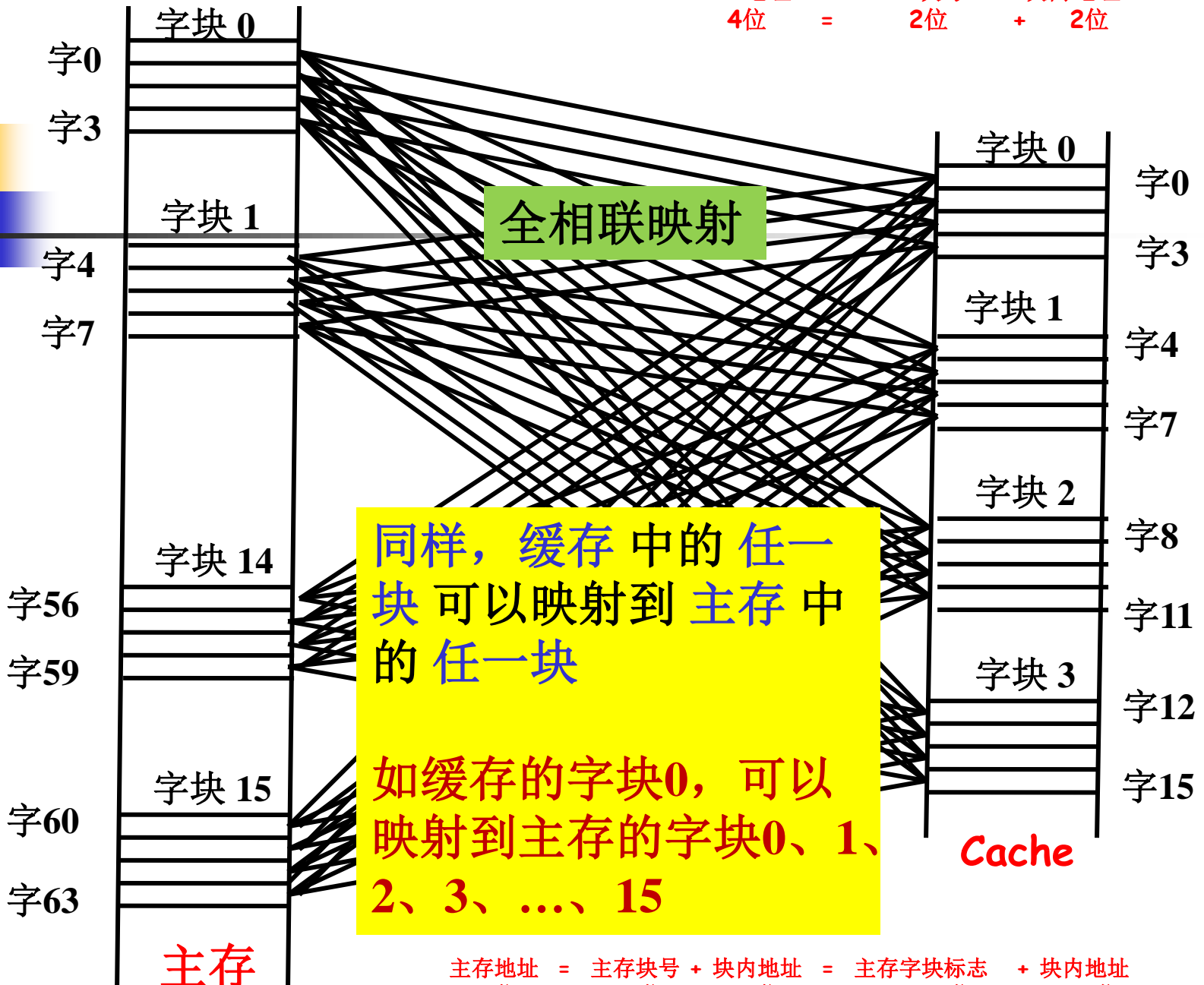


$$\begin{aligned} \text{主存地址} &= \text{主存块号} + \text{块内地址} = \text{主存字块标志} + \text{块内地址} \\ 6\text{位} &= 4\text{位} + 2\text{位} = 2+2\text{位} + 2\text{位} \end{aligned}$$

$$\begin{aligned} \text{Cache地址} &= \text{Cache块号} + \text{块内地址} \\ 4\text{位} &= 2\text{位} + 2\text{位} \end{aligned}$$



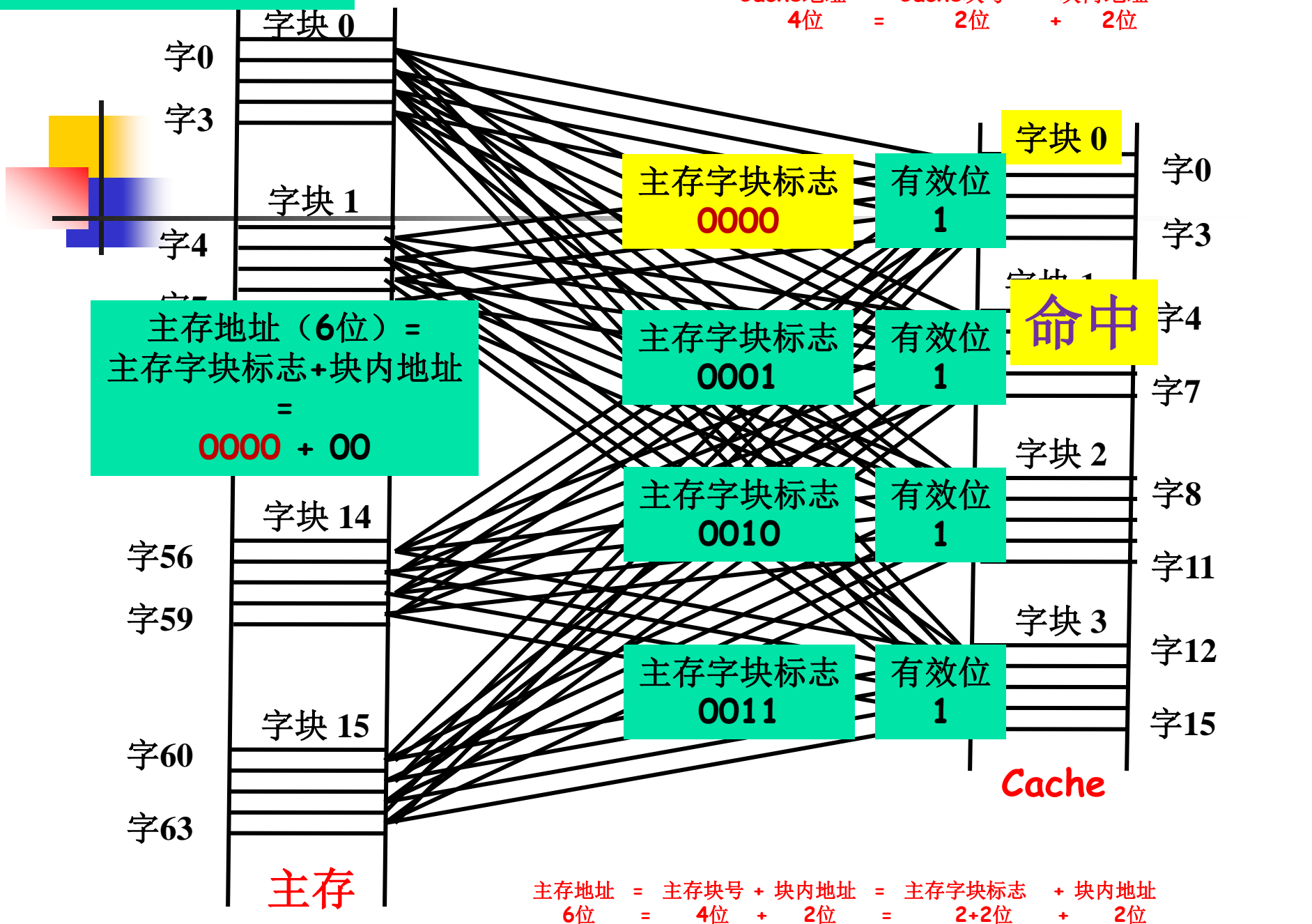
$$\begin{aligned} \text{Cache地址} &= \text{Cache块号} + \text{块内地址} \\ 4\text{位} &= 2\text{位} + 2\text{位} \end{aligned}$$



$$\begin{aligned} \text{主存地址} &= \text{主存块号} + \text{块内地址} = \text{主存字块标志} + \text{块内地址} \\ 6\text{位} &= 4\text{位} + 2\text{位} = 2+2\text{位} + 2\text{位} \end{aligned}$$

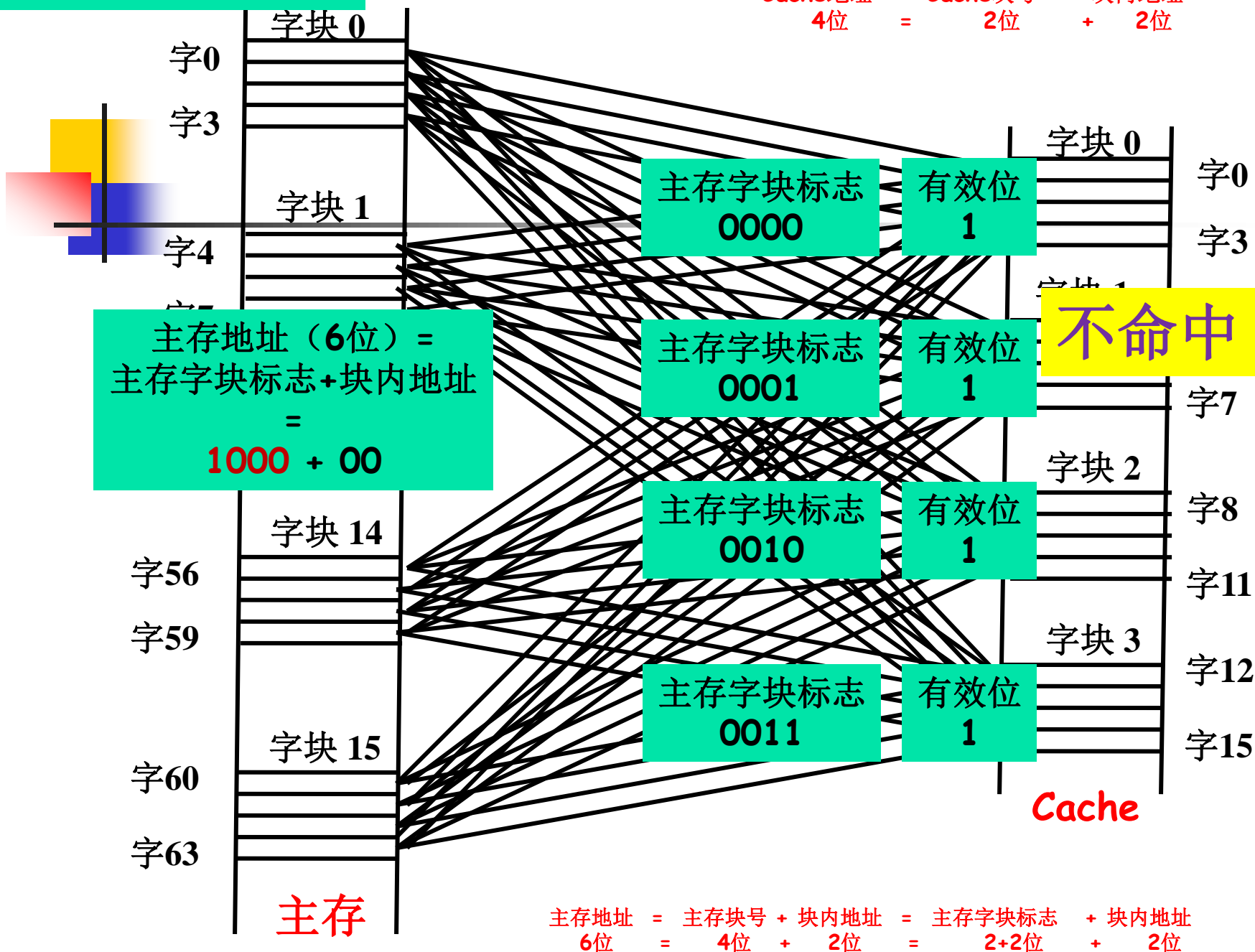
全相联映射的实现过程

Cache地址 = Cache块号 + 块内地址
4位 = 2位 + 2位



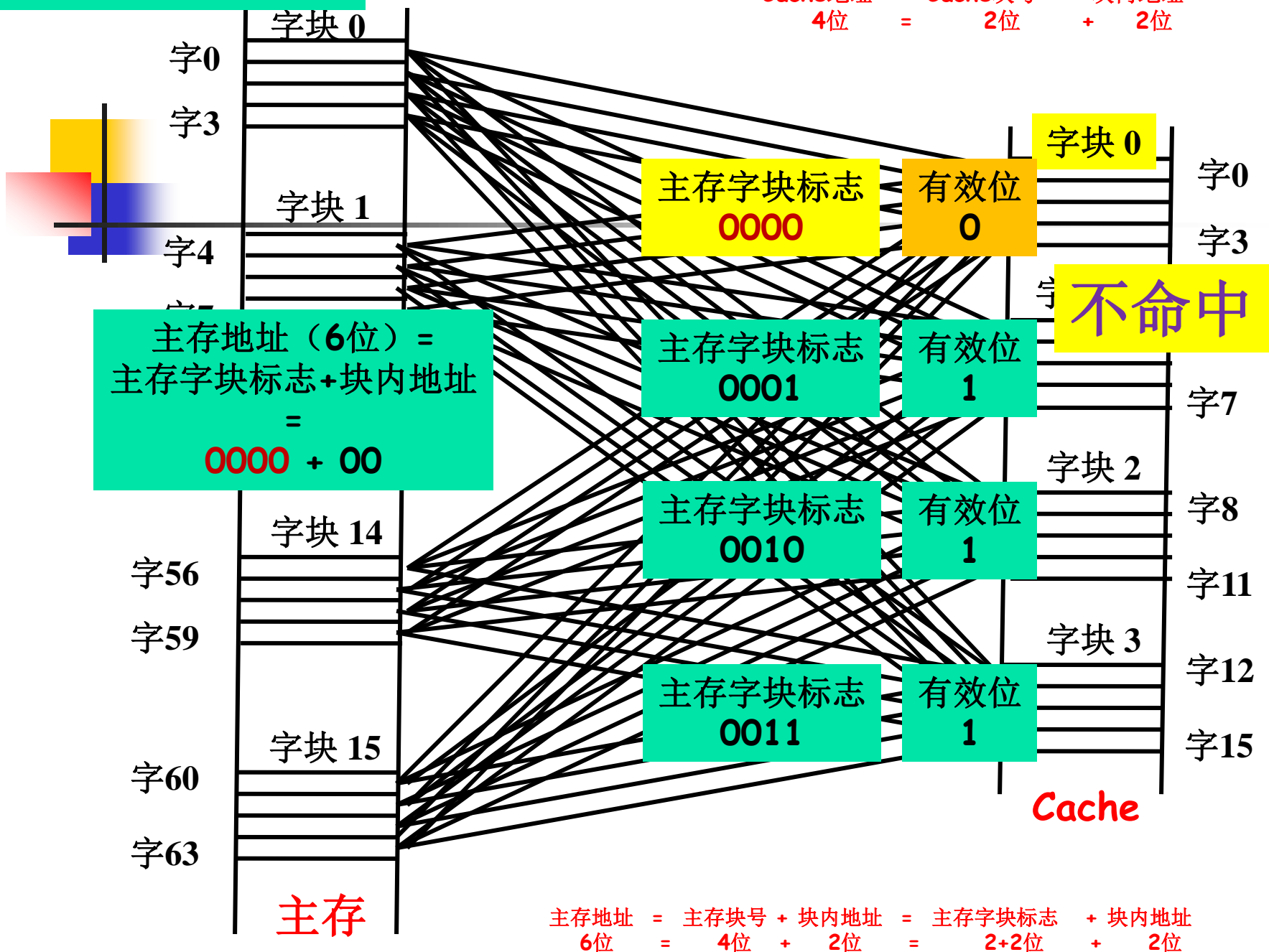
全相联映射的实现过程

Cache地址 = Cache块号 + 块内地址
4位 = 2位 + 2位



全相联映射的实现过程

Cache地址 = Cache块号 + 块内地址
4位 = 2位 + 2位



全相联映射

Cache (字块)	主存 (字块)
0	0、1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
1	0、1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
2	0、1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
3	0、1、2、3、4、5、6、7、8、9、10、11、12、13、14、15

$t=2$

$c=2$

$m=4$

3. 组相联映射

共32块，分为16组、每组2块

1,024个块

组 Cache 共 Q 组，每组内两块 ($r = 1$)

0	标记	字块 0	标记	字块 1
1	标记	字块 2	标记	字块 3
	⋮	⋮	⋮	⋮
$2^{c-r}-1$	标记	字块 2^c-2	标记	字块 2^c-1

主存地址

主存字块标记	组地址	字块内地址
$s = t + r$ 位	$q = c - r$ 位	b 位
m 位		

全相联映射

直接映射

$$i = j \bmod Q$$

某一主存块 j 按模 Q 映射到 缓存 的第 i 组中的 任一块

主存储器

字块 0

字块 1

⋮

字块 $2^{c-r}-1$

字块 2^c

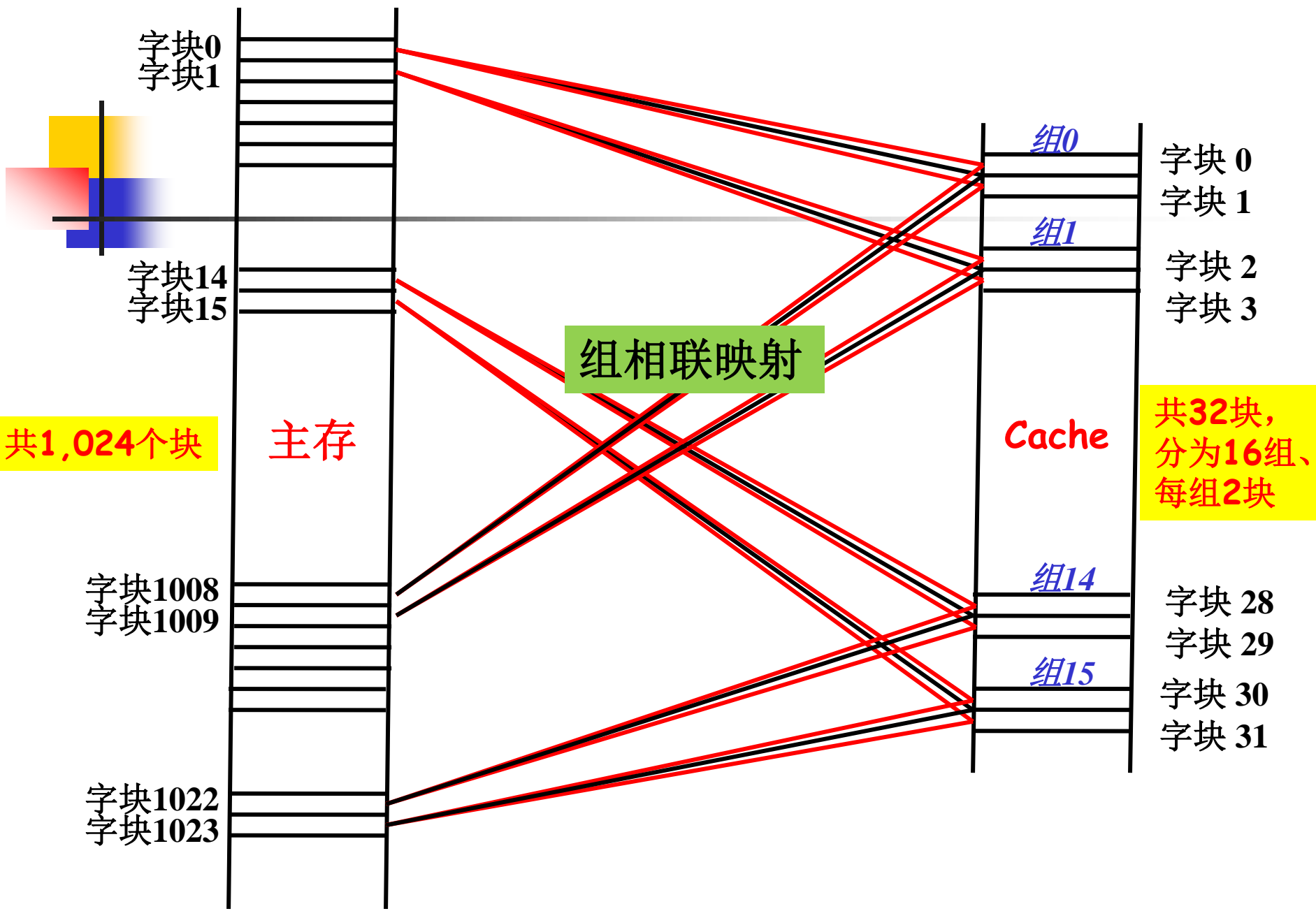
字块 $2^{c-r}+1$

⋮

字块 $2^{c-r}+1$

⋮

字块 2^m-1



字块0
字块1

组相联映射

字块14
字块15

组0

字块 0
字块 1

组1

字块 2
字块 3

主存的第0、16、...、1008字块可以映射到Cache的第0组（直接映射）中2个字块的任一块（全相联映射）

Cache

共32块，分为16组、每组2块

字块1008
字块1009

组14

字块 28
字块 29

组15

字块 30
字块 31

字块1022
字块1023

主存

共1,024个块

字块0
字块1

组相联映射

字块14
字块15

组0

字块 0

字块 1

组1

字块 2

字块 3

主存的第1、
17、...、1009字
块可以映射到
Cache的第1组
(直接映射) 中2
个字块的任一块
(全相联映射)

Cache

共32块，
分为16组、
每组2块

字块1008
字块1009

组14

字块 28

字块 29

组15

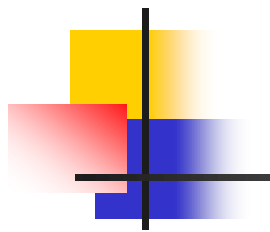
字块 30

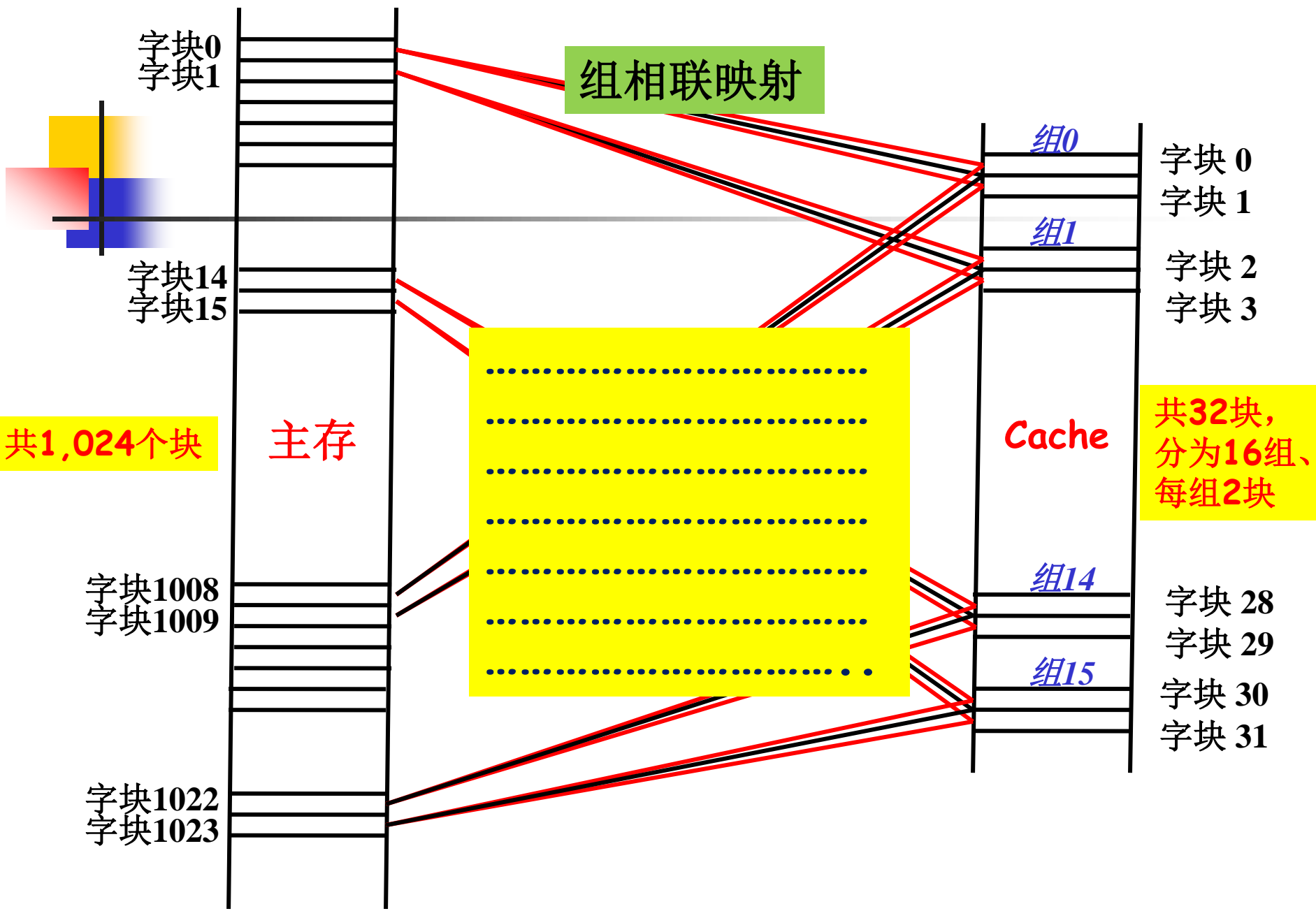
字块 31

字块1022
字块1023

主存

共1,024个块





字块0
字块1

组相联映射

字块14
字块15

组0

字块 0
字块 1

组1

字块 2
字块 3

Cache

共32块，
分为16组、
每组2块

组14

字块 28
字块 29

组15

字块 30
字块 31

共1,024个块

主存

主存的第14、
30、...、1022字
块可以映射到
Cache的第14组
(直接映射) 中2
个字块的任一块
(全相联映射)

字块1008
字块1009

字块1022
字块1023

字块0
字块1

组相联映射

字块14
字块15

组0

字块 0

字块 1

组1

字块 2

字块 3

Cache

共32块，
分为16组、
每组2块

字块1008
字块1009

组14

字块 28

字块 29

组15

字块 30

字块 31

主存

主存的第15、
31、...、1023字
块可以映射到
Cache的第15组
(直接映射) 中2
个字块的任一块
(全相联映射)

字块1022
字块1023

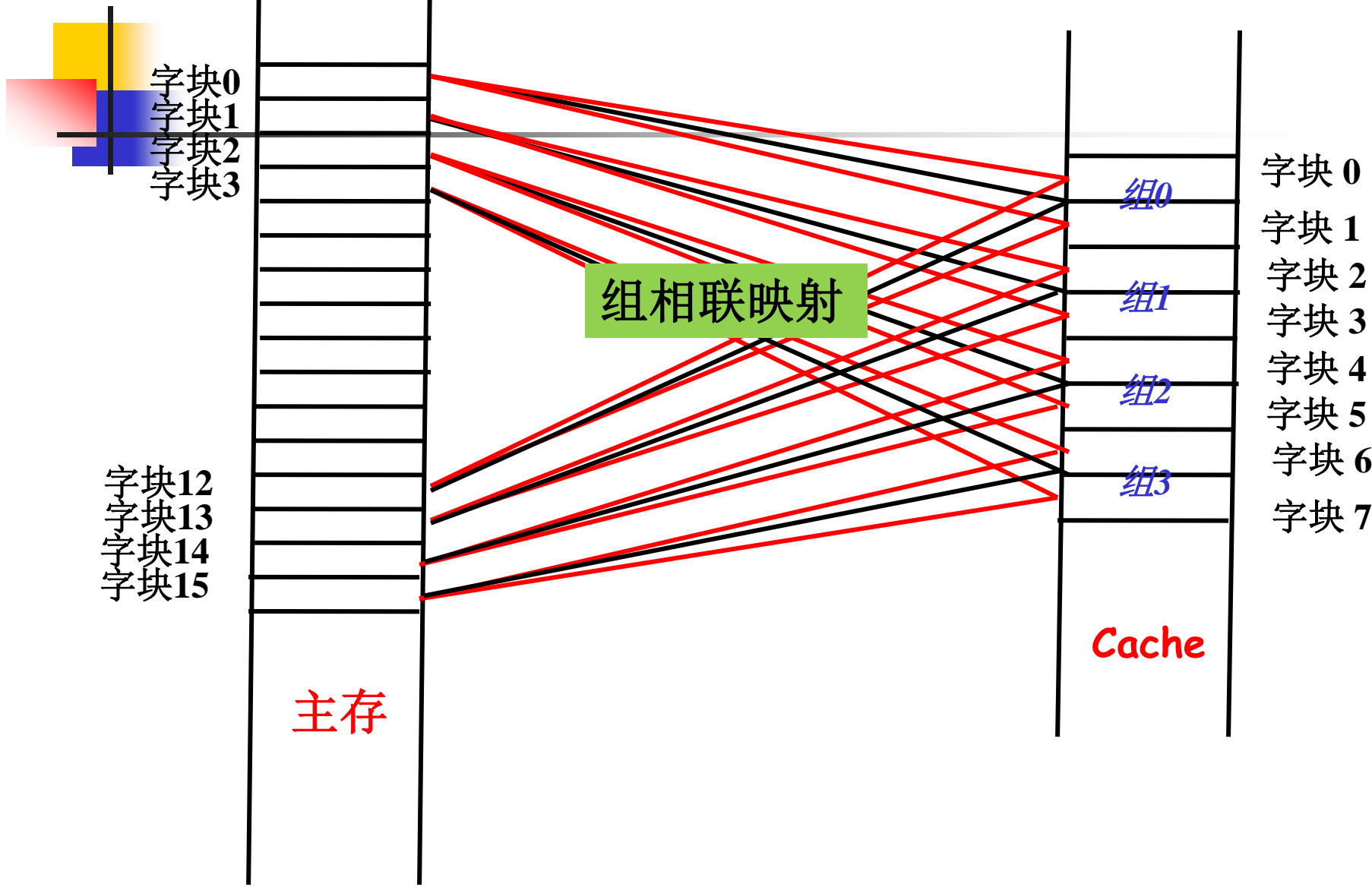
共1,024个块

组相联映射

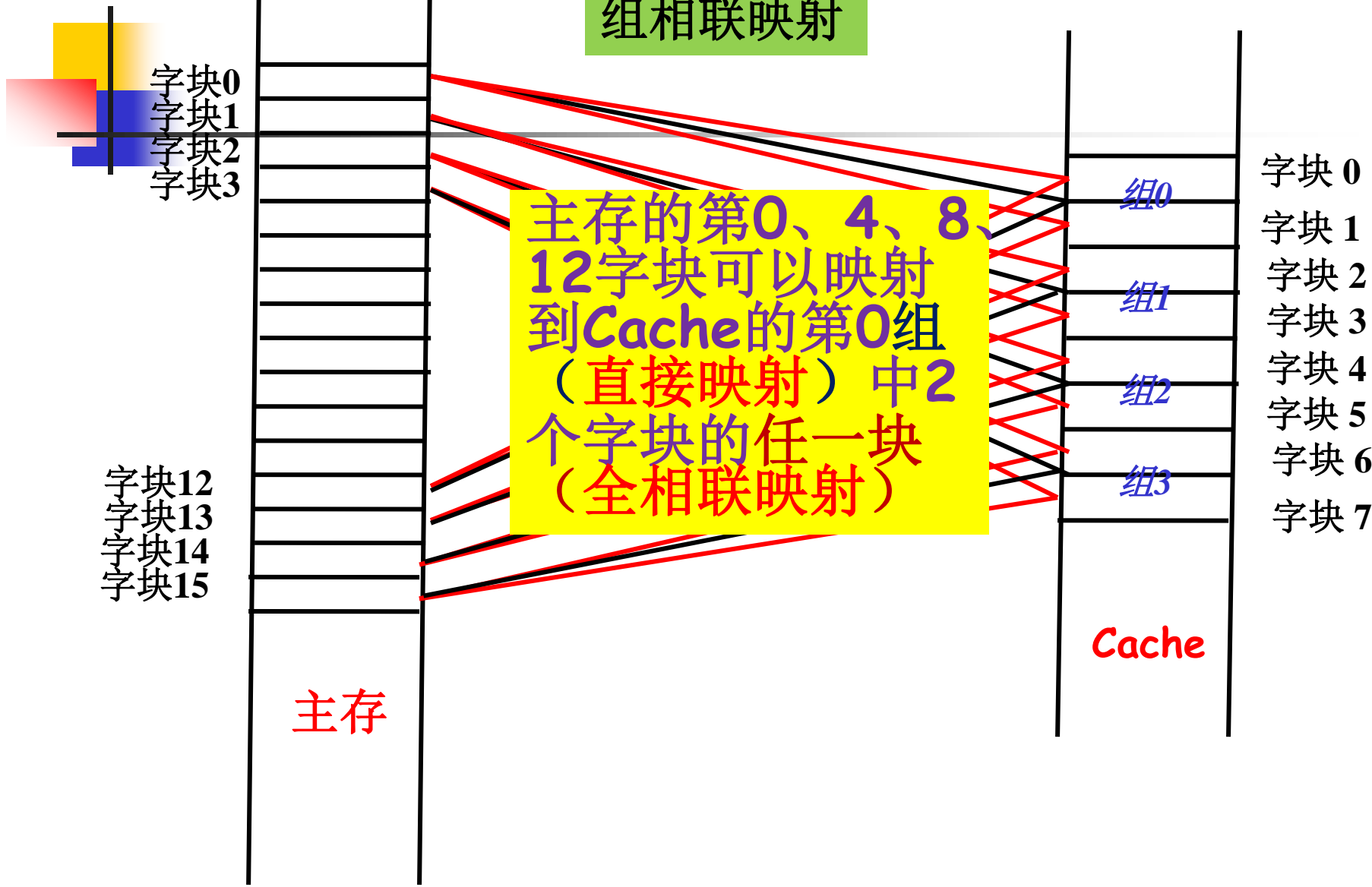
主存有1024块

Cache有32块（16组、每组2块）

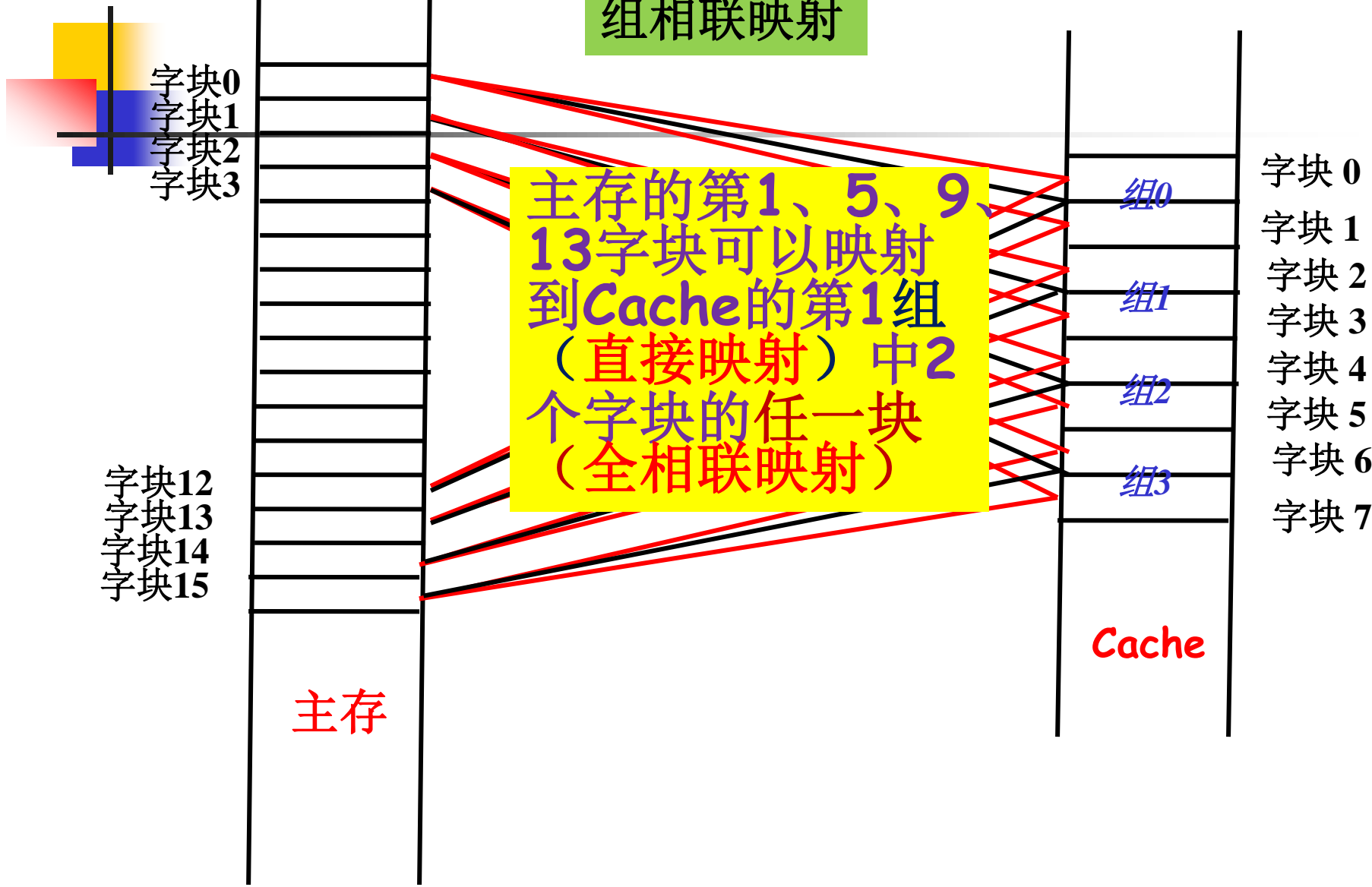
Cache	主存 64个
0组 0、 1	0、 16、 32、、 1008
1组 2、 3	1、 17、 33、、 1009
.....
14组 28、 29	14、 30、 46、、 1022
15组 30、 31	15、 31、 47、、 1023



组相联映射



组相联映射



主存的第1、5、9、13字块可以映射到Cache的第1组 (直接映射) 中2个字块的任一块 (全相联映射)

字块0
字块1
字块2
字块3

字块12
字块13
字块14
字块15

字块 0
字块 1
字块 2
字块 3
字块 4
字块 5
字块 6
字块 7

组0

组1

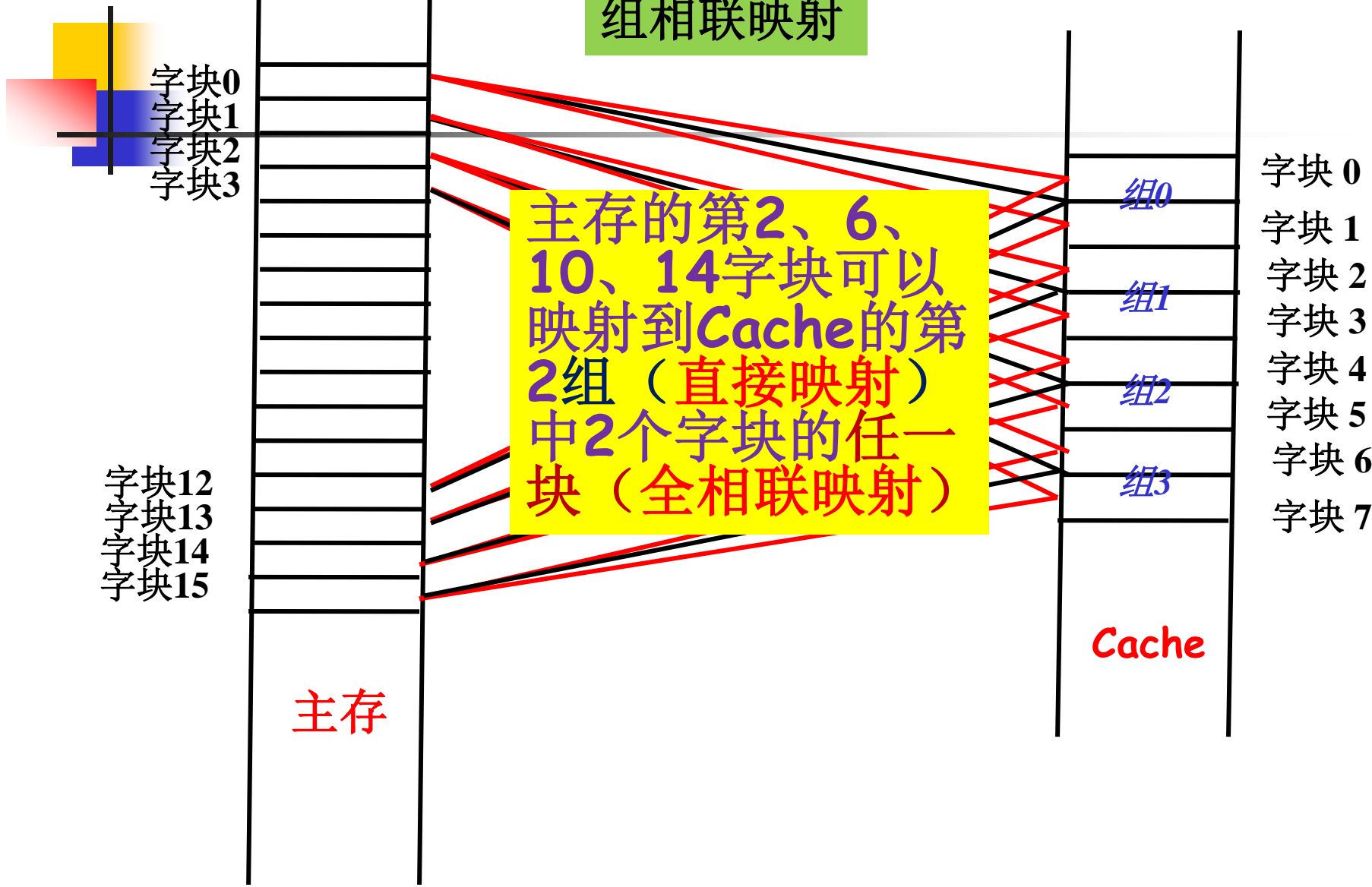
组2

组3

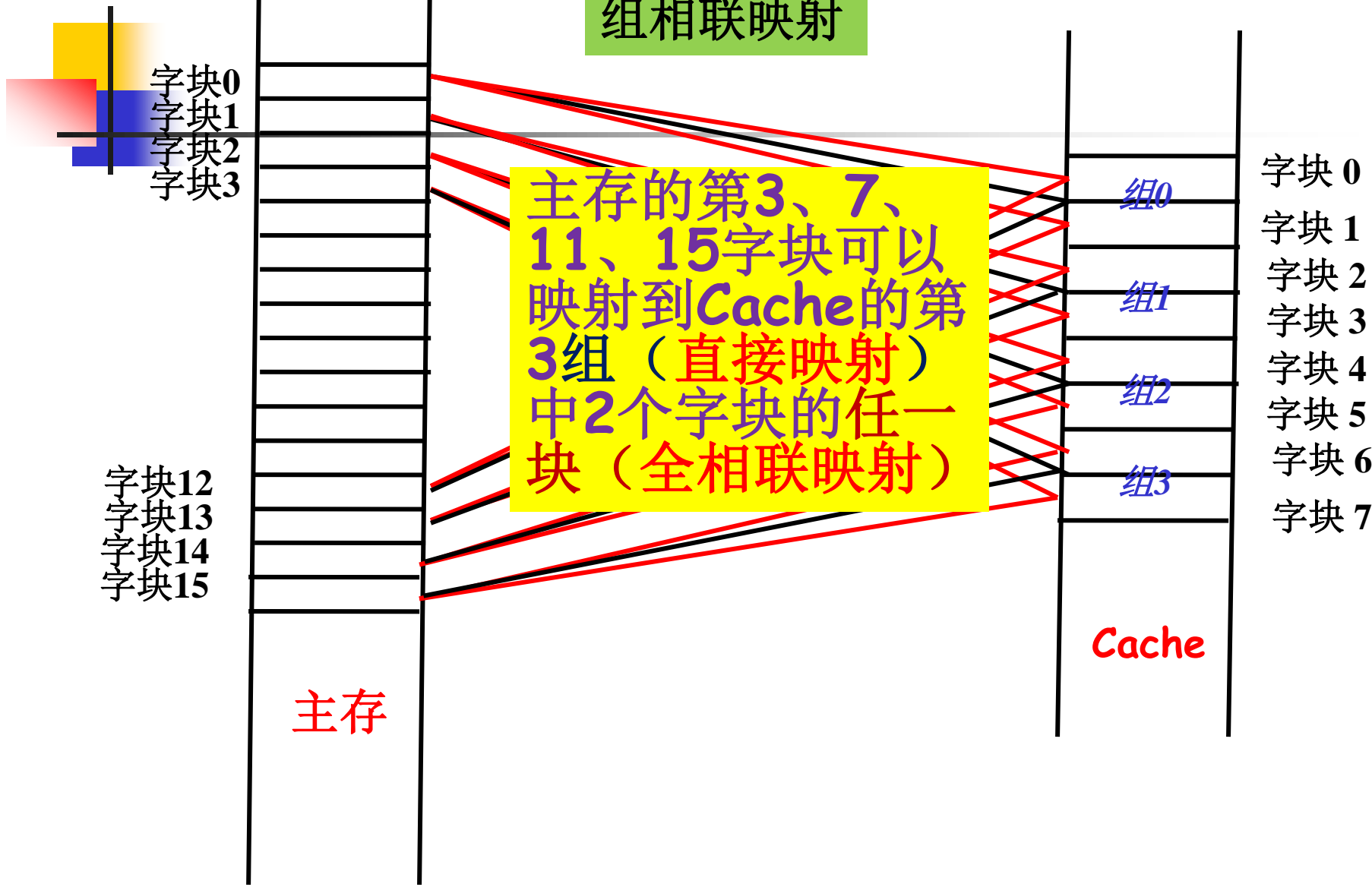
Cache

主存

组相联映射



组相联映射



组相联映射

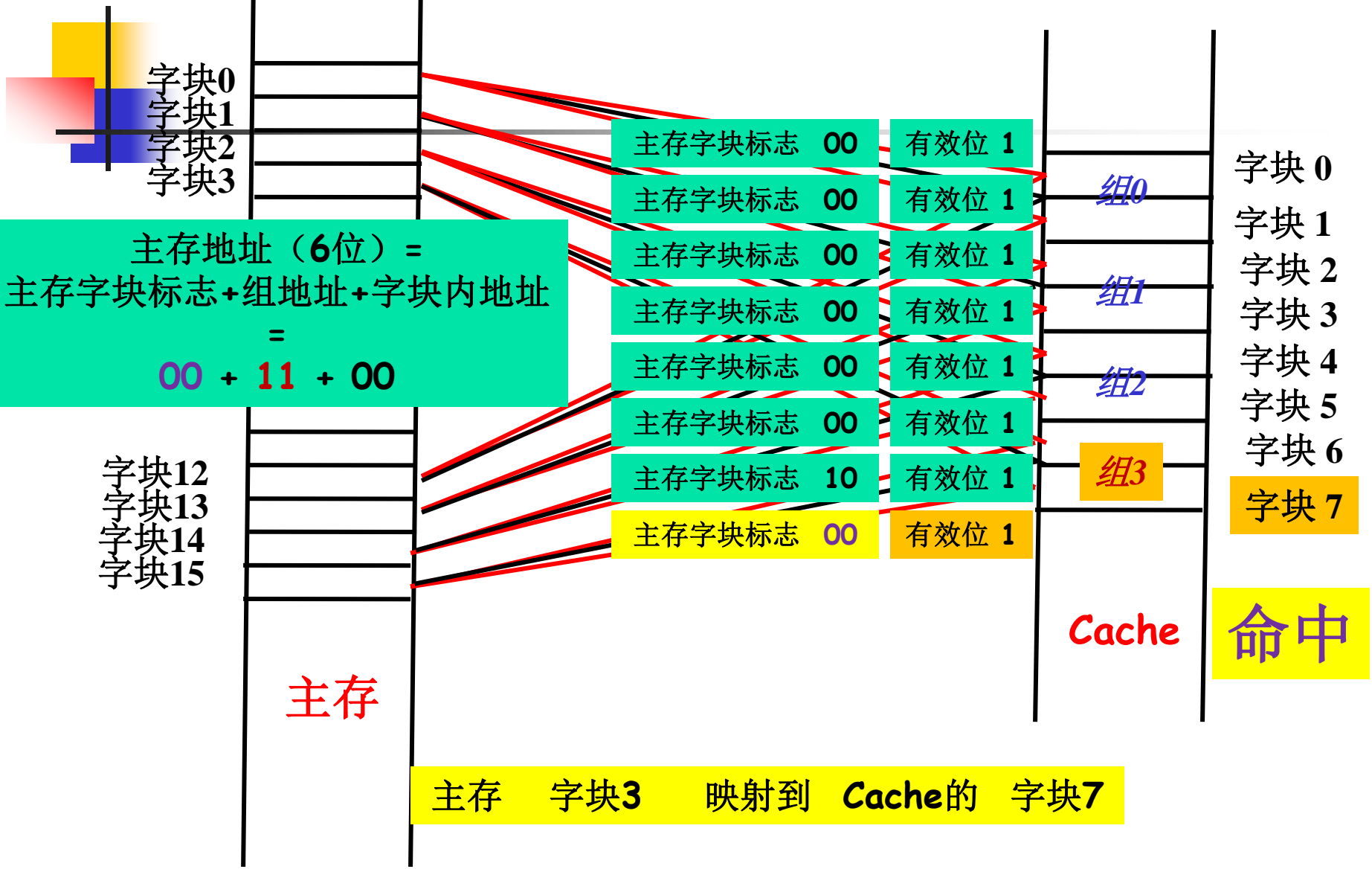
主存有16块

Cache有8块（4组、每组2块）

Cache		主存
0组	0、1	0、4、8、12
1组	2、3	1、5、9、13
2组	4、5	2、6、10、14
3组	6、7	3、7、11、15

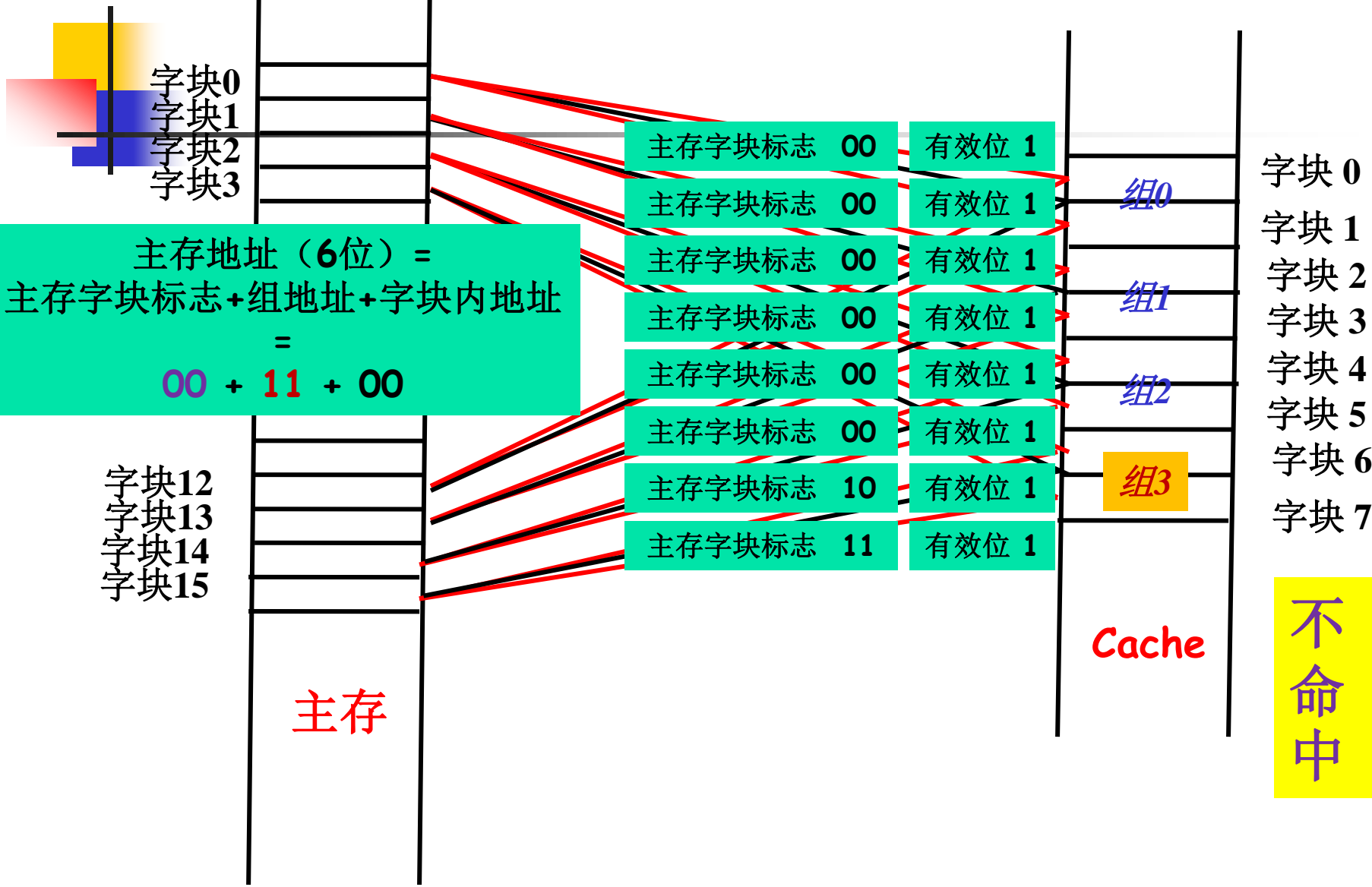
组相联映射的实现过程

1个字块=4个字



组相联映射的实现过程

1个字块=4个字



主存地址（6位）= 主存字块标志+组地址+字块内地址

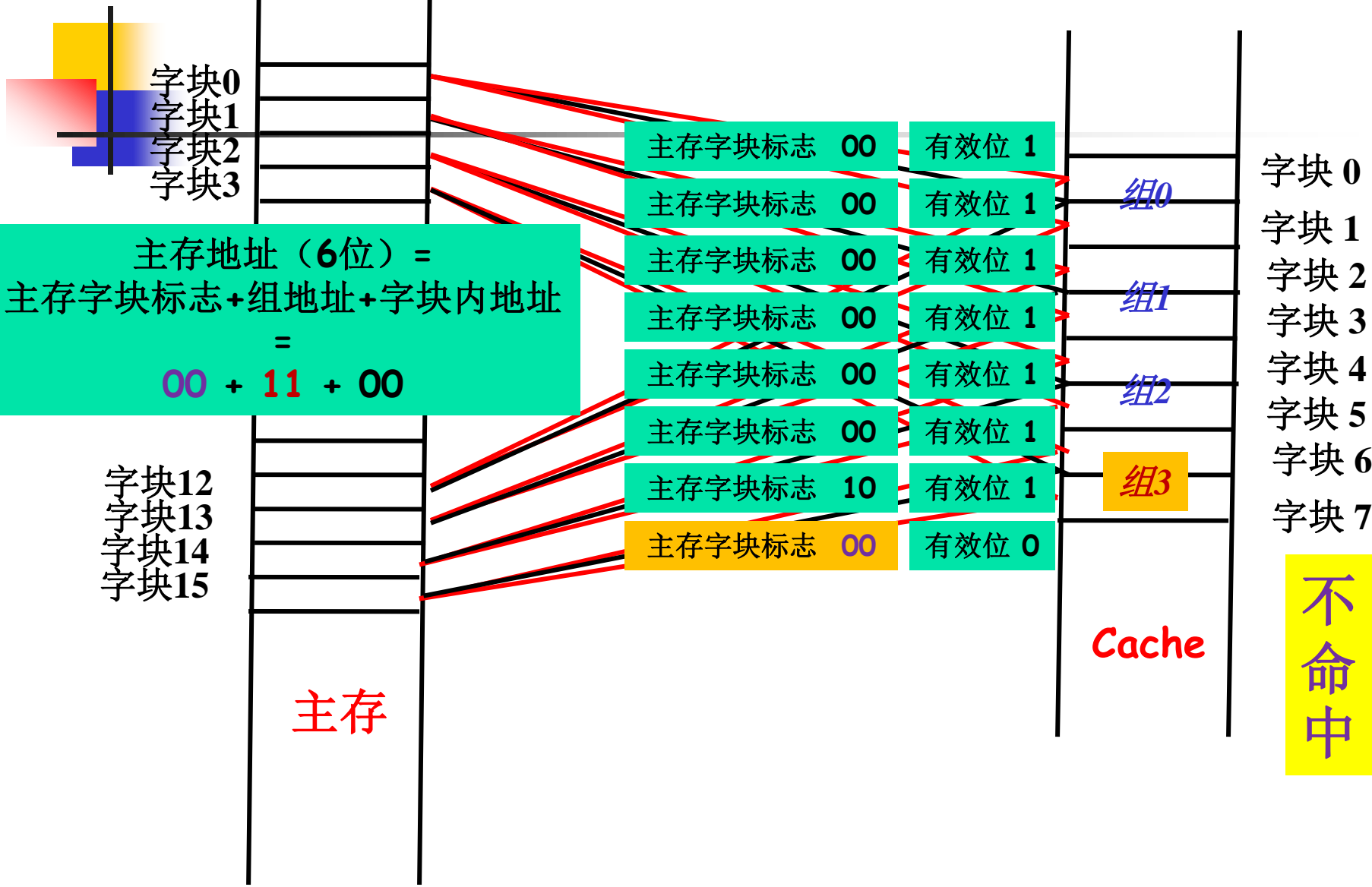
= 00 + 11 + 00

- 字块 0
- 字块 1
- 字块 2
- 字块 3
- 字块 4
- 字块 5
- 字块 6
- 字块 7

不命中

组相联映射的实现过程

1个字块=4个字



主存地址（6位） =
主存字块标志 + 组地址 + 字块内地址
= 00 + 11 + 00

不命中

例4.8: 假设主存容量为**512KB**, **Cache**容量为**4KB**, 每个字块为**16**个字, 每个字**32**位。

(1)**Cache**地址有多少位? 可容纳多少块?

(2)主存地址有多少位? 可容纳多少块?

(3)在直接映射方式下, 主存的第几块映射到**Cache**的第**5**块 (假设起始字块为第**1**块)?

(4)画出直接映射方式下主存地址字段中各段的位数。

解:

(1)**Cache**容量为**4KB**, 则地址=**12位** ($2^{12}=4K$)

Cache有**64块** ($4KB/16/4B=64$) ($32位=4B$)

(2)主存容量为**512KB**, 则地址=**19位** ($2^{19}=512K$)

主存有**8,192块** ($512KB/16/4B=8,192$) ($32位=4B$)

(3)直接映射方式下, 主存的第**5**、**64+5**、**2X64+5**、...、**127x64+5**映射到**Cache**的第**5**块

每个字块为**16**个字, 每个字**32**位

(4) 主存地址 = 主存字块标志+**Cache**字块地址+块内地址

主存地址 = **19位** = **7位** + **6位** + (**4位** + **2位**)

Cache地址 = **12位** = **6位** + (**4位** + **2位**)

图4.57

Cache有**64**块, 字块地址=**6**位

主存有**8192**块, 字块地址=**13**位=**7**位+**6**位

直接映射方式

主存字块标记	缓存字块地址	字块内地址
7 位	6 位	6 位

图 4.57 例 4.8 主存地址各字段的分配

直接映射方式

主存字块标志用于确定这128个中的哪一个？
(7位)

128个

Cache	主存
0	0、64、128、.....、8128
1	1、65、129、.....、8129
2	2、66、130、.....、8130
3	3、67、131、.....、8131
4	4、68、132、.....、8132
5	5、69、133、.....、8133
.....
63	63、127、191、.....、8191

例4.9：假设主存容量为**512KX16**位，**Cache**容量为**4,096X16**位，块长为**4**个**16**位的字，**访存地址为字地址**。

(1)在直接映射方式下，设计主存的地址格式；

(2)在全相联映射方式下，设计主存的地址格式；

(3)在二路组相联映射方式下，设计主存的地址格式；

(4)若主存容量为**512KX32**位，块长不变，在四路组相联映射方式下，设计主存的地址格式。

解： Cache容量为**4,096X16**位

每块 = 4个字 = **4X16**位

分为**1,024**块

$$(4,096 \times 16) / (4 \times 16) = 1,024$$

主存容量为**512KX16**位

分为**128K**个块

$$(512 \text{K} \times 16) / (4 \times 16) = 128 \text{K}$$

(1)直接映射方式

访存地址为字地址

主存地址=**19**位 ($2^{19}=512\text{K}$)

Cache地址=**12**位 ($2^{12}=4096$)

主存地址格式: **7**(主存字块标记)+**10**(Cache字块地址)+**2**(块内地址)

Cache有**1,024**块

块长为**4**个字

(2)全相联映射方式

主存地址格式: **17**(主存字块标记) + **2**(块内地址)

- (3)二路组相联映射方式

访存地址为字地址

Cache分为512组，每组2块

主存地址格式：8(主存字块标记) + 9(Cache组地址) + 2(块内地址)

Cache分为512组

- (4)主存容量为512KX32位

主存地址=20位 ($2^{20}=512K \times 2 \times 16$ 位)

块长不变，在四路组相联映射方式下，Cache分为256组，每组4块

主存地址格式：10(主存字块标记) + 8(Cache组地址) + 2(块内地址)

Cache分为256组

主存字块标记	Cache 字块地址	字块内地址
7	10	2

(a) 直接映射方式主存地址格式

主存字块标记	字块内地址
17	2

(b) 全相联映射方式主存地址格式

主存字块标记	组地址	字块内地址
8	9	2

(c) 二路组相联映射方式主存地址格式

主存字块标记	组地址	字块内地址
10	8	2

(d) 四路组相联映射方式双字宽主存地址格式

图 4.58 例 4.9 主存地址格式

直接映射方式

主存字块标志用于确定这128个中的哪一个
(7位)

Cache	主存
0	0、1024、2048、.....、130048
1	1、1025、2049、.....、130049
2	2、1026、2050、.....、130050
.....
1023	1023、2047、3071、.....、131071

128个

1,024块

128K块=131072块

全相联映射方式

主存字块标志用于确定这131072个中的哪一个
(17位) $2^{17}=131072$

131072个

Cache	主存
0	0、1、2、.....、131071
1	0、1、2、.....、131071
2	0、1、2、.....、131071
.....
1023	0、1、2、.....、131071

1,024块

128K块=131072块

二路组相联映射方式

主存字块标志用于确定这256个中的哪一个
(8位)

256个

Cache		主存
0组	0, 1	0、512、1024、.....、130559
1组	2, 3	1、513、1025、.....、130560
2组	4, 5	2、514、1026、.....、130561
.....	
511组	1022, 1023	511、1023、1535、.....、131071

1,024块
分为512组、每组2块

128K块=131072块

四路组相联映射方式

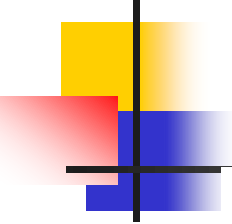
主存字块标志用于确定这**1024**个中的哪一个
(**10**位)

1024个

Cache		主存
0组	0, 1, 2, 3	0、256、512、.....、261887
1组	4, 5, 6, 7	1、257、513、.....、261888
2组	8, 9, 10, 11	2、258、514、.....、261889
.....	
255组	1020, 1021, 1022, 1023	255、511、767、.....、262143

1,024块
分为**256组**、每组**4块**

256K块=262144块



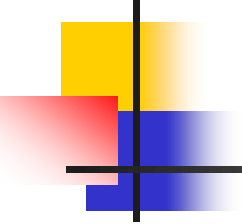
例4.10：假设**Cache**的工作速度是主存的**5**倍，且**Cache**被访问命中的概率为**95%**，则采用**Cache**后，存储器性能提高多少？

$$\text{平均访问时间 } t_a = h t_c + (1-h) t_m$$

解：系统的平均访问时间为：

$$t_a = 0.95 \times t + 0.05 \times 5t = 1.5t$$

性能为原来的 $5t / 1.5t = 3.33$ 倍，提高 **2.33** 倍

- 
- **例4.11**：设某机主存容量为**16MB**，**Cache**的容量为**8KB**。每字块有**8**个字，每字**32**位。设计一个四路组相联映射的**Cache**组织。
 - 画出主存地址字段中各段的位数；
 - 设 **Cache** 初 态 为 空 ， **CPU** 依 次 从 主 存 第 **0,1,2,...,99**号单元读出**100**个字（主存一次读出一个字），并重复此次序读**10**次，问命中率是多少？
 - 若**Cache**的速度是主存速度的**5**倍，试问有**Cache**和无**Cache**相比，速度提高多少倍？
 - 系统的效率为多少？

■ **解：** 主存地址=24位 ($2^{24}=16\text{M}$)
Cache地址=13位 ($2^{13}=8\text{K}$)

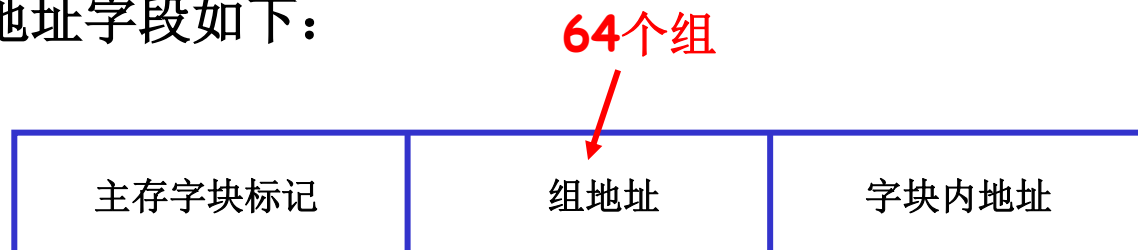
每字块有8个字，每字32位

主存分为512K块 ($16\text{MB}/8/4\text{B} = 512\text{K}$) ($32\text{位}=4\text{B}$)
Cache分为256块 ($8\text{KB}/8/4\text{B} = 256$) ($32\text{位}=4\text{B}$)

(1)四路组相联：

Cache分为64组，每组4块 ($64=256/4$)

主存的地址字段如下：



13位

6位

5位

$24-6-5=13$

每块8个字、每个字32位

($5 = 3 + 2$)

四路组相联映射方式

8192个

Cache	主存
0组 0, 1, 2, 3	0、64、128、.....、524223
1组 4, 5, 6, 7	1、65、129、.....、524224
2组 8, 9, 10, 11	2、66、130、.....、524225
.....
63组 252, 253, 254, 255	63、127、191、.....、524287

256块
分为64组、每组4块

512K块 = 524288块

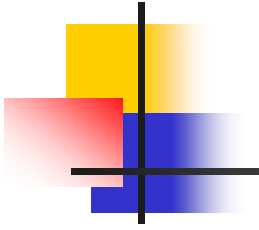
0, 8, 16, 24, 32,
40, 48, 56, 64,
72, 80, 88, 96

因为每块8个字

第0块中是0-7号字（0-7号单元）

- (2) 因为Cache的初态为空，因此CPU读0号单元时为未命中，必须访问主存，同时将该字所在的主存块调入Cache第0组中的任一块内。接着CPU读1-7号单元时均命中。
 - 同理，读第8,16,...,96号单元时均未命中
 - CPU连续读100个字有13次未命中，而后9次循环则全部命中
 - 命中率 $h = (100 \times 10 - 13) / (100 \times 10) = 0.987$
- (3) 没有Cache时访问存储器的时间为 $1,000 \times 5t$ ；有Cache后，访问存储器的时间为 $(1,000 - 13) \times t + 13 \times 5t = 1,052t$ 。速度提高： $5,000 / 1,052 - 1 = 3.75$ 倍
- (4) 效率 $e = t / [0.987 \times t + (1 - 0.987) \times 5t] = 95\%$

$$\text{访问效率 } e = t_c / (ht_c + (1 - h) \times t_m)$$



直接映射 某一主存块只能固定映射到某一缓存块

全相联映射 某一主存块能映射到任一缓存块

组相联映射 某一主存块只能映射到某一缓存组中的任一块



三、替换算法

- 当新的主存块需要调入**Cache**并且它的可用空间位置又被占满时，需要替换掉**Cache**中的数据，这就产生了替换策略(算法)问题：
 - **直接映射**方式的**Cache**，主存块与**Cache**块唯一对应，替换策略很简单；
 - **全相联映射**和**组相联映射**方式的**Cache**，主存的一个块对应**Cache**的多个块，存在替换算法问题。

三、替换算法（续）

1. 先进先出（FIFO）算法

选择最早调入Cache的字块进行替换，优点是算法简单，缺点是没有利用访存的局部性原理

First In First Out

三、替换算法（续）

2. 近期最少使用（LRU）算法

Least Recently Used, LRU

LRU算法比较好地利用访存的局部性原理，替换出近期用得最少的字块

优点：利用存储器的局部性原理，提高了命中率

缺点：算法比较复杂



三、替换算法（续）

3. 随机法（**RAND**）

随机地确定被替换的块，算法比较简单，没有利用访存的局部性原理，不能提高**Cache**的命中率

Random



第6次作业——习题(P150-153)

- 4.25
- 4.28
- 4.29
- 4.30
- 4.31
- 4.32
- 4.33



关于作业的提交

- **1周内**必须提交（上传到学院的**FTP**服务器上），否则认为是迟交作业；如果期末仍然没有提交，则认为是未提交作业
 - 作业完成情况成绩=第**1**次作业提交情况*第**1**次作业评分+第**2**次作业提交情况*第**2**次作业评分+.....+第**N**次作业提交情况*第**N**次作业评分
 - 作业评分：**A**（好）、**B**（中）、**C**（差）三挡
 - 作业提交情况：按时提交（**1.0**）、迟交（**0.5**）、未提交（**0.0**）
- 请采用电子版的格式（**Word**文档）上传到**FTP**服务器上，文件名取“学号+姓名+第**X**次作业.doc”
 - 例如：**11920192203642+袁佳哲+第6次作业.doc**
- 第**6**次作业提交的截止日期为：**2021年4月9日晚上24点**



The End

Thanks