计算机组织与体系结构

第二十二讲

计算机科学与技术学院

张展

第八章　存储层次

降低Cache失效率的方法

优化技术	不命中率	不命中开销	命中时间	硬件复杂度	说 明
增加块大小	+	–		0	实现容易；Pentium 4 的第二级Cache采用了128字节的块
提高相联度	+		–	1	被广泛采用
牺牲Cache	+			2	AMD Athlon采用了8个项的Victim Cache
伪相联Cache	+			2	MIPS R10000的第二级Cache采用
硬件预取指令和数据	+			2～3	许多机器预取指令，UltraSPARC Ⅲ预取数据
编译器控制的预取	+			3	需同时采用非阻塞Cache；有几种微处理器提供了对这种预取的支持
用编译技术减少Cache不命中次数	+			0	向软件提出了新要求；有些机器提供了编译器选项

本章内容

8.4 减少Cache失效开销

- 写缓冲及写合并
- 让读失效优先于写
- 请求字处理技术
- 多级Cache
- 非阻塞Cache技术

8.4.1 写缓冲及写合并

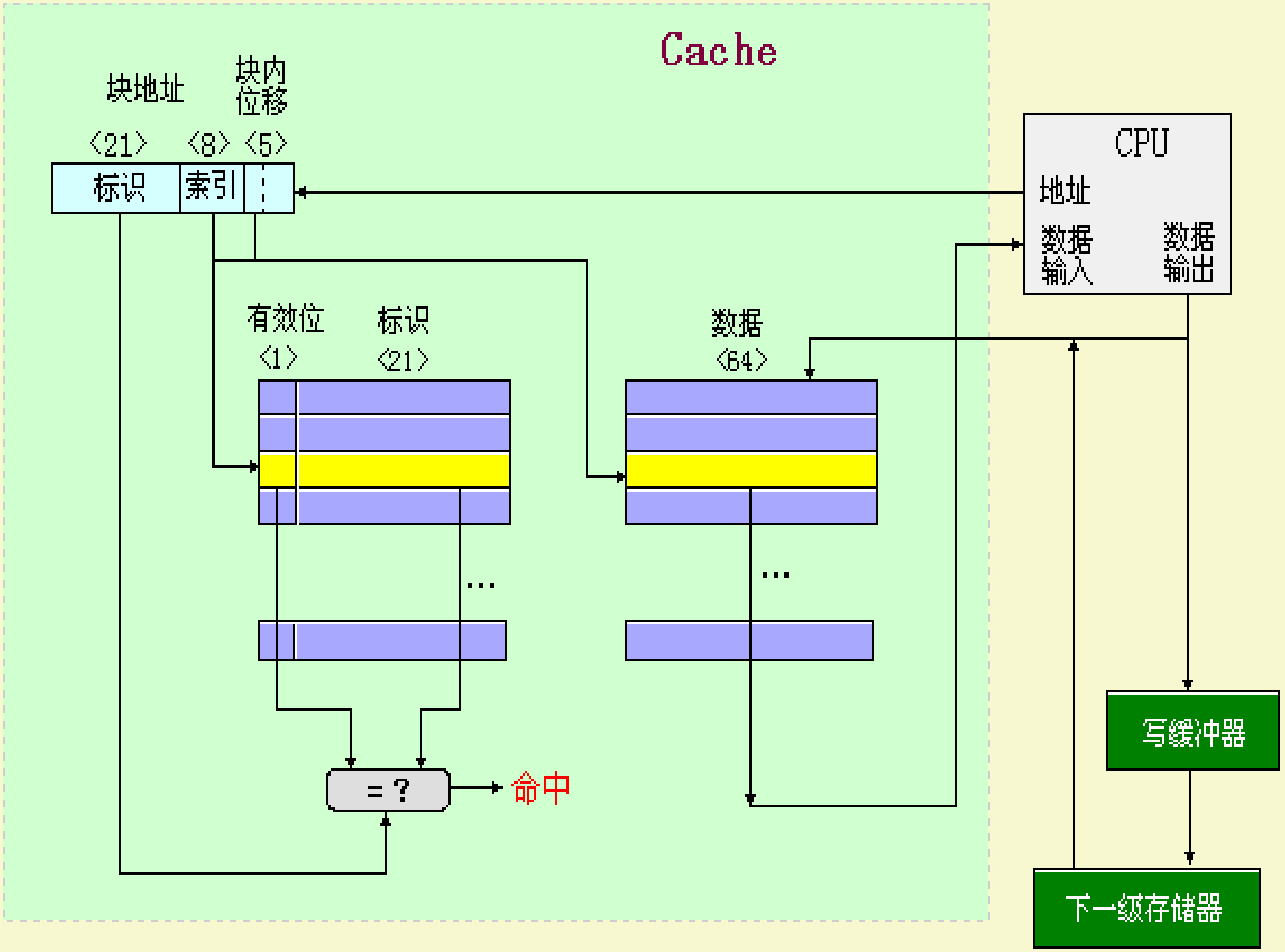
1. 写直达Cache中，因为所有的写请求都必须发送到下级存储层次中，所以经常使用一个写缓冲来降低失效开销。

写直达Cache

依靠写缓冲来减少对下一级存储器写操作的时间。

如果写缓冲器为空，就把数据和相应地址写入该缓冲器。

从CPU的角度来看，该写操作就算是完成了。


Cache
块地址
块内
位移
〈21〉
〈8〉
〈5〉
标识
索引
CPU
地址
数据
输入
数据
输出
有效位
〈1〉
标识
〈21〉
数据
〈64〉
...
...
= ?
命中
写缓冲器
下一级存储器

8.4.1 写缓冲及写合并

2. 如何提高写缓冲的效率和利用率?

写合并

提高了写缓冲器的空间利用率，而且还能减少因写缓冲器满而要进行的等待时间。

- 如果写缓冲器中已经有了待写入的数据，就要把这次的写入地址与写缓冲器中已有的所有地址进行比较，看是否有匹配的项。如果有地址匹配而对应的位置又是空闲的，就把这次要写入的数据与该项合并。这就叫写缓冲合并。
- 如果写缓冲器满且又没有能进行写合并的项，就必须等待。

3. 在写回法Cache中，也可采用写缓冲器

写地址	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

（a）不采用写合并

写地址	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

（b）采用了写合并

8.4.2 让读失效优先于写

1. Cache中的写缓冲器导致对存储器访问的复杂化

例 看下面的代码序列：

```
Sw R3, 512（R0）    ; M[512]←R3    （Cache index 0）
Lw R1, 1024（R0）   ; R1←M[1024]   （Cache index 0）
Lw R2, 512（R0）    ; R2←M[512]    （Cache index 0）
```

假设有一个直接映射的写通过的 Cache，它把 512 和 1024 这两个地址映射到 Cache 的一个相同块中，它带有容量为四个字的写缓存，那么，R2 中的值总是同 R3 中的值相等么？

解

使用第三章的术语，这是一个存储器中先写后读的数据冲突。我们来跟踪一次 Cache 访问来看一下隐患。R3 中的数据在执行存操作之后进入到写缓存中。接下来的取操作由于使用了相同的 Cache 索引，因而导致了缺失。第二个取指令想把位置 512 处的值写到寄存器 R2 中；这也导致了一次缺失。如果写缓存还没有完成向 512 处的写操作，则对 512 的读操作就把原先的错误值调入到 Cache 块中，接着调入到 R2 中。如果没有正确的防范措施，R3 值将与 R2 的值不相等！

2. 解决问题的方法(读失效的处理)

- 推迟对读失效的处理及到写缓冲排空
  (缺点：读失效的开销增加)
- 优先读操作

  检查写缓冲器中的内容：增加硬件

8.4.3 请求字处理技术

1. 请求字

从下一级存储器调入Cache的块中，只有一个字是立即需要的。这个字称为**请求字**。

2. 应尽早把请求字发送给CPU

- **尽早重启动**：调块时，从块的起始位置开始读起。一旦请求字到达，就立即发送给CPU，让CPU继续执行。
- **请求字优先**：调块时，从请求字所在的位置读起。这样，第一个读出的字便是请求字。将之立即发送给CPU。

3. 这种技术在以下情况下效果不大：

- Cache块较小
- 下一条指令正好访问同一Cache块的另一部分。

8.4.4 多级Cache

1. 应把Cache做得更快？还是更大？

答案：二者兼顾，再增加一级Cache

- 第一级Cache(L1)小而快
  - 与快速的CPU运行时钟周期时间相匹配
- 第二级Cache(L2)容量大
  - 捕捉到对主存进行的大多数访问

2. 性能分析

平均访问时间＝命中时间$_{L1}$＋失效率$_{L1}$×失效开销$_{L1}$

＝命中时间$_{L1}$＋失效率$_{L1}$×

(命中时间$_{L2}$＋失效率$_{L2}$×失效开销$_{L2}$)

3. 局部失效率与全局失效率

局部失效率＝该级Cache的失效次数/到达该级Cache的访问次数

例如：上述式子中的失效率$_{L2}$

全局失效率＝该级Cache的失效次数/CPU发出的访存的总次数

全局失效率＝失效率$_{L1}$×失效率$_{L2}$

评价多级Cache时，应使用全局失效率这个指标

它指出了在CPU发出的访存中，究竟有多大比例是穿过各级Cache，最终到达存储器的

4. 采用两级Cache时，每条指令的平均访存停顿时间：

每条指令的平均访存停顿时间

＝ 每条指令的平均不命中次数$_{L1}$×命中时间$_{L2}$＋

每条指令的平均不命中次数$_{L2}$×不命中开销$_{L2}$

例 考虑某一两级Cache：第一级Cache为L1，第二级Cache为L2。

（1）假设在1000次访存中，L1的不命中是40次，L2的不命中是20次。求各种局部不命中率和全局不命中率。

（2）假设L2的命中时间是10个时钟周期，L2的不命中开销是100时钟周期，L1的命中时间是1个时钟周期，平均每条指令访存1.5次，不考虑写操作的影响。问：平均访存时间是多少？每条指令的平均停顿时间是多少个时钟周期？

解（1）

第一级Cache的不命中率（全局和局部）是40/1000，即4%；

第二级Cache的局部不命中率是20/40，即50%；

第二级Cache的全局不命中率是20/1000，即2%。

（2）平均访存时间＝命中时间$_{L1}$＋不命中率$_{L1}$×（命中时间$_{L2}$＋不命中率$_{L2}$×不命中开销$_{L2}$）

＝1＋4%×（10＋50%×100）

＝1＋4%×60＝3.4个时钟周期

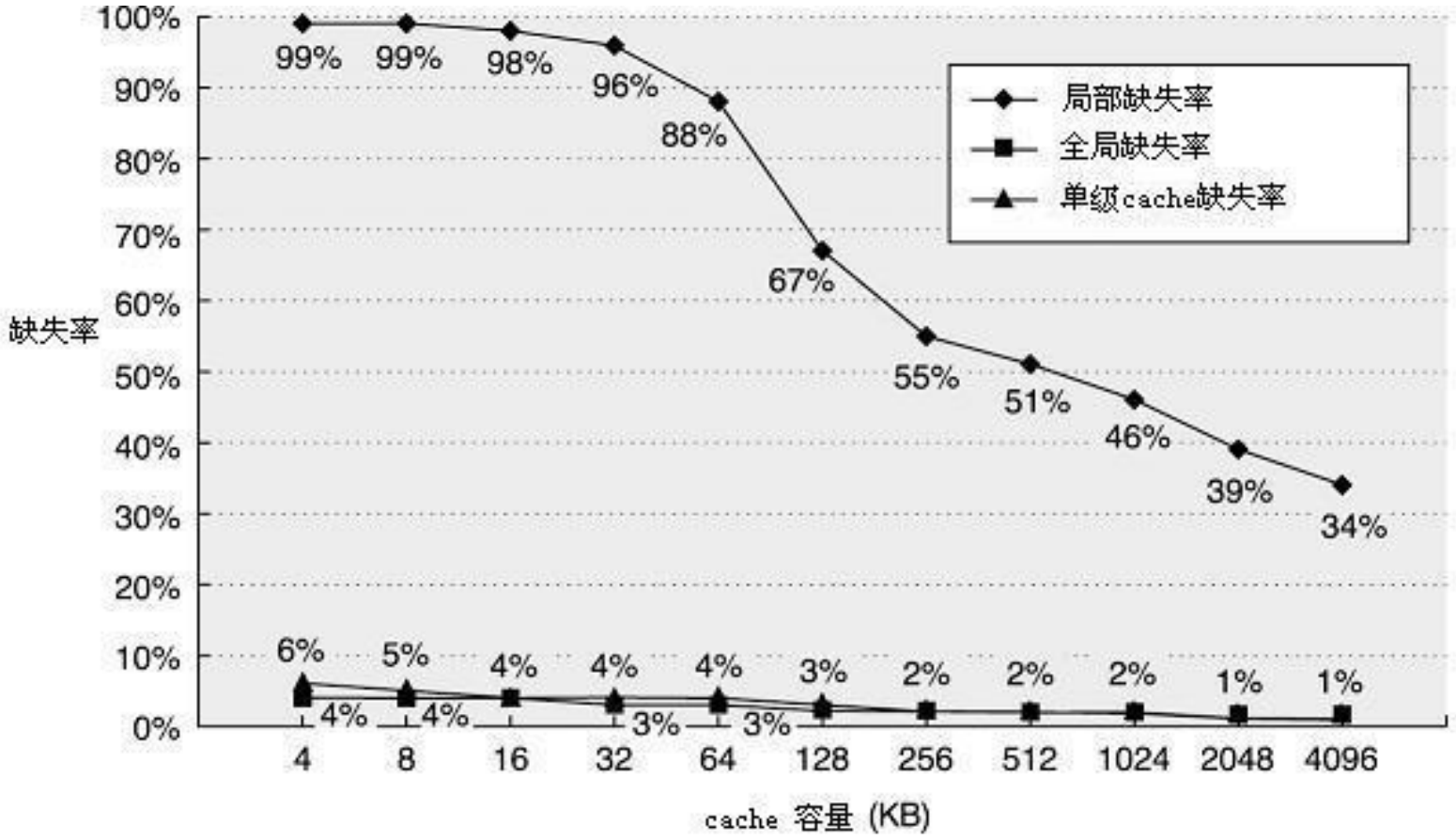
由于平均每条指令访存1.5次，且每次访存的平均停顿时间为：

3.4－1.0＝2.4

所以：

每条指令的平均停顿时间＝2.4×1.5＝3.6个时钟周期

5. 当第二级Cache比第一级Cache大得多时，两级Cache的全局失效率与容量和第二级Cache相同的单级Cache的失效率非常接近。



6. 第二级Cache的参数

第二级Cache不会影响CPU的时钟频率，因此其设计有更大的考虑空间。

两个问题：

- 能否降低CPI中的平均访存时间部分?
- 成本是多少?

(1) 容量　第二级Cache的容量一般比第一级的大许多，如512KB

大容量意味着第二级Cache可能实际上没有容量不命中，只剩下一些强制性不命中和冲突不命中。

(2) 相联度　第二级Cache可采用较高的相联度或伪相联方法

例8.12

给出有关第二级Cache的以下数据：

⑴对于直接映象，命中时间$_{L2}$＝10个时钟周期

⑵两路组相联使命中时间增加0.1个时钟周期，即为10.1个时钟周期。

⑶ 对于直接映象，局部失效率$_{L2}$＝25%

⑷ 对于两路组相联，局部失效率$_{L2}$＝20%

⑸ 失效开销$_{L2}$＝200个时钟周期

试问第二级Cache的相联度对失效开销的影响如何?

解 对一个直接映象的第二级Cache来说，第一级Cache的不命中开销为：

不命中开销$_{\text{直接映象，L1}}$ ＝ 10＋25%×200 ＝ 60 个时钟周期

对于两路组相联第二级Cache来说，命中时间增加了10%（0.1）个时钟周期，故第一级Cache的不命中开销为：

不命中开销$_{\text{两路组相联，L1}}$ ＝ 10.1＋20%×200 ＝ 50.1个时钟周期

把第二级Cache的命中时间取整，得10或11，则：

不命中开销$_{\text{两路组相联，L1}}$ ＝ 10＋20%×200 ＝ 50个时钟周期

不命中开销$_{\text{两路组相联，L1}}$ ＝ 11＋20%×200 ＝ 51 个时钟周期

故对于第二级Cache来说，两路组相联优于直接映象。

减少第二级Cache的缺失率，可以降低缺失代价

(3) 多级包容和多级互斥

需要考虑的另一个问题：

第一级Cache中的数据是否总是同时存在于第二级Cache中。

(4)块大小

- 第二级Cache可采用较大的块，如 64、128、256字节。

为减少平均访存时间，可以让容量较小的第一级Cache采用较小的块，而让容量较大的第二级Cache采用较大的块。

8.4.5 非阻塞Cache技术

1. **非阻塞Cache：Cache失效时仍允许CPU进行其它的命中访问。即允许“失效下命中”**
2. **进一步提高性能：多重失效下命中**
   - “多重失效下命中”
   - “失效下失效”

   （存储器必须能够处理多个不命中）
3. **可以同时处理的不命中次数越多，所能带来的性能上的提高就越大。**

3. 重叠失效个数对平均访问时间的影响

4. 模拟研究

非阻塞Cache的平均存储器等待时间（以周期为单位）与阻塞Cache平均存储器等待时间的比值

- 测试条件：8K直接映象Cache，块大小为32字节
- 测试程序：SPEC92（14个浮点程序，4个整数程序）
- 结果表明

在重叠不命中个数为1、2和64的情况下

浮点程序的平均比值分别为：76%、51%和39%

整数程序的平均比值则分别为：81%、78%和78%

对于整数程序来说，重叠次数对性能提高影响不大，简单的“一次不命中下命中”就几乎可以得到所有的好处。

- 对非阻塞cache的性能评价很难，因为存在重叠停顿时间

优化技术	不命中率	不命中开销	命中时间	硬件复杂度	说 明
使读不命中优先于写		+	–	1	在单处理机上实现容易，被广泛采用
写缓冲写合并		+		1	与写直达合用，广泛应用，例如21164，UltraSPARC III
尽早重启动和请求字优先		+		2	被广泛采用
非阻塞Cache		+		3	在支持乱序执行的CPU中使用
两级Cache		+		2	硬件代价大；两级Cache的块大小不同时实现困难；被广泛采用

8.5 减少命中时间

- 采用容量小、结构简单的Cache
- 虚拟Cache
- Cache访问流水化

命中时间直接影响到处理器的时钟频率。在当今的许多计算机中，往往是Cache的访问时间限制了处理器的时钟频率。

8.5.1 容量小、结构简单的Cache

1. 硬件越简单，速度就越快；

2. 应使Cache足够小，以便可以与CPU一起放在同一块芯片上。

3. 一种折中方案：把Cache的标识放在片内，而把Cache的数据存储器放在片外，这样既可以实现快速标识检测，又能利用独立的存储芯片来提供更大的容量

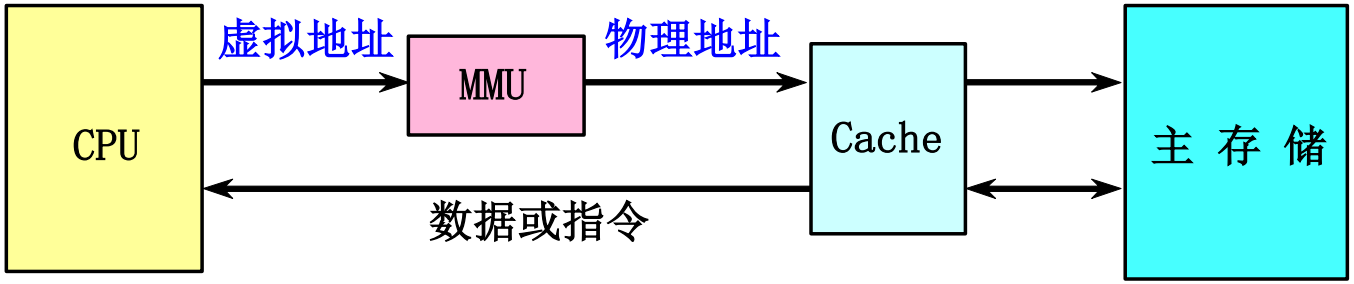
4. 多级Cache，一级Cache大多采用2路组相联或直接映像。

8.5.2 虚拟Cache

1. 物理Cache

- 使用物理地址进行访问的传统Cache。
- 标识存储器中存放的是物理地址，进行地址检测也是用物理地址。

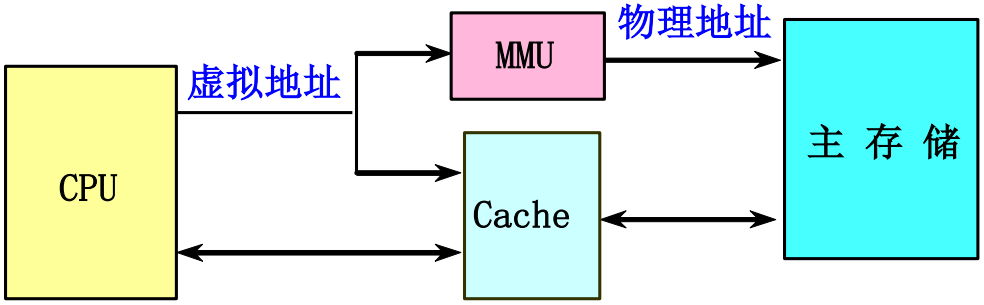
– 缺点：地址转换和访问Cache串行进行，访问速度很慢。



物理Cache存储系统

2. 虚拟Cache

- 可以直接用虚拟地址进行访问的Cache。标识存储器中存放的是虚拟地址，进行地址检测用的也是虚拟地址。
  - 优点：
    - 在命中时不需要地址转换，省去了地址转换的时间。即使不命中，地址转换和访问Cache也是并行进行的，其速度比物理Cache快很多。



3. 并非都采用虚拟Cache(为什么?)

➢ 虚拟Cache的清空问题（进程切换）

- 解决方法：在地址标识中增加PID字段(进程标识符)
- 三种情况下不命中率的比较
  - 单进程，PIDs，清空
  - PIDs与单进程相比：＋0.3%～＋0.6%
  - PIDs与清空相比：　－0.6%～－4.3%

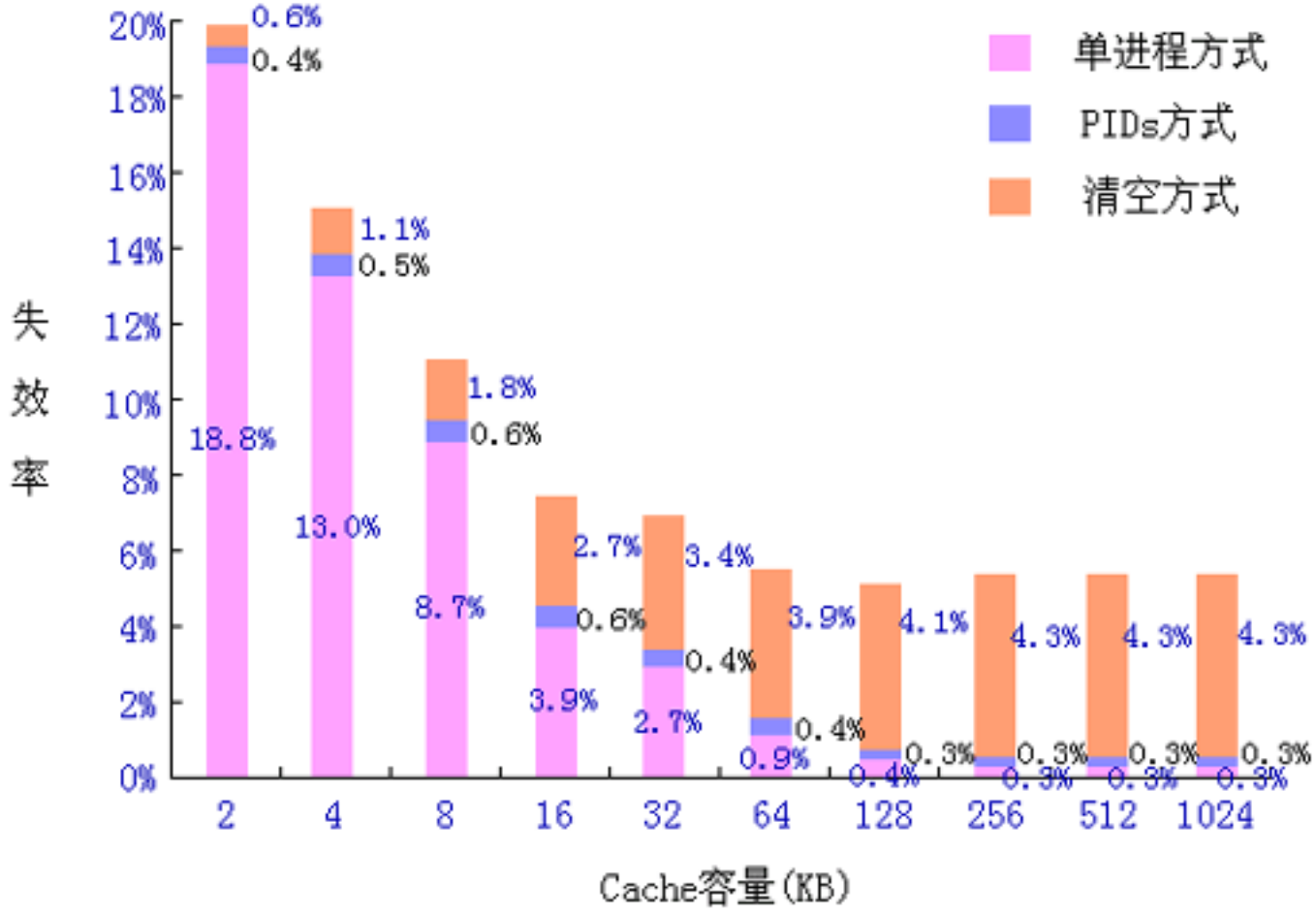
➢ 同义和别名

- 操作系统和用户程序为了共享某些空间，对于同一个物理地址可能采用两种以上不同形式的虚拟地址来访问，这些地址称为同义（synonym）或别名(alias)

解决方法：

页着色:通过软件强制别名的某些位相同

三种方式下，虚拟Cache的失效率



5. 虚拟索引＋物理标识

利用虚拟地址找到索引读cache，同时进行虚实地址转换：前提是虚拟索引=物理索引

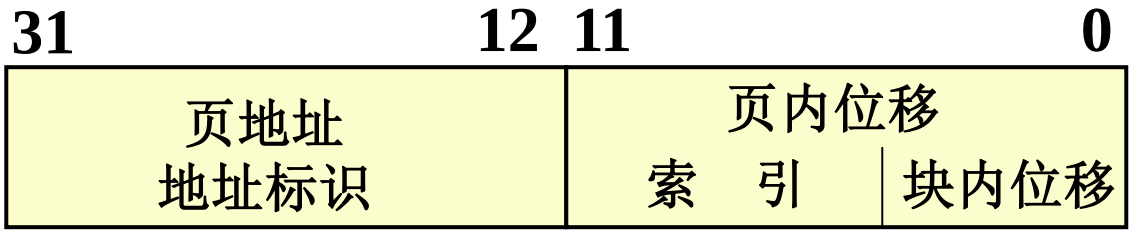
而：页内位移在虚实地址变换时不变

优点：兼得虚拟Cache和物理Cache的好处

局限性：Cache容量受限≤页大小×相联度

6. 举例：IBM3033的Cache

页大小 = 4KB　　相联度 = 16



Cache容量 = 16×4KB = 64KB

8.5.3 Cache访问流水化

1. 对第一级Cache的访问按流水方式组织

2. 访问Cache需要多个时钟周期才可以完成

例如

- Pentium访问指令Cache需要一个时钟周期
- Pentium Pro到Pentium Ⅲ需要两个时钟周期
- Pentium 4 和Core i7则需要4个时钟周期

访问Cache周期数的增加使得流水线的站数也增加了，进一步增加了分支预测失败的代价，也增加了从load指令发射到使用该数据间的时间间隔

减小命中时间的方法

优化技术	不命中率	不命中开销	命中时间	硬件复杂度	说 明
容量小且结构简单的Cache	–		+	0	实现容易，被广泛采用
对Cache进行索引时不必进行地址变换			+	2	对于小容量Cache来说实现容易，已被Alpha 21164和UltraSPARC III采用
流水化Cache访问			+	1	被广泛采用

共7种

增加块大小

提高相联度

Victim Cache

伪相联 Cache

硬件预取

编译器控制的预取

用编译器技术减少Cache失效

降低失效率

共5种

使读失效优于写

子块放置技术

尽早重启动和关键字优先

非阻塞Cache

第二级Cache

减少失效开销

共3种

容量小且结构简单的Cache

对Cache索引时，不必进行地址变换

流水化写

减少命中时间

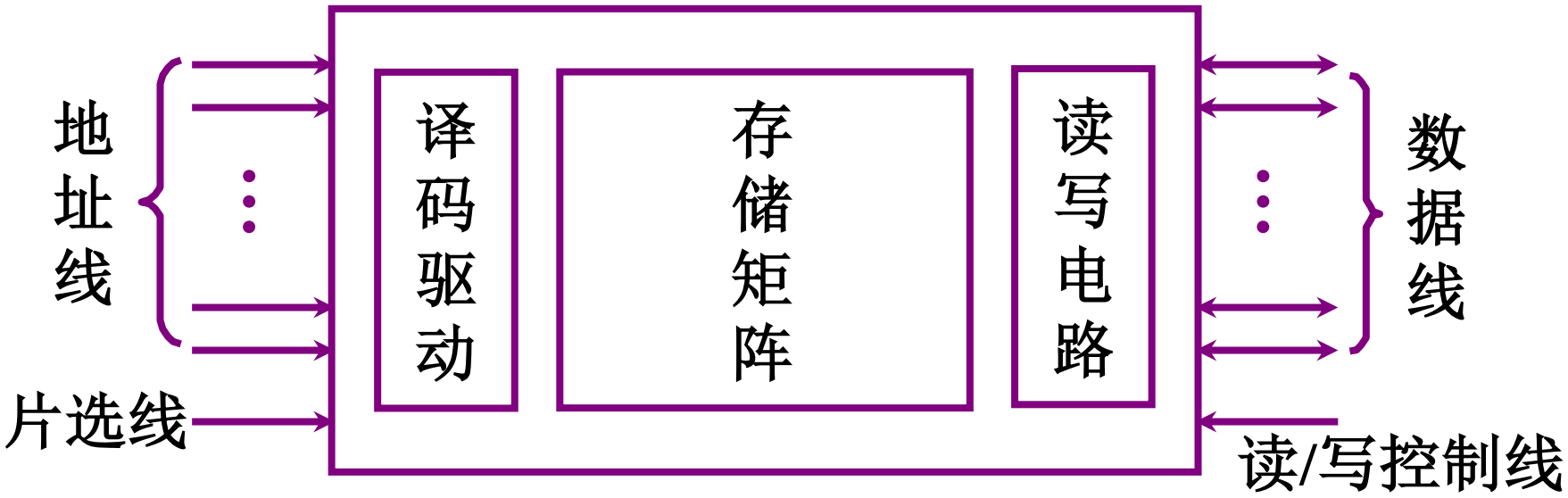
本章内容

8.6 主存

名称	寄存器	Cache	主存	磁盘
典型大小	<1KB	<16MB	<512G	>1TB
实现技术	定制多端口存储器，CMOS	片上或片外CMOS SRAM	CMOS DRAM	磁介质盘
访问时间(ns)	0.25-0.5	0.5-25	50-250	5,000,000
带宽(MB/s)	50,000-500,000	5000-20,000	2500-10,000	50-500
管理	编译器	硬件	操作系统	操作系统和用户
后备	Cache	主存	磁盘	CD或磁带

8.6.1 存储器芯片技术

半导体存储芯片的基本结构



地址线（单向）	数据线（双向）	芯片容量
10	4	1K×4位
14	1	16K×1位
13	8	8K×8位

8.6.1 存储器芯片技术

DRAM与SRAM

容量：4～8：1

存储周期：8～16:1

价格：1：8～16

一般来说：主存：DRAM　Cache：SRAM

各代DRAM的典型参数 （表5-11）

动态 RAM 和静态 RAM 的比较

主存

缓存

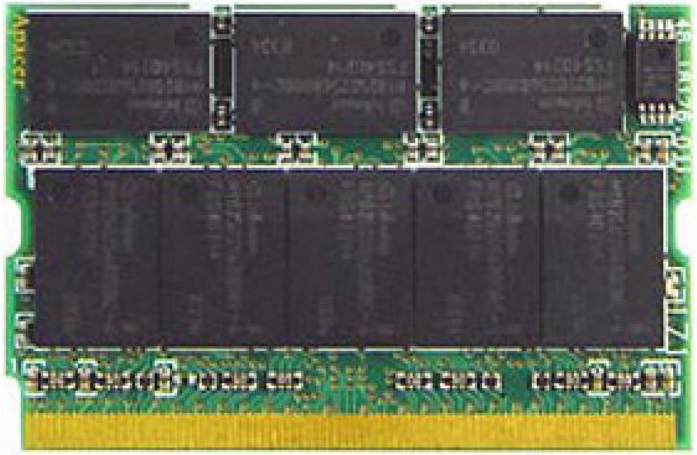
	DRAM	SRAM
存储原理	电容	触发器
集成度	高	低
芯片引脚	少	多
功耗	小	大
价格	低	高
速度	慢	快
刷新	有	无

8.6.1 存储器芯片技术

DIMM（dual inline memory modules）

多个DRAM芯片经常被组装在称为条的小型板上，构成“双列直插式存储模块”。

一个DIMM通常包含4~16片DRAM芯片，这些芯片常被组织成8字节宽的主存（带ECC校验）



8.6.1 存储器芯片技术

2. DRAM芯片优化技术

- 芯片内部优化技术是提高主存系统性能的一个重要方面。
- SDRAM：Synchronous DRAM：DRAM接口增加一个时钟信号可使DRAM能针对一个请求连续同步地传输多个数据而不需同步开销。
- DDR（double data rate）：在DRAM时钟的上沿和下沿都进行数据传输，可把数据传输率提高一倍。
- CDRAM（Cache DRAM）:带Cache的DRAM，DRAM芯片里集成一个小的SRAM，暂存最后读出行数据。

8.6.2 存储器组织技术

主存的主要性能指标：延迟和带宽

以往：Cache主要关心延迟，I/O主要关心带宽

现在：Cache关心两者

本节讨论几种提高主存性能的存储器组织技术

在下面的讨论中，以处理Cache失效开销为例来说明各种存储器组织结构的好处。

8.6.2 存储器组织技术

- **为了减少失效开销T_M，应该：**
  - **减少主存延迟**
  - **提高主存带宽**

- **增加Cache块大小能利用主存带宽增加所带来的好处。**

8.6.2 存储器组织技术

假设基本存储器结构的性能为：

- **送地址需4个时钟周期**
- **每个字的访问时间为24个时钟周期**
- **传送一个字（4个字节）的数据需4个时钟周期**

如果Cache大小为4个字，则：

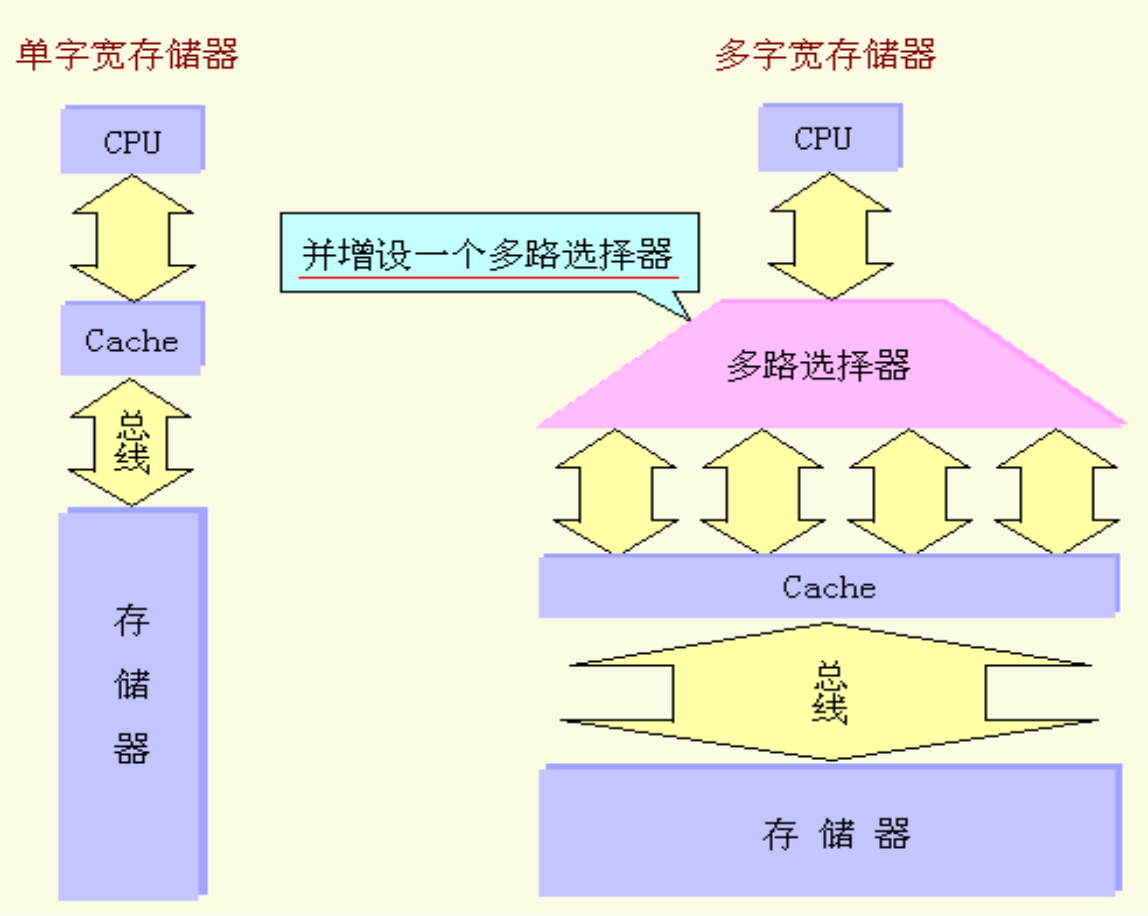
失效开销 = 4×(4 + 24 + 4)

= 4×32 = 128(时钟周期)

带宽 = 16/128 = 0.0125(字节/时钟周期)

8.6.2 存储器组织技术

1. 增加存储器的宽度



8.6.2 存储器组织技术

- **性能分析　(参照前面的假设)**

  当宽度为4个字时：失效开销 = 1×32(周期)

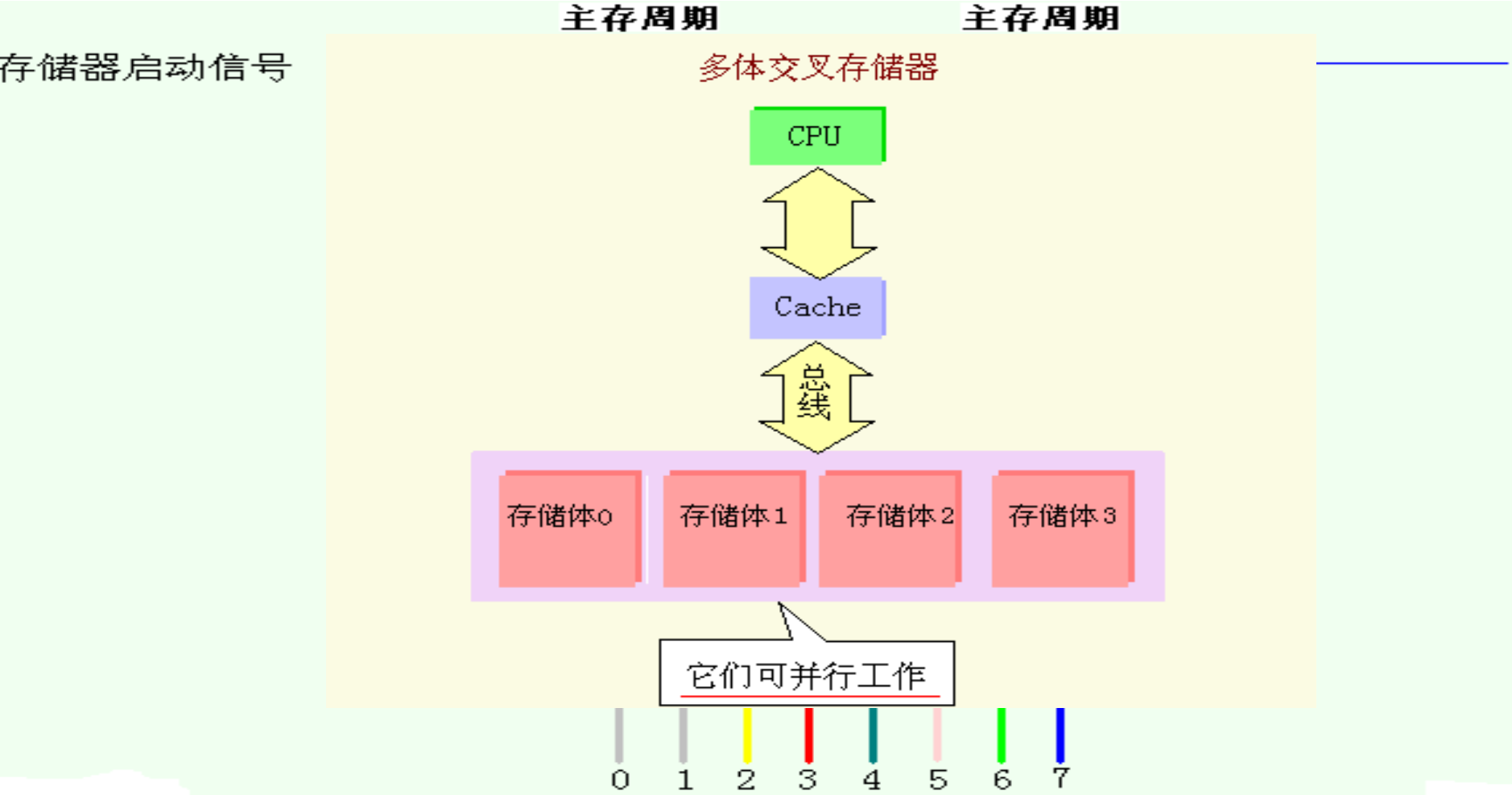
  带宽 = 0.5(字节/周期)

- **缺点：**
  - **增加CPU和存储器之间的连接通路宽度**
  - **CPU和Cache之间有一个多路选择器**
  - **扩充主存的最小增量增加了相应的倍数**
  - **写入有可能变得复杂**
- **实例：DEC的Alpha Axp21064：256位宽**

8.6.2 存储器组织技术

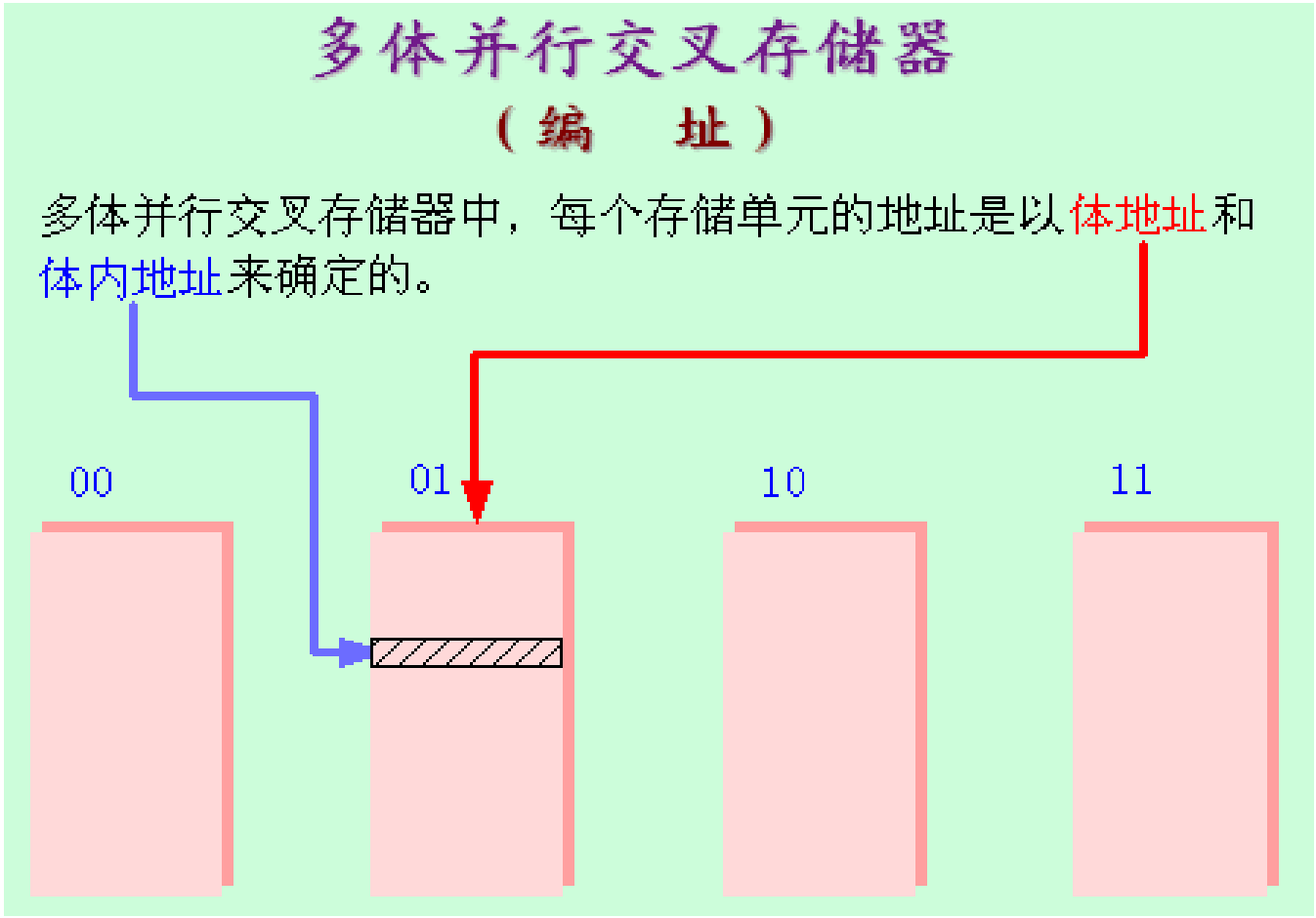
2. 采用简单的多体交叉存储器

存储系统中采用多个DRAM，利用它们潜在的并行性



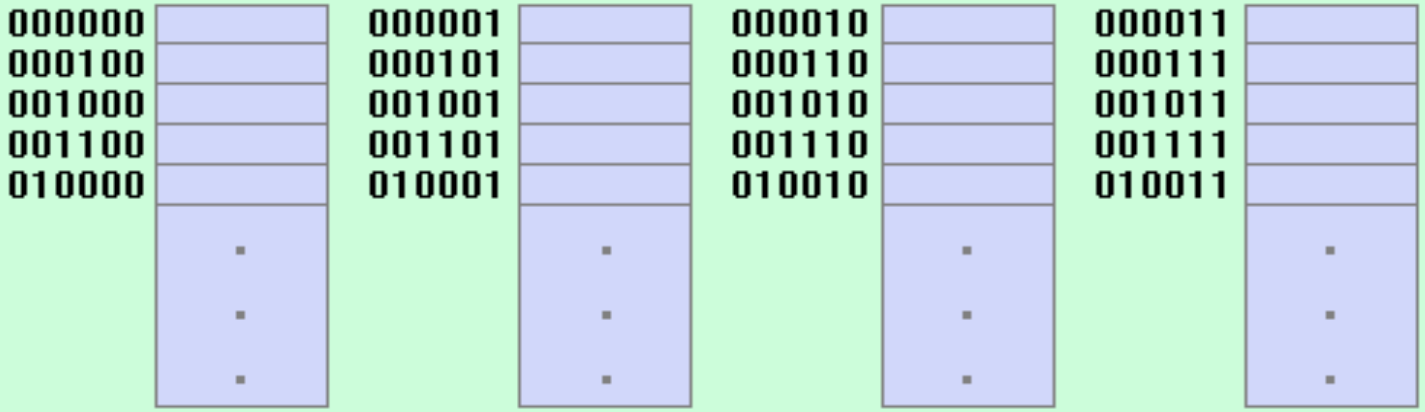
8.6.2 存储器组织技术

- 高位交叉与低位交叉编址



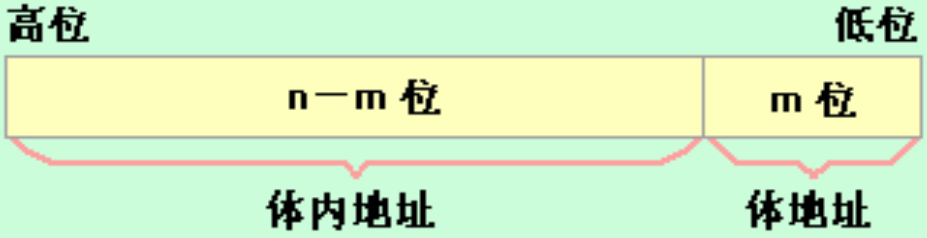
8.6.2 存储器组织技术

- 低位交叉编址



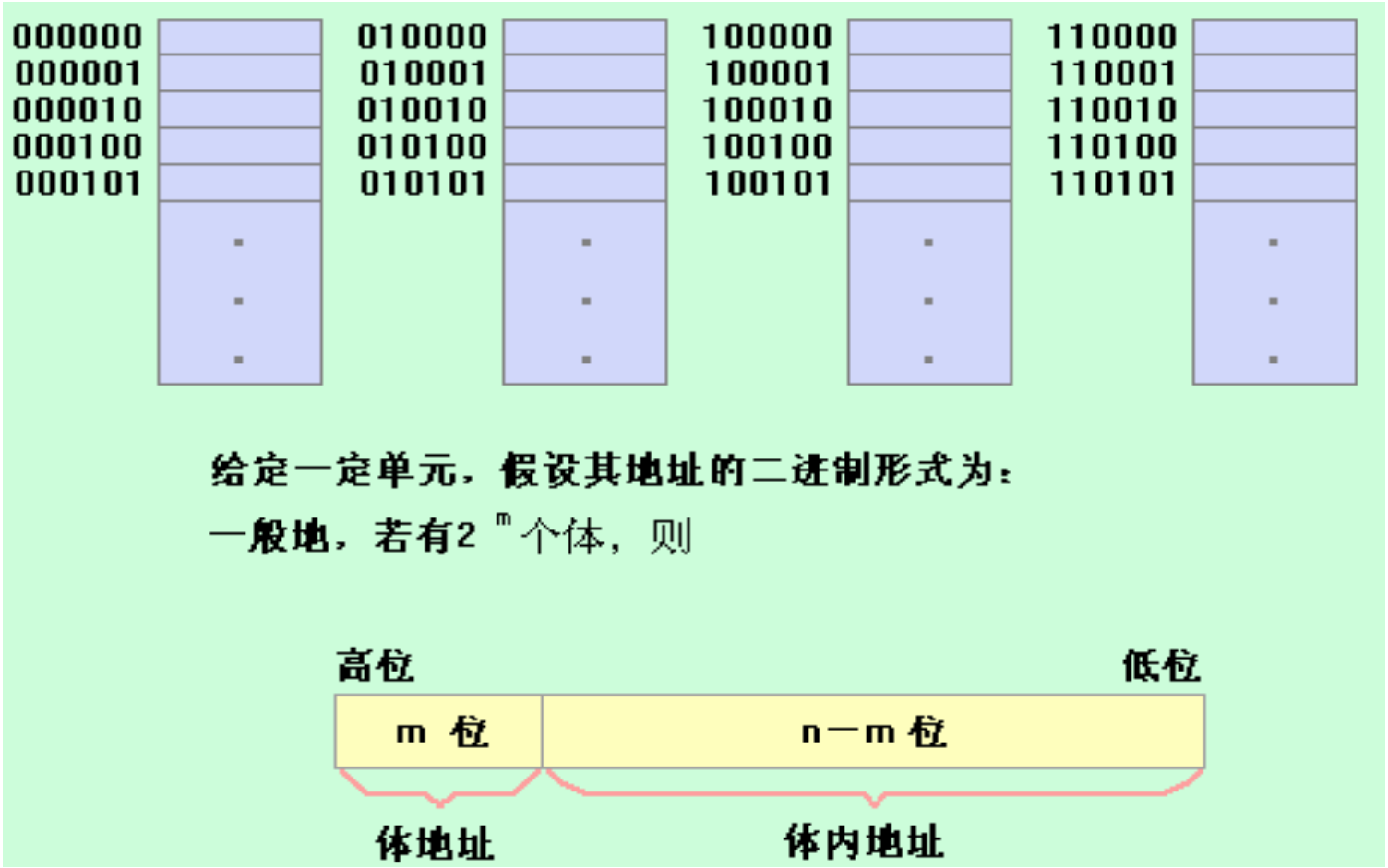
给定一定单元，假设其地址的二进制形式为：

一般地，若有2^m个体，则：



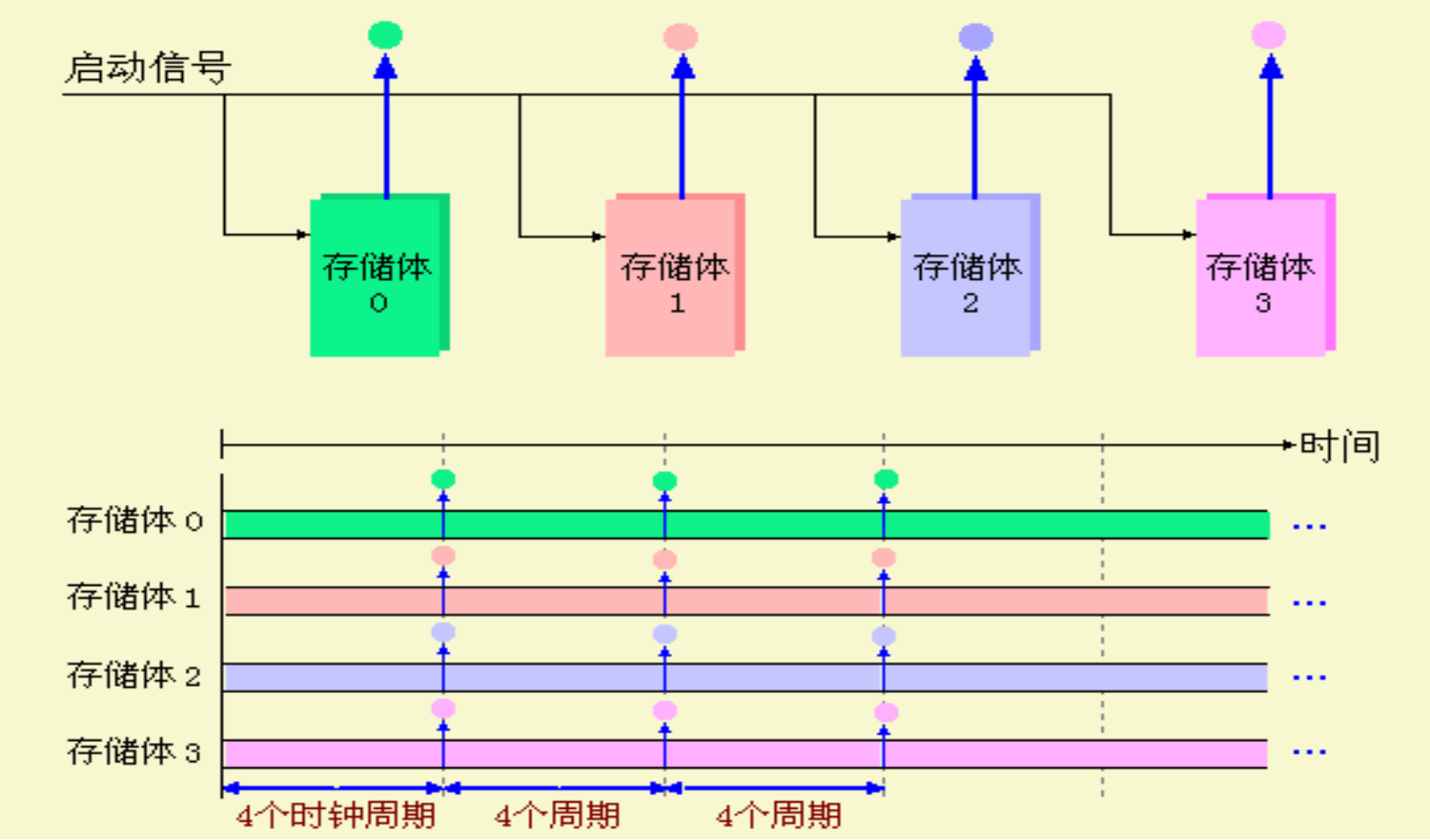
8.6.2 存储器组织技术

- 高位交叉编址



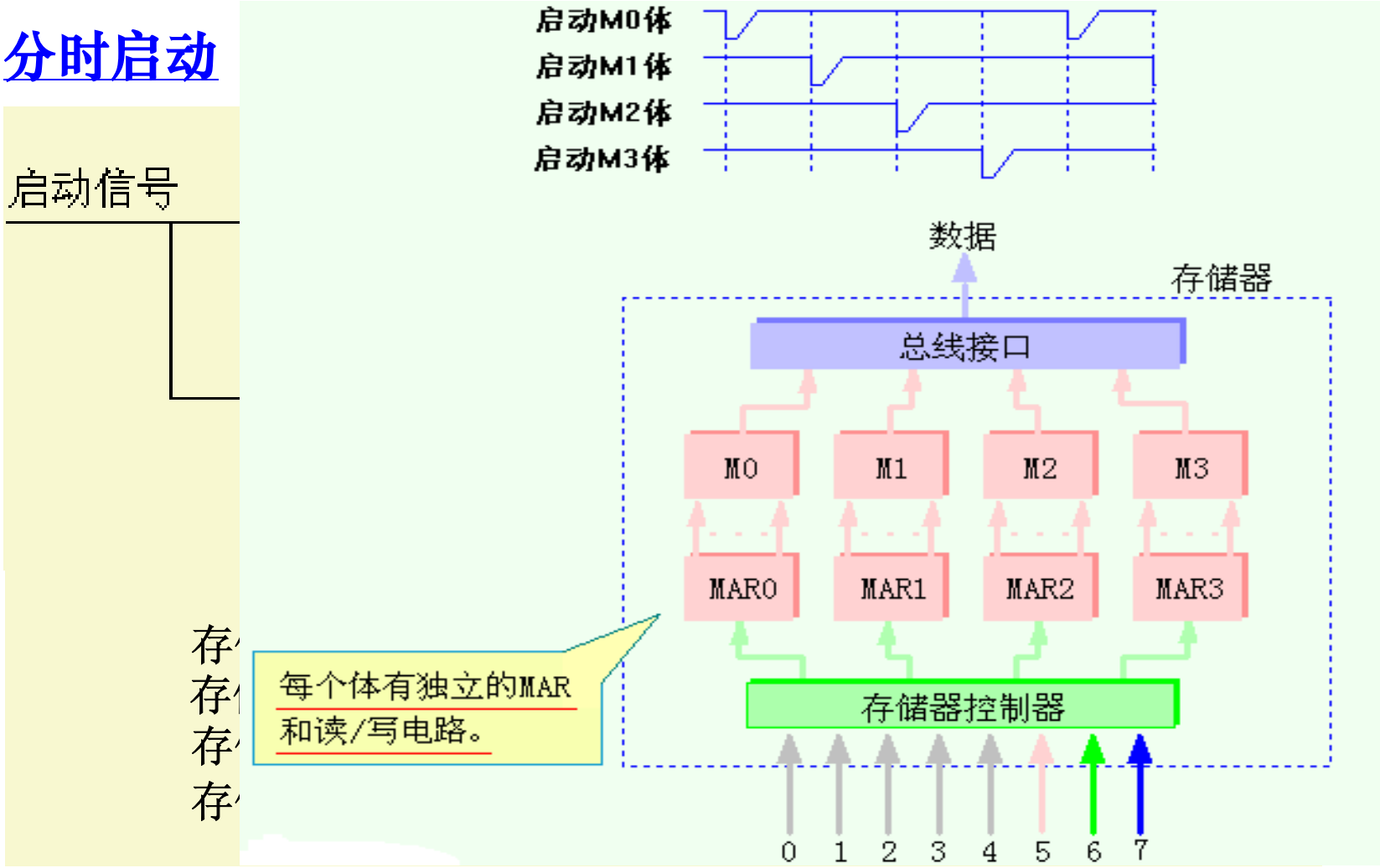
8.6.2 存储器组织技术

同时启动

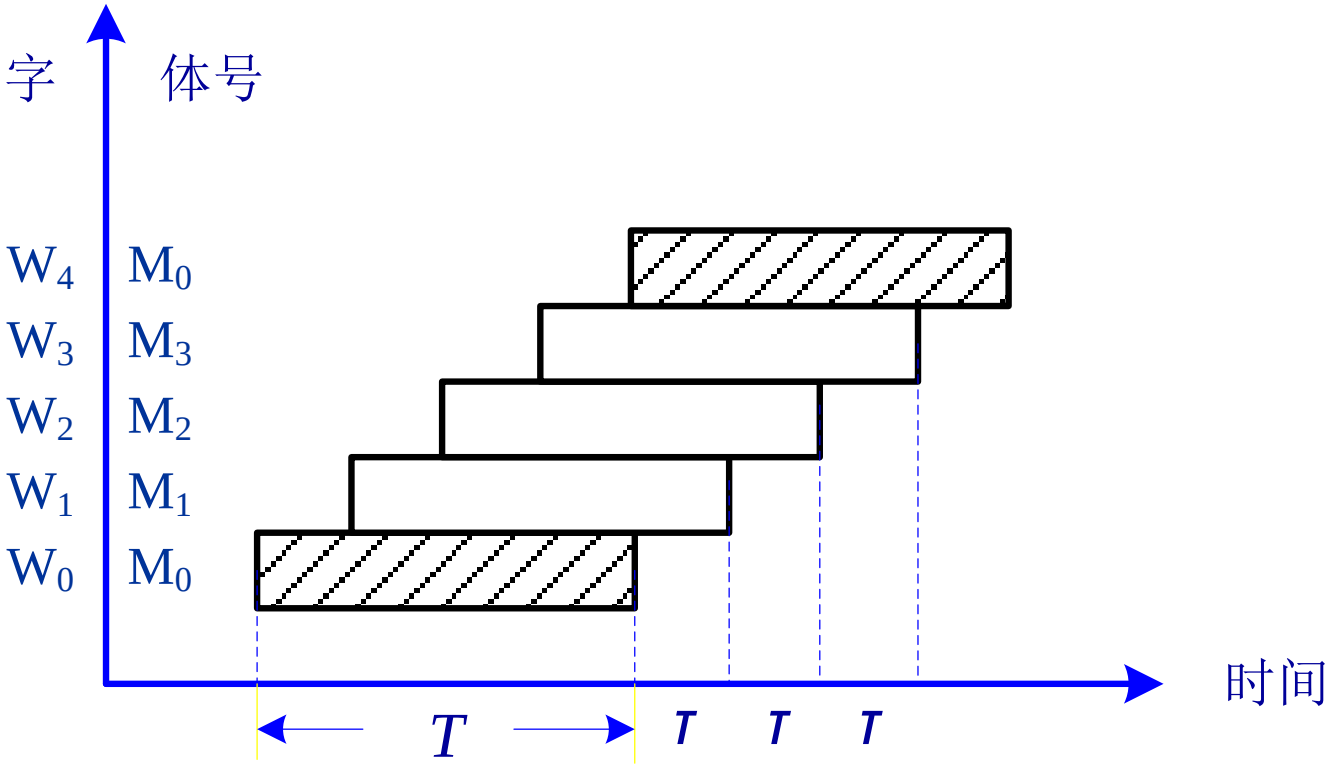


8.6.2 存储器组织技术

分时启动



设四体低位交叉存储器，存取周期为T，总线传输周期为τ，为实现流水线方式存取，应满足 $T = 4\tau$。



连续读取 4 个字所需的时间为 $T+(4-1)\tau$

8.6.2 存储器组织技术

性能举例：(参照前面的假设)

失效开销 = 4 + 24 + 4×4 = 44(周期)

带宽 = 0.4(字节/周期)

- 存储器的各个体一般是按字交叉的，低位字交叉存储器非常适合于处理：

  Cache读失效，写回法Cache中的写回

8.6.2 存储器组织技术

例8.14

假设某台机器的特性及其Cache的性能为:

- **块大小为1个字**
- **存储器总线宽度为1个字（32位）**
- **Cache失效率为3 %**
- **平均每条指令访存1.2次**
- **Cache失效开销为32个时钟周期(和上面相同)**
- **平均CPI(忽略Cache失效)为2**

试问多体交叉和增加存储器宽度对提高性能各有何作用?

8.6.2 存储器组织技术

如果当把Cache块大小变为2个字时，失效率降为2%；块大小变为4个字时，失效率降为1%。
根据8.6.2小节中给出的访问时间，求在采用2路、4路多体交叉存取以及将存储器和总线宽度增加一倍时，性能分别提高多少?

解：在改变前的机器中，Cache块大小为一个字，其CPI为：

$$2 + (1.2 \times 3\% \times 32) = 3.15$$

8.6.2 存储器组织技术

当将块大小增加为2个字时，在下面三种情况下的CPI分别为：

32位总线和存储器，不采用多体交叉：

$2 + (1.2 \times 2\% \times 2 \times 32) = 3.54$

32位总线和存储器，采用多体交叉：

$2 + (1.2 \times 2\% \times (4 + 24 + 8)) = 2.86$

性能提高了10%

64位总线和存储器，不采用多体交叉：

$2 + (1.2 \times 2\% \times 1 \times 32) = 2.77$

性能提高了14%

8.6.2 存储器组织技术

如果将块大小增加到4个字，则：

32位总线和存储器，不采用多体交叉：

$2 + (1.2 \times 1\% \times 4 \times 32) = 3.54$

32位总线和存储器，采用多体交叉：

$2 + (1.2 \times 1\% \times (4 + 24 + 16)) = 2.53$

性能提高了25%

64位总线和存储器，不采用多体交叉：

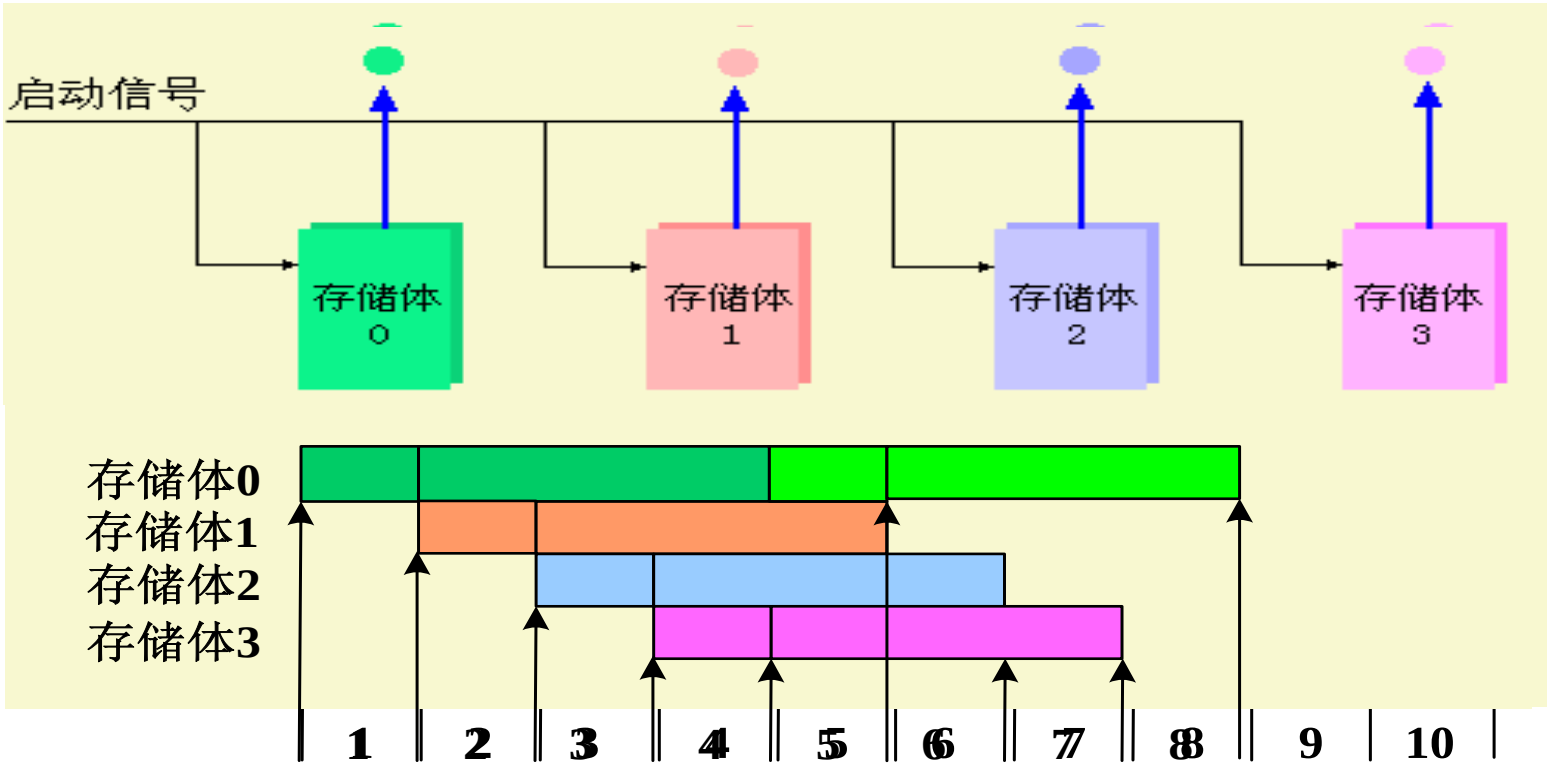
$2 + (1.2 \times 1\% \times 2 \times 32) = 2.77$

性能提高了14%

8.6.2 存储器组织技术

2. 采用简单的多体交叉存储器

◆ 存储体的数目



体的数目≥访问体中一个字所需的时钟周期

8.6.2 存储器组织技术

3. 独立存储体

设置多个存储控制器，使多个体能独立操作，以便能同时进行多个独立的访存。

- 每个体有独立的地址线
- 非阻塞Cache与多体结构

  Cache失效时仍允许CPU进行其它的访存。这样的设计对于独立多体结构才有意义。

  IO访存

谢谢！