计算机体系结构

周学海
xhzhou@ustc.edu.cn
0551-63606864
中国科学技术大学

关于本章的术语

- **Miss： 缺失||失效**
- **Miss rate：缺失率||失效率**
- **Miss penalty：缺失代价||失效开销**
- **multilevel inclusive: 多级包容**
- **Multilevel exclusive：多级不包容||多级互斥**
- **Stall：停顿**
- **Write-through: 写直达||直写**
- **Write-back： 写回**

review-高级Cache优化方法

- **缩短命中时间**
  - 1、小而简单的第一级Cache
  - 2、路预测方法
- **增加Cache带宽**
  - 3、Cache访问流水化
  - 4、无阻塞Cache
  - 5、多体Cache
- **减小失效开销**
  - 6、关键字优先和提前重启
  - 7、合并写
- **降低失效率**
  - 8、编译优化
- **通过并行降低失效开销或失效率**
  - 9、硬件预取
  - 10、编译器控制的预取

Summary

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined cache access	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Banked caches		+			+	1	Used in L2 of both i7 and Cortex-A8
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware.
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs

Figure 2.11 Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity. Although generally a technique helps only one factor, prefetching can reduce misses if done sufficiently early; if not, it can reduce miss penalty. + means that the technique improves the factor, – means it hurts that factor, and blank means it has no impact. The complexity measure is subjective, with 0 being the easiest and 3 being a challenge.

第4章　存储层次结构设计

4.1 Cache的基本概念

存储系统的层次结构

Cache基本知识

4.2 Cache的基本优化方法

4.3 Cache的高级优化方法

4.4 存储器技术与优化

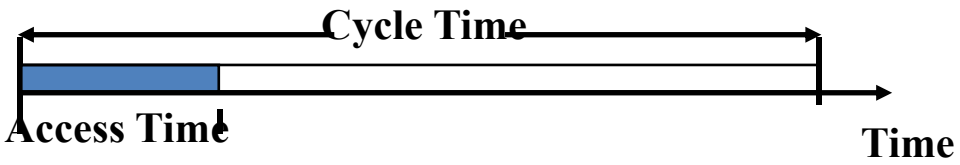
4.5 虚拟存储器 - 基本原理

4.4 存储器技术及优化

DRAM芯片 → 存储器模块

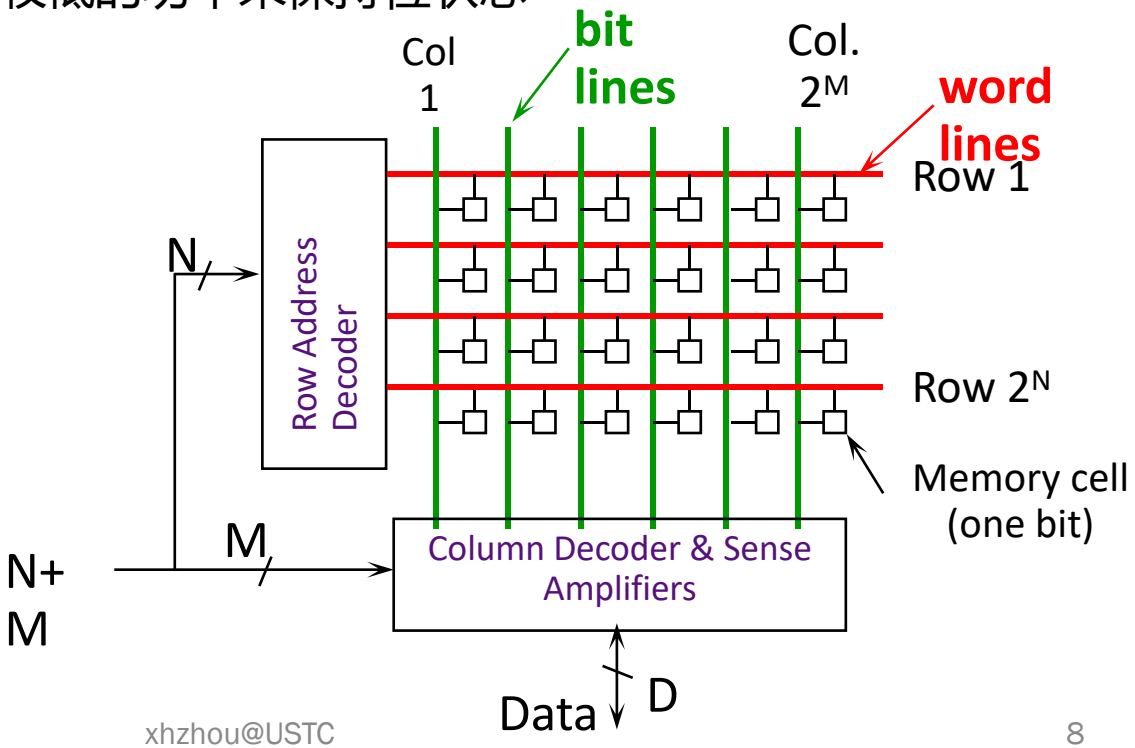
存储器技术与优化

- **存储器的访问源**
  - 取指令、取操作数、写操作数和I/O
- **存储器性能指标**
  - 容量、速度和每位价格
  - 访问时间（Access Time)
  - 存储周期（Cycle Time)
- **种类：DRAM和SRAM**
  - Memory：DRAM，Cache：SRAM



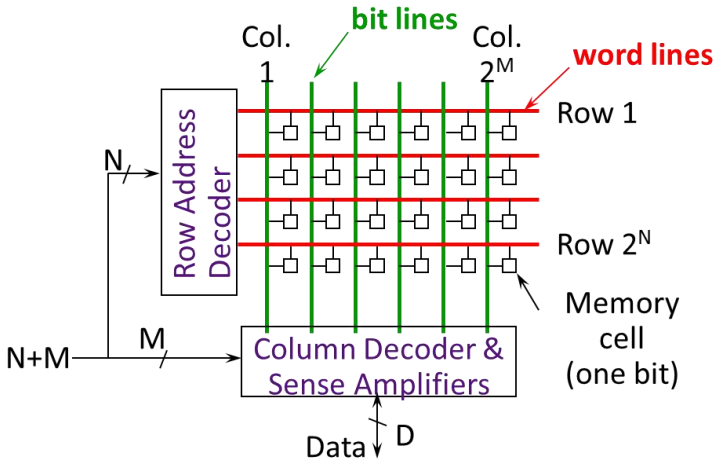
DRAM

- **DRAM**
  - 破坏性读：读后需要重新写回,必须要周期性的刷新；每位1个 transistor
  - 地址线复用:
    - Lower half of address: column access strobe (CAS)
    - Upper half of address: row access strobe (RAS)
- **SRAM**
  - 每位6个transistors；只需较低的功率来保持位状态



DRAM

- **DRAM是由单个信息位构成的阵列**
  - 通过行选择线和列选择线访问
  - 所有DRAM都由这些阵列构成
  - 不同的结构根据性能的需求选择的阵列数可能不同
- **所有DRAM的访问至少三个阶段**
  - Precharge， row access, column access
- **DRAM 的性能**
  - Latency
    - 地址信号有效到第一组数据信号有效所需要的时间
    - 处理器发出请求到所请求的第一组数据到达处理器输入引脚所需要的cycle数
  - Bandwidth
    - 第一组数据到达后，后续数据到达的速率



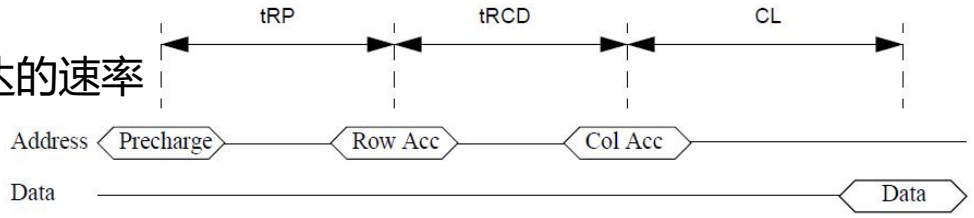
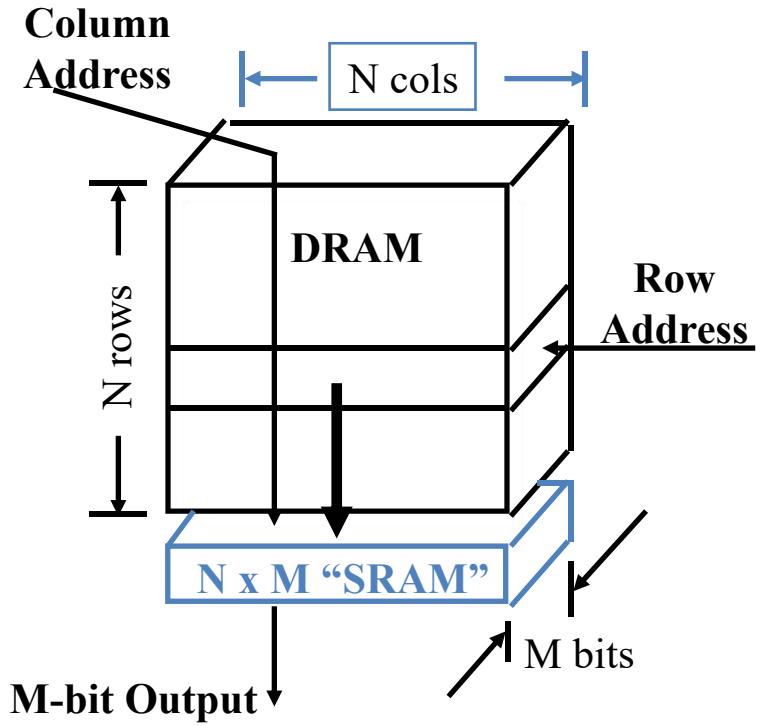
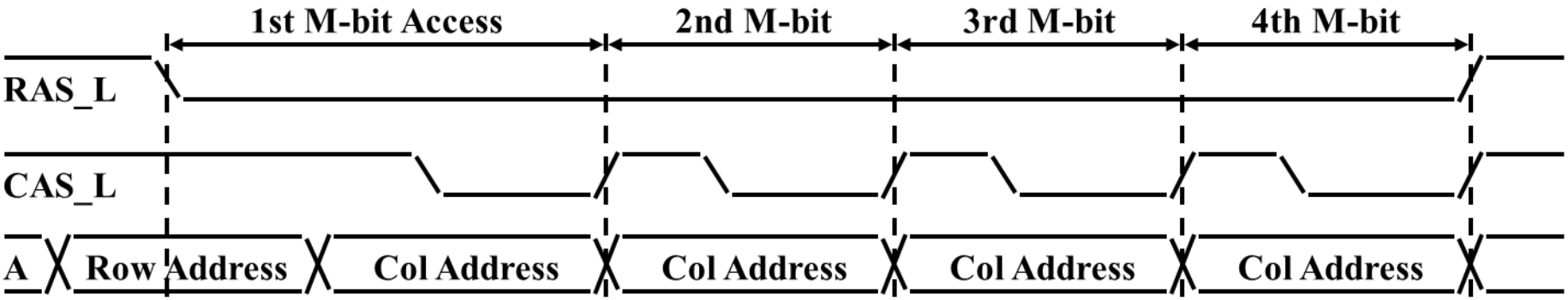


Figure 3.2: Timing Phases of a DRAM Access

Memory 优化

- Some optimizations:
  - Fast Page Mode Operation
    - Multiple accesses to same row
  - Synchronous DRAM
    - Added clock to DRAM interface
    - Burst mode with critical word first
  - Double data rate (DDR)
    - Wider interfaces
    - Multiple banks on each DRAM device

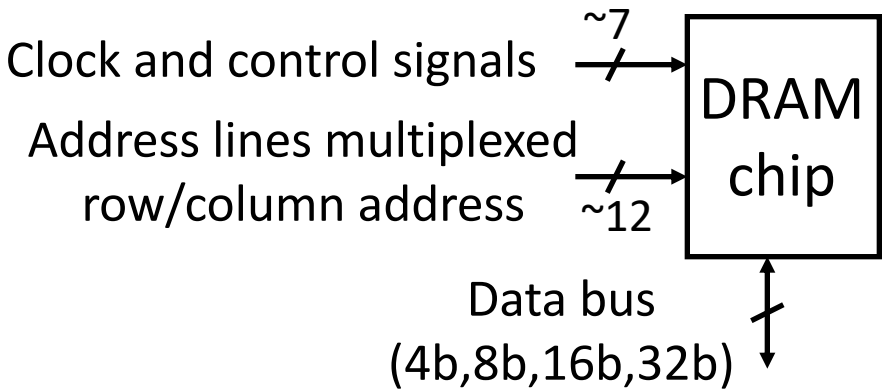




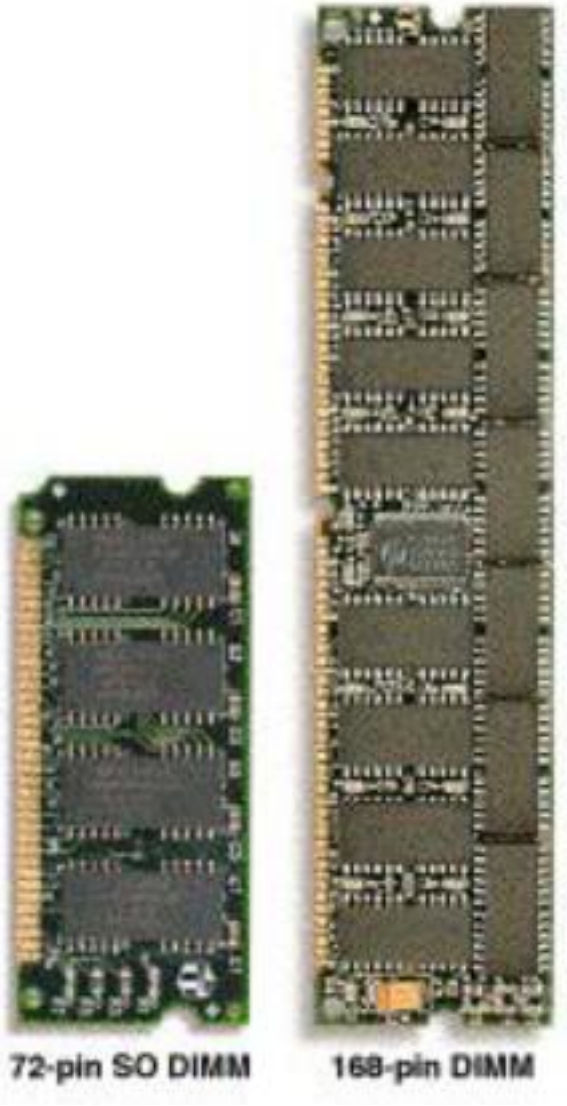
Production year	Chip size	DRAM Type	Row access strobe (RAS)		Column access strobe (CAS)/ data transfer time (ns)	Cycle time (ns)
			Slowest DRAM (ns)	Fastest DRAM (ns)		
1980	64K bit	DRAM	180	150	75	250
1983	256K bit	DRAM	150	120	50	220
1986	1M bit	DRAM	120	100	25	190
1989	4M bit	DRAM	100	80	20	165
1992	16M bit	DRAM	80	60	15	120
1996	64M bit	SDRAM	70	50	12	110
1998	128M bit	SDRAM	70	50	10	100
2000	256M bit	DDR1	65	45	7	90
2002	512M bit	DDR1	60	40	5	80
2004	1G bit	DDR2	55	35	5	70
2006	2G bit	DDR2	50	30	2.5	60
2010	4G bit	DDR3	36	28	1	37
2012	8G bit	DDR3	30	24	0.5	31

Figure 2.13 Times of fast and slow DRAMs vary with each generation. (Cycle time is defined on page 95.) Performance improvement of row access time is about 5% per year. The improvement by a factor of 2 in column access in 1986 accompanied the switch from NMOS DRAMs to CMOS DRAMs. The introduction of various burst transfer modes in the mid-1990s and SDRAMs in the late 1990s has significantly complicated the calculation of access time for blocks of data; we discuss this later in this section when we talk about SDRAM access time and power. The DDR4 designs are due for introduction in mid- to late 2012. We discuss these various forms of DRAMs in the next few pages.

DRAM Packaging (Laptops/Desktops/Servers)



- **DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)**
- **Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)**



Memory Optimizations

Standard	Clock rate (MHz)	M transfers per second	DRAM name	MB/sec /DIMM	DIMM name
DDR	133	266	DDR266	2128	PC2100
DDR	150	300	DDR300	2400	PC2400
DDR	200	400	DDR400	3200	PC3200
DDR2	266	533	DDR2-533	4264	PC4300
DDR2	333	667	DDR2-667	5336	PC5300
DDR2	400	800	DDR2-800	6400	PC6400
DDR3	533	1066	DDR3-1066	8528	PC8500
DDR3	666	1333	DDR3-1333	10,664	PC10700
DDR3	800	1600	DDR3-1600	12,800	PC12800
DDR4	1066–1600	2133–3200	DDR4-3200	17,056–25,600	PC25600

Figure 2.14 Clock rates, bandwidth, and names of DDR DRAMS and DIMMs in 2010. Note the numerical relationship between the columns. The third column is twice the second, and the fourth uses the number from the third column in the name of the DRAM chip. The fifth column is eight times the third column, and a rounded version of this number is used in the name of the DIMM. Although not shown in this figure, DDRs also specify latency in clock cycles as four numbers, which are specified by the DDR standard. For example, DDR3-2000 CL 9 has latencies of 9-9-9-28. What does this mean? With a 1 ns clock (clock cycle is one-half the transfer rate), this indicate 9 ns for row to columns address (RAS time), 9 ns for column access to data (CAS time), and a minimum read time of 28 ns. Closing the row takes 9 ns for precharge but happens only when the reads from that row are finished. In burst mode, transfers occur on every clock on both edges, when the first RAS and CAS times have elapsed. Furthermore, the precharge in not needed until the entire row is read. DDR4 will be produced in 2012 and is expected to reach clock rates of 1600 MHz in 2014, when DDR5 is expected to take over. The exercises explore these details further.

- **DDR:**
  - DDR2：Lower power (2.5 V -> 1.8 V)，Higher clock rates (266 MHz, 333 MHz, 400 MHz)
  - DDR3：1.5 V，800 MHz
  - DDR4：1-1.2 V，1600 MHz
- **GDDR5 is graphics memory based on DDR3**
- **Graphics memory:**
  - Achieve 2-5 X bandwidth per DRAM vs. DDR3
    - Wider interfaces (32 vs. 16 bit)
    - Higher clock rate

https://zhuanlan.zhihu.com/p/335685399

Memory 功耗

Reducing power in SDRAMs:

- Lower voltage
- Low power mode (ignores clock, continues to refresh)

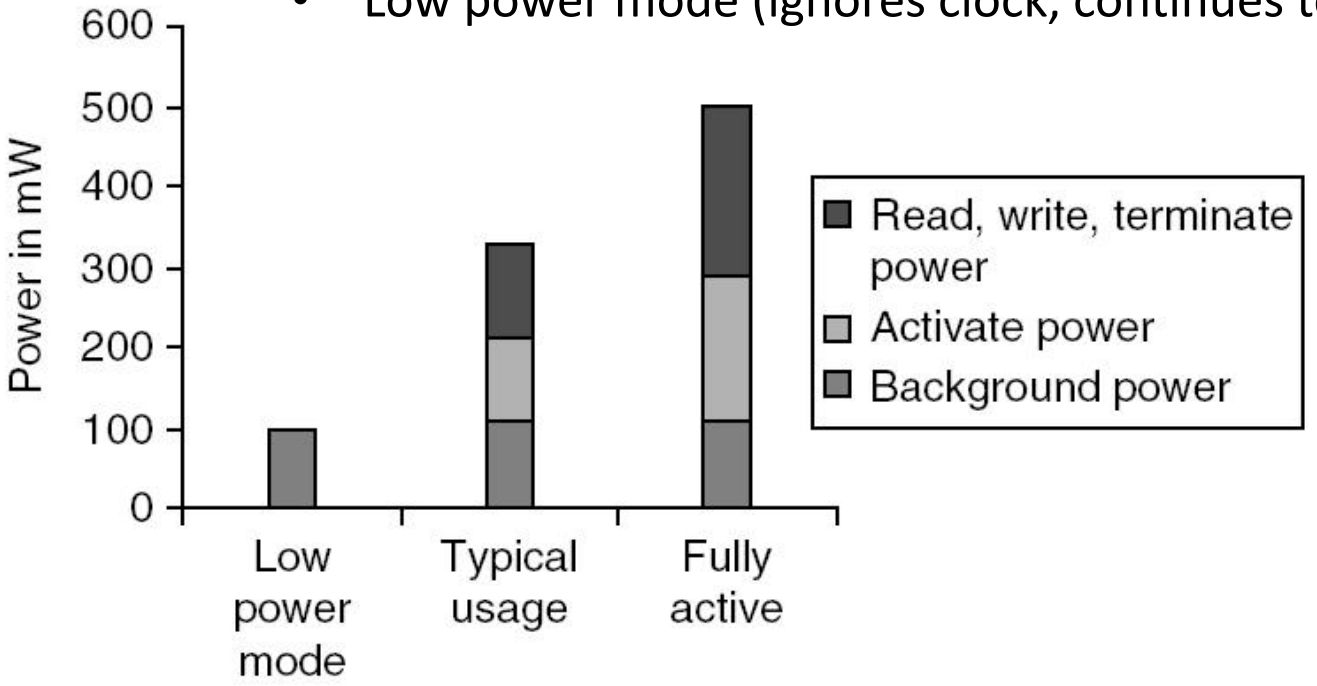
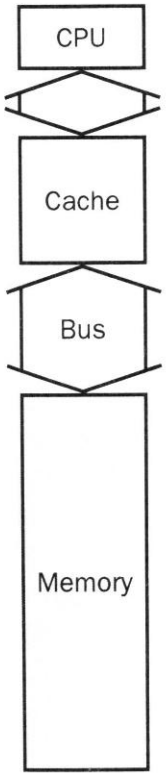


Figure 2.6 Power consumption for a DDR3 SDRAM operating under three conditions: low-power (shutdown) mode, typical system mode (DRAM is active 30% of the time for reads and 15% for writes), and fully active mode, where the DRAM is continuously reading or writing. Reads and writes assume bursts of eight transfers. These data are based on a Micron 1.5V 2GB DDR3-1066, although similar savings occur in DDR4 SDRAMs

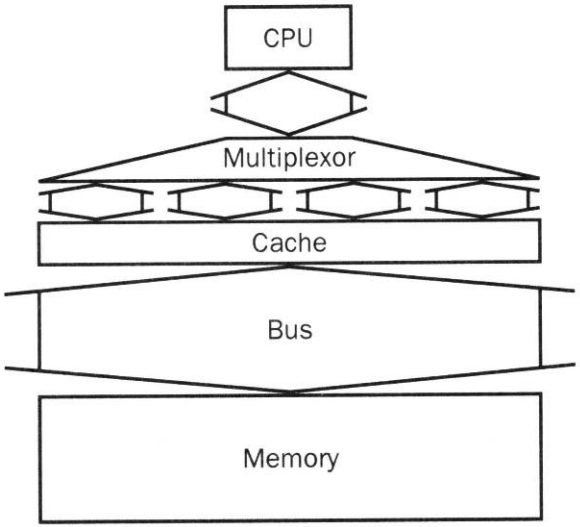
4.4 存储器技术及优化

DRAM芯片 → **存储器模块**

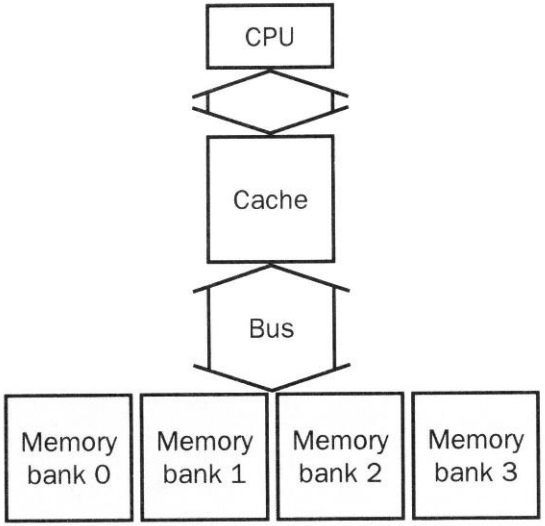
三种存储器组织方式



a. One-word-wide memory organization



b. Wide memory organization



c. Interleaved memory organization

Simple:

CPU, Cache, Bus, Memory same width
(32 bits)

Wide:

CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)

Interleaved:

**CPU, Cache, Bus 1 word: Memory N Modules
(4 Modules); example is *word interleaved***

提高主存性能的方法-增大存储器的宽度（并行访问存储器）

- **最简单直接的方法**
- **优点：简单、直接，可有效增加带宽**
- **缺点**
  - 增加了CPU与存储器之间的连接通路的宽度，实现代价提高
  - 主存容量扩充时，增量应该是存储器的宽度
  - 写操作问题（部分写操作）
- **冲突问题**
  - 取指令冲突，遇到程序转移时，一个存储周期中读出的n条指令中，后面的指令将无用
  - 读操作数冲突。一次同时读出的几个操作数，不一定都有用
  - 写操作冲突。这种并行访问，必须凑齐n个字之后一起写入。如果只写一个字，必须先把属于同一个存储字的数据读到数据寄存器中，然后在地址码的控制下修改其中一个字，最后一起写。
  - 读写冲突。当要读写的字在同一个存储字内时，无法并行操作。
- **冲突的原因**
  - 从存储器本身看，主要是地址寄存器和控制逻辑只有一套。

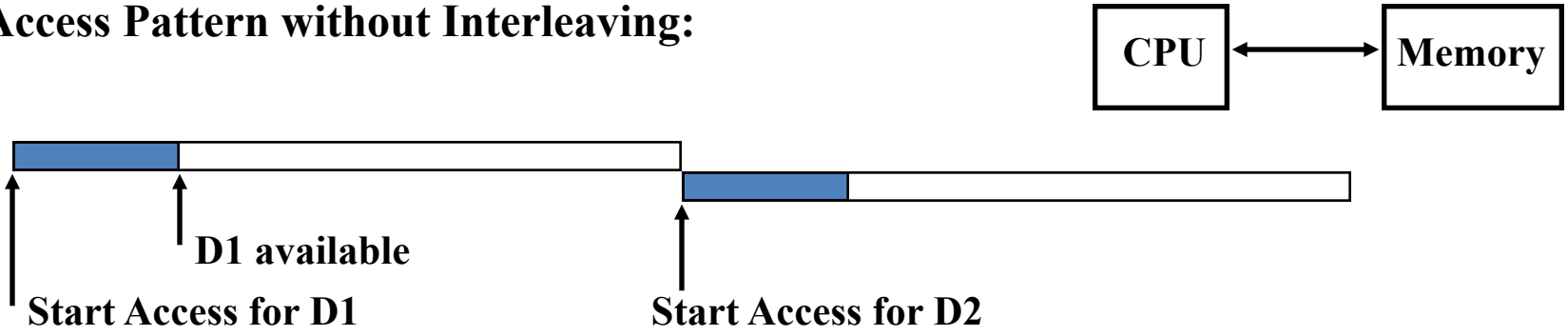
采用简单的多体交叉存储器

- **一套地址寄存器和控制逻辑**
- **存储器组织为多个体（Bank）**
- **存储体的宽度，通常为一个字，不需要改变总线的宽度**
- **目的：在总线宽度不变的情况下，完成多个字的并行读写**

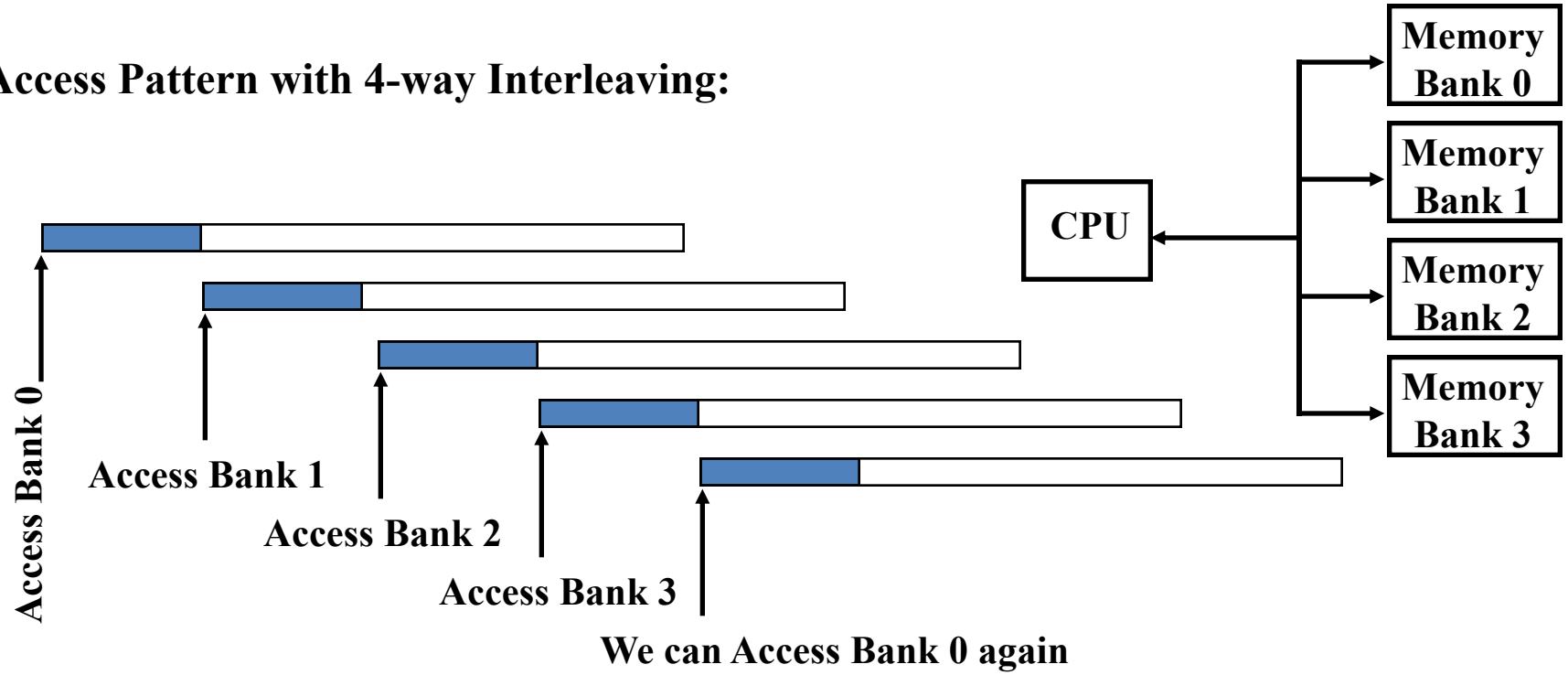
- **存储模块中所包含的体数，为避免访问冲突，基本原则为：**
  - 体的数目 >= 访问体中一个字所需的时钟周期数
  - 例如：某一向量机的存储系统，CPU发出访存请求10个时钟周期后，CPU将从存储体0得到一个字，随后体0开始读该存储体的下一个字，而CPU依次从其余7个存储体中得到后继的7个字。在第18个周期，CPU 将需要存储体0提供下一个字，但该字要到第20个时钟周期才被读出，CPU只好等待。
- **缺陷：不能对单个体单独访问，对解决冲突没有帮助，逻辑上是一种宽存储器，对各个存储体的访问被安排在不同的时间段**

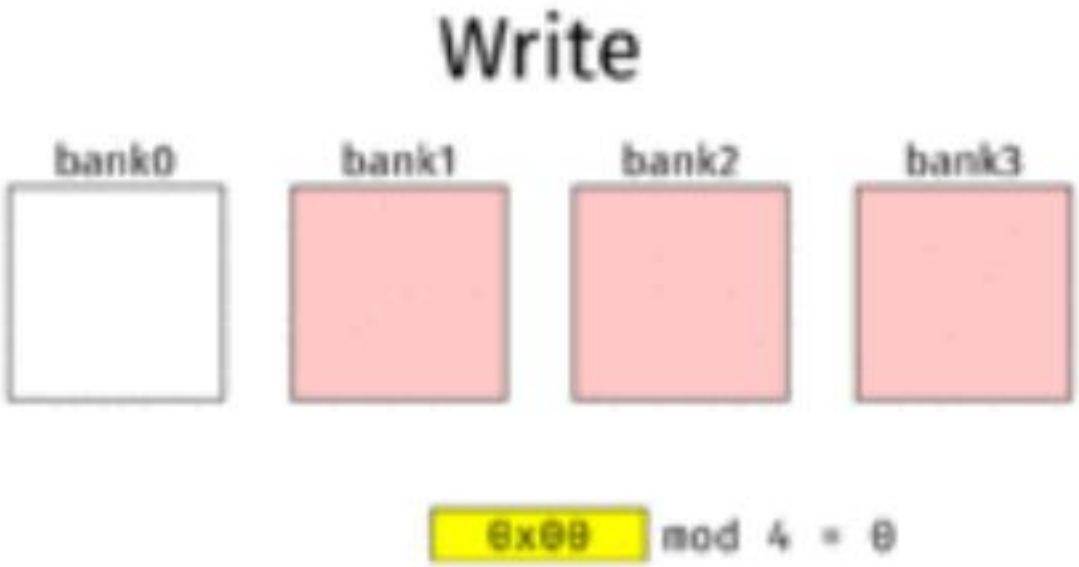
Increasing Bandwidth - Interleaving

Access Pattern without Interleaving:



Access Pattern with 4-way Interleaving:




Write
bank0
bank1
bank2
bank3
0x08
mod 4 = 0

地址映射方法（m个存储体，每个存储体容量为n)

- **高位交叉编址：**相当于对存储单元矩阵按列优先的方式进行编址

  考虑处于第 i 行第 j 列的单元，其线性地址为：

  $A = j \times n + i$　　// 其中：j =0~m-1 ；i = 0~n-1

  已知：一个单元的线性地址为A， 则：

  j = $\lfloor A/n \rfloor$　　//A / n 取下限

  i = $A \bmod n$

  取线性地址高位 $\log_2 m$位就是体号，其余低位部分就是体内地址

- **低位交叉编址：**相当于对存储单元矩阵按行优先的方式进行编址

  考虑处于第 i 行第 j 列的单元，其线性地址为：

  $A = i \times m + j$　　//其中：i = 0~n-1 ；j = 0~m-1

  已知：一个单元的线性地址为A， 则：

  i = $\lfloor A/m \rfloor$　　j = A mod m

  取线性地址低位 $\log_2 m$位就是体号，其余高位部分就是体内地址

例题：

- **举例：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
  - 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
  - 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?**

- **假设：**
  - 送地址：4cycles
  - 每个字的访问时间（存取周期）为24cycles,
  - 传送一个字的数据需要4cycles

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%。根据前面给出的访问时间，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?

在改变前的机器中，Cache块大小为一个字，其CPI为:

$$2 + (1.2 \times 3\% \times (4+24+4)) = 3.15$$

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：

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

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%

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：

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

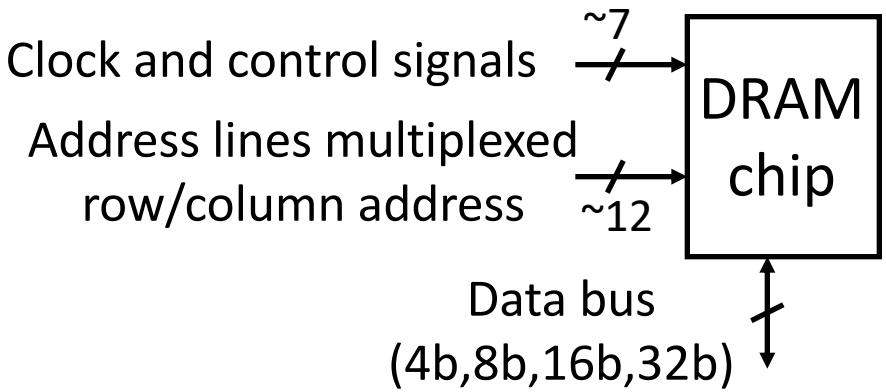
性能提高了25%

64位总线和存储器，不采用多体交叉：

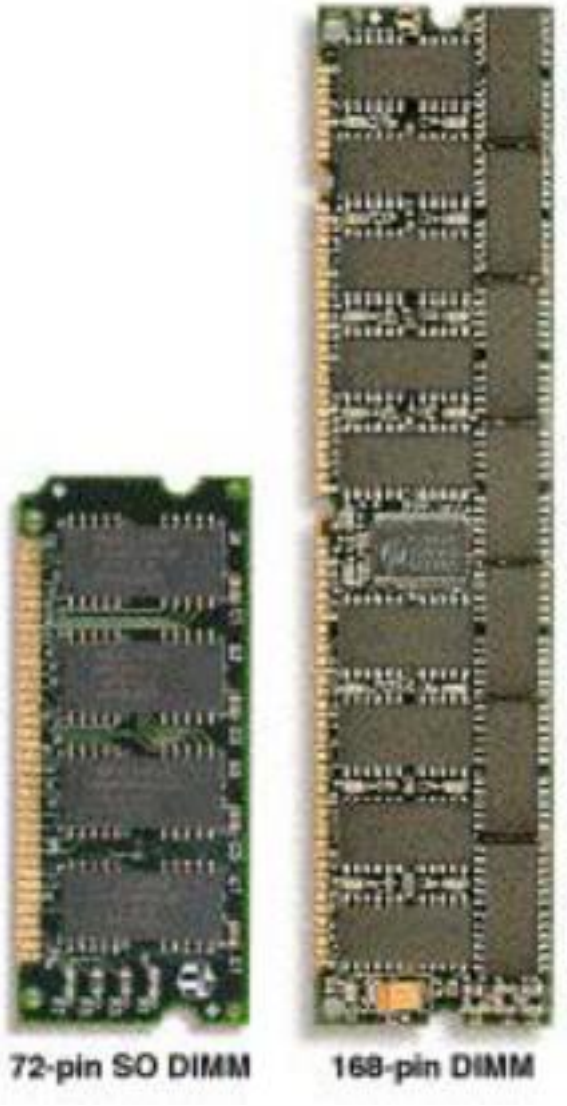
$2 + (1.2 \times 1\% \times 2 \times 32) = 2.77$

性能提高了14%

DRAM Packaging (Laptops/Desktops/Servers)



- **DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)**
- **Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)**



- **存储器组织**
  - 存储系统结构

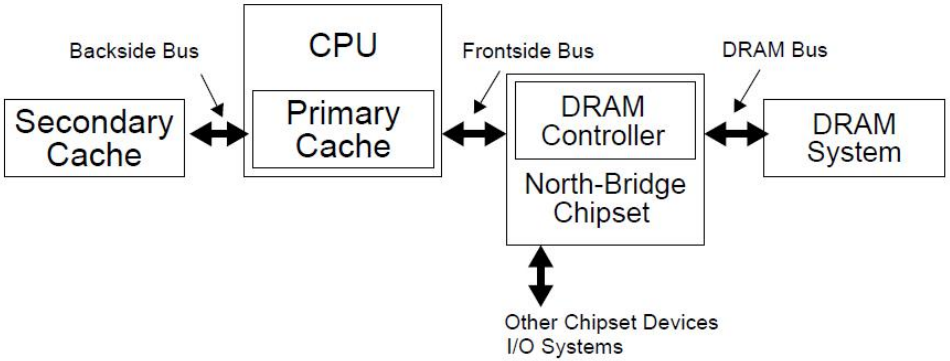


Figure 4.1: Memory System Architecture

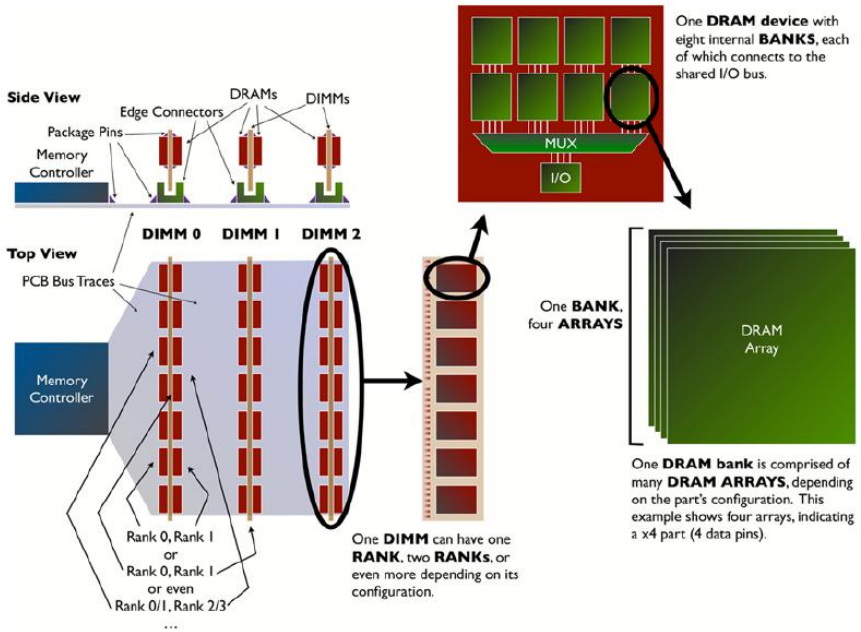


Figure 1.5: DIMMs, ranks, banks, and arrays. A system has potentially many DIMMs, each of which may contain one or more ranks. Each rank is a set of ganged DRAM devices, each of which has potentially many banks. Each bank has potentially many constituent arrays, depending on the parts data width.

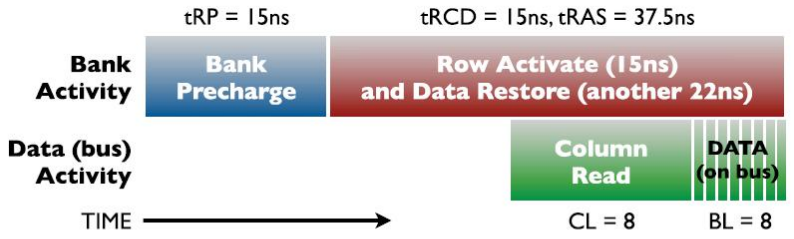


Figure 1.7: The costs of accessing DRAM. Before a DRAM cell can be read or written, the device must first be placed into a state that allows read/write access to the cells. This is a two step process: the **precharge** command initializes the banks sense amplifiers (sets the voltage on the bitlines appropriately), and the **activate** command reads the row data from the specified storage capacitors into the sense amplifiers. Once the sense amplifiers have read the data, it can be read/written by the memory controller. Precharge and activate together take several tens of nanoseconds; reading or writing takes roughly a nanosecond. Timing values taken from a Micron 2Gb DDR3-1066 part.

参考文献

Davis, B. T. (2001). Modern dram architectures, University of Michigan: 221.

Jacob, B. (2009). The Memory System, Morgan & Claypool.

3D-Stacked DRAM and Processing in Memory (PIM)

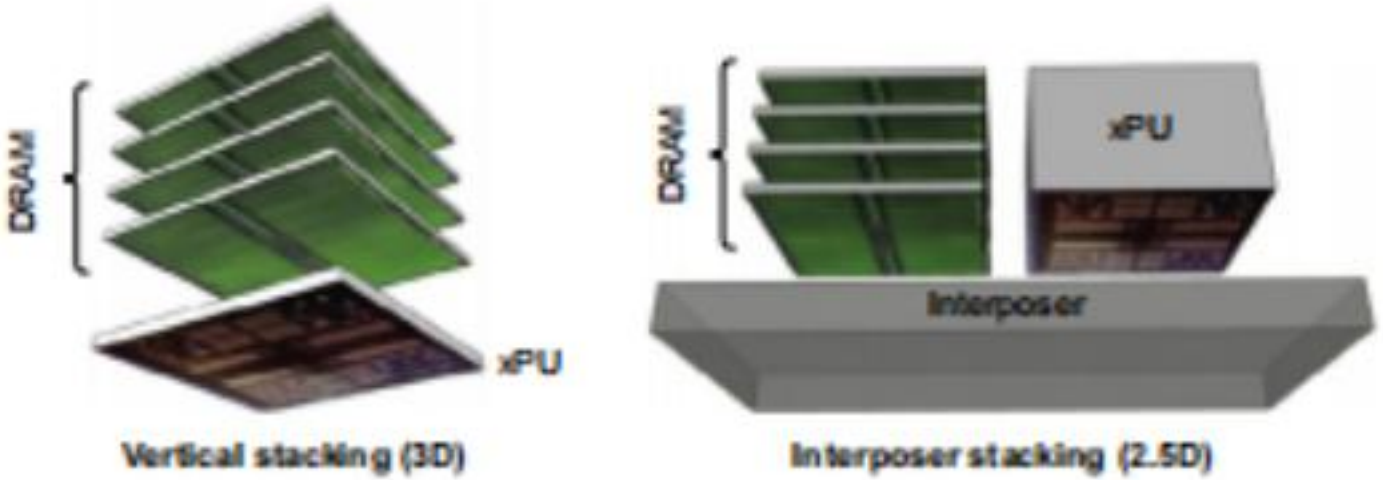


Figure 2.7 Two forms of die stacking. The 2.5D form is available now. 3D stacking is under development and faces heat management challenges due to the CPU.

Vertical stacking (3D)

Interposer stacking (2.5D)

4.5 虚拟存储

虚拟存储回顾

TLB

虚拟存储器 - 基本原理

- **允许应用程序的大小，超过主存容量。目的是提高存储系统的容量**
- **帮助OS进行多进程管理**
  - 每个进程可以有自己的地址空间
  - 提供多个进程空间的保护
  - 可以将多个逻辑块映射到共享的物理存储器上
  - 静态重定位和动态重定位
    - 应用程序运行在虚地址空间
    - 虚实地址转换对用户是透明的
- **虚拟存储管理的是主存 - 辅助存储器这个层面上**
  - 失效：页失效或地址失效
  - 块：页或段

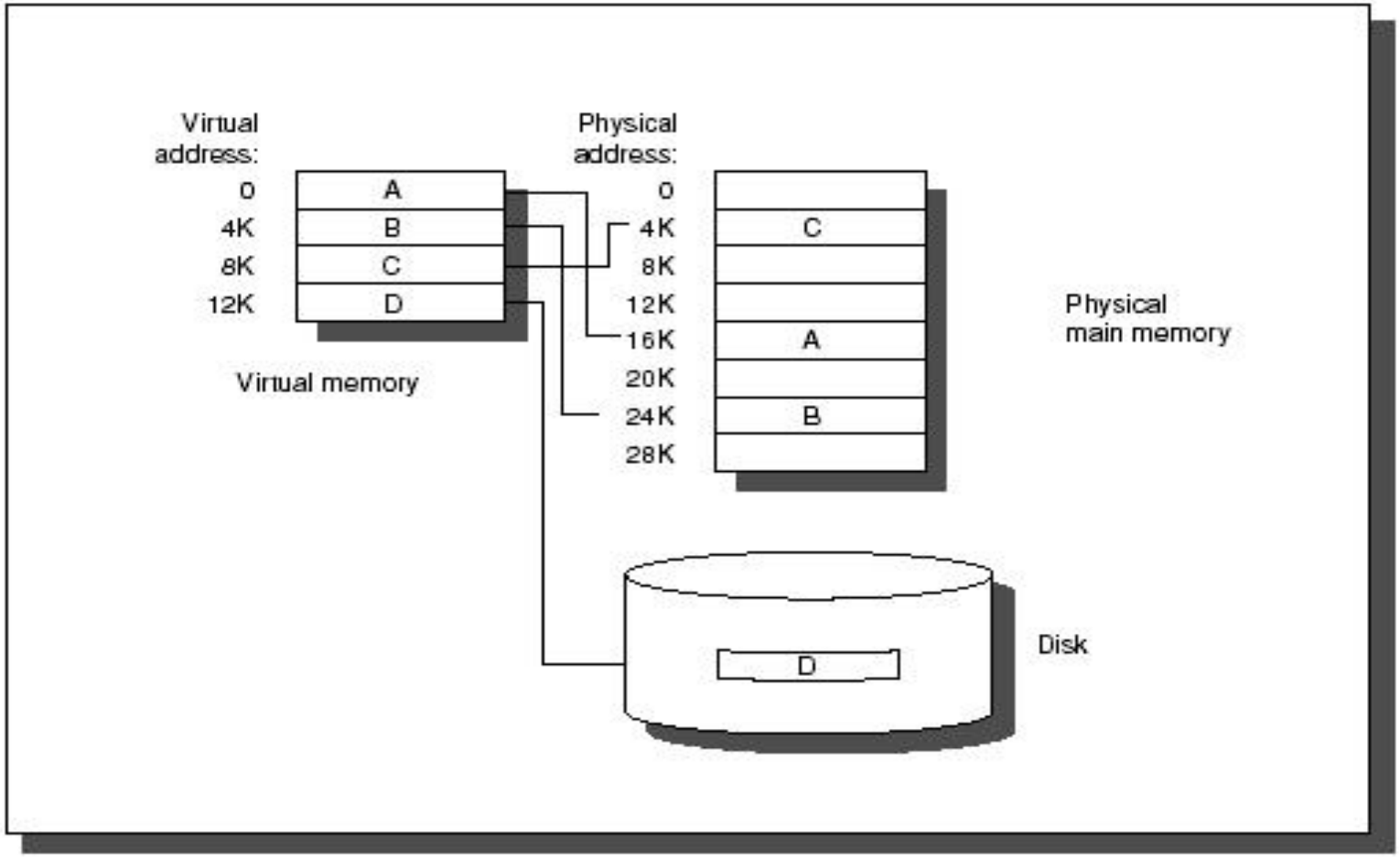


FIGURE 5.31 The logical program in its contiguous virtual address space is shown on the left. It consists of four pages A, B, C, and D. The actual location of three of the blocks is in physical main memory and the other is located on the disk.

Cache与VM的区别

- **目的不同**
  - Cache是为了提高访存速度
  - VM是为了提高存储容量
- **替换的控制者不同**
  - Cache失效由硬件处理
  - VM的页失效通常由OS处理
    - 一般页失效开销很大，因此替换算法非常重要
- **地址空间**
  - VM空间由CPU的地址尺寸确定
  - Cache的大小与CPU地址尺寸无关
- **下一级存储器**
  - Cache下一级是主存
  - VM下一级是磁盘，大多数磁盘含有文件系统，文件系统寻址与主存不同，它通常在I/O空间中，VM的下一级通常称为SWAP空间

虚拟存储器页式管理的典型参数与Cache的比较

Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–3 clock cycles	100–200 clock cycles
Miss penalty	8–200 clock cycles	1,000,000–10,000,000 clock cycles
(access time)	(6–160 clock cycles)	(800,000–8,000,000 clock cycles)
(transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.1–10%	0.00001–0.001%
Address mapping	25–45 bit physical address to 14–20 bit cache address	32–64 bit virtual address to 25–45 bit physical address

Figure C.19 Typical ranges of parameters for caches and virtual memory. Virtual memory parameters represent increases of 10–1,000,000 times over cache parameters. Normally first-level caches contain at most 1 MB of data, while physical memory contains 256 MB to 1 TB.

- 从表中看 （与Cache参数相比）
  - **除了失效率较低，其他参数都比Cache大**

页式管理和段式管理

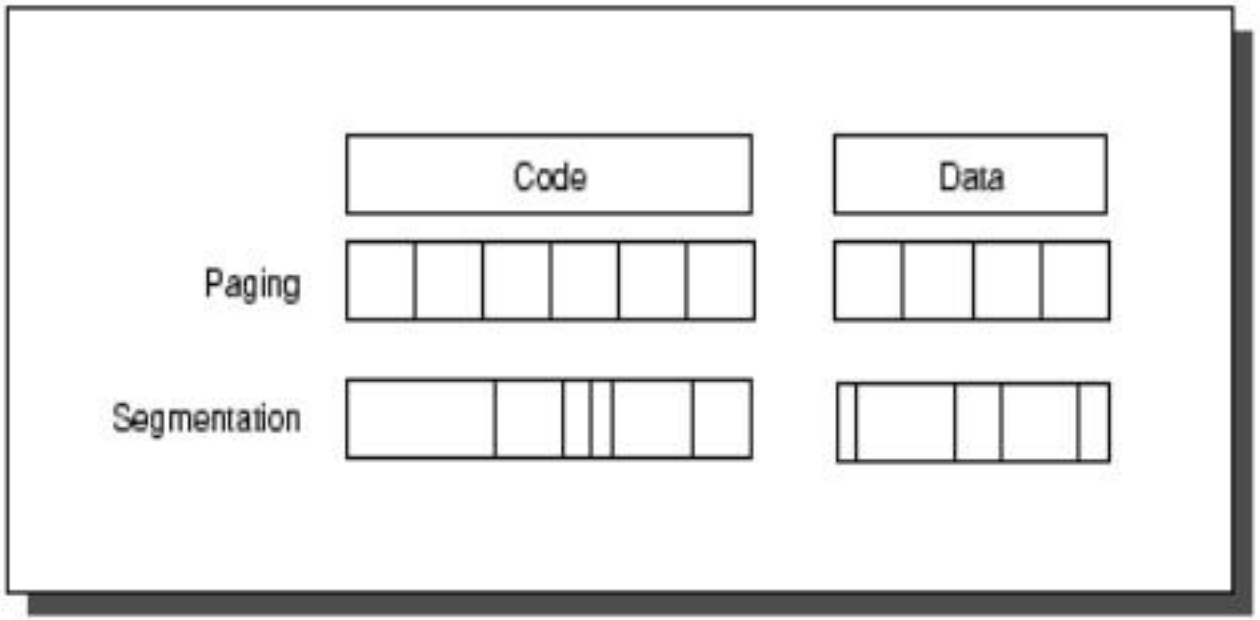


FIGURE 5.33 Example of how paging and segmentation divide a program.

页式管理和段式管理

Aspect	Page	Segment
Words/Address	One - contains page and offset	Two - possible large max-size hence need Seg and offset address words
Programmer visible	No	Sometimes yes
Replacement	Trivial - due to fixed size	Hard - need to find contiguous space ==> GC necessary or wasted memory
Memory Inefficiency	Internal fragmentation - wasted part of a page	External fragmentation - due to variable size blocks
Disk Efficiency	Yes - adjust page size to balance access and transfer time	Not always - segment size varies

- VM可分为两类：页式和段式
  - 页式：每页大小固定
  - 段式：每段大小不等
  - 两者区别：

VM的四个问题 (1/2)

- **映象规则**
  - 选择策略：低失效率（复杂映象算法） vs. 高失效率（简单映射方法），**由于失效开销很大，一般选择低失效率方法，即全相联映射**
- **查找算法 - 用附加数据结构**
  - 固定页大小 - 用页表
    - VPN - > PPN
    - Tag标识该页是否在主存
  - 可变长段 - 段表
    - 段表中存放所有可能的段信息
    - 段号 - >段基址 再加段内偏移量
    - 可能存在许多小尺寸段
  - 页表
    - 页表中所含项数：一般为虚页的数量
    - 功能: VPN - >PPN，方便页重新分配，有一位标识该页是否在内存

VM的四个问题（2/2)

- **替换规则**
  - LRU是最好的
  - 但真正的LRU方法，硬件代价较大
  - 用硬件简化，通过OS来完成
    - 为了帮助OS寻找LRU页，每个页面设置一个 use bit
    - 当访问主存中一个页面时，其use bit置位
    - OS定期复位所有使用位，这样每次复位之前，使用位的值反映了从上次复位到现在的这段时间中，哪些页曾被访问过。
    - 当有失效冲突时，由OS来决定哪些页将被换出去。
- **写策略**
  - 总是用写回法，因为访问硬盘速度很慢。

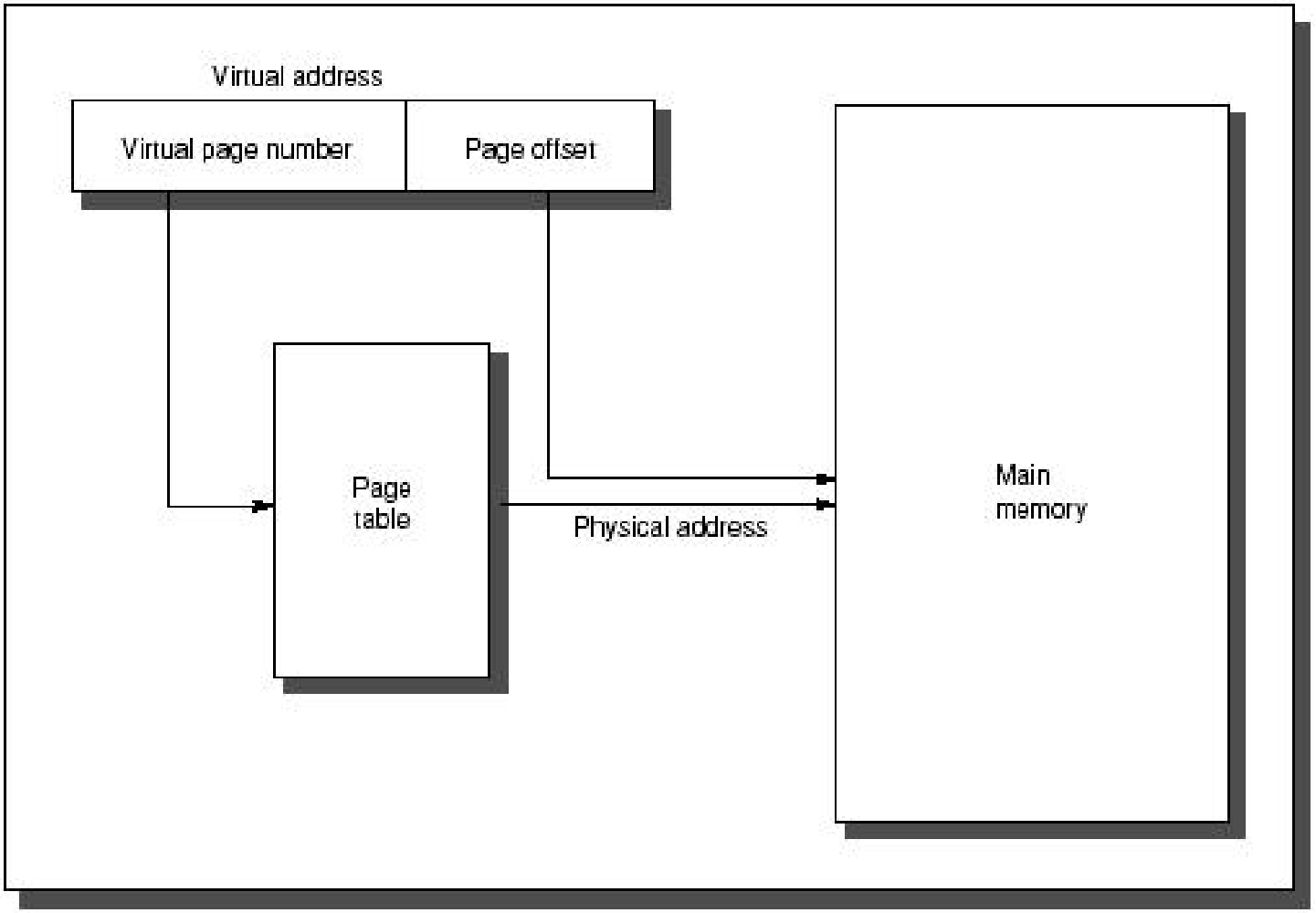
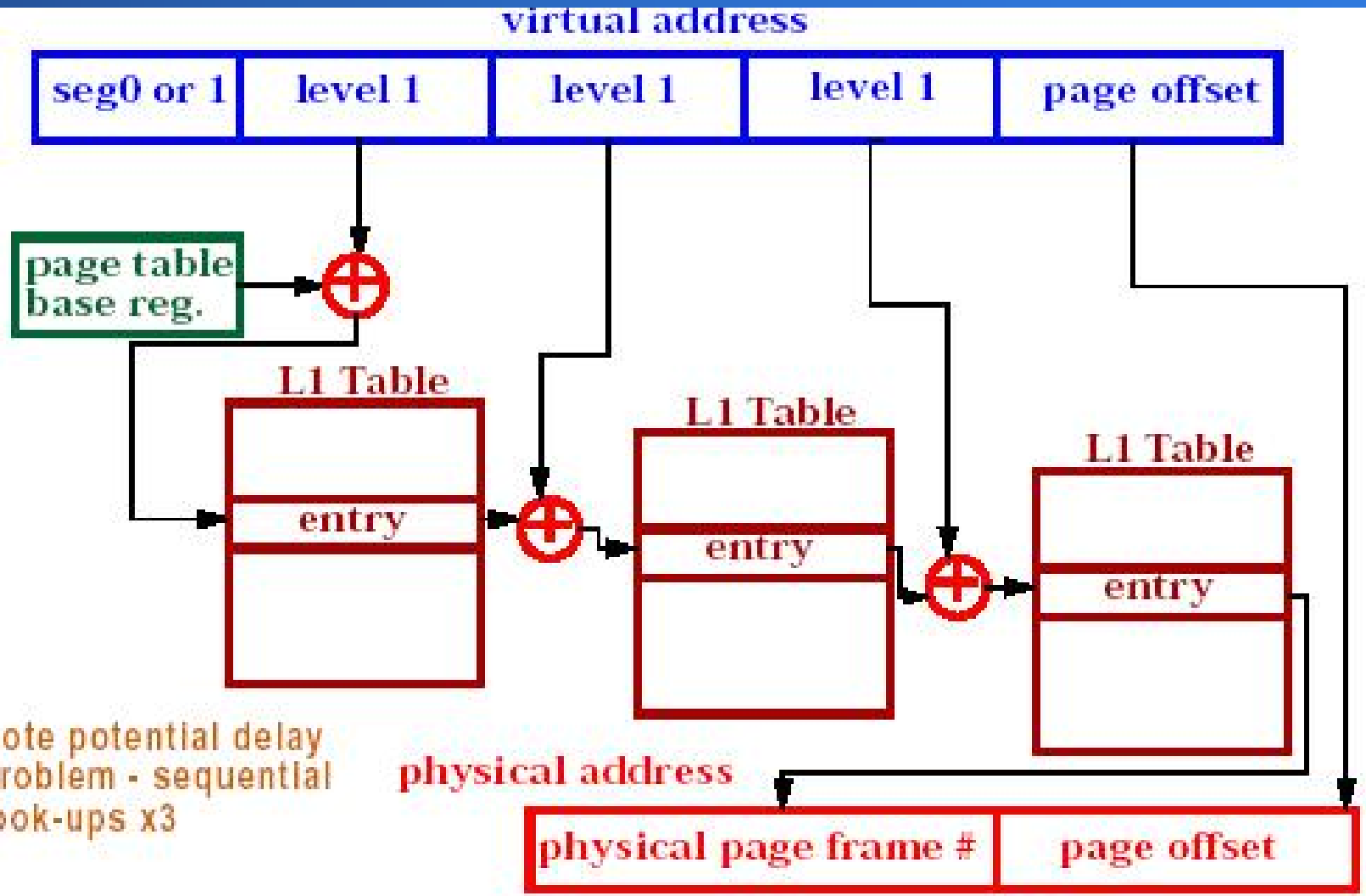


FIGURE 5.35 The mapping of a virtual address to a physical address via a page table.

页面大小的选择

- **页面选择较大的优点**
  - 减少了页表的大小
  - 如果局部性较好，可以提高命中率
- **页面选择较大的缺点**
  - 内存中的碎片较多，内存利用率低
  - 进程启动时间长
  - 失效开销加大

Alpha VPN - >PPN



4.5 虚拟存储

虚拟存储回顾

TLB

TLB (Translation look-aside Buffer)

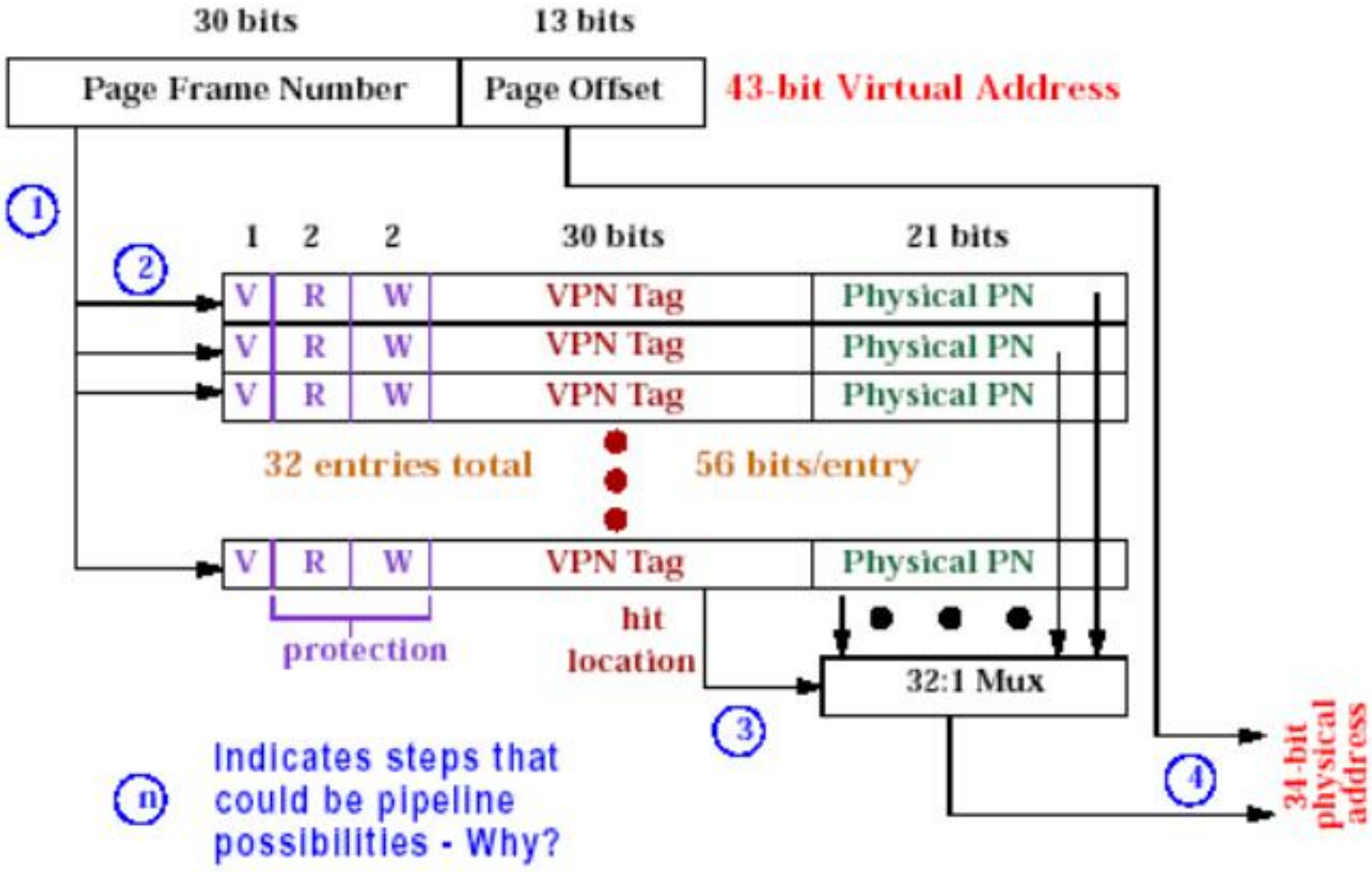
- **页表一般很大，存放在主存中。**
  - 导致每次访存可能要两次访问主存，一次读取页表项，一次读写数据
  - 解决办法：采用 TLB
- **TLB**
  - 存放近期经常使用的页表项，是整个页表的部分内容的副本。
  - 基本信息：

    VPN##PPN##Protection Field##use bit ## dirty bit
  - OS修改页表项时，需要刷新TLB，或保证TLB中没有该页表项的副本
  - TLB必须在片内
    - 速度至关重要
    - TLB过小，意义不大
    - TLB过大，代价较高
    - 相联度较高（容量小）

TLB的典型参数

- **block size - same as a page table entry - 1 or 2 words**
- **hit time - 1 cycle**
- **miss penalty - 10 to 30 cycles**
- **miss rate - .1% to 2%**
- **TLB size - 32 B to 8 KB**

举例：Alpha 21064的TLB



Summary of Virtual Memory and Caches

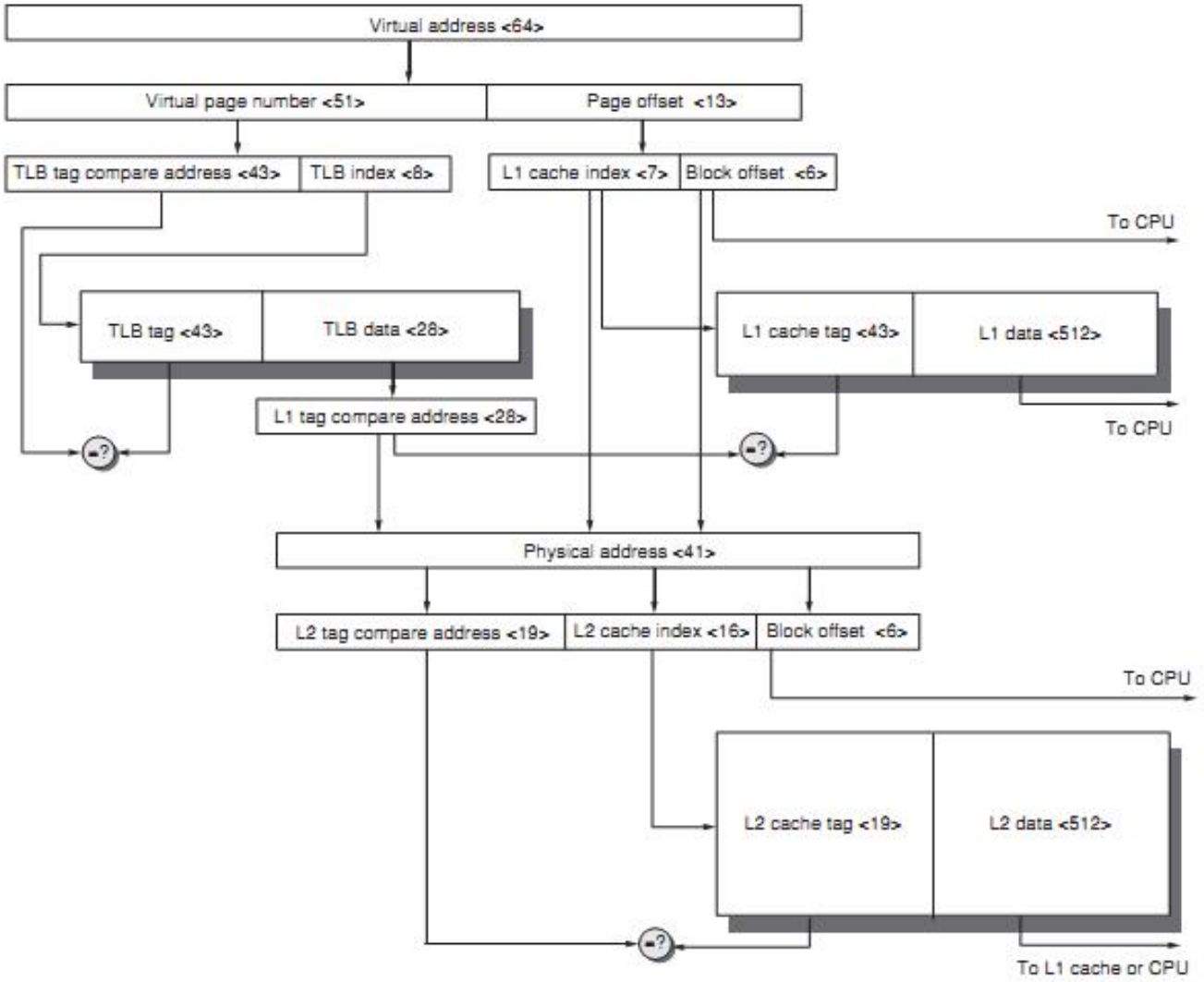


Figure C.24 The overall picture of a hypothetical memory hierarchy going from virtual address to L2 cache access. The page size is 8 KB. The TLB is direct mapped with 256 entries. The L1 cache is a direct-mapped 8 KB, and the L2 cache is a direct-mapped 4 MB. Both use 64-byte blocks. The virtual address is 64 bits and the physical address is 41 bits. The primary difference between this simple figure and a real cache is replication of pieces of this figure.

Acknowledgements

- **These slides contain material developed and copyright by:**
  - John Kubiatowicz (UCB)
  - Krste Asanovic (UCB)
  - John Hennessy (Standford)and David Patterson (UCB)
  - Chenxi Zhang (Tongji)
  - Muhamed Mudawar (KFUPM)
- **UCB material derived from course CS152、CS252、CS61C**
- **KFUPM material derived from course COE501、COE502**