计算机组成原理

第 10 讲

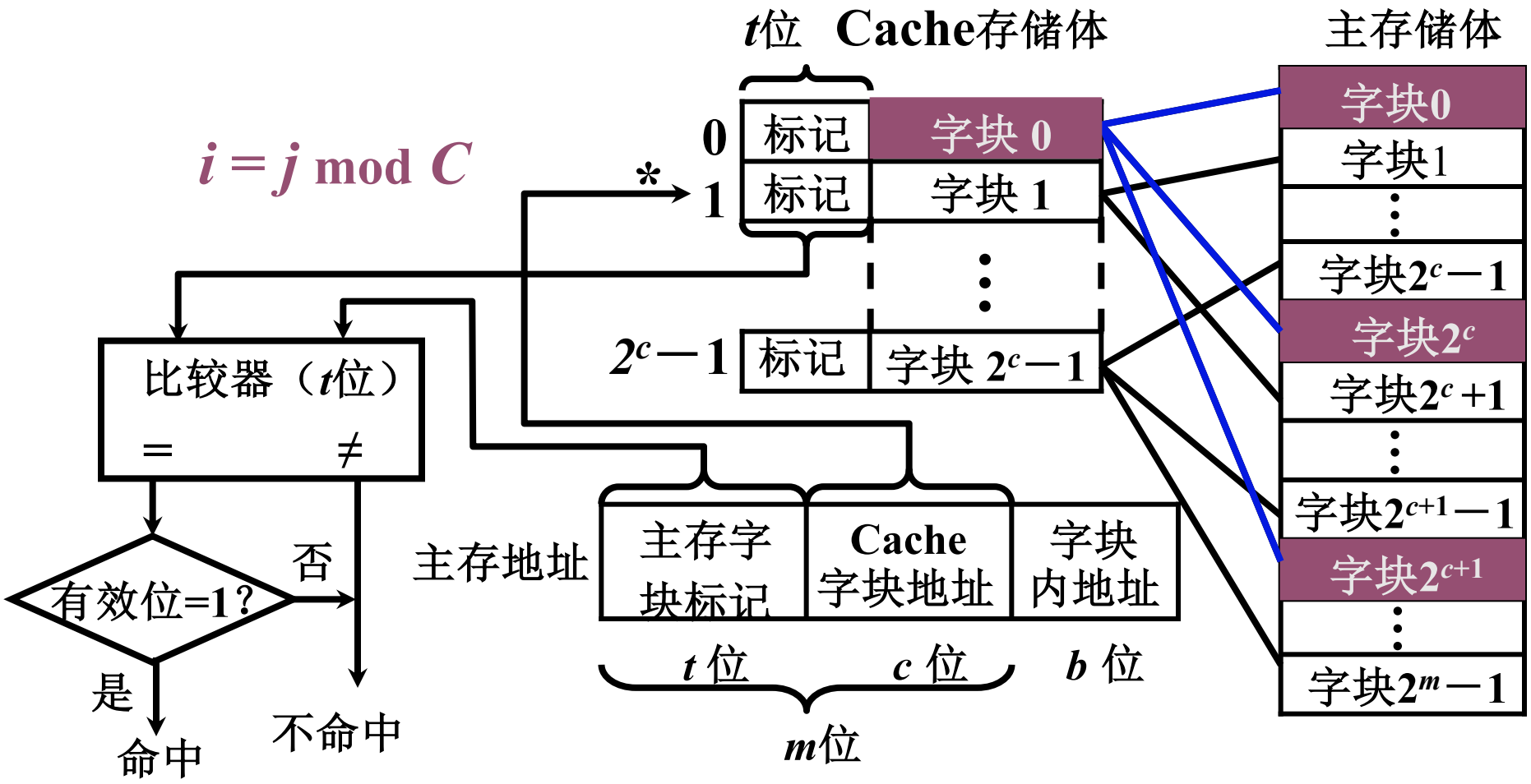
左德承

哈尔滨工业大学计算学部

容错与移动计算研究中心

1. 直接映射

4.3



每个缓存块 i 可以和 若干 个 主存块 对应

每个主存块 j 只能和 一 个 缓存块 对应

2. 全相联映射

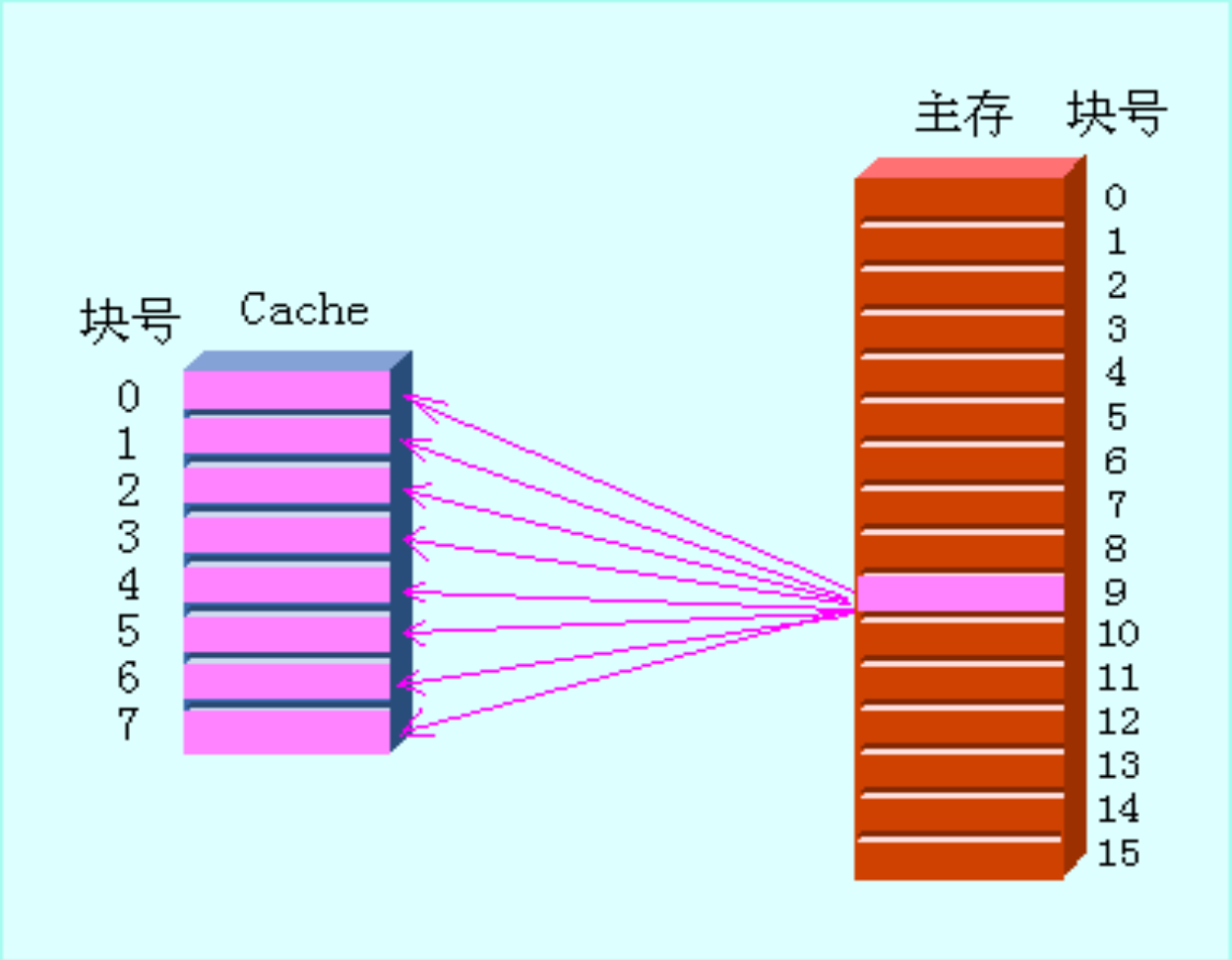
全相联：主存中的任一块可以被放置到Cache中的任意一个位置。

对比：　阅览室位置--随便坐

特点：　空间利用率最高，冲突概率最低，实现最复杂。

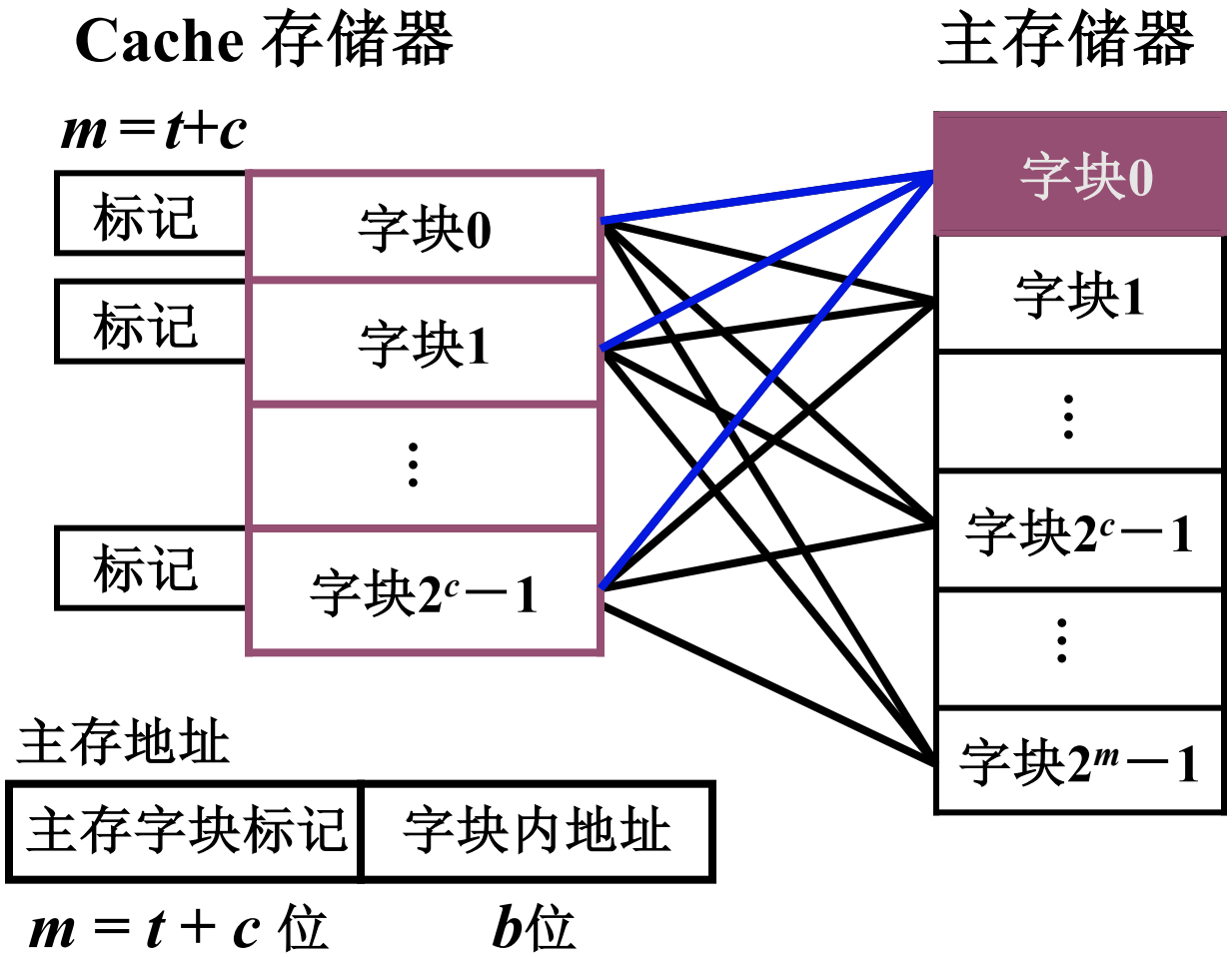
全相联映射

（举例）



2. 全相联映射

4.3

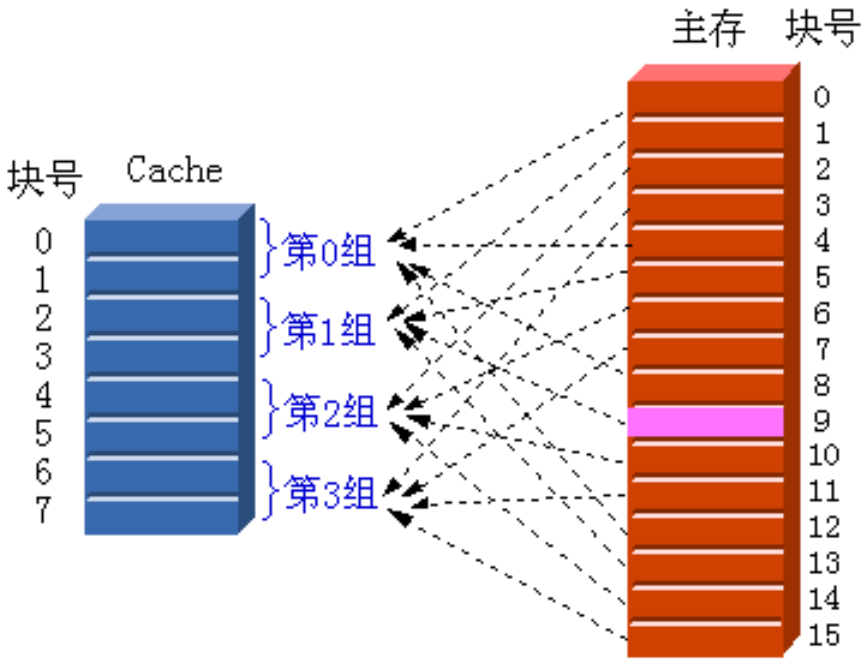


主存 中的 任一块 可以映射到 缓存 中的 任一块

3. 组相联映射

组相联：主存中的每一块可以被放置到Cache中唯一　的一个组中的任何一个位置。

组相联是直接映象和全相联的一种折中



3. 组相联映射

主存储器

组　Cache　共 Q 组，每组内两块（$r=1$）

组	标记	字块 0	标记	字块 1
0				
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 →		

主存储器

- 字块0
- 字块1
- ⋮
- 字块$2^{c-r}-1$
- 字块2^{c-r}
- 字块$2^{c-r}+1$
- ⋮
- 字块2^{c-r+1}
- ⋮
- 字块2^m-1

$i=j \bmod Q$

直接映射　全相联映射

某一主存块 j 按模 Q 映射到 缓存 的第 i 组中的 任一块

- *n* 路组相联：每组中有*n* 个块($n=M/G$)，*n* 称为相联度　相联度越高，Cache空间的利用率就越高，块冲突概率就越低，失效率也就越低。

	n (路数)	*G* (组数)
全相联	*M*	1
直接映象	1	*M*
组相联	$1<n<M$	$1<G<M$

- 大多数计算机的Cache: $n \leq 4$

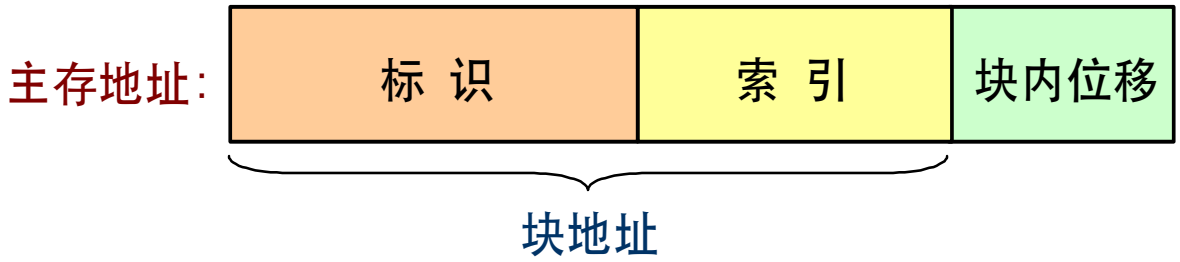
想一想：相联度是否越大越好？

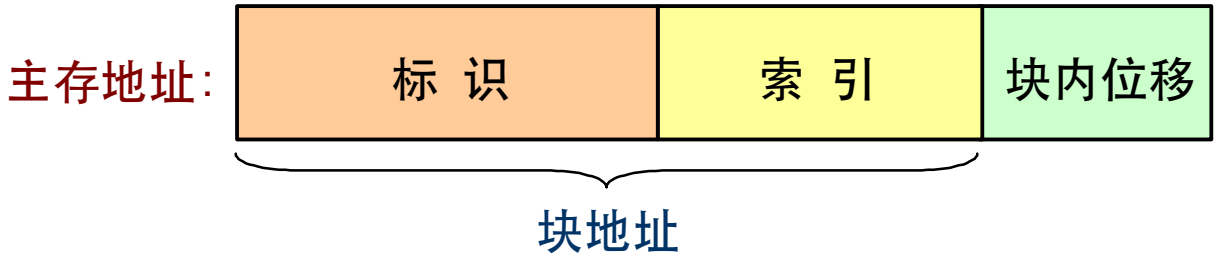
四、查找方法

4.3

- 当CPU访问Cache时，如何确定Cache中是否有所要访问的块？
- 若有的话，如何确定其位置？

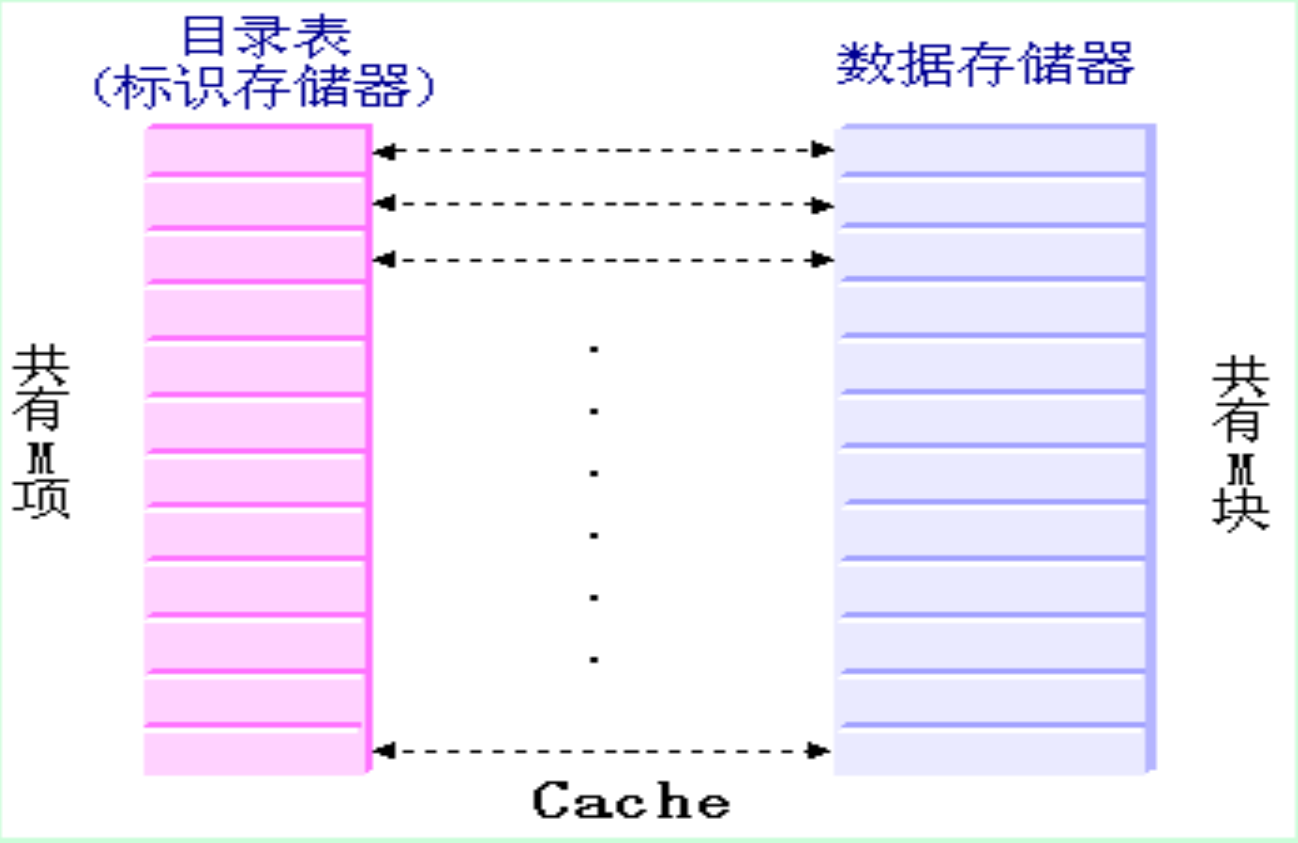
- 通过查找目录表来实现
  - 目录表的结构
    - 主存块的块地址的高位部分，称为标识。
    - 每个主存块能唯一地由其标识来确定
    - 每一项有一个有效位，指出Cache中的块是否有效
  - 只需查找候选位置所对应的目录表项



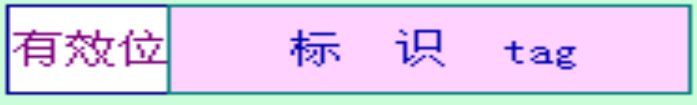


- 块内地址偏移用来从块中选出需要的数据
- 索引域用来选择组
- 通过比较标识域来判断是否发生命中
- 块内偏移是没有必要比较的，这是因为Cache命中与否的单位是整个块
- 由于索引域是用来选择被检查的组， 所以对索引域的比较是多余的

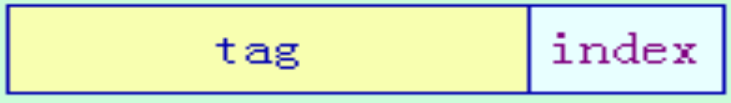
Cache目录表的结构



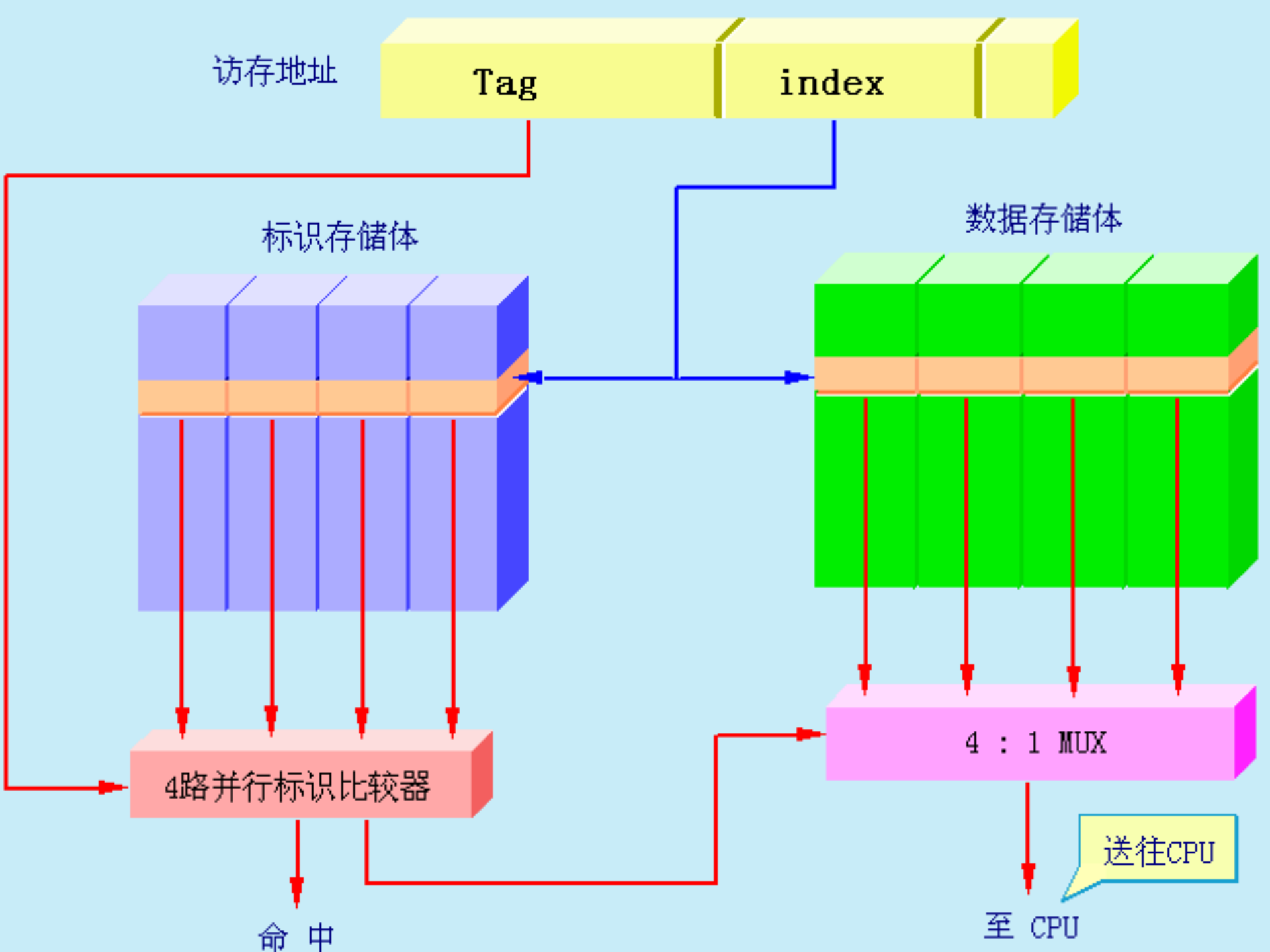
目录表项：

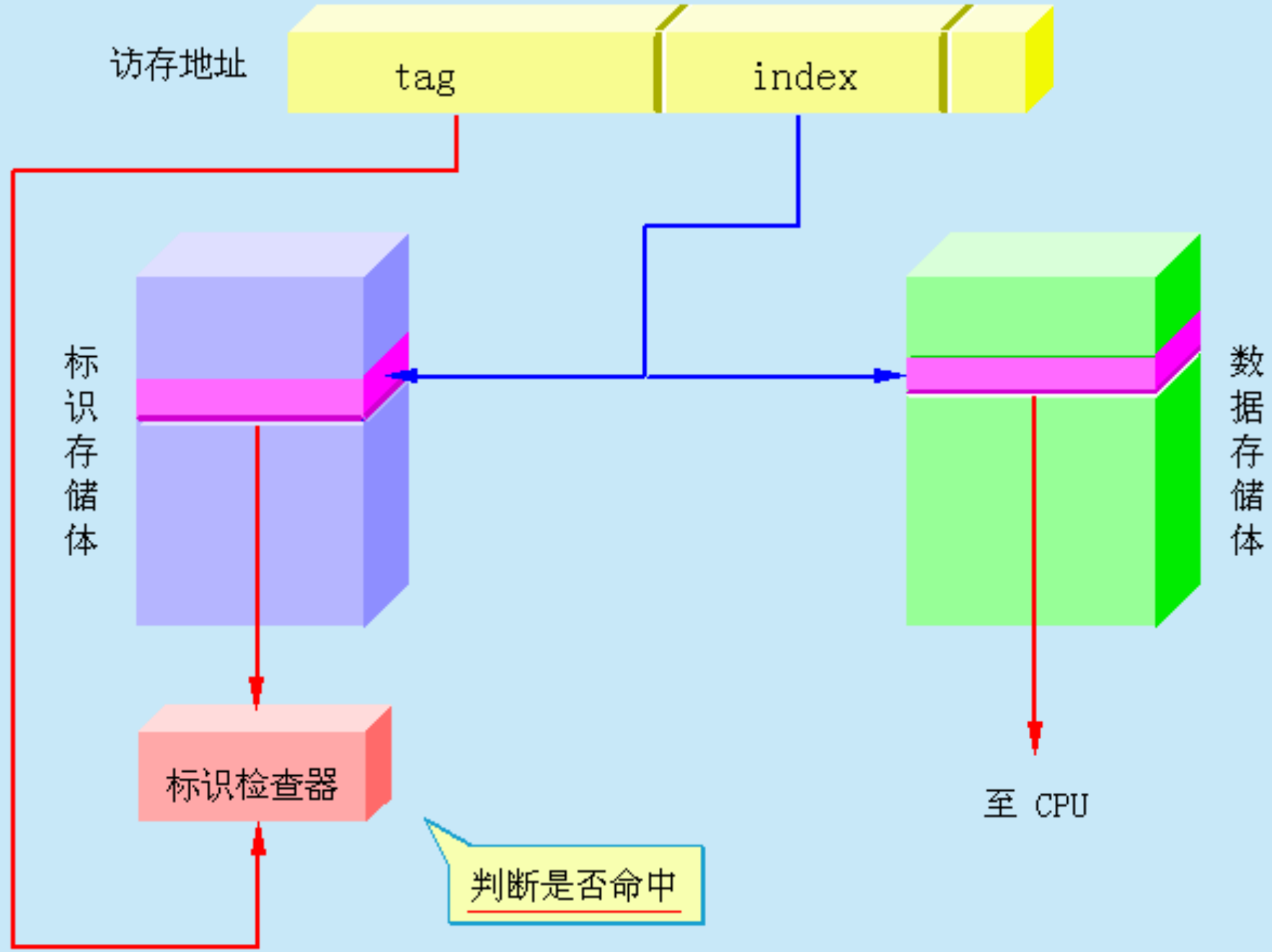


访存地址：

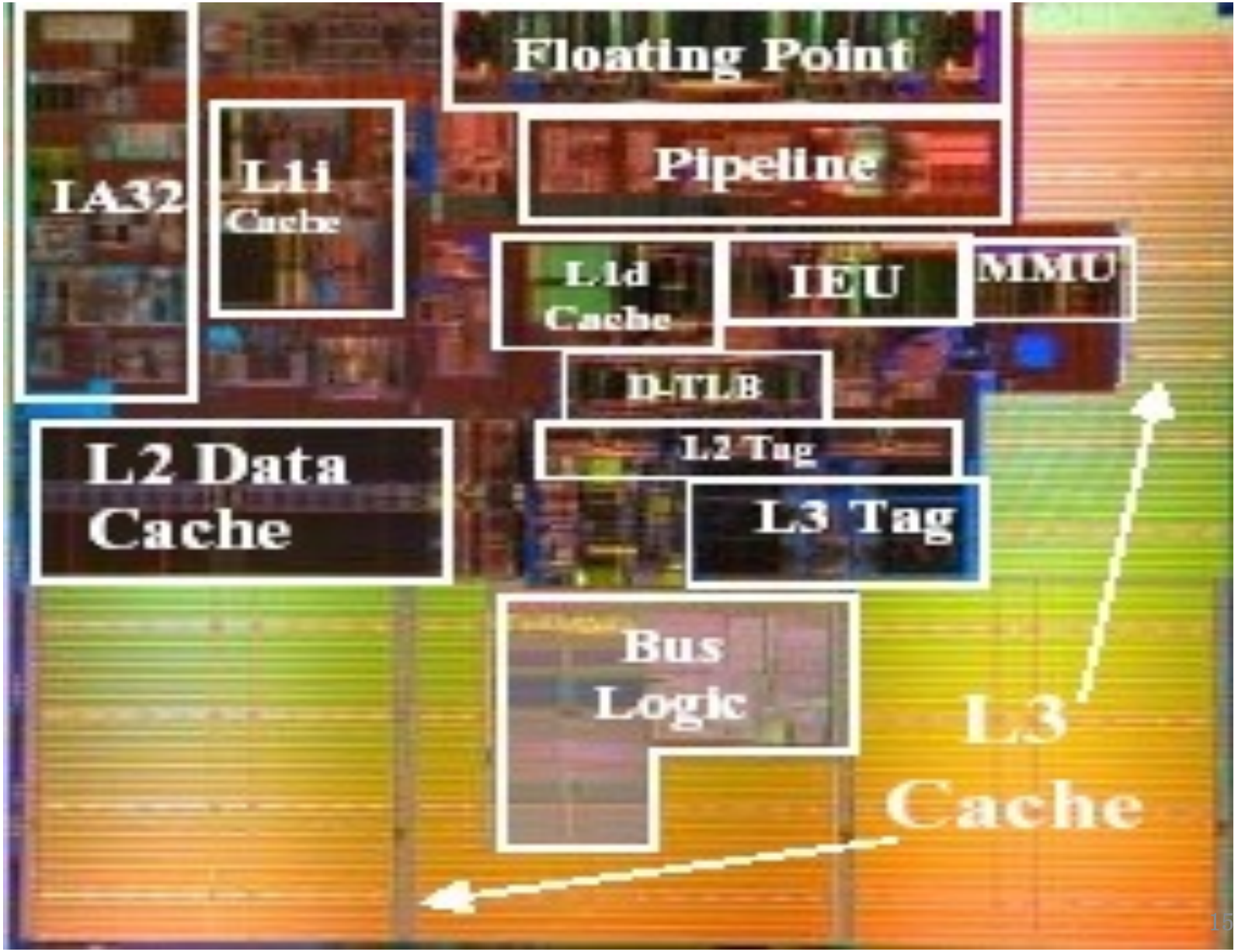


- 4 路组相联Cache的查找过程
  - 优缺点
    - 不必采用相联存储器，而是用按地址访问的存储器来实现。
    - 当相联度n增加时，不仅比较器的个数会增加，而且比较器的位数也会增加。
- 直接映象Cache的查找过程


访存地址
Tag
index
标识存储体
数据存储体
4路并行标识比较器
4 : 1 MUX
送往CPU
命 中
至 CPU


访存地址
tag
index
标识存储体
数据存储体
标识检查器
至 CPU
判断是否命中

Intel Itanium2（版图布局）



例—相联度对命中率影响

4.3

- 四个块组成的Cache
- 比较在直接相联、2路组相联，全相联命中情况
- 主存块到来顺序: 0, 8, 0, 6, 8

直接相联

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

例—相联度对命中率影响

4.3

- 2路-组相联

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- 全相联

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

例题 4.3

- 某计算机字长32位，采用直接映射cache,主存容量4MB，cache数据存储体容量为4KB，字块长度为8个字。
  1. 画出直接映射方式下主存地址划分情况。
  2. 设cache初始状态为空，若CPU顺序访问0-99号单元，并从中读出100个字，假设访问主存一次读一个字，并重复此顺序10次，请计算cache命中率。
  3. 如果cache的存取时间是20ns，主存访问时间是200ns，平均访问时间是多少。
  4. cache-主存系统访问效率。

三、替换算法

4.3

所要解决的问题：当新调入一块，而该块能够占用的Cache位置已被占满时，替换哪一块?

- **直接映象Cache中的替换很简单**

  因为只有一个块，别无选择。

- **在组相联和全相联Cache中，则有多个块供选择。**

– 主要的替换算法有三种

- **随机法**

  优点：实现简单

- **先进先出法FIFO**
- **最近最少使用法LRU**

四、写策略

4.3

1. “写”操作所占的比例

Load指令：26%　　Store指令：9%

“写”在所有访存操作中所占的比例：

9%/(100%＋26%＋9%)≈7%

“写”在访问数据Cache操作中所占的比例：

9%/(26%＋9%)≈25%

2. “写”操作必须在确认是否命中后才可进行

3. “写”访问有可能导致Cache和主存内容的不一致

4. 两种写策略

- 写直达法：执行“写”操作时，不仅写入Cache，而且也写入下一级存储器。
- 写回法：执行“写”操作时，只写入Cache。仅当Cache中相应的块被替换时，才写回主存。（设置“脏位”）

5. 两种写策略的比较

- 写回法的优点：速度快，占用存储器频带低
- 写直达法的优点：易于实现，一致性好

6. 写缓冲器

7. “写”操作时的调块

- 按写分配(写时取)：写失效时，先把所写单元所在的块调入Cache，再行写入。
- 不按写分配(绕写法)：写失效时，直接写入下一级存储器而不调块。

8. 写策略与调块

写回法 —— 按写分配

写直达法 —— 不按写分配

五、Cache结构举例

例子：DEC的Alpha AXP21064中的内部数据Cache

1. 简介

容量：8KB

块大小：32B

块数：256

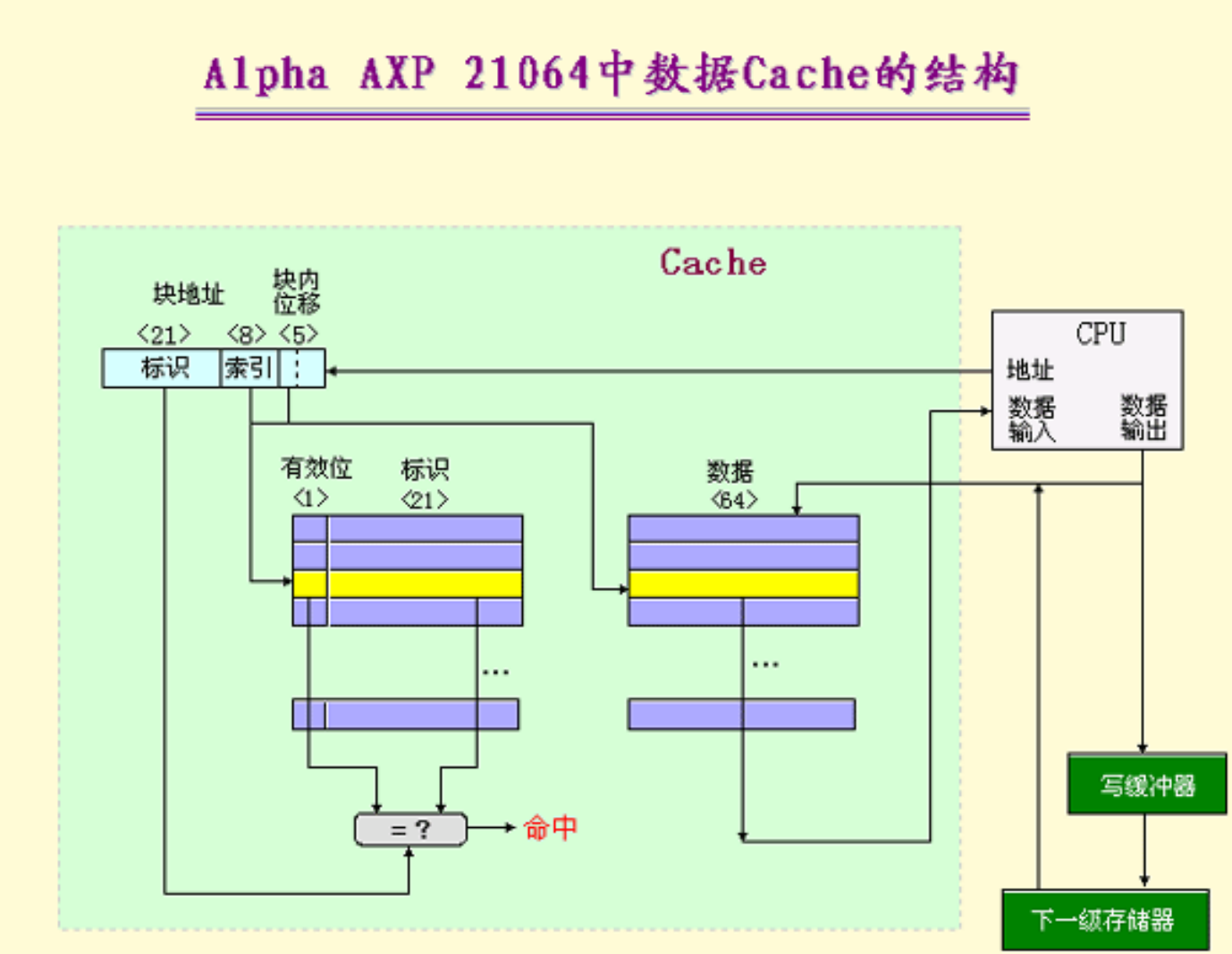
映射方法：直接映射

“写”策略：写直达—不按写分配

写缓冲器大小：4个块

内存地址：34位（块地址29位，块内地址5位）

结构图


Alpha AXP 21064中数据Cache的结构
Cache
块地址
块内位移
〈21〉
〈8〉
〈5〉
标识
索引
有效位
〈1〉
标识
〈21〉
数据
〈64〉
CPU
地址
数据输入
数据输出
写缓冲器
下一级存储器
= ?
命中

3. 工作过程

① 处理器传送给Cache物理地址

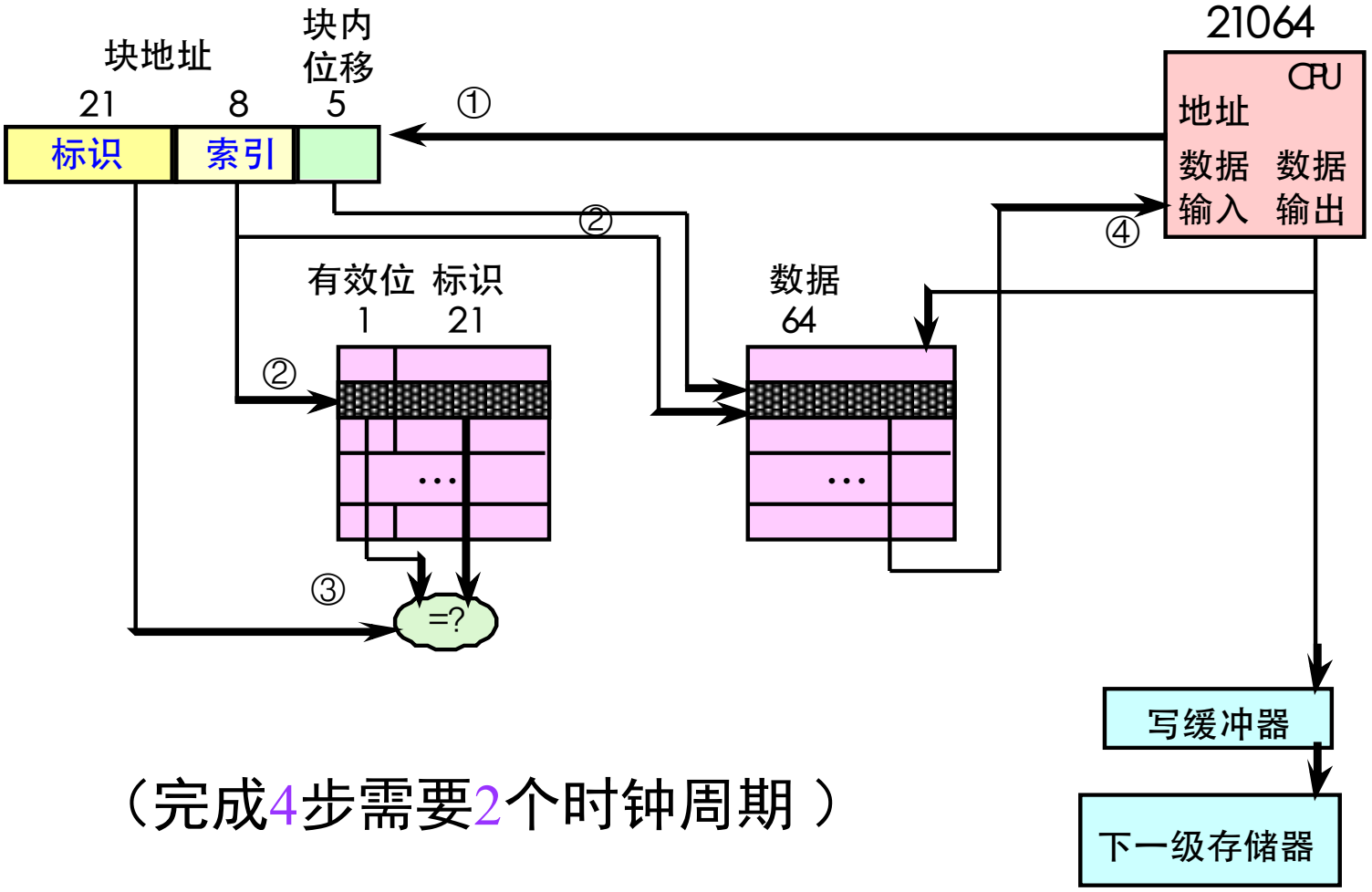
- Cache的容量与索引index、相联度、块大小之间的关系

  Cache的容量=2^{index}×相联度×块大小

把容量为8192、相联度为1、块大小为32（字节）代入：

索引index：8位　　标识：29－8＝21位

3. 工作过程



（完成4步需要2个时钟周期）

3. 工作过程

① 处理器传送给Cache物理地址

② 由索引选择标识的过程

- 根据索引从目录项中读出相应的标识和有效位

③ 从Cache中读出标识之后，用来同从CPU发来的块地址中标志域部分进行比较

- 为了保证包含有效的信息，必须要设置有效位

④ 如果有一个标识匹配，且标志位有效，则此次命中

- 通知CPU取走数据

- “写”访问命中
  - 前三步一样，只有在确认标识匹配后才把数据写入
- 设置了一个写缓冲器

  （提高“写”访问的速度）
  - 按字寻址的，它含有4个块，每块大小为4个字。
  - 当要进行写入操作时，如果写缓冲器不满，那么就把数据和完整的地址写入缓冲器。对CPU而言，本次“写”访问已完成，CPU可以继续往下执行。由写缓冲器负责把该数据写入主存。
  - 在写入缓冲器时，要进行写合并检查。即检查本次写入数据的地址是否与缓冲器内某个有效块的地址匹配。如果匹配，就把新数据与该块合并 。

发生读不命中与写不命中时的操作

- **读不命中：** 向CPU发出一个暂停信号，通知它等待，并从下一级存储器中新调入一个数据块（32字节）
  - Cache与下一级存储的数据通路宽度为16B，传送一次需5个周期，因此，一次传送需要10个周期
- **写不命中：** 将使数据“绕过”Cache，直接写入主存。
  - 写直达 –不按写分配
- 因为是写直达，所以替换时不需要写回

六、改进Cache性能

平均访存时间＝命中时间＋失效率×失效开销

可以从三个方面改进Cache的性能：

(1) 降低失效率

例如：增加块大小、提高相联度

(2) 减少失效开销

例如：多级Cache、写缓冲及写合并、

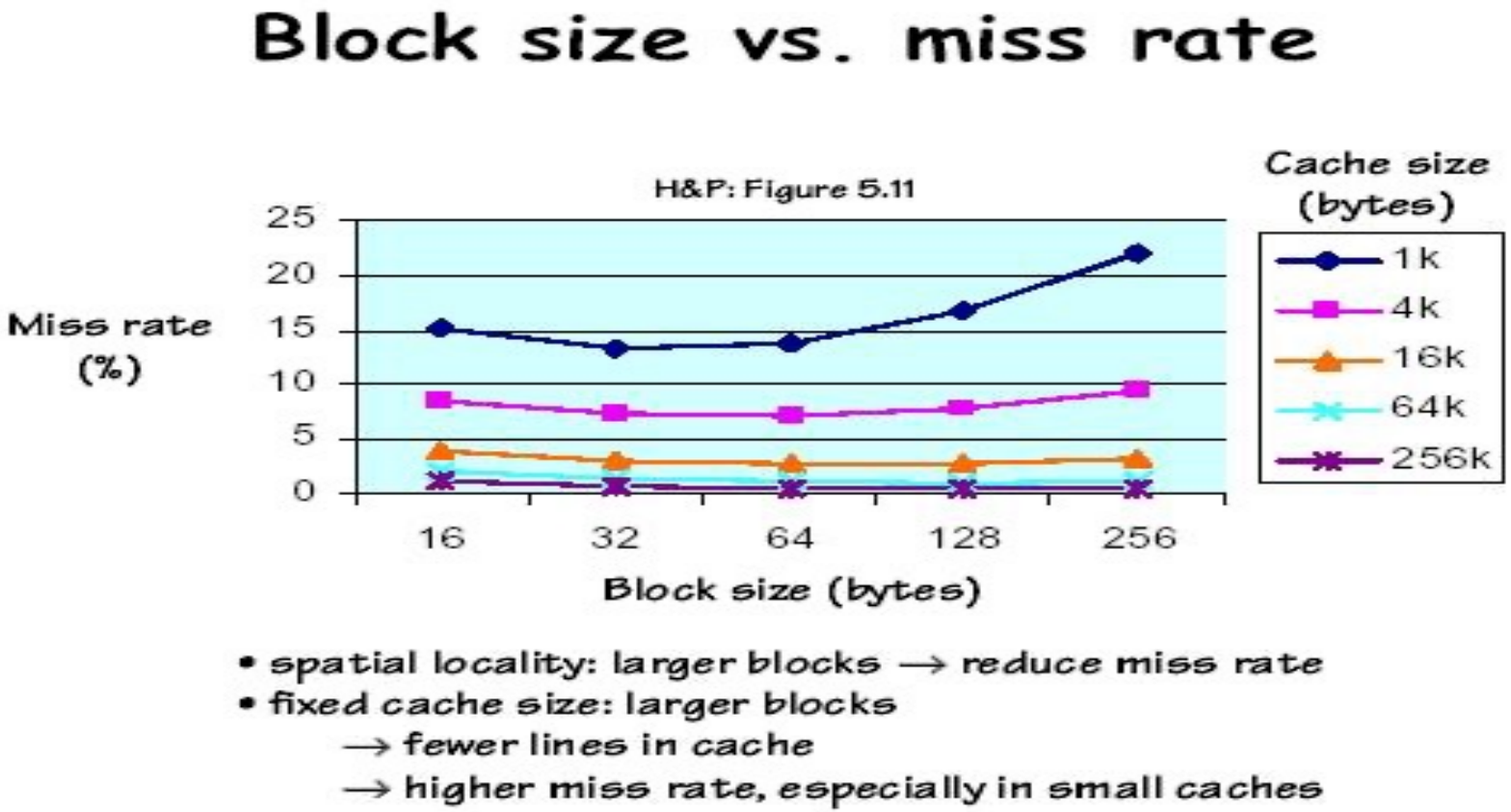
请求字优先

(3) 减少Cache命中时间

例如：容量小且结构简单的Cache

Cache大小、Block大小和缺失率的关系 4.3

缺失率与Cache大小、Block大小等有关



Cache大小：Cache越大，Miss率越低，但成本越高！

Block大小：Block大小与Cache大小有关，且不能太大，也不能太小！

访存时间(时钟周期数）和Cache容量及相联度的关系（举例）

Cache容量（KB）	相联度（路）			
	1	2	4	8
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44

Cache系统

4.3

- 多Cache系统中，需考虑两个方面：

[1] 单级/多级？

外部(Off-chip)Cache:不做在CPU内而是独立设置一个Cache

片内(On-chip)Cache: 将Cache和CPU作在一个芯片上

单级Cache：只用一个片内Cache

多级Cache：同时使用L1 Cache和L2 Cache，甚至有L3 Cache，
L1 Cache更靠近CPU，其速度比L2快，其容量比L2小

[2] 联合/分立？

分立：指数据和指令分开存放在各自的数据和指令Cache中

一般L1 Cache都是分立Cache

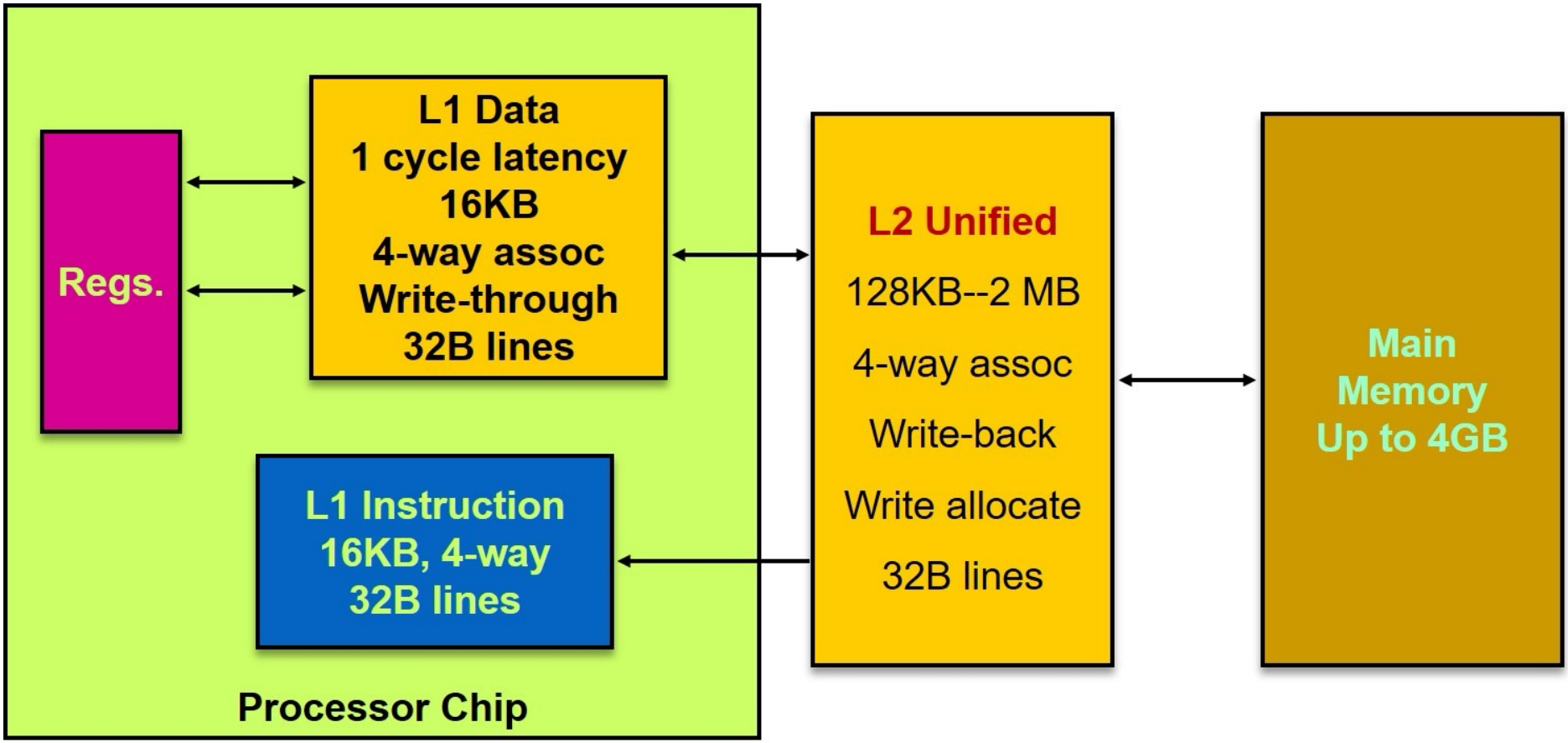
L1 Cache的命中时间比命中率更重要！

联合：指数据和指令都放在一个Cache中

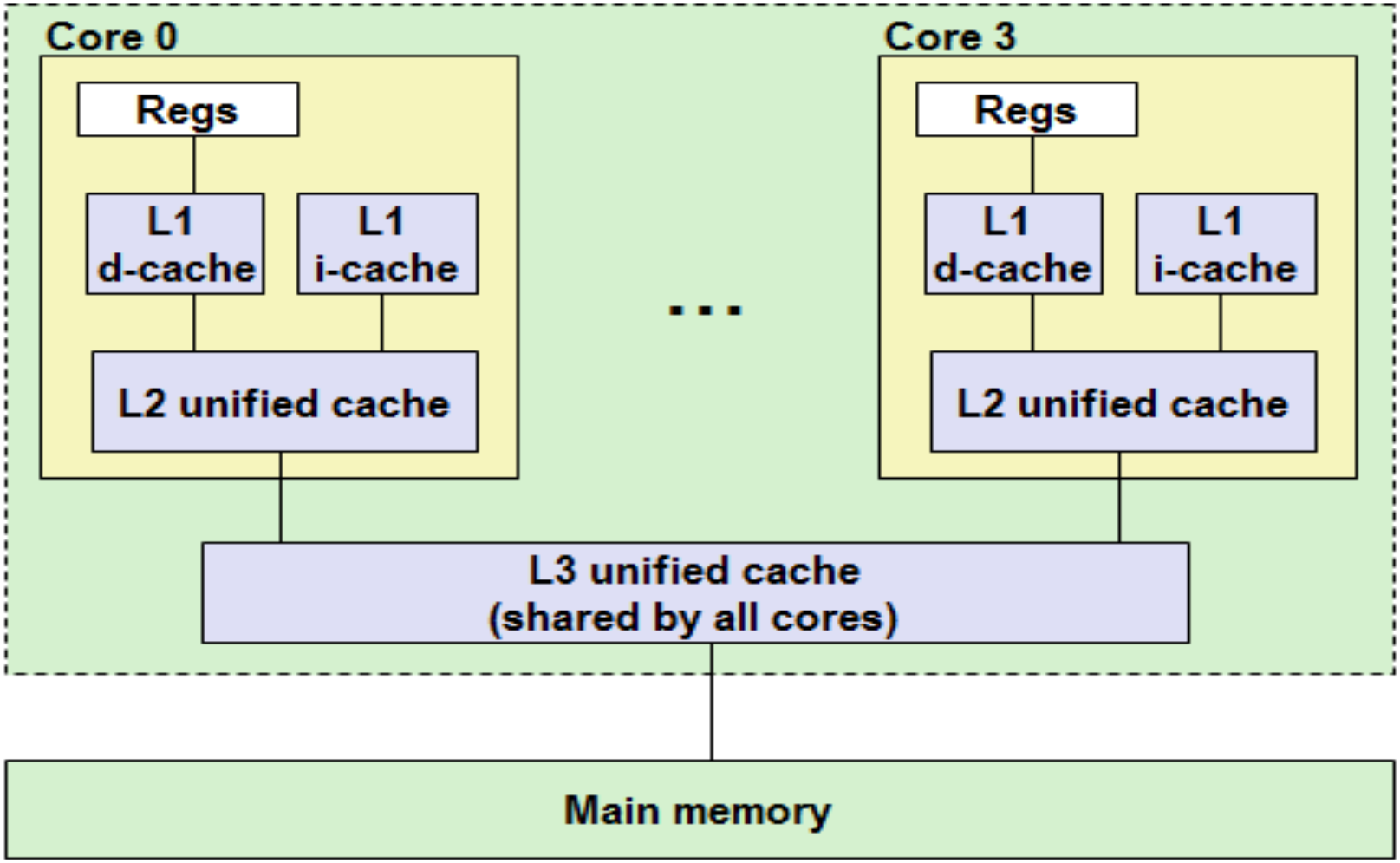
一般L2 Cache都是联合Cache

L2 Cache的命中率比命中时间更重要！

Intel Pentium cache hierarchy



实例：Intel Core i7处理器的cache结构 4.3



i-cache和d-cache都是32KB、8路、4个时钟周期；L2 cache：256KB、8路、11个时钟周期。所有核共享的L3 cache：8MB、16路、30~40个时钟周期。Core i7中所有cache的块大小都是64B

4.4 虚拟存储器

- **虚拟存储器概述**
- 虚拟存储器地址映射与变换
- 举例

虚拟存储系统的基本概念

- 虚拟存储技术的实质
  - 程序员在比实际主存空间大得多的逻辑地址空间中编写程序
  - 程序执行时，把当前需要的程序段和相应的数据块调入主存，其他暂不用的部分存放在磁盘上
  - 指令执行时，通过硬件将逻辑地址（也称虚拟地址或虚地址）转化为物理地址（也称主存地址或实地址）
  - 在发生程序或数据访问失效(缺页)时，由操作系统进行主存和磁盘之间的信息交换
- 虚拟存储器机制由硬件与操作系统共同协作实现，涉及到操作系统中的许多概念，如进程、进程的上下文切换、存储器分配、虚拟地址空间、缺页处理等。

虚拟存储器管理方式 4.4

- 页式管理
  - 页式虚存把主存与外存均划分成等长的页面。
  - 常用页大小为**4KB～64KB**。
  - 便于维护，便于生成页表，类似于**cache**的块表
  - 不容易产生碎块，存储空间浪费小
  - 页不是独立的实体，处理、保护和共享不方便
- 段式管理
- 段页式管理

“Cache－主存”与“主存－辅存”层次的区别 4.4

存储层次 / 比较项目	“Cache －主存”层次	“主存－辅存”层次
目　　的	为了弥补主存速度的不足	为了弥补主存容量的不足
存储管理实现	主要由专用硬件实现	主要由软件实现
访问速度的比值(第一级和第二级)	几比一	几百比一
典型的块(页)大小	几十个字节	几百到几千个字节
CPU对第二级的访问方式	可直接访问	均通过第一级
失效时CPU是否切换	不切换	切换到其他进程

存储层次中cache和虚存的典型指标

4.4

参数	一级cache	虚拟存储器
块（页）大小	16~128字节	4096~65536字节
命中时间	1~3个时钟周期	50~150个时钟周期
失效开销	8~150个时钟周期	10^6~10^7个时钟周期
失效率	0.1%~10%	0.00001%~0.001%
地址映射关系	25~45位物理地址映射到14~20位cache地址	32~64位虚地址映射到25~45位物理地址

有关虚拟存储器的四个问题

4.4

◆映象规则

全相联。以降低失效率为主要目标。

◆ 查找算法

页表，段表，TLB。

◆ 替换算法

LRU等。

尽可能减少页故障，OS为每个页面设置“使用位”。

◆ 写策略

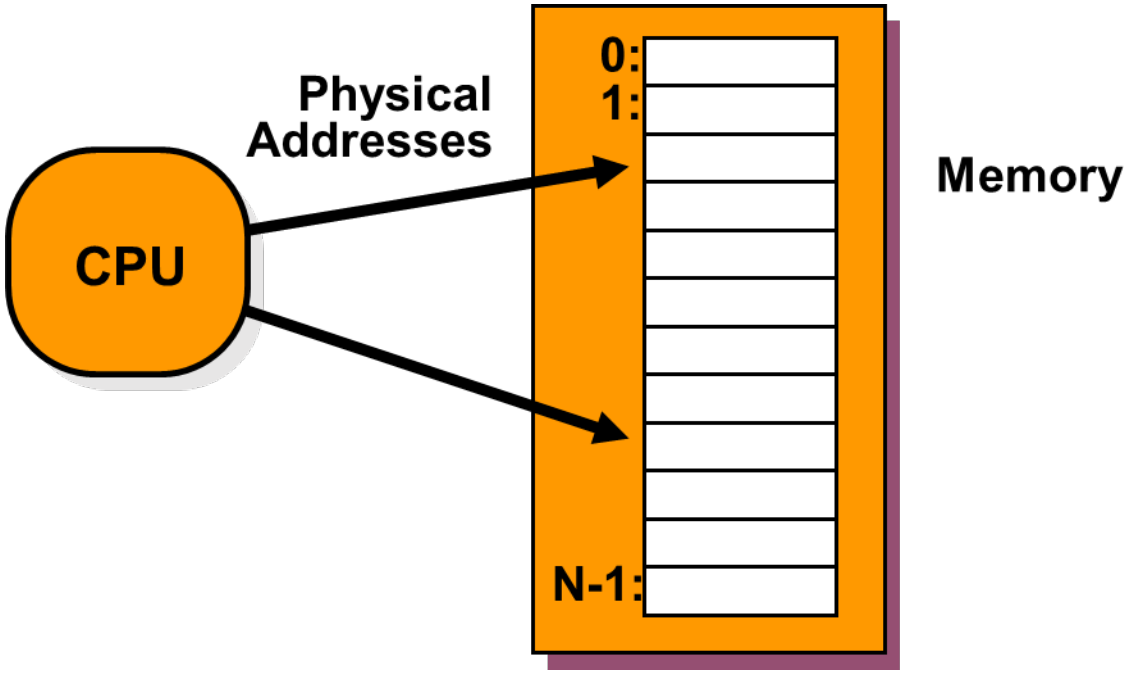
写回法

4.4 虚拟存储器

- 虚拟存储器概述
- **虚拟存储器地址映射与变换**
- 举例

实地址计算机系统

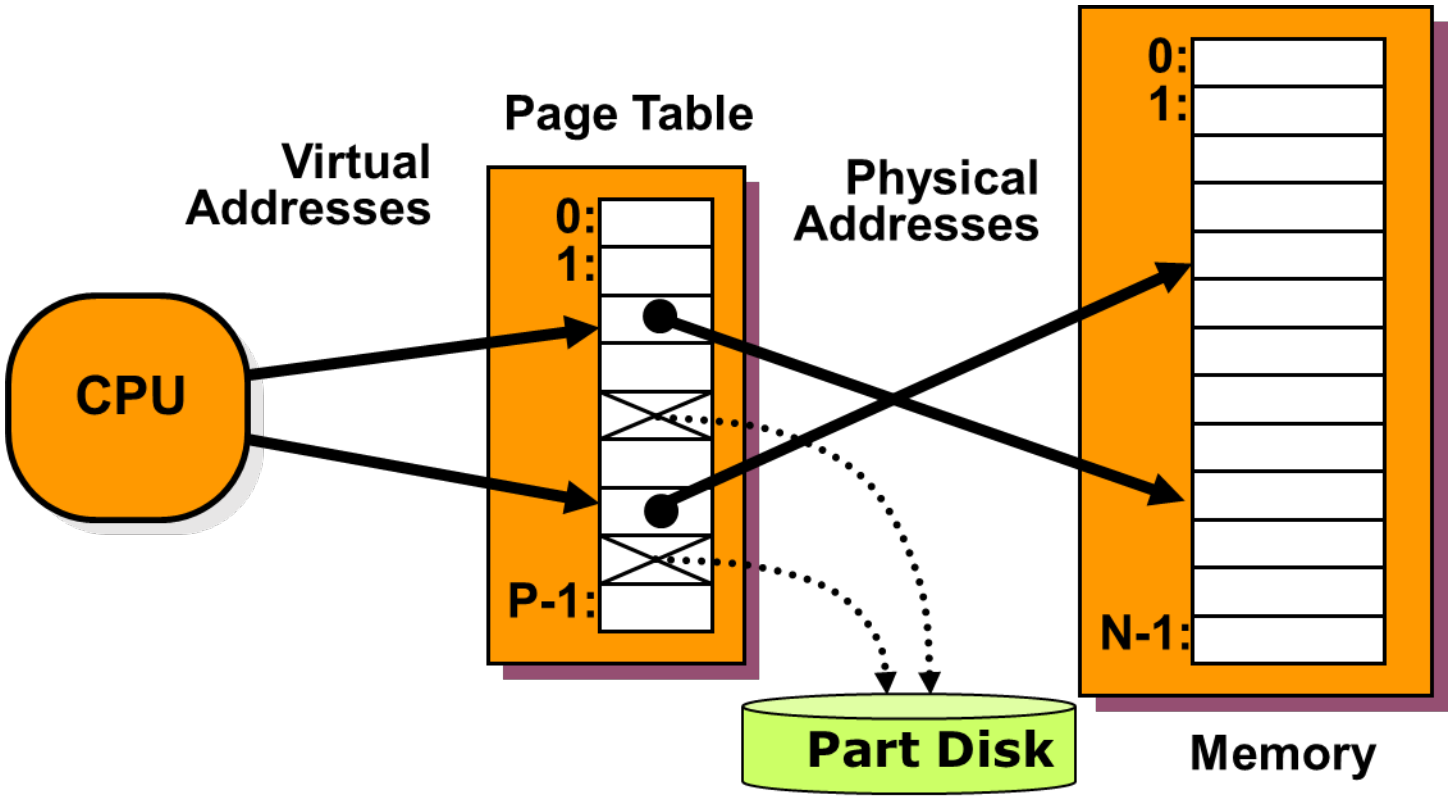
- **CPU地址：物理内存地址**
  - 大多数Cray计算机，早期PC，大多数嵌入式系统



虚地址计算机系统

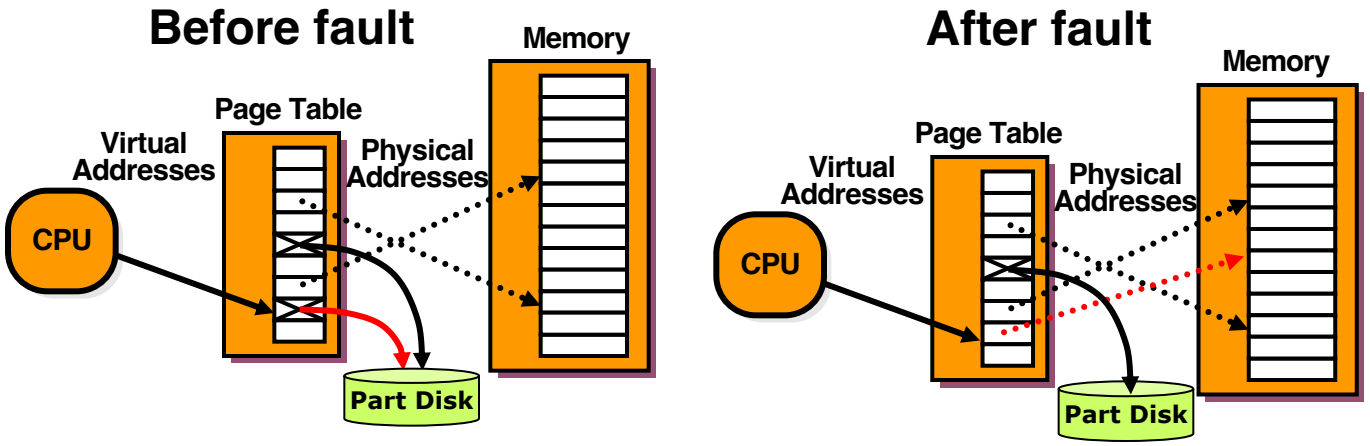
4.4

- **CPU地址：虚拟地址**
  - 需要虚实变换，硬件通过OS维护的页表将虚拟地址转换为物理地址
  - 工作站，服务器，现代PC

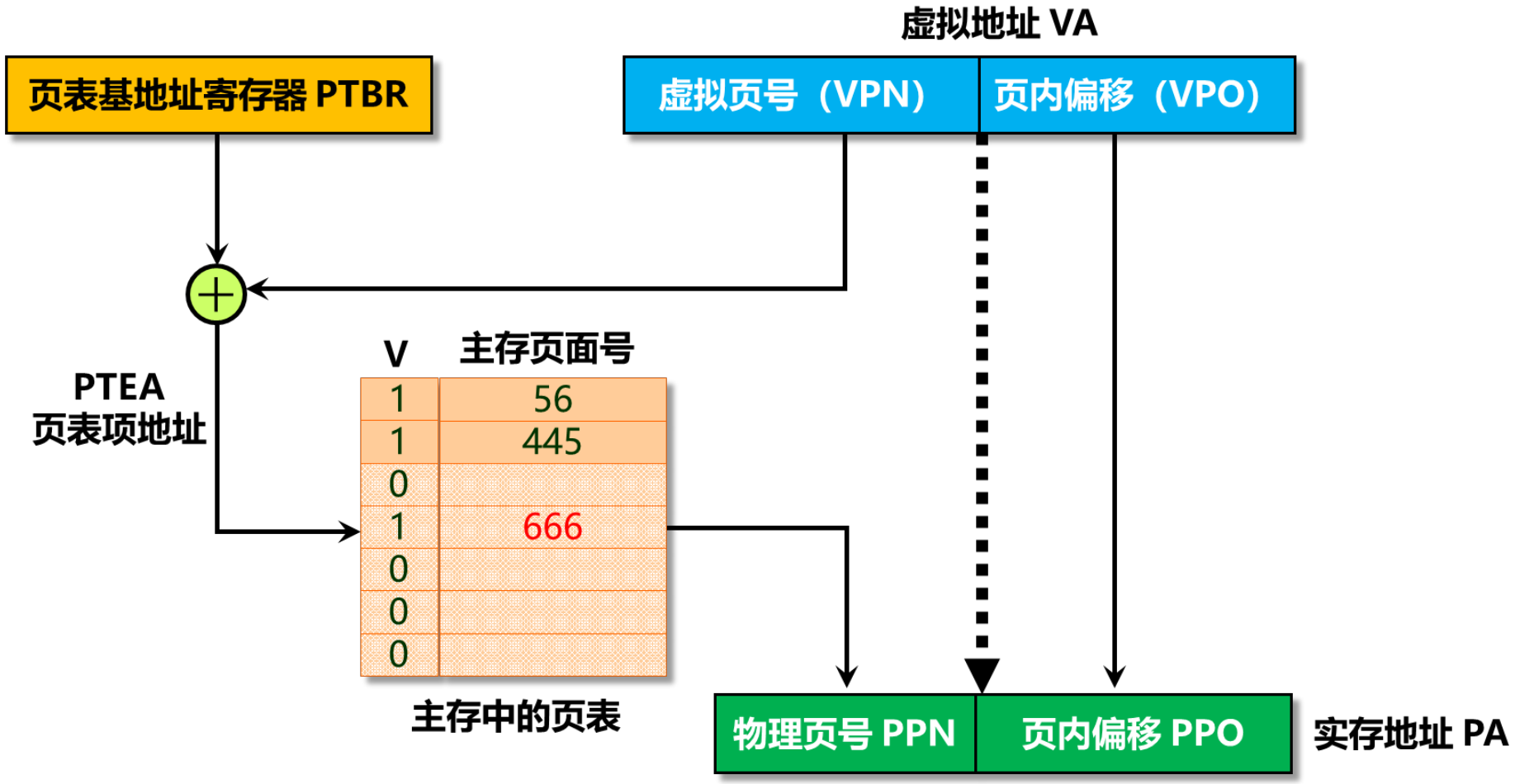


缺页 Page Faults

- 页表指示虚拟地址不在内存中
  - 操作系统负责将数据从磁盘迁移到内存中
  - 当前进程挂起
  - 操作系统负责所有的替换策略
  - 唤醒挂起进程



页式虚拟存储器结构


虚拟地址 VA
页表基地址寄存器 PTBR
虚拟页号（VPN）
页内偏移（VPO）
PTEA
页表项地址
V
主存页面号
1 56
1 445
0
1 666
0
0
0
主存中的页表
物理页号 PPN
页内偏移 PPO
实存地址 PA

页表结构

4.4

◆ **页表首址记录在页表基址寄存器中**

页表首地址

	装入位	修改位	替换控制位	其他	实页号（8 进制）
0 虚页	1				11
1 虚页	1				13
2 虚页	1				16
3 虚页	1				10
4 虚页	1				14

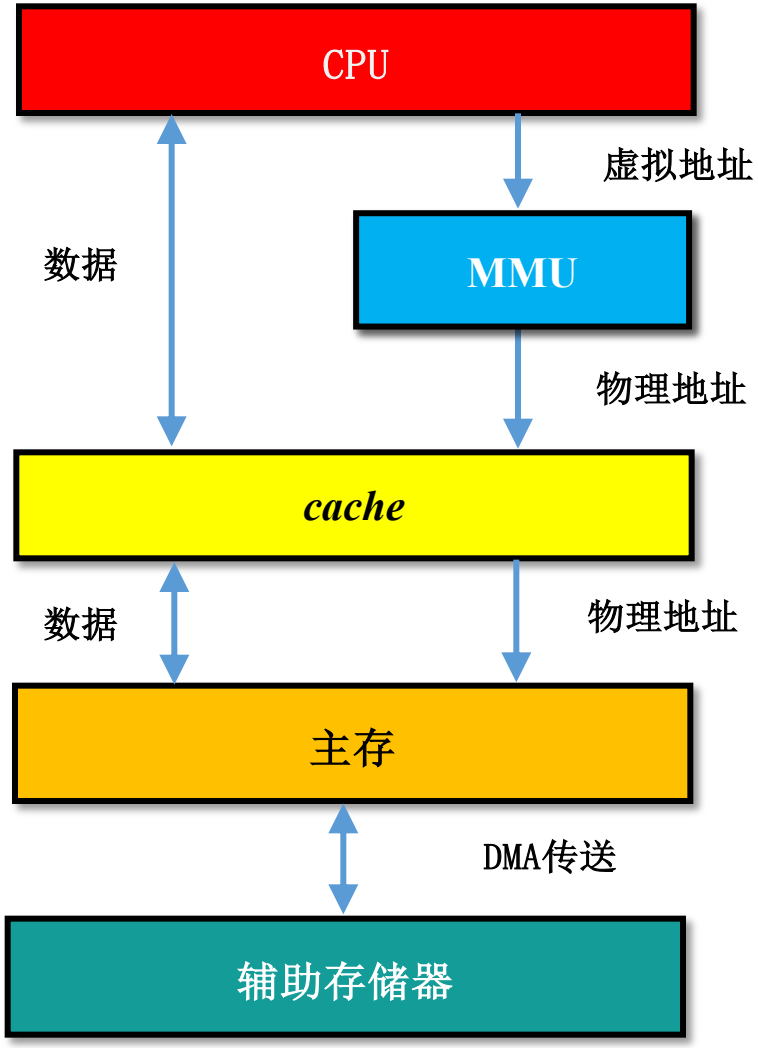
- 每个进程有一个页表，其中有装入位、修改（Dirt）位、替换控制位、访问权限位、禁止缓存位、实页号。
- 一个页表的项数由什么决定？

理论上由虚拟地址空间大小决定。

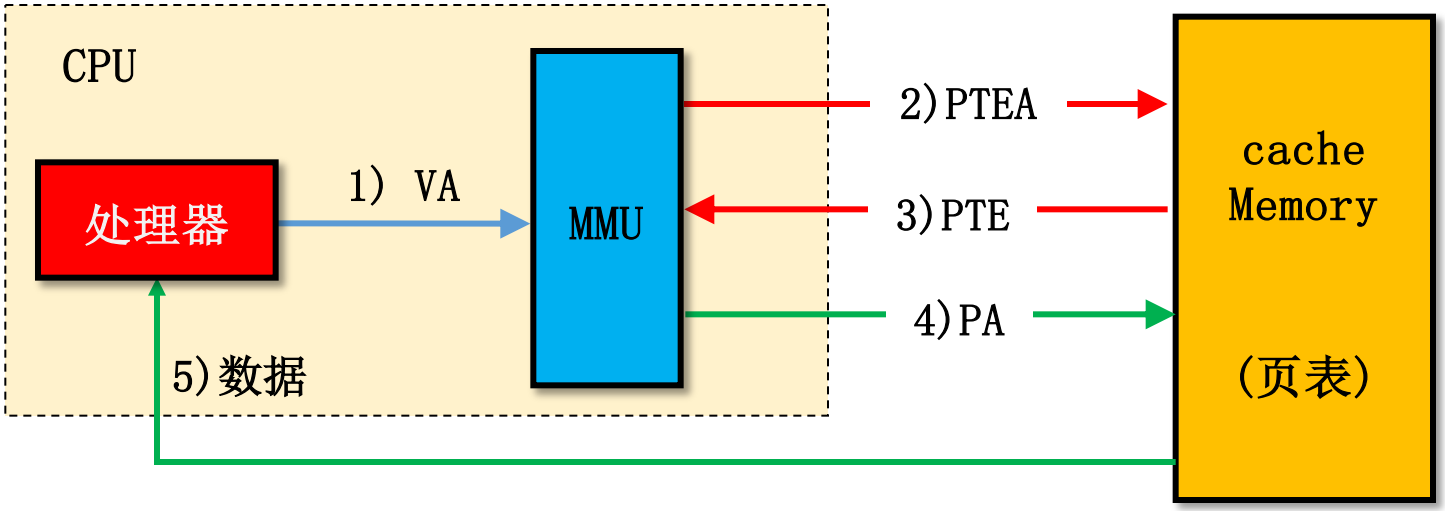
- 每个进程的页表大小一样吗？

各进程有相同虚拟空间，故理论上一样。实际大小看具体实现方式，如“空洞”页面如何处理等

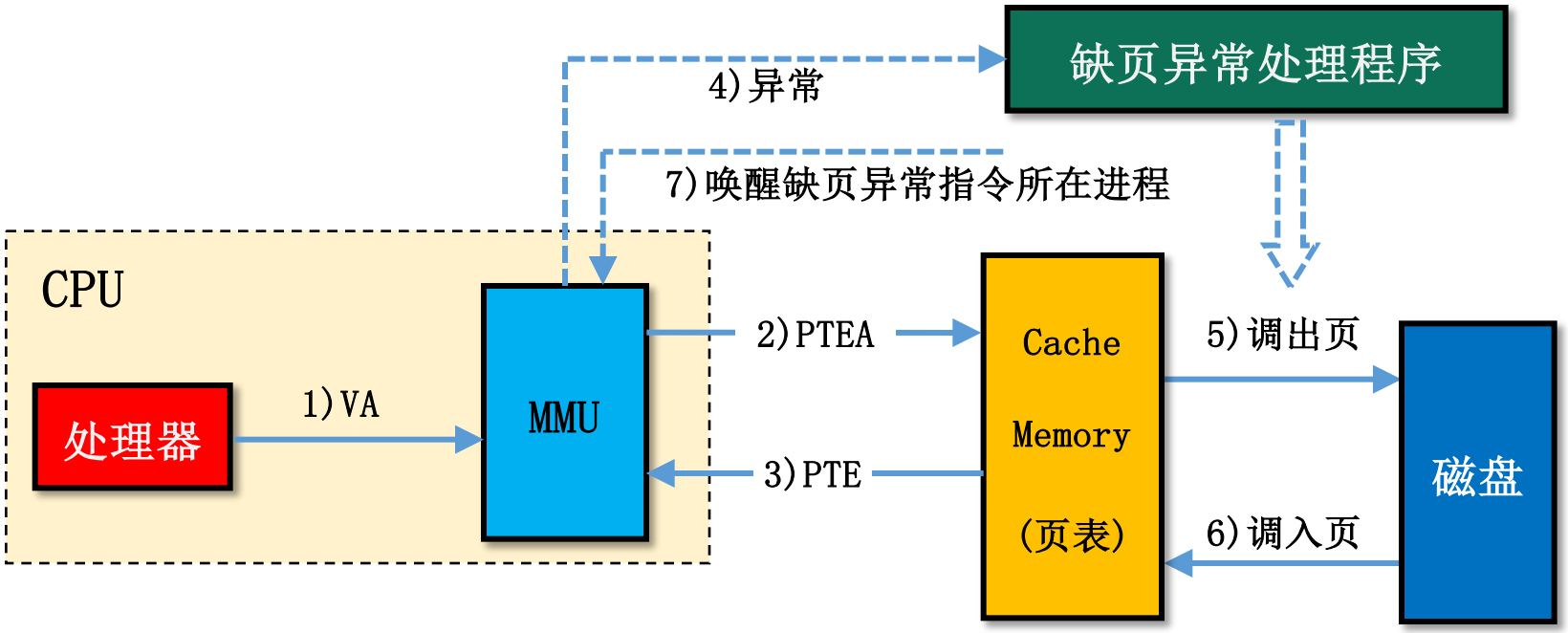
层次性结构



虚拟地址→物理地址 （页命中）



虚拟地址→物理地址（缺页）


缺页异常处理程序
4)异常
7)唤醒缺页异常指令所在进程
CPU
处理器
1)VA
MMU
2)PTEA
3)PTE
Cache
Memory
(页表)
5)调出页
6)调入页
磁盘

快表提高地址转换速度

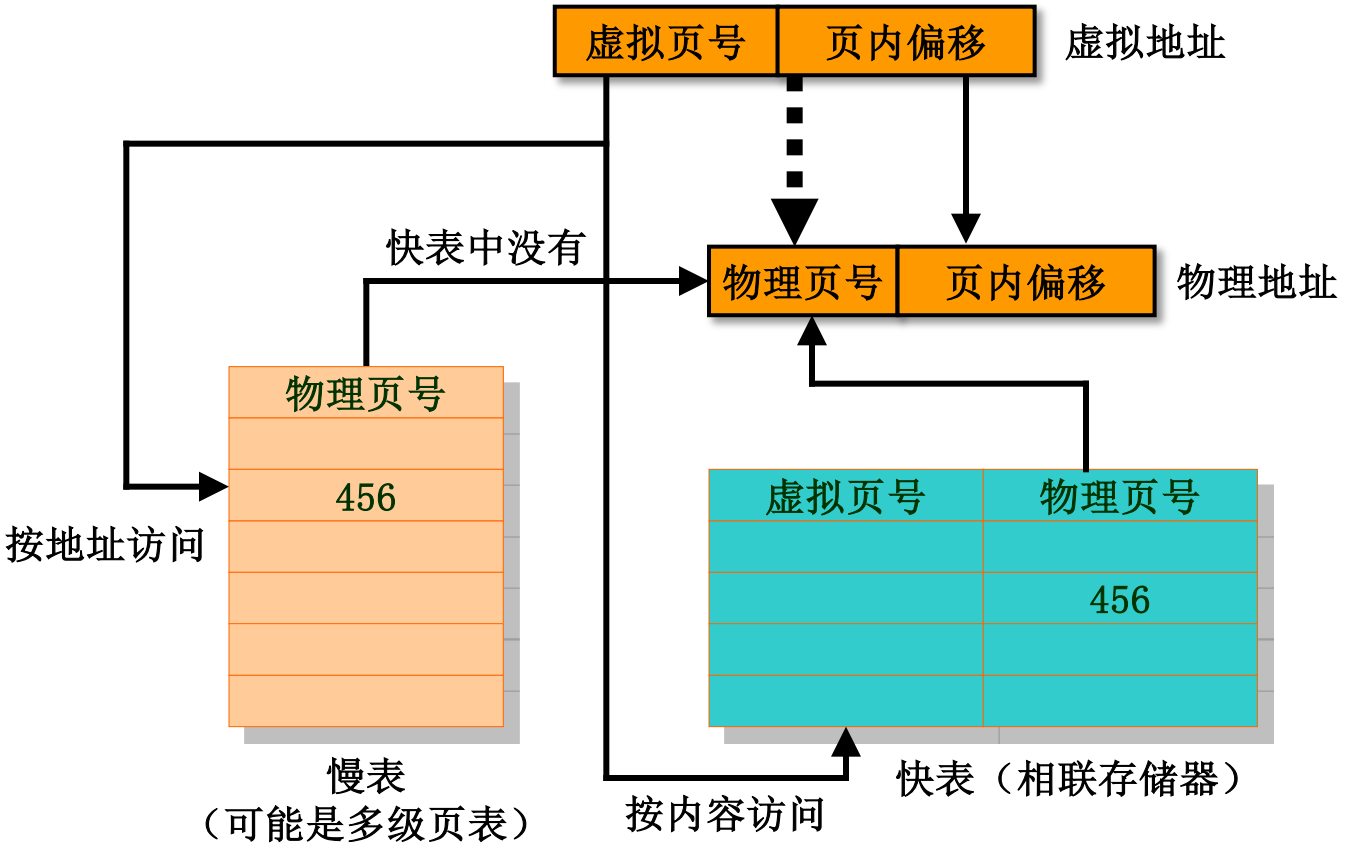
4.4

- **地址转换速度慢**
  - 访问页表，访问数据，需2次访存，速度慢
  - 为缩小页表大小，OS普遍采用多级页表结构，速度更慢
- **加速方法：**

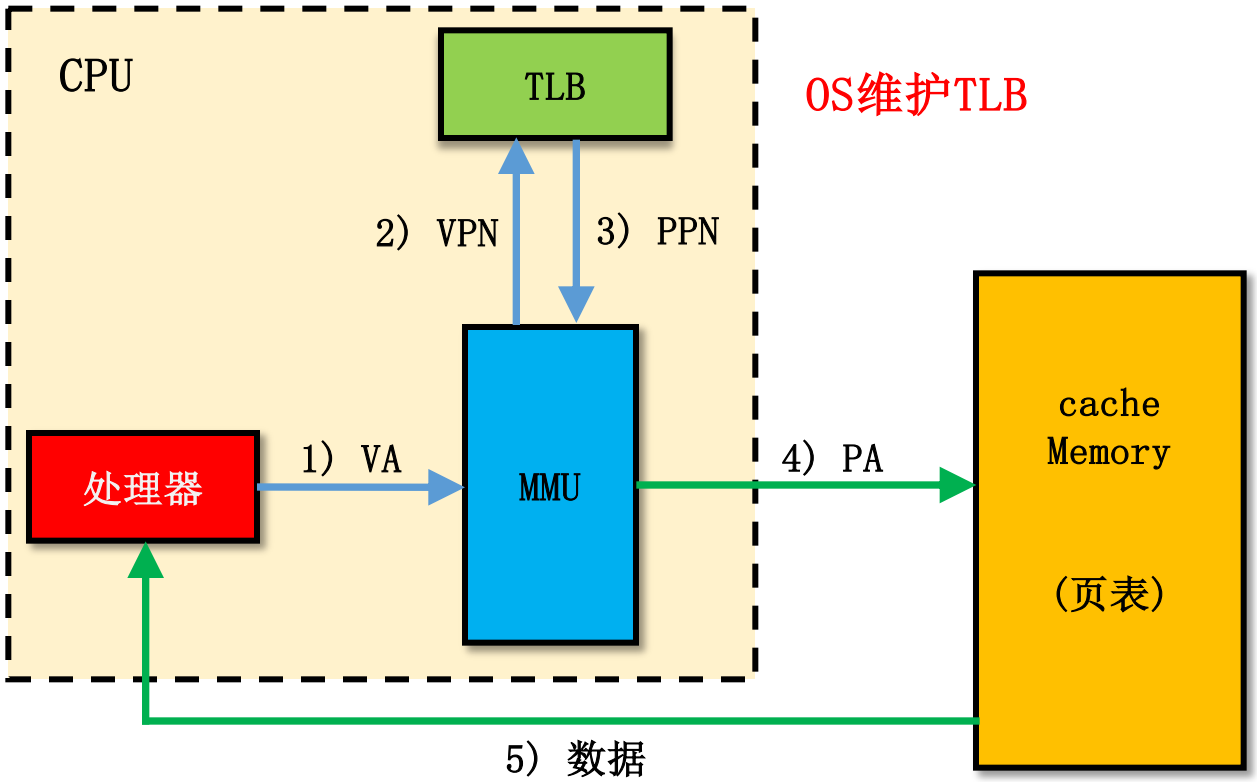
引入一个体积小的快表TLB（ Translation Look-Aside Buffers ）

  - 缓存页表中经常被访问的表项
  - (Valid，VPN，PPN)
- **快表引入相联存储器机制，提高查找速度**
  - 采用随机替换算法

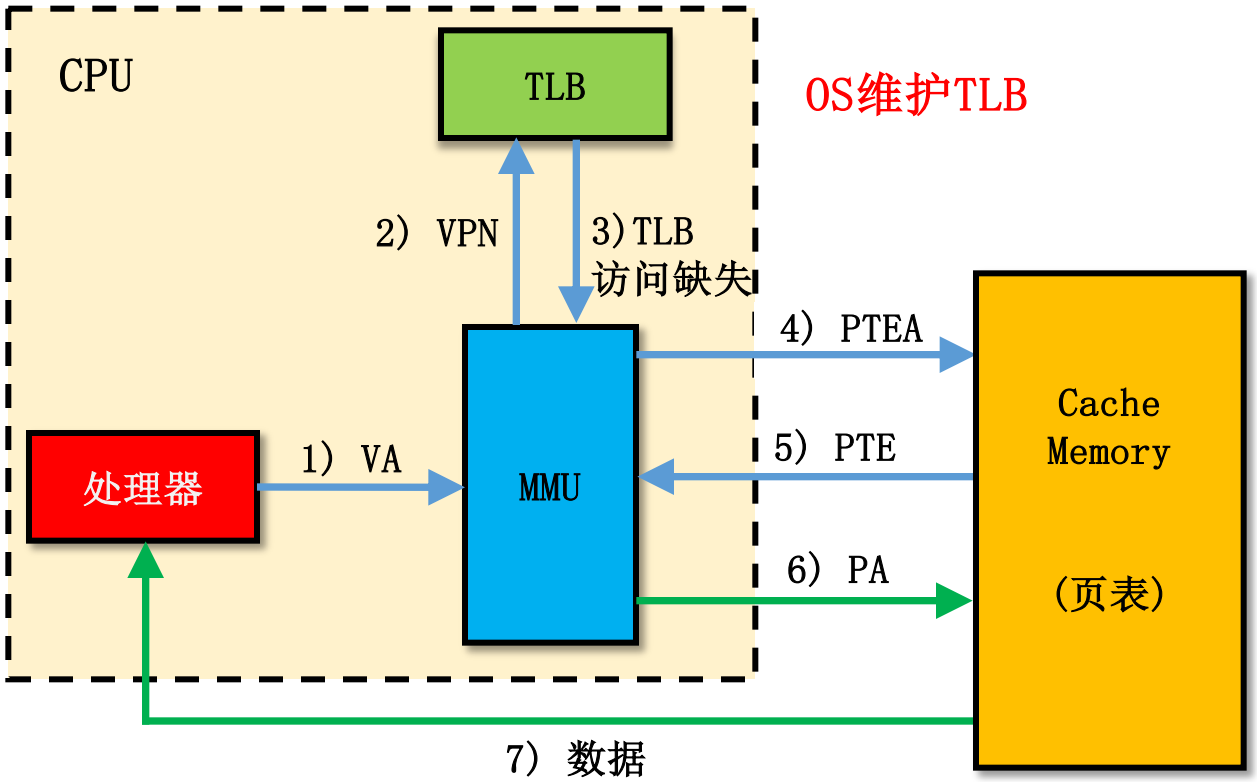
经快慢表实现内部地址转换



TLB 命中

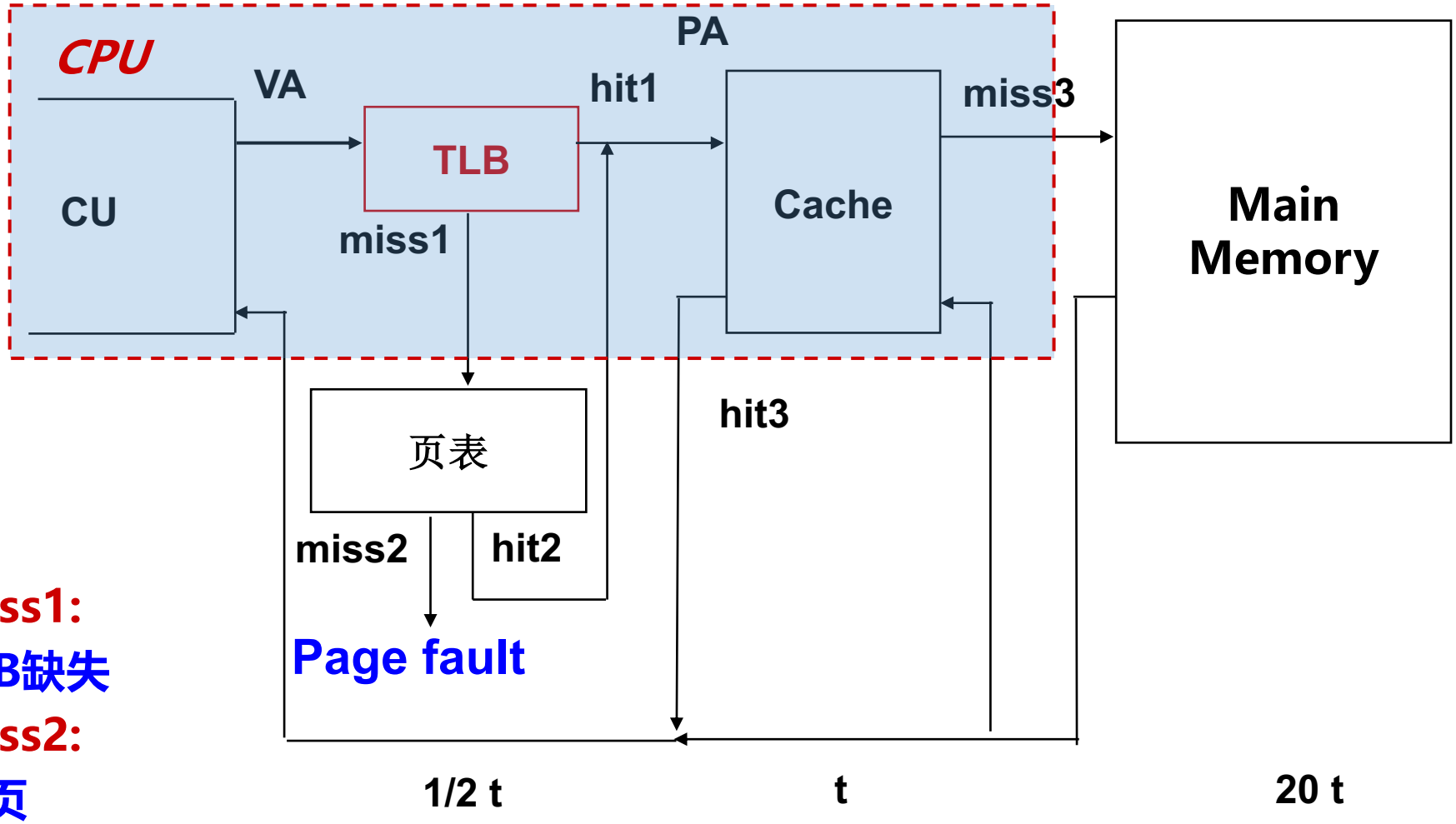


TLB 缺失


CPU
TLB
OS维护TLB
2) VPN
3)TLB
访问缺失
4) PTEA
处理器
1) VA
MMU
5) PTE
Cache
Memory
6) PA
(页表)
7) 数据

Translation Look-Aside Buffers

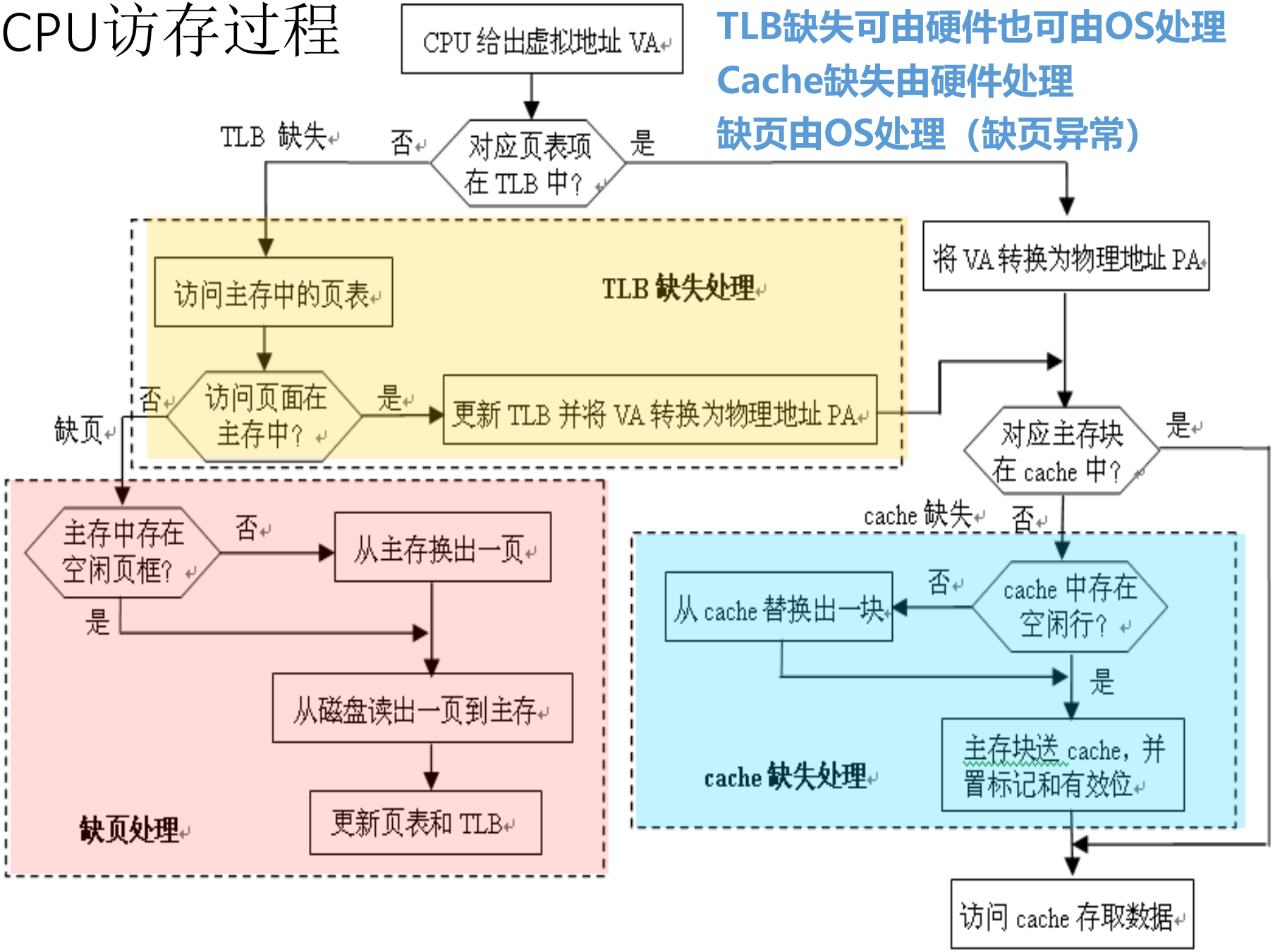
4.4



Miss1:
TLB缺失
Miss2:
缺页
Miss3:
PA 在主存中，但不在Cache中

TLB冲刷指令和Cache冲刷指令都是操作系统使用的特权指令

CPU访存过程



TLB缺失可由硬件也可由OS处理

Cache缺失由硬件处理

缺页由OS处理（缺页异常）

举例：三种不同缺失的组合

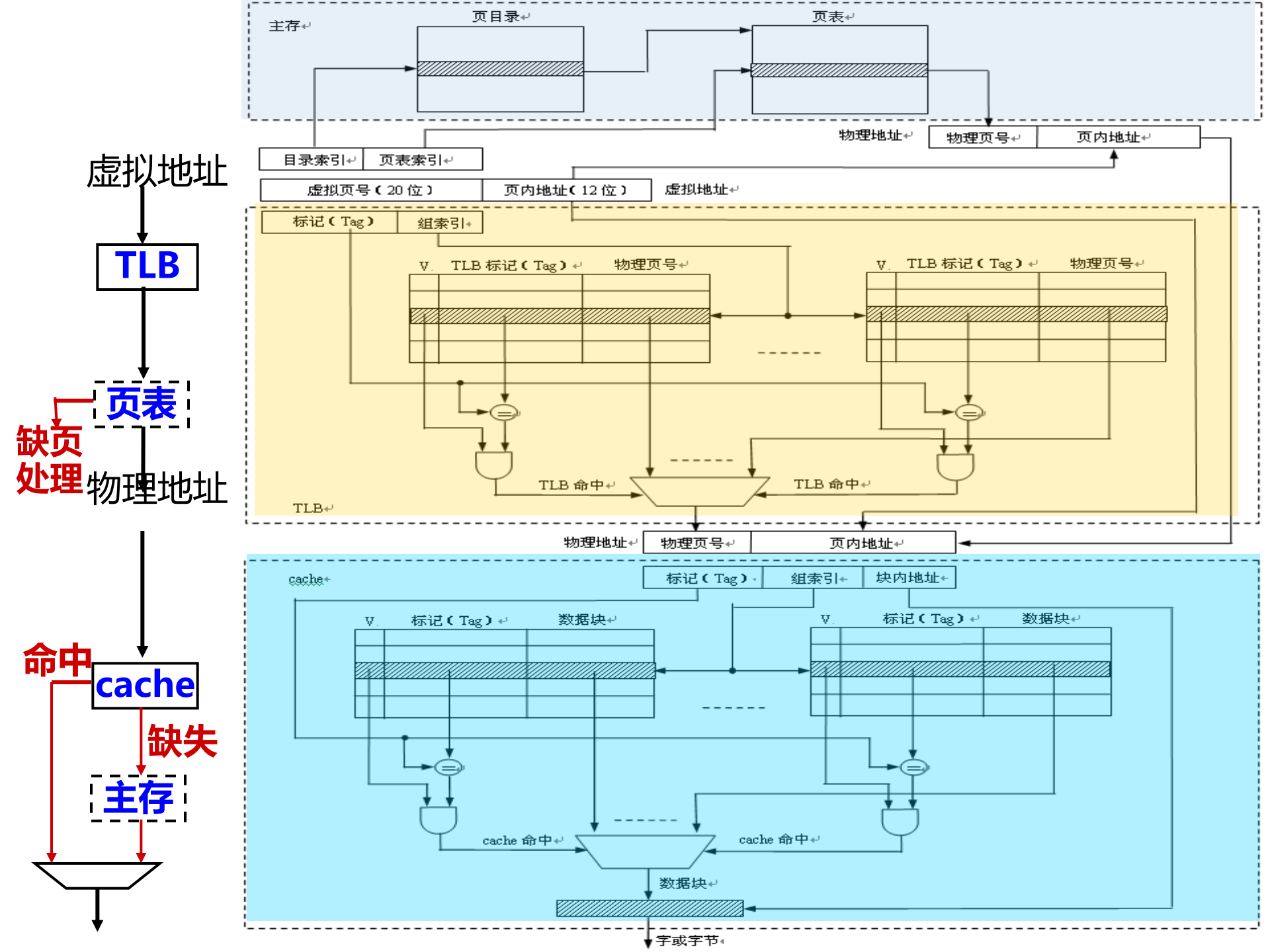
TLB	Page table	Cache	Possible? If so, under what circumstance?
hit	hit	miss	可能，TLB命中则页表一定命中，但实际上不会查页表
miss	hit	hit	可能，TLB缺失但页表命中，信息在主存，就可能在Cache
miss	hit	miss	可能，TLB缺失但页表命中，信息在主存，但可能不在Cache
miss	miss	miss	可能，TLB缺失页表缺失，信息不在主存，一定也不在Cache
hit	miss	miss	不可能，页表缺失，信息不在主存，TLB中一定没有该页表项
hit	miss	hit	同上
miss	miss	hit	不可能，页表缺失，信息不在主存，Cache中一定也无该信息

最好的情况是hit、hit、hit，此时，访问主存几次？ 不需要访问主存！

以上组合中，最好的情况是？ hit、hit、miss和miss、hit、hit 访存1次

以上组合中，最坏的情况是？ miss、miss、miss 需访问磁盘、并访存至少2次

介于最坏和最好之间的是？ miss、hit、miss 不需访问磁盘、但访存至少2次


虚拟地址
TLB
页表
缺页处理
物理地址
命中
cache
缺失
主存
主存
页目录
页表
物理地址
物理页号
页内地址
目录索引
页表索引
虚拟页号（20 位）
页内地址（12 位）
虚拟地址
标记（Tag）
组索引
V
TLB 标记（Tag）
物理页号
TLB 命中
TLB
物理地址
物理页号
页内地址
cache
标记（Tag）
组索引
块内地址
V
标记（Tag）
数据块
cache 命中
数据块
字或字节

4.4 虚拟存储器

- 虚拟存储器概述
- 虚拟存储器地址映射与变换
- **举例**

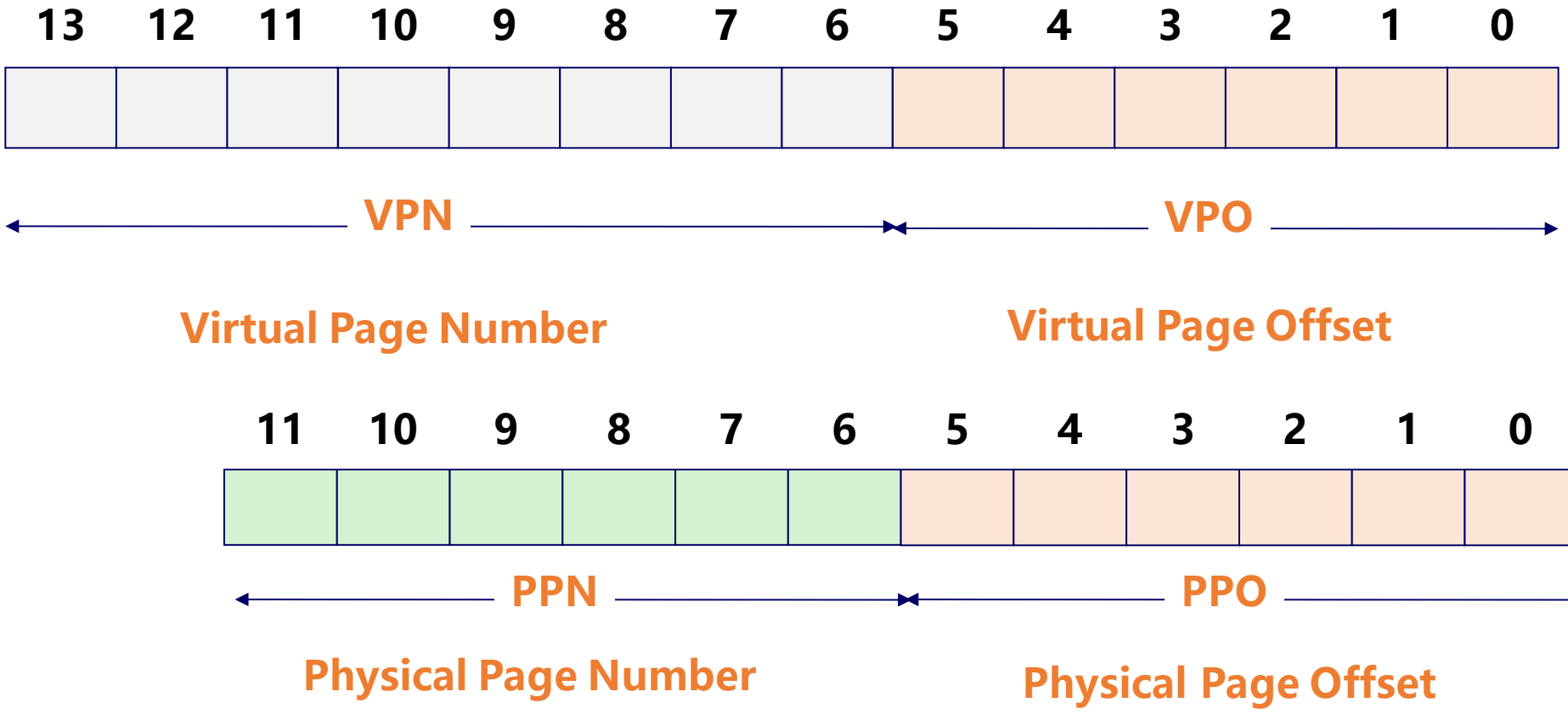
缩写的含义

- 基本参数（按字节编址）
  - $N = 2^n$: 虚拟地址空间大小
  - $M = 2^m$: 物理地址空间大小
  - $P = 2^p$: 页大小
- 虚拟地址 (VA)中的各字段
  - TLBI: TLB index（TLB索引）
  - TLBT: TLB tag（TLB标记）
  - VPO: Virtual page offset （页内偏移地址）
  - VPN: Virtual page number （虚拟页号）
- 物理地址(PA)中的各字段
  - PPO: Physical page offset (页内偏移地址)
  - PPN: Physical page number（物理页号）
  - CO: Byte offset within cache line（块内偏移地址）
  - CI: Cache index（cache索引）
  - CT: Cache tag（cache标记）

一个简化的存储系统举例

- 假定以下参数，则虚拟地址和物理地址如何划分？共多少页表项?
  - 14-bit virtual addresses（虚拟地址14位）
  - 12-bit physical address（物理地址12位）
  - Page size = 64 bytes（页大小64B）

页表项数应为：
$2^{14-6}=256$



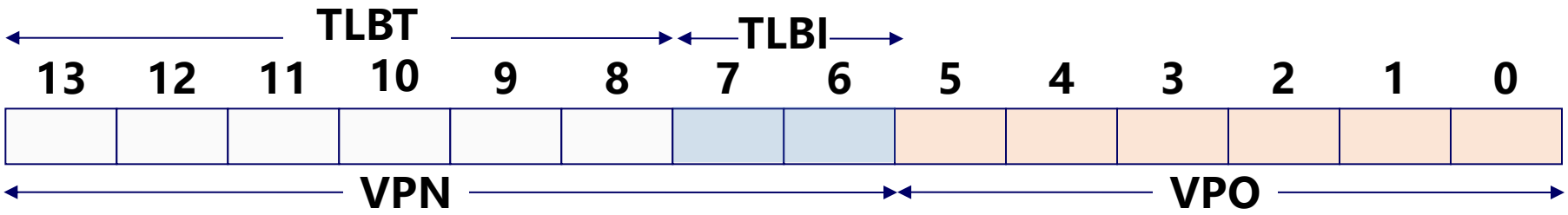
一个简化的存储系统举例（续）

假定部分页表项内容（十六进制表示）如右：

VPN	PPN	Valid
000	28	1
001	–	0
002	33	1
003	02	1
004	–	0
005	16	1
006	–	0
007	–	0

VPN	PPN	Valid
028	13	1
029	17	1
02A	09	1
02B	–	0
02C	–	0
02D	2D	1
02E	11	1
02F	0D	1

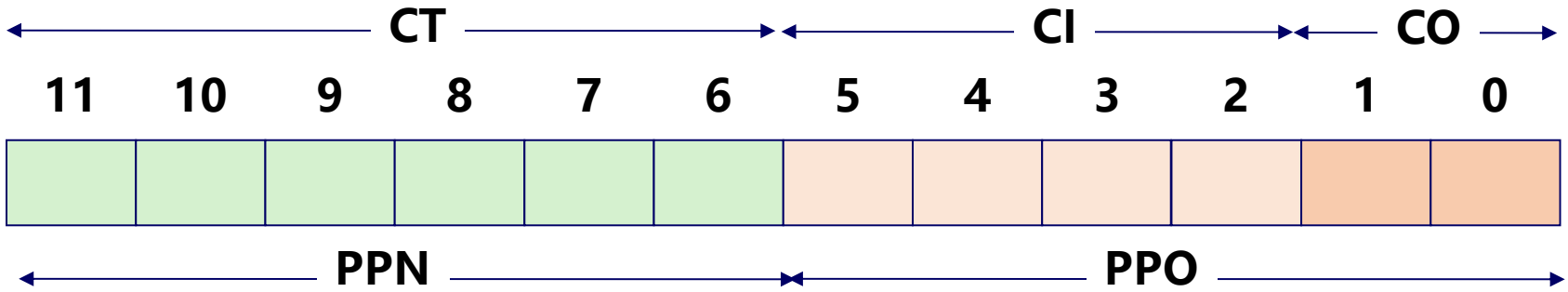
假定TLB如下：16 个TLB项，4路组相联，则TLBT和TLBI各占几位?



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

一个简化的存储系统举例（续）

假定Cache的参数和内容（十六进制）如下：16 行，主存块大小为4B，直接映射，则主存地址如何划分?

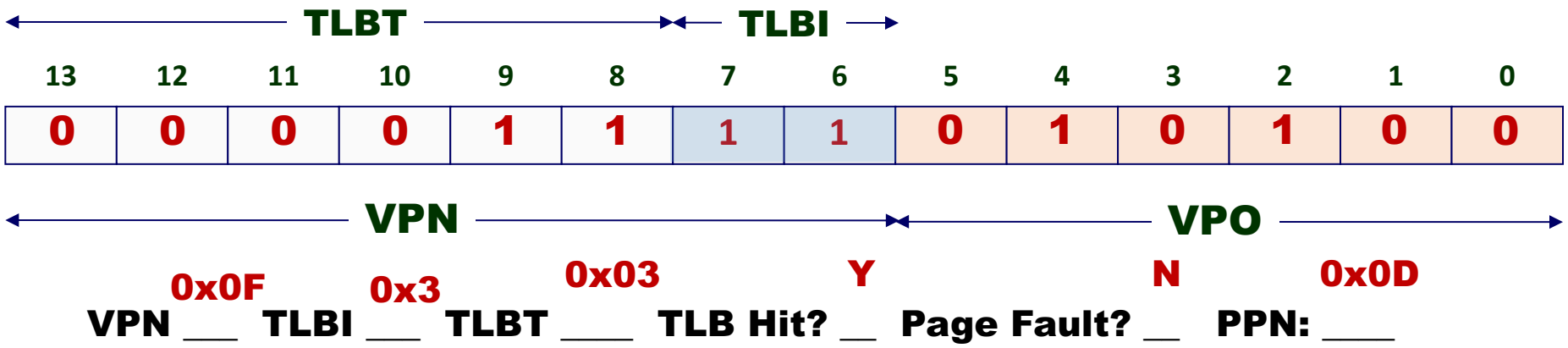


Idx	Tag	V	B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Idx	Tag	V	B0	B1	B2	B3
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

一个简化的存储系统举例（续）

假设该存储系统所在计算机采用小端方式， CPU执行某指令过程中要求访问一个16位数据，给出的逻辑地址为0x03D4，说明访存过程。



物理地址为

问题：逻辑地址为0x0A7A、0x0507时的访存过程如何？
TLB缺失/cache缺失、TLB缺失/缺页