第2 - 1章：
存储层次结构

存储层次结构
- 缓存基础
- 主存组织

第B & 2章：内存层次结构

□内存层次结构ABC

□内存技术与优化□缓存性能优化□虚拟内存与虚拟机□内存层次结构设计

□ARM和英特尔酷睿i7 6700中的内存层次结构

2.1 引言

□为什么设计师需要了解内存技术?

＞ 处理器性能通常受内存带宽限制

＞ 随着集成电路密度的增加，大量内存将可集成到处理器芯片上

□ 应用需求：

＞ 无限容量的内存

＞ 更快的内存，更高的带宽

＞ 每字节成本更低

＞ 若用于嵌入式系统：更低的功耗

□这些要求相互矛盾。

＞ 规模越大，越难实现高速运行

＞ 速度越快，成本越高

＞ 速度越快，功耗越大。

内存技术

□随机存取存储器

» 动态随机存取存储器（DRAM）：高密度、低功耗、价格便宜、速度慢——动态：需要定期“刷新”

＞ 静态随机存取存储器（SRAM）

•低密度、高功耗、价格昂贵、速度快

•静态：内容将“永久”保留（直到断电）

□不同场景使用哪种存储器?

＞ 主存是动态随机存取存储器（DRAM）：你需要它容量大，所以你需要它价格便宜

＞ CPU缓存是静态随机存取存储器（SRAM）：你需要它速度快，所以它更贵，并且由于资源限制，它比你通常期望的尺寸要小

□相对性能

» 大小：DRAM/SRAM：DRAM比SRAM大4 - 8倍

» 成本/周期时间：SRAM/DRAM：SRAM速度快8 - 16倍，价格更贵

内存层次结构：一种自然的解决方案

□ 我们如何提供一种访问时间短、容量大且价格低的内存?

□第一条原则：让常见情况快速执行!

＞ 常见情况是什么?

□回想：引用局部性原则!

＞ 程序在任何时刻都只访问地址空间中相对较小的一部分。

＞ 好的，我们应该让这些访问操作更快。

＞ 我们可以将最近访问过的项目存放在高速内存中。

□ 对呀：更小的内存速度会更快!

＞ 我们可以使用更昂贵、容量更小的内存来存放最近使用过的项目。

＞ 小容量内存的成本和功耗影响会降低。

什么是内存层次结构?

□内存层次结构分为多个级别：

＞ 每一级都比下一级更小、更快，且每字节成本更高。

＞ 时间局部性（时间上的局部性）：

⇒ 使最近访问的数据项更靠近处理器

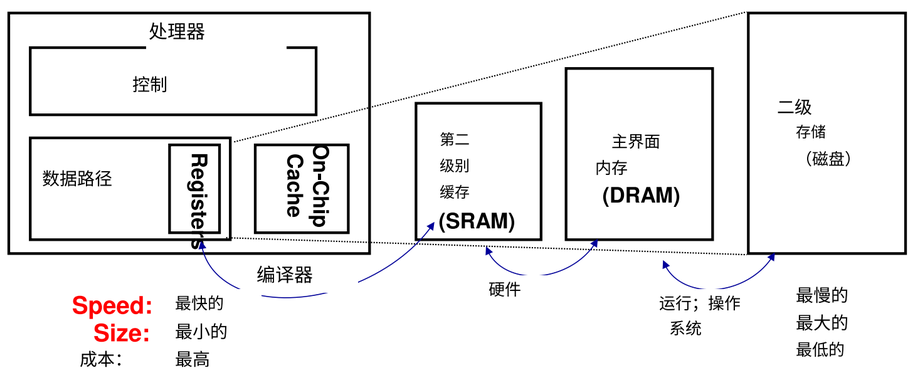
＞ 空间局部性（空间上的局部性）：

⇒ 将由连续字组成的块移动到更快的级别

内存层次结构

□ 目标：提供一种存储系统，其成本几乎与最便宜的存储级别一样低，速度几乎与最快的存储级别一样快。 □ 解决方案：

＞ 最大、最慢的存储器中提供整个可寻址存储空间
＞ 逐渐变小且速度更快的存储器，每个存储器都包含其下一级存储器的一个子集，逐步向处理器靠近



不同的关注点在于

桌面计算机、服务器和嵌入式计算机

□ 桌面计算机：

＞ 主要为单用户运行一个应用程序

＞ 更关注内存层次结构的平均延迟。

□ 服务器计算机：

＞ 可能有数百个用户同时运行潜在的数十个应用程序。

＞ 关注内存带宽。

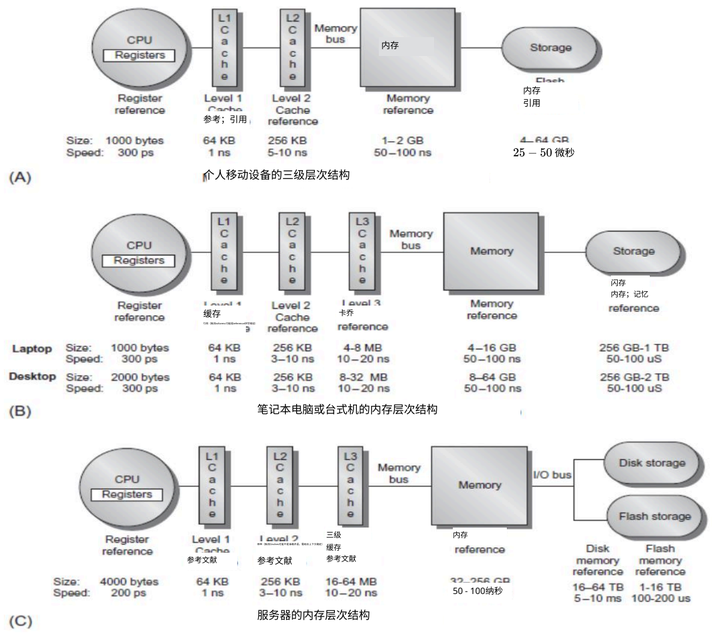
□嵌入式计算机：

＞ 用于实时应用，因此最坏情况下的性能是关注重点

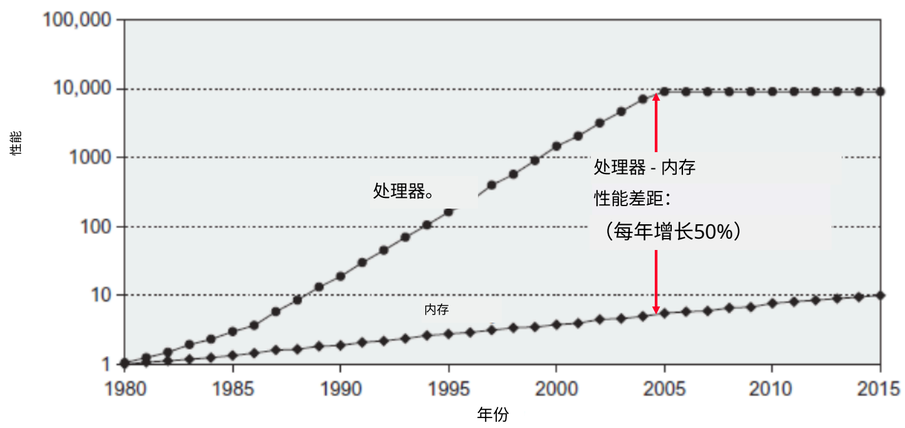
＞ 功率和电池续航，可能不选择硬件优化

＞ 使用非常简单的操作系统仅运行一个应用程序，因此内存的保护作用通常会减弱。

内存层次结构



内存性能差距



□1980：μ 处理器中无缓存；
□1989 首款在芯片上带有缓存的英特尔 μ 处理器
□2001：2 级片上缓存

内存层次结构设计

□ 随着近期多核处理器的发展，内存层次结构设计变得更加关键：

» 聚合峰值带宽随核心数量增加而增长：
-英特尔酷睿i7每个时钟周期每个核心可产生两次引用
-四个核心和 $3.2\mathrm{GHz}$ 时钟
＞ 25.6 十亿次64位数据引用/秒 +
＞ 12.8 十亿次128位指令引用/秒
$>= 409.6\mathrm{GB/s}$!
-动态随机存取存储器（DRAM）带宽仅为此的 8% （34.1 GB/秒）

要求：
＞ 多端口、流水线式缓存
＞ 每个核心有两级缓存
＞ 芯片上有共享的三级缓存

性能与功耗

□高端微处理器的片上缓存 >10 MB ＞ 占用大量面积和功耗预算

缓存基础知识回顾

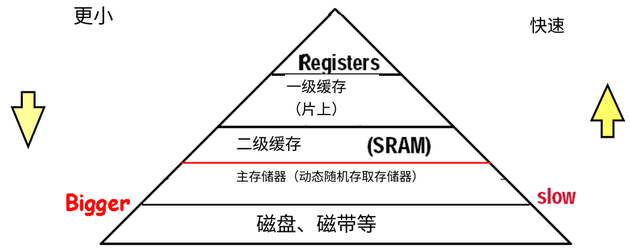
缓存的36个术语

缓存	虚拟内存	
数据缓存	指令缓存	统一缓存
块	页	标签字段
块地址	索引字段	块偏移
全相联	组相联	直接映射
n路组相联	组	地址跟踪
每条指令的缺失次数	内存停顿周期	缺失代价
有效位	脏位	局部性
缓存命中	命中时间	
缓存未命中	未命中率	缺页中断
直写	回写	写分配
随机替换	最近最少使用	非写分配
写缓冲区的平均内存访问时间		写停顿

什么是缓存?

□用于提高对慢速内存平均访问时间的小型快速存储设备。

□在计算机体系结构中，几乎所有东西都是缓存！
＞ 寄存器——变量的“缓存”，由软件管理
＞ 一级缓存——二级缓存的缓存
＞ 二级缓存——内存的缓存
＞ 内存——磁盘（虚拟内存）的缓存
＞ 快表（TLB）：页表缓存
＞ 分支预测：预测信息缓存?



内存层次结构设计者的四个问题

□问题1：一个块可以放置在上层的什么位置?
（块放置）
＞ 全相联、组相联、直接映射
□问题2：如果一个块在上层，如何找到它?
（块识别）
>标签/块
□问题3：未命中时应替换哪个块？（块替换）

>随机、最近最少使用（LRU）、先进先出（FIFO）

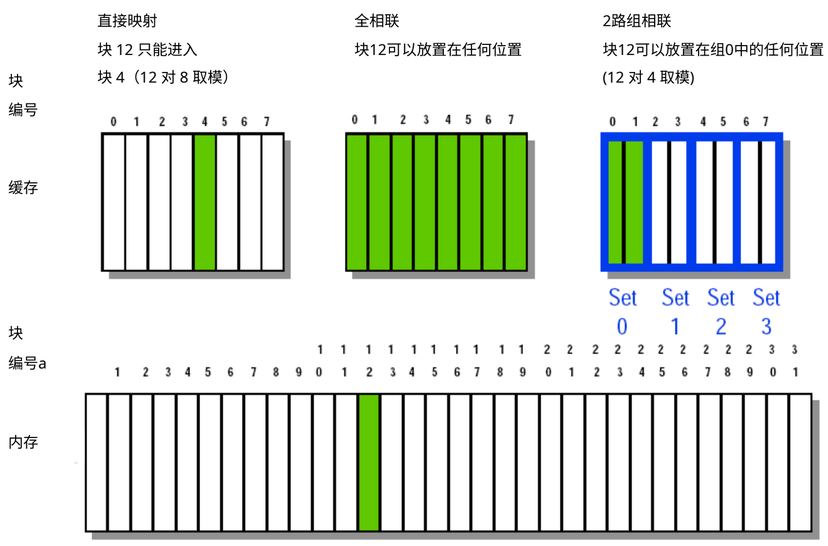
□问题4：写入时会发生什么?
（写入策略）
＞ 写回或直写（带写缓冲区）

问题1：块放置

□ **直接映射**
＞ 块只能存放在缓存中的一个位置
通常为 (地址) 模 (缓存中的块数)
□ 全相联
＞ 块可以存放在缓存中的任意位置。
□ 组相联
＞ 块可以存放在缓存中某一组位置中的任意一处。
＞ 组是缓存中的一组块。
(块地址) 模 (缓存中的组数)
＞ 如果组中有 n 个块，则称该缓存为 n 路组相联。

•请注意，直接映射等同于 1 路组相联，而全相联是 m 路组相联（对于有 m 个块的缓存）。

8 - 32块放置

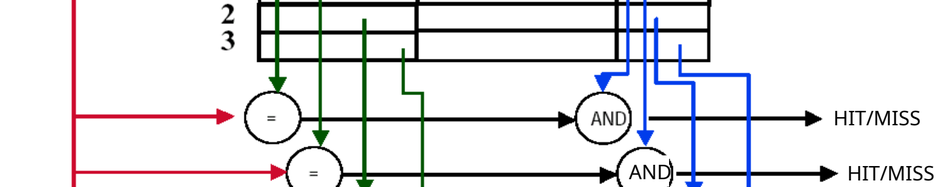


问题2：块标识

□每个块都有一个地址标签，用于存储该块中所存储数据的主存地址。
在检查缓存时，处理器会将请求的内存地址与缓存标签进行比较——如果两者相等，则发生缓存命中，且数据存在于缓存中

□ 通常，每个缓存块还有一个有效位，用于指示缓存块的内容是否有效

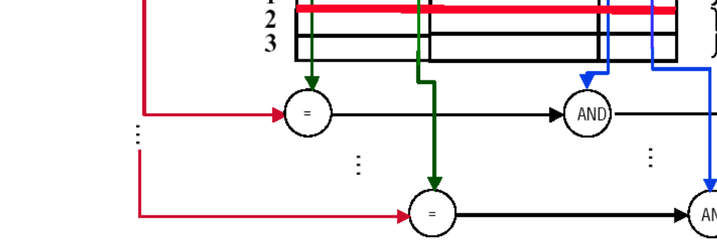
物理地址的格式



□索引字段选择
　＞ 在组相联缓存的情况下，选择组 ＞
在直接映射缓存的情况下，选择块□字节偏移字段选择 ＞ 块内的字节 ＞ 位数
与 $\log_2$（块大小）相同
□标签用于在组内或缓存中查找匹配的块

　＞ 位数与
（地址大小） – （索引大小） – （字节偏移大小）

直接映射缓存示例（单字块）



全相联缓存示例（单字块）

二路组相联缓存

□ 假设缓存有4个块，且每个块为1个字

□ 每组2个块，因此每个缓存有2组

示例：组相联缓存

□内存大小：4G，缓存8K，2路组相联

问题3：块替换

□ 在直接映射缓存中，只有一个块可以被替换

❑ In set-associative and fully-associative caches, there are N blocks
(其中 N 是关联度

块替换策略

☐随机替换 - 随机选择任何块
> 在硬件中易于实现，仅需一个随机数生成器
在缓存中均匀分配
•可能会淘汰即将被访问的块
☐ 最近最少使用（LRU）——选择集合中最近最少被访问的块
> 假设最近被访问的块更有可能再次被引用
> 这需要在缓存中使用额外的位来跟踪访问情况。
☐先进先出（FIFO）——从最早进入缓存的组中选择一个块

替换算法的实现

☐伪最近最少使用（LRU）☐ 示例：

	V	NV
A	1	0
B	0	1
C	0	0
D	0	0

☐ 当发生缺失时：
☐ 踢出受害者
☐ 使下一受害者成为受害者
☐ 并从剩下的两个方块中选择一个作为下一受害者

另一种伪LRU

☐一组（四路）用3位

☐ 用一位表示AB中哪个是最近最少使用（LRU）的
☐ 用一位表示CD中哪个是最近最少使用（LRU）的
☐ 用于表示AB / CD中哪个是最近最少使用（LRU）的一位

问题4：撰写策略

[当数据写入缓存（存储时），数据是否也会写入主内存?

☐直写：信息同时写入缓存中的块和低速内存中的块
•缓存控制位：仅有效位
•内存（或其他处理器）始终拥有最新数据
•始终与写缓冲区结合使用，以免等待低速内存
☐回写：信息仅写入缓存中的块。仅当修改后的缓存块被替换时，才会将其写入主内存

•缓存控制位：有效位和脏位
•带宽低得多，因为重复写访问时无需写入慢速内存

写入策略的优缺点

☐直写优点：
> 读缺失不会导致写入，
> 内存层次结构一致，且易于实现。

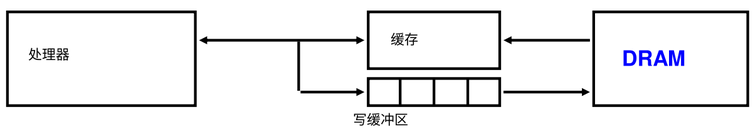
☐回写优点：
> 写入操作以缓存速度进行
> 当对同一块进行多次写入时，主存带宽会变小。

写停顿

☐写停顿.....在直写过程中，CPU必须等待写操作完成时

☐写缓冲区
> 一个小缓存，可保存几个等待写入主存的值，以避免写操作时停顿
>当写入操作集中进行时，此缓冲区会起到帮助作用。
>它并不能完全消除停顿，因为如果突发写入量大于缓冲区容量，缓冲区有可能会被填满。

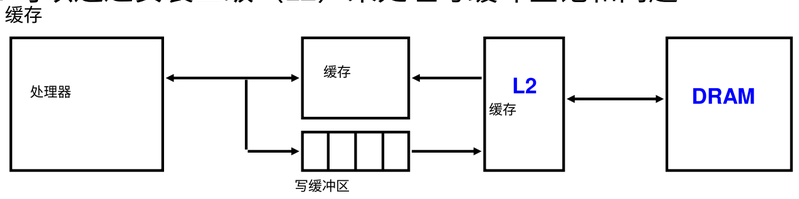
通过缓冲进行直写



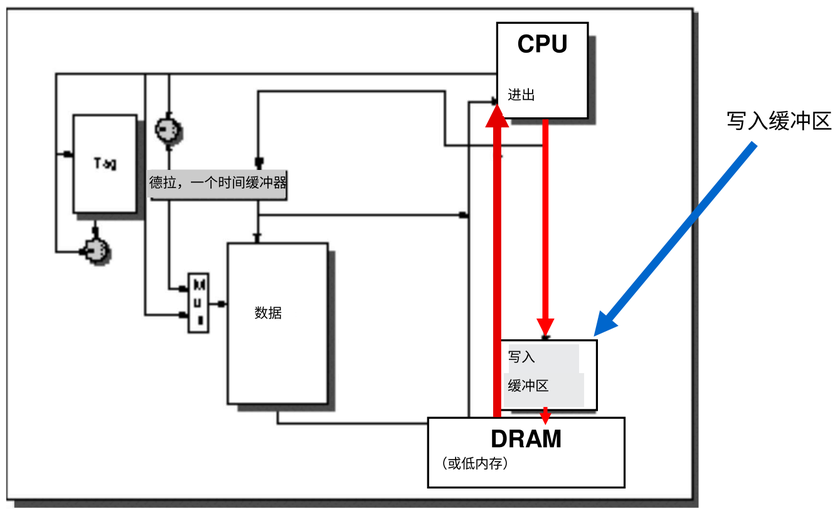
处理器将数据写入缓存和写缓冲区，内存控制器将缓冲区的内容写入内存。写入频率增加会导致写缓冲区饱和。如果CPU周期时间过快和/或连续有太多存储指令：

> 无论存储缓冲区设置得多大，它都会溢出
•CPU周期时间接近DRAM写入周期时间
☐ 可以通过安装二级（L2）来处理写缓冲区饱和问题

缓存



写入缓冲区



未命中时写入策略

如果写入时发生未命中（块不存在）

有两种选择。

□写入分配

•发生未命中时，在其他操作之前将块加载到缓存中。

□绕过写入（不进行写入分配）

•该块仅写入主内存

•它不存储在缓存中。

□一般来说，回写式缓存使用写分配策略，并且

直写式缓存使用绕写策略。

示例

¬ 假设一个全相联写回缓存，有许多缓存项且初始为空。以下是一个包含五个内存操作的序列（地址用方括号表示）：

1 写入 Mem[100];
2 写入 Mem[100];
3 读取 Mem[200];
4 写入内存[200];
5 写入内存[100];

命中次数是多少
以及在不使用……时的未命中次数
写分配与写
分配?

答案：

对于非写分配
未命中次数：1,2,3,5
命中次数： 4

写分配
未命中次数：1,3
命中次数： 2,4,5

分离式缓存与统一式缓存

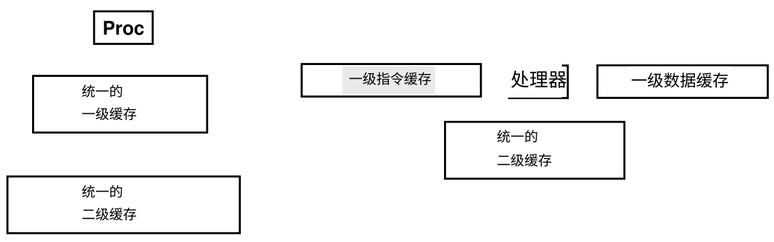
□统一式缓存

＞ 所有内存请求都通过单个缓存。

>这需要较少的硬件，但命中率也较低□分离指令和数据缓存

>指令和数据分别使用单独的缓存。

•这需要额外的硬件，不过也有一些简化之处（指令缓存是只读的）。



分离式缓存与混合式缓存

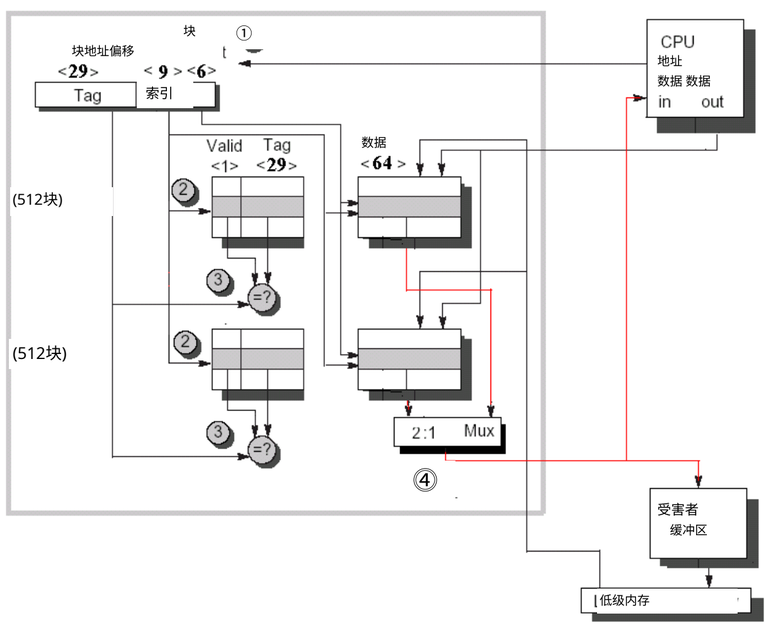
□两路组相联缓存每1000条指令的缺失次数。

大小	指令缓存	数据缓存	统一缓存
8KB	8.16	44.0	63.0
16KB	3.82	40.9	51.0
32KB	1.36	38.4	43.3
64KB	0.61	36.9	39.4
128KB	0.30	35.3	36.2
256KB	0.02	32.6	32.9

□ 平均缺失率 = 指令 %× 指令缺失率 + 数据 %× 数据缺失率

□ 分离式：消除因指令块和数据块冲突导致的未命中问题，但指令和数据的缓存空间是固定的。

示例：Alpha 21264数据缓存



监控器缓存 / 用户缓存

□指令缓存

□监控器/用户空间位

•1：仅监控器可访问

•0：监控器/用户均可访问

5.3 缓存性能

CPU执行时间=

$= (\text{CPU clock cycles} + \text{Memory stall cycles}) \times$ 时钟周期时间

内存停顿周期 $= \text{IC} \times$ 每条指令的内存引用次数 $\times$ 缺失率 $\times$ 缺失代价

CPU时间 $= \text{IC} \times \left(\text{CPI}_{\text{Execution}} + \frac{\text{MemAccess}}{\text{Inst}} \times \text{MissRate} \times \text{MissPenalty}\right) \times$ 时钟周期时间

CPU时间 $= \text{IC} \times \left(\text{CPI}_{\text{Execution}} + \frac{\text{MemMisses}}{\text{Inst}} \times \text{MissPenalty}\right) \times$ 时钟周期时间

$\text{CPI}_{\text{Execution}}$ 包括ALU和内存指令

平均内存访问时间

$$\text{平均内存访问时间} = \frac{\text{Whole accesses time}}{\text{All memory accesses in program}}$$

$$= \frac{\textbf{Accesses time on hitting+ Accesses time on}}{\textbf{All memory accesses in program}}$$

$$= \text{命中时间} + (\text{失效率} \times \text{失效开销})$$

$$= (HitTime_{Inst} + MissRate_{Inst} \times MissPenalty_{Inst}) + (HitTime_{Data} + MissRate_{Data} \times MissPenalty_{Data})$$

$$CPUtime = IC \times \left(\frac{AluOps}{Inst} \times CPI_{AluOps} + \frac{MemAccess}{Inst} \times AMAT\right) \times CycleTime$$

缓存性能指标

□失效率

＞ 与硬件速度无关。

□平均内存访问时间（AMAT）

＞ 优于失效率，但

＞ 性能的间接度量 □CPU时间

练习1：对性能的影响

□假设一个处理器以

» 时钟频率 = 200 MHz（每个周期 5 纳秒），理想（无缺失）CPI = 1.1

>50% 为算术/逻辑运算，30% 为加载/存储操作，20% 为控制操作

□假设 10% 的内存操作会产生 50 个周期的缺失惩罚

□假设 1% 的指令会产生相同的缺失惩罚

□计算平均访存时间（AMAT）和实际CPI。

-答案：$\mathrm{CPI}=$ 理想 $\mathrm{CPI}+$ 每条指令的平均停顿次数

$=1.1$（周期/指令）$+[0.30$（数据访存操作/指令）

x 0.10（缺失/数据访存操作）x 50（周期/缺失）] +

1（指令访存操作/指令）x 0.01（缺失/指令访存操作）x 50（周期/缺失)]

$=(1.1+1.5+.5)$ 周期/指令 $=3.1$

64%的时间里，进程因等待内存而停滞！

· 平均访存时间 $=(1/1.3)\times[1.1+0.01\times 50]+(0.3/1.3)\times[1.1+0.1\times 50]$

- =2.54

示例2：对性能的影响

假设：理想CPI = 1（无缺失）

□$\mathrm{L/S}$ 的结构。 50% 的指令为数据访问 □缺失代价为25个时钟周期

□缺失率为2%

□ 如果所有指令都能在缓存中命中，计算机的速度会快多少?

□ 答案：首先计算始终命中时的性能：

$\mathrm{CPU_{time}}=(\mathrm{CPU\ clock\ cycles}+\mathrm{memory\ stall\ cycles})\times$ 时钟周期 $=(\mathrm{IC}\times\mathrm{CPI}+0)\times$ 时钟周期

$=1\mathrm{C}\times 1.0\times$ 时钟周期

示例2的答案（续）

现在来看带有真实缓存的计算机，首先计算内存停顿周期：内存停顿周期 $=\mathrm{IC}\times\frac{\text{Memory accesses}}{\text{Instruction}}\times$ 失效率 $\times$ 失效代价

$=IC\times(1+0.5)\times 0.02\times 25=IC\times 0.75$

因此，总体性能如下：

CPU执行时间缓存 $=(\mathrm{IC}\times 1.0+\mathrm{IC}\times 0.75)\times$ 时钟周期 $=1.75\times\mathrm{IC}\times$ 时钟周期性能比是执行时间的倒数 $\frac{CPU\ execution\ time}{}=\frac{1.75\times IC\times Clockcycle}{}$ CPU执行时间 $1.0\times 10\times$ 时钟周期无缓存缺失的计算机速度快1.75倍。

示例3：对性能的影响

假设：统一缓存： 32 K 统一缓存

□分离缓存：16K数据缓存和16K指令缓存

□36% 的指令是数据传输指令

□一次命中需要1个时钟周期

□未命中惩罚为100个时钟周期

□在统一缓存上，一次加载/存储操作额外需要1个时钟周期

□采用写缓冲的直写策略

并且忽略写缓冲导致的停顿

每种情况下的平均内存访问时间是多少?

统一缓存和分离缓存的缺失率

两路组相联缓存每1000条指令的缺失情况

大小	指令缓存	数据缓存	统一缓存
8KB	8.16	44.0	63.0
16KB	3.82	40.9	51.0
32KB	1.36	38.4	43.3
64KB	0.61	36.9	39.4
128KB	0.30	35.3	36.2
256KB	0.02	32.6	32.9

示例3的答案

答案：首先，让我们将每1000条指令的缺失次数转换为缺失率。

$$\text{Miss rate}=\text{缺失}\ \frac{1000\ \text{Instructions}}{\text{Memory accesses}}$$

instructions

由于每次指令访问恰好有一次内存访问来获取指令，根据图5.8，指令缺失率为

$$\text{Miss rate}_{\text{16KB instruction}}=\frac{3.82/1000}{1.0}=0.0038$$

由于36%的指令是数据传输，根据图5.8，数据缺失率为

$$\text{Miss rate}_{\text{16KB data}}=\frac{40.9/1000}{0.36}=0.1136$$

示例3的答案（续）

根据图5.8，统一缺失率需要考虑指令和数据访问情况：

$$\text{Miss rate}_{\text{32KB unified}}=\frac{43.3/1000}{1.00+0.36}=0.0318$$

根据第138页的图2.32，在分离缓存中指令引用占比为74%。分离缓存的平均缺失率为：

$$(74\%\times 0.0038)+(26\%\times 0.1136)=0.0323$$

因此，一个32KB的统一缓存的有效缺失率略低于两个16KB的缓存。

示例3的答案（续）

•平均内存访问时间可分为指令访问和数据访问：

平均内存访问时间

$= \%$ 说明 $\times (\mathrm{HitTime}_{\mathrm{Inst}} + \mathrm{MissRate}_{\mathrm{Inst}} \times \mathrm{MissPenalty}_{\mathrm{Inst}}) + \%$ 数据 $\times (\mathrm{HitTime}_{\mathrm{Data}} + \mathrm{MissRate}_{\mathrm{Data}} \times \mathrm{MissPenalty}_{\mathrm{Data}})$ - 因此，每个组织的时间是平均内存访问时间 $_{\text{split}}$

$= 74\% \times (1 + 0.0038 \times 100) + 26\% \times (1 + 0.1136 \times 100)$

$= (74\% \times 1.38) + (26\% \times 12.36) = 1.021 + 3.214 = 4.25$ 平均内存访问时间 $_{\text{unified}}$

$= 74\% \times (1 + 0.0318 \times 100) + 26\% \times (1 + 1 + 0.0318 \times 100)$

=(74%×4.18)+(26%×5.18)=3.093+1.347=4.40

示例4：对性能的影响

假设：按序执行的计算机，如Ultra SPARC III。

缺失代价：100个时钟周期
缺失率：2%
每条指令的内存引用次数：1.5
每1000条指令的平均缓存未命中次数：30
时钟周期数（CPI） = 1.0（忽略内存停顿）

当考虑缓存行为时，对性能有何影响（使用每条指令的未命中次数和未命中率来计算影响）？

示例4的答案

答案：包括缓存未命中在内的性能为

$$CPU_{time} = IC \times \left(CPI_{exexution} + \frac{Mem\,stall\,clock\,cycles}{Instruction} \right) \times Clock\,cycle\,time$$

带有缓存的CPU时间 $=$

$= \mathrm{IC} \times (1.0 + (30/1000 \times 100)) \times$ 时钟周期时间

$= \mathrm{IC} \times 4.00 \times$ 时钟周期时间

现在使用缺失率计算性能：

$\mathrm{CPU}_{\mathrm{time}} = \mathrm{IC} \times \left(\mathrm{CPI}_{\mathrm{execution}} + \mathrm{Missrate} \times \frac{\text{Mem accesses}}{\text{Instruction}} \times \mathrm{Misspenalty} \right) \times$ 时钟周期时间带缓存的CPU时间 $= = \mathrm{IC} \times (1.0 + (1.5 \times 2\% \times 100)) \times$ 时钟周期时间 $= \mathrm{IC} \times 4.00 \times$ 时钟周期时间

示例4的答案（续）

无论有无缓存，时钟周期时间和指令数量都是相同的。因此，CPU时间增加了四倍，时钟周期数（CPI）从“完美缓存”的1.00增加到可能发生未命中的缓存的4.00。

□如果完全没有内存层次结构，时钟周期数（CPI）将再次增加到 $1.0 + 100 \times 1.5$ ，即比有缓存的系统长近40倍。

缓存缺失对CPU有双重影响

Q $\mathrm{CPI}_{\mathrm{execution}}$ 越低，固定数量的缓存缺失时钟周期的相对影响就越高。

□在计算CPI时，缓存未命中代价以未命中时的CPU时钟周期来衡量。因此，即使两台计算机的内存层次结构相同，时钟频率较高的CPU每次未命中的时钟周期数更多，因此CPI中的内存部分也更高。

示例5：对性能的影响

假设： $\mathrm{CPI} = 2$ （完美缓存）时钟周期时间 $= 1.0\mathrm{ns}$ - 每条指令的内存引用次数（MPI） = 1.5

- •两个缓存的大小均为 $64\ \mathrm{K}$ ，块大小为64字节
- •一个缓存是直接映射的，另一个是两路组相联的。前者的失效率为1.4%，后者的失效率为1.0%
- •选择多路复用器使CPU时钟周期时间延长至1.25倍
- •失效代价为75纳秒，命中时间为1个时钟周期

□两种不同的缓存组织方式对CPU性能有何影响（首先，计算平均内存访问时间，然后计算CPU性能）？

示例5的答案

答案：平均内存访问时间为

平均内存访问时间 $=$ 命中时间 + 未命中率 $\times$ 未命中代价

因此，每个组织的时间为

平均内存访问时间 $_{\text{1-way}} = 1.0 + (0.014 \times 75) = 2.05$ ns 平均内存访问时间 $_{\text{2-way}} = 1.0 \times 1.25 + (0.01 \times 75)$

$= 2.00$ ns

对于两路组相联缓存，平均内存访问时间更优。

示例5的答案（续）

CPU性能为

$$CPUtime = IC \times \left(CPI_{execution} + \frac{Misses}{Instruction} \times Misspenalty \right) \times Clock\,cycle\,time$$

$$= IC \times \left[\left(CPI_{execution} \times Clock\,cycle\,time \right) + \left(Miss\,rate \times \frac{Memory\,accesses}{Instruction} \times Misspenalty \times Clock\,cycle\,time \right) \right]$$

将75纳秒代入（缺失损失 $\times$ 时钟周期时间），每种缓存组织方式的性能为

CPU时间 $_{\text{1-way}} = \mathrm{IC} \times (2 \times 1.0 + (1.5 \times 0.014 \times 75)) = 3.58 \times \mathrm{IC}$

CPU时间 $_{\text{2-way}} = \mathrm{IC} \times (2 \times 1.0 \times 1.25 + (1.5 \times 0.010 \times 75)) = 3.63 \times \mathrm{IC}$

示例 5 的答案（续）

相对性能为

$$\frac{CPU\,time_{2-way}}{CPU\,time_{1-way}} = \frac{3.63 \times Instruction\,count}{3.58 \times Instruction\,count} = \frac{3.63}{3.58} = 1.01$$

与平均内存访问时间的结果相反，直接映射方式带来了稍好的平均性能。因为 CPU 时间是我们的最终评估指标。

缺失代价与乱序执行处理器

如何定义“缺失代价”?
□它是缺失访问内存的全部延迟，还是
这仅仅是处理器必须停顿的“暴露”或非重叠延迟吗?

□ 对于按序处理器来说，这毫无疑问，但这里却并非如此。
□细化内存停顿，以将缺失代价重新定义为非重叠延迟：

$$\frac{\text{Memory stall cycles}}{\text{instruction}} = \frac{\text{Misses}}{\text{instruction}} \times \text{（总缺失延迟 - 重叠缺失延迟）}$$

两个定义

□内存延迟时长——在乱序处理器中，应将什么视为一次内存操作的开始和结束。
□ 重叠延迟时长——与处理器的重叠从何时开始（或者等效地说，我们何时认为一次内存操作.....）
操作正在使处理器停滞)

示例6：乱序处理器的性能

假设：75纳秒（52.5）缺失代价的30%可以重叠；其他参数与示例5（上一个示例）相同
直接映射缓存对乱序（OOO）CPU的性能有什么影响?
答案：乱序计算机的平均内存访问时间为：平均内存访问时间 $_{1-\text{way,OOO}} = 1.0^{*}1.25 + (0.014 \times 52.5)$

$$= 1.74 \text{ ns}$$

乱序缓存的性能为：
CPU时间 $_{1-\text{way,OOO}} = \text{IC} \times (2 \times 1.0 * 1.25 + (1.5 \times 0.014 \times 52.5))$
$= 3.10 \times IC$

因此，尽管时钟周期时间慢得多且直接映射缓存的失效率较高，但如果乱序计算机能够隐藏30%的失效代价，它可能会稍微快一些。

内存层次结构基础

六项基本缓存优化措施：

＞ 更大的块大小
- •减少强制性缺失
- •增加容量缺失和冲突缺失，增加缺失代价

＞ 更大的总缓存容量以降低缺失率
- •增加命中时间，增加功耗

＞ 更高的关联性
- •减少冲突缺失
- •增加命中时间，增加功耗

＞ 更多的缓存层级
- •减少总体内存访问时间

＞ 对读缺失给予比写更高的优先级——减少缺失代价
＞ 避免在缓存索引中进行地址转换——减少命中时间

如何提高缓存性能?

平均内存访问时间（AMAT）= 命中时间 + 失效率 × 失效代价

□减少命中时间(4)
＞ 小型且简单的一级缓存、路径预测 ＞ 避免地址转换、跟踪缓存□增加带宽(3) ＞ 流水线缓存、多体缓存、非阻塞缓存□减少缺失代价(5) ＞ 多级缓存、读缺失优先于写操作、 ＞ 关键字优先、合并写缓冲区和牺牲缓存□降低缺失率(4) ＞ 更大的块大小、更大的缓存容量、更高的关联性 ＞ 编译器优化□通过并行化减少缺失代价或缺失率(1) ＞ 硬件或编译器预取