# 第4章 存储器-层次结构设计

- 4.1 引言
- 4.2 Cache 性能
- 4.3 Cache优化技术
- 4.4 主存储器技术
- 4.5 虚拟存储器
- 4.6 谬误和易犯的错误
- 4.7 小结

# 4.4 主存储器

本节描述在存储器芯片内部的技术。

2个测量指标—访问时间、访存周期时间

- 访问时间----- 一个读请求开始到读出的字到达所需时间
- 访存周期时间 ----- 访问存储器2个请求之间的时间

访存周期比访问时间更大，原因是在两次访问之间存储器需要地址线是稳定的。

# 4.4 主存储器

- 主存储器存储层次结构中cache的下一层。

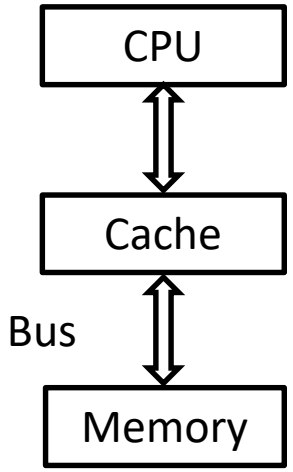
  也称为内存

  主存通常采用DRAM，caches采用SRAM。

- 主存的性能
  - 延迟Latency：减小更困难

    影响cache的缺失代价
  - 带宽Bandwidth：用新的组织更容易改善*带宽*

    *（单位B/CLK）* 也改善cache的缺失代价




- 前几节描述的cache优化可以减小平均访存时间。
- 本节分析如何组织主存以改善带宽，也减小了缺失代价。

# 基本主存组织的性能

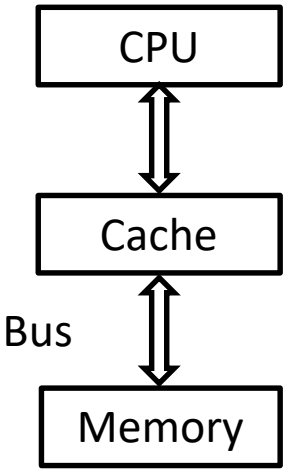
第1级cache通常用1个字的物理带宽组织，因为大多数CPU访问是以字为单位。Cache失效时访问主存。

假设： 一个字8字节，64位总线

① 4 个时钟周期发送地址

② 访问每个字花费56 时钟周期

③ 4 个时钟周期发送一个字数据




Cache块32B（256位），4个字（字64位，8B）

缺失代价：

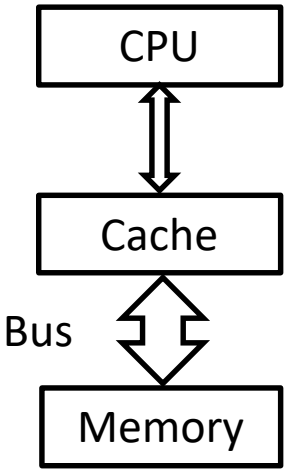
4×(4+56+4)＝256 CLKs

带宽：

$$\frac{4\times 8}{256} = \frac{1}{8} = 0.125 \quad \text{B/CLK}$$

# 第1种提高带宽的技术：增加主存储器带宽

- 主存容量应该与CPU速度增长成线性比例。
- 回顾
  - 存储器容量增加：4倍/每3年
  - 但DRAM性能改善：7%/每年，CPU-DRAM性能差距一直是一个问题。
  - 现在，看看更具成本效益的主存组织改进。
- 增加主存带宽
  - 双倍或四倍cache与主存之间的带宽
  - 主存带宽也将增加双倍或四倍




缺点： 增加bus宽度、增加存储器控制器的硬件

# 单体多字存储器的性能

- 主存宽度2个字（128位，块32B），4个字/块

  缺失代价：

  $$2\times(4+56+4)=128\ \text{CLKs}$$

  带宽：

  $$\frac{4\times 8}{128}=\frac{1}{4}=0.25\ \text{B/CLK}$$

- 主存宽度4个字（256位，块32B），4个字/块

  缺失代价：

  $$1\times(4+56+4)=64\ \text{CLKs}$$

  带宽：

  $$\frac{4\times 8}{64}=\frac{1}{2}=0.5\ \text{B/CLK}$$

# 单体存储器性能比较

- 主存宽度为1个字（64位）的主存：

  缺失代价 256 CLKs，带宽 1/8 B/CLK

- 主存宽度为2个字（128位）的主存：

  缺失代价 128 CLKs，带宽 1/4 B/CLK

- 主存宽度为4个字（256位）的主存：

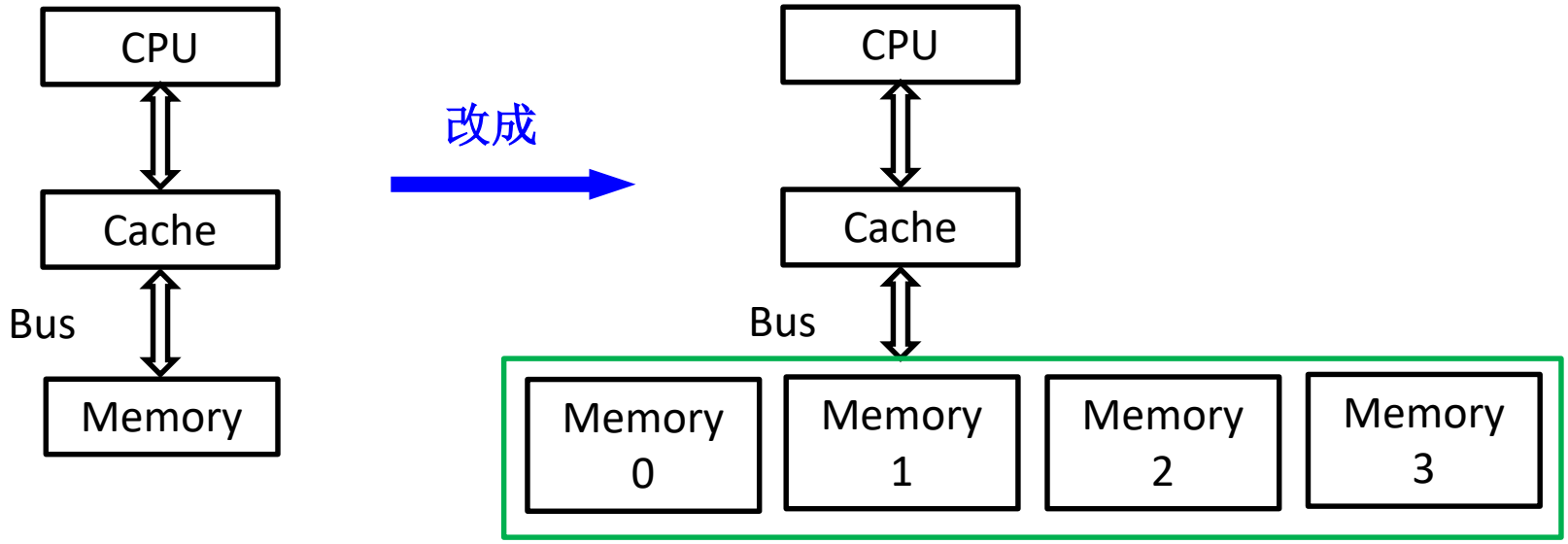
  缺失代价 64CLKs，带宽 1/2 B/CLK

单体多字存储器的性能随着主存宽度的增加而改善，缺失代价减少，带宽增加。代价是增加硬件。

# 第2种提高带宽的技术：简单交叉存储器

优点：

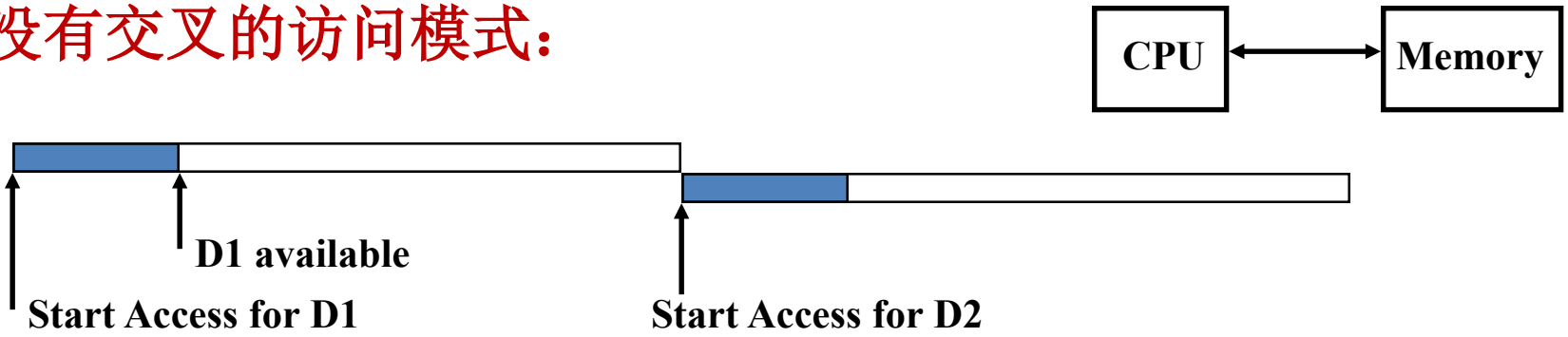
利用了存储系统中具有多个DRAM体潜在的并行性。
这种主存组织对连续访问的写直达策略尤其重要。



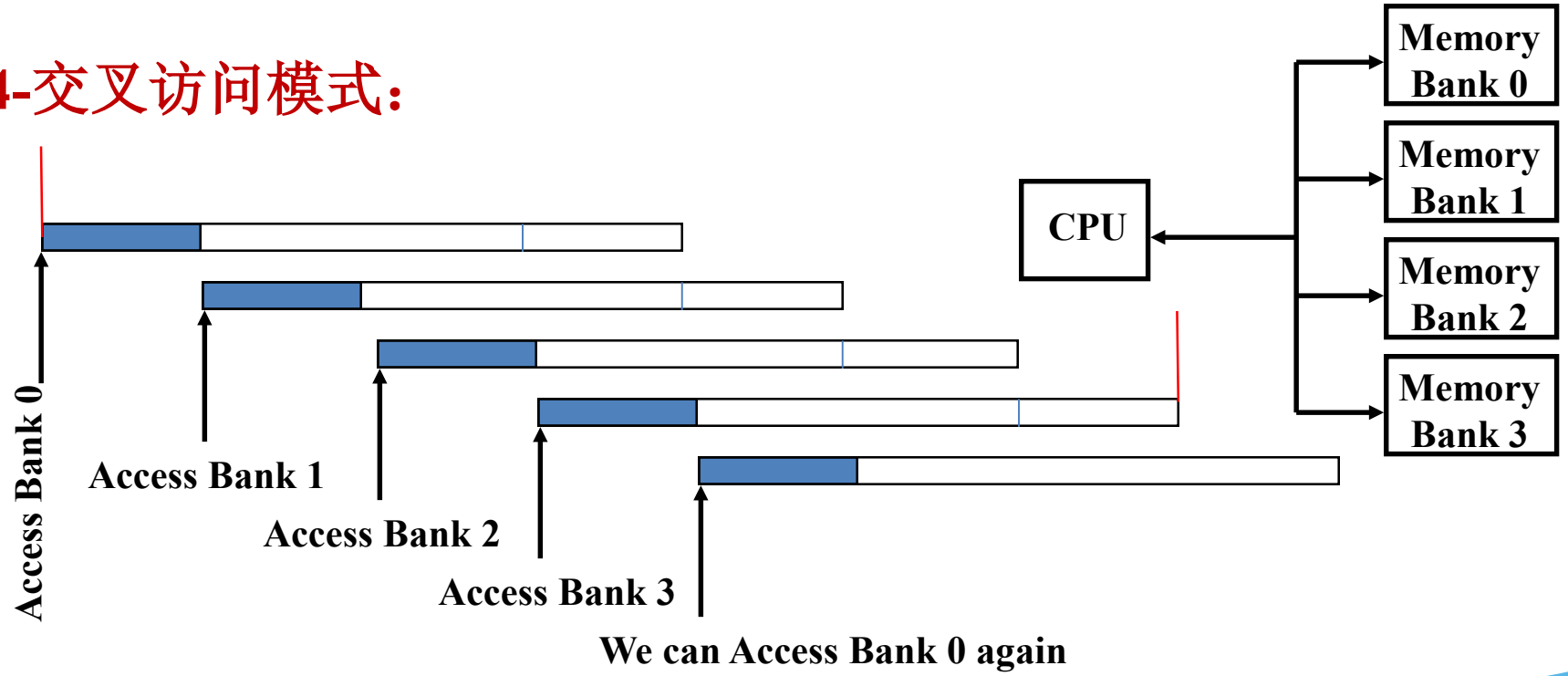

存储器芯片按*多个体组织*以便并行*读或写多个字*。
每个体通常是*1个字*宽度，因此总线与cache宽度不需要改变。
但是，要*分时发送地址*到多个体，使得它们几乎都同时读数据。

# 怎样访问存储体

## 没有交叉的访问模式：




## 4-交叉访问模式：



# 交叉存储器的性能

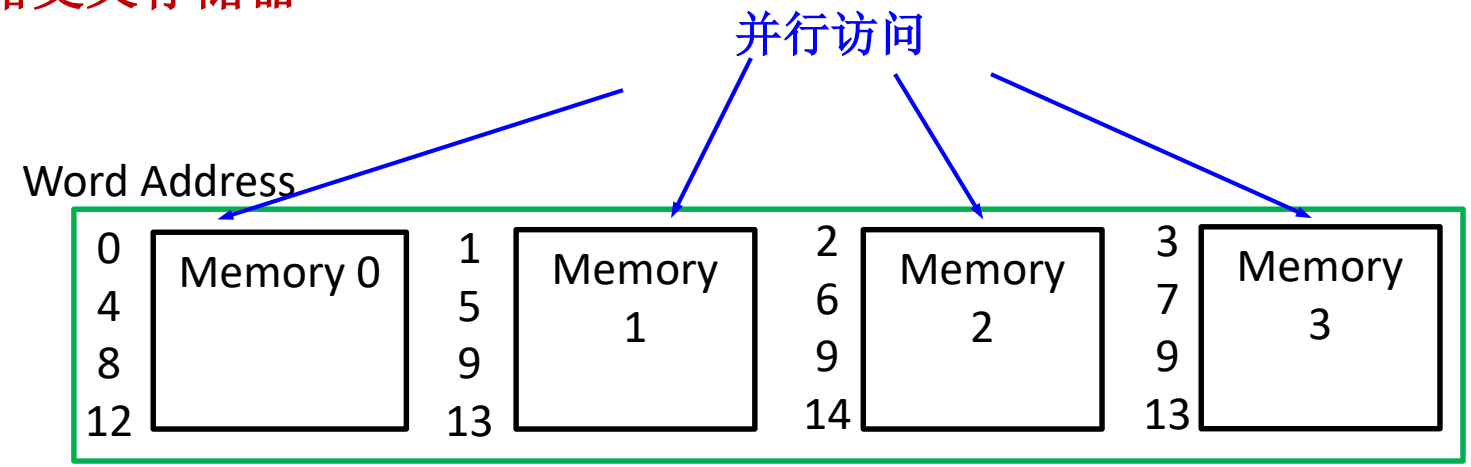
- **4个体的交叉存储器，4个字/块**

  **缺失代价：**

$$4+56+(4\times 4)=76$$

**带宽：** $\dfrac{4\times 8}{76}=0.421$ **B/CLK**

**4-路交叉存储器**




优化连续地址访问模式

# 例 交叉存储器

- 假设：

  块大小 = 1 个字

  存储器总线宽度 = 1 个字

  缺失率 = 3%

  平均每条指令访存次数 = 1.2

  Cache 缺失代价 = 4 + 56 + 4= 64 个时钟周期

  理想CPI（忽略cache缺失）= 2

- 当块大小增加到2个字、4个字后，用CPI分析交叉存储器与宽存储器的性能？

答案： 使用1个字1块机器的实际CPI

$$CPI_{real} = 2 + (1.2 \times 3\% \times 64) = 4.304$$

可以通过计算CPI比较不同方案的性能改善。

# 例 交叉存储器

## 答案

- 增加块容量到2个字，给出以下选择方案（缺失率2%）：

- 64-bit 总线和存储器，没有交叉

$$CPI = 2 + (1.2 \times 2\% \times 2 \times 64) = 5.072$$

- 64-bit 总线和存储器，有交叉

$$CPI = 2 + (1.2 \times 2\% \times (4 + 56 + 8)) = 3.632$$

- 128-bit 总线和存储器，没有交叉

$$CPI = 2 + (1.2 \times 2\% \times 1 \times 64) = 3.536$$

块容量增大1倍后：直接实现会减慢（5.072 对比 4.304）；

交叉 64位存储器快 1.185 倍（比 4.304 ）；

128位加宽存储器快1.217 倍（比 4.304 ）。

- 如果增加块容量到4个字，性能会怎样？

- 增加块容量到4个字，缺失率 1.2%，给出以下选择方案：

- 64-bit 总线和存储器，没有交叉

$$CPI = 2 + (1.2 \times 1.2\% \times 4 \times 64) = 5.686$$

- 64-bit 总线和存储器，有交叉

$$CPI = 2 + (1.2 \times 1.2\% \times (4 + 56 + 16)) = 3.094$$

- 128-bit 总线和存储器，没有交叉

$$CPI = 2 + (1.2 \times 1.2\% \times 2 \times 64) = 3.843$$

- 增加块容量到4个字：简单实现会降低性能（5.686对比4.304）；

交叉64-bit 存储器快1.391倍（比 4.304 ）；

128位宽存储器及总线快1.12倍（比 4.304 ）。

# 第4章 存储器-层次结构设计

- 4.1 引言
- 4.2 Cache 性能
- 4.3 Cache优化技术
- 4.4 主存储器技术
- 4.5 虚拟存储器
- 4.6 谬误和易犯的错误
- 4.7 小结

# 4.5 虚拟存储器

什么是虚拟存储器？

- 允许执行的一个程序
  - 可以放在不连续的存储位置
  - 不必全部放在主存中
- 允许计算机“欺骗”一个程序相信
  - 存储器是连续的
  - 存储空间比物理存储器更大，提供了很大存储空间的*假象*

为什么VM 很重要？

- 便宜 – 不必买许多RAM
- 解脱了程序员管理存储器资源的负担
- 允许多道程序设计，分时共享，保护

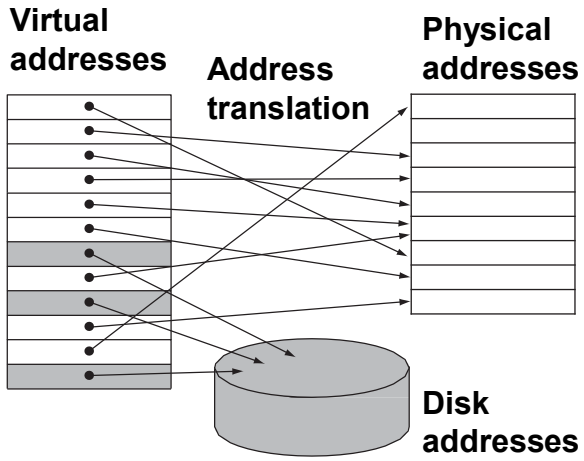
# 优点

- 主存（物理存储器）作为辅助存储器（磁盘）的一个Cache
- 具有更大和连续物理存储器的假象
- 程序按“页”或“段”重定位
- 多道程序中的保护

*虚拟地址*：
程序员使用的地址

*虚拟地址空间*：
虚拟地址的集合

*存储器地址*：
物理存储器中的字地址
也称为“物理地址”
或“实际地址”



# VM怎样工作

## 2个存储器“空间”

- 虚拟存储器空间-- 程序员所“看见”
- 物理存储器空间—程序运行时所在（RAM的大小）

## 在程序启动时

- OS 将程序复制到RAM
- 如果没有足够的RAM空间，OS 停止复制，开始运行已经装入RAM的部分程序
- 当程序需要运行不在RAM中的那部分时，OS 将从磁盘复制这部分程序到RAM
- 为了从磁盘复制部分程序到RAM，OS 需要替换已经在RAM中的部分程序
  - 如果需要移走程序的页是脏的（被修改过），OS 则要复制有脏标志的页到磁盘

# 虚拟存储器的存储层次参数

不同存储层次的术语有区别

- 块 ——*页*或*段*
- 缺失——*页失效*或*地址失效*
- *存储器映射*或*地址转换* ——对于虚拟存储器， CPU 用*硬件和软件*的组合将 *虚拟地址*转换成*物理地址*。—— 操作系统转换

虚拟存储器控制的2层存储结构层次：

*DRAMs* 和*磁盘*。

# 再来审视
# 存储层次结构的4个问题

## 问题1：一块可以放在主存什么位置?（映射）

- 高缺失代价
  - 相当高
    - 访问旋转的磁盘，延迟大，上百万时钟周期
- 必须有更低的缺失率
  - 选择一种更简单的放置算法
  - 为了避免过高的缺失代价，操作系统设计者通常选择更低的缺失率
- 全相联策略
  - 因此，OS允许块放在主存的任意位置

# 问题2：如何在主存中查找一个块？

页式和段式都依赖数据结构表（页表或段表）

- 数据结构包含块的物理地址。
- 按页号或段号查找块

- 段式：
  - 在段的物理地址上加上偏移地址就得到物理地址。
- 页式：
  - 在物理页地址后简单拼接偏移地址就得到物理地址。
- 表的大小
  - 虚拟地址空间里的页数。
  - 假设：32-bit 虚拟地址，4-KB 页，每页4 bytes 表项；页表的大小为：

$$\frac{2^{32}}{2^{12}} \times 2^2 = 2^{22}\text{B} = 4\ \text{MB}$$

# 问题3：发生虚拟存储器缺失时应该替换哪一块？

## 最小化页缺失率

- 几乎所有操作系统都替换最近最少使用（LRU-least-recently used）的块
- 按照局部性原则，被替换的块正是将来很少可能用到的块。

## 实现机制

- 很多处理器为一页提供一位使用位或参考位
  - 逻辑上是只要一页被访问该位就置位。
  - 操作系统周期性清除使用位，随后记录它们。
  - 按照这种方式保持跟踪，操作系统可以选择出最近最少使用过的一页。

# 问题4：写操作时会发生什么？

## 写策略

- 在主存的下一级是旋转的磁盘，访问磁盘需要花费上百万个时钟周期时间。
  - 因此，写策略采用写回方式。

## 脏位

  - 由于磁盘访问代价过高，仅仅只有在替换时，被修改的块才写回磁盘。

# 快速地址转换技术

## 页表存放主存中：

- 页表很大，可能分页；
- 从主存取数据、存结果、取指令，至少需要2次访问主存：

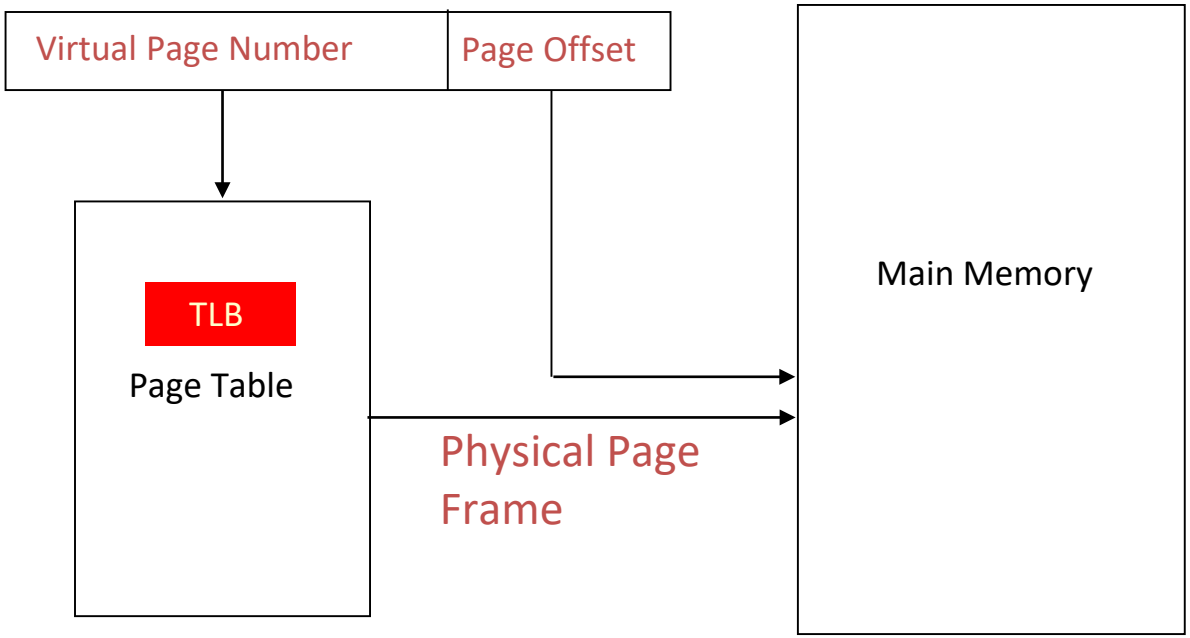
  一次按虚拟地址访存从页表获得物理地址，二次访存获得数据。
- 访存时间太长！减少虚拟地址转换为物理地址的时间。

## 解决方法：

使用一个特殊的cache，存放最近用过的页表项 。这个cache就是TLB（ *Translation Lookaside Buffer* ）变换旁路缓冲器。

# 虚实地址转换与页表

- 在页的范围内，虚实地址相等
- TLB是页表的cache



# TLB

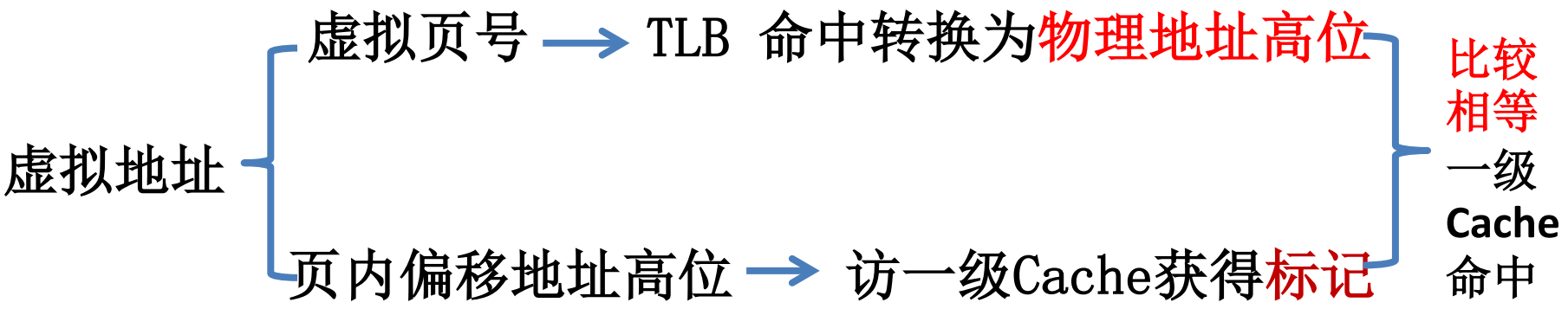
- **TLB实际上是操作系统中页表的Cache**
  - TLB主要负责完成用户空间到物理空间的转化
  - 一般与Cache访问同时进行
  - TLB访问失效时需要把相应的页表项从内存取进TLB
  - TLB中还有保护位

# 虚拟存储器和Cache的综合

**64位虚拟地址 ⟹ 40位物理地址**

- 页大小16KB = $2^{14}$，页内偏移地址14位，虚页号64-14 = 50位。
- TLB二路组相联，256项，128组，索引7位。
- 物理地址40位，物理页号 40 -14 = 26位。
- 一级Cache采用虚拟索引物理标记，16KB直接映像，块64B，256块，8位索引，6位块偏移量，标记26位。
- 二级Cache大小4MB，块64B（512位），$2^{16}$个块，4路组相联，$2^{14}$=16384组，14位索引，6位块偏移量，标记20位。

虚拟地址到物理地址的转化：




如果一级Cache缺失，物理地址将继续用于二级Cache访问。

- 一个程序运行于拥有四项全相联变换旁路缓冲区（TLB）的计算机上

| VP# | PP# | Entry Valid |
|---|---|---|
| 5 | 30 | 1 |
| 7 | 1 | 0 |
| 10 | 10 | 1 |
| 15 | 25 | 1 |

TLB内容

| Virtual page index | Physical page# | Present |
|---|---|---|
| 0 | 3 | Y |
| 1 | 7 | N |
| 2 | 6 | N |
| 3 | 5 | Y |
| 4 | 14 | Y |
| 5 | 30 | Y |
| 6 | 26 | Y |
| 7 | 11 | Y |
| 8 | 13 | N |
| 9 | 18 | N |
| 10 | 10 | Y |
| 11 | 56 | Y |
| 12 | 110 | Y |
| 13 | 33 | Y |
| 14 | 12 | N |
| 15 | 25 | Y |

页表内容

**Exercise B.12**

- 下面是一组由程序访问的虚拟页号。指出每个访问是否会发生 TLB命中/缺失, 如果访问页表, 它是发生页命中还是缺页错误。如果未访问页表, 则在页表列下画一个 X。

| Virtual Page Accessed | TLB (hit or miss) | Page Table (hit or fault) |
|---|---|---|
| 1 | | |
| 5 | | |
| 9 | | |
| 14 | | |

TLB内容

| VP# | PP# | Entry Valid |
|---|---|---|
| 5 | 30 | 1 |
| 7 | 1 | 0 |
| 10 | 10 | 1 |
| 15 | 25 | 1 |

| Virtual Page Accessed | TLB (hit or miss) | Page Table (hit or fault) |
|---|---|---|
| 1 | miss | fault |
| 5 | hit | hit |
| 9 | miss | fault |
| 14 | miss | fault |

页表内容

| Virtual page index | Physical page# | Present |
|---|---|---|
| 0 | 3 | Y |
| 1 | 7 | N |
| 2 | 6 | N |
| 3 | 5 | Y |
| 4 | 14 | Y |
| 5 | 30 | Y |
| 6 | 26 | Y |
| 7 | 11 | Y |
| 8 | 13 | N |
| 9 | 18 | N |
| 10 | 10 | Y |
| 11 | 56 | Y |
| 12 | 110 | Y |
| 13 | 33 | Y |
| 14 | 12 | N |
| 15 | 25 | Y |

**Exercise B.12**

- A program is running on a computer with a four-entry fully associative translation lookaside buffer (TLB):
- The following is a trace of virtual page numbers accessed by a program. For each access indicate whether it produces a TLB hit/miss and, if it accesses the page table, whether it produces a page hit or fault. Put an X under the page table column if it is not accessed.

| VP# | PP# | Entry Valid |
|---|---|---|
| 5 | 30 | 1 |
| 7 | 1 | 0 |
| 10 | 10 | 1 |
| 15 | 25 | 1 |

TLB Content

| Virtual page index | Physical page# | Present |
|---|---|---|
| 0 | 3 | Y |
| 1 | 7 | N |
| 2 | 6 | N |
| 3 | 5 | Y |
| 4 | 14 | Y |
| 5 | 30 | Y |
| 6 | 26 | Y |
| 7 | 11 | Y |
| 8 | 13 | N |
| 9 | 18 | N |
| 10 | 10 | Y |
| 11 | 56 | Y |
| 12 | 110 | Y |
| 13 | 33 | Y |
| 14 | 12 | N |
| 15 | 25 | Y |

Page Table Content

# Exercise B.12

# 第4章 存储器-层次结构设计

- 4.1 引言
- 4.2 Cache 性能
- 4.3 Cache优化技术
- 4.4 主存储器技术
- 4.5 虚拟存储器
- 4.6 谬误和易犯的错误
- 4.7 小结

# 谬误

- **由一个程序的 cache 性能推测另一个程序的 cache 性能:**需要具体分析，否则其结论可能是错的。

# 易犯错误

- **地址空间太小：**地址大小限制了程序长度。 地址大小之所以很难修改，原因在于它确定了所有 与地址相关的最小宽度：PC、寄存器、存储器字和实际地址运算。如果从一开始就没有扩展地址的计划，那么成功改变地址大小的机会微乎其微，通常也就意味着该计算机系列的终结。

# 易犯错误

- **忽视操作系统对存储器层次结构性能的影响：**大约 25%的停顿时间消耗在操作系统的缺失部分中，或者是因为应用程序与操作系统互相干扰而导的缺失部分中。

| Workload | % Time OS misses and application conflicts |
|---|---|
| Pmake | 25.8% |
| Multipgm | 24.9% |
| Oracle | 26.8% |

Figure B.29 Misses and time spent in misses for applications and operating system (1992).

# 易犯错误

- **依靠操作系统来改变页面大小：**因为TLB缺失率极高，所以向TLB中增加了多种较大的页面大小。希望操作系统程序员会将一个对象分配到最大页中，从而保留TLB条目。 在尝试了10来年之后，大多数操作系统仅在精心选择的功能中使用了这些 “超级页面”，比如映射显示存储器或其他I/O设备，或者为数据库代码使用这种极大页面。