存储层次

100076202： 计算机系统导论

任课教师：

计卫星　　宿红毅　　张艳

原作者：

Randal E. **Bryant and** David R. O’Hallaron

提纲

- 存储技术与趋势/Storage technologies and trends
- 访存局部性原理/Locality of reference
- 存储层次中的缓存/Caching in the memory hierarchy

随机访问内存/Random-Access Memory (RAM)

- **主要特点/Key features**
  - RAM通常封装为一个芯片/RAM is traditionally packaged as a chip.
  - 基本存储单元是一个cell（每个cell是一个bit）Basic storage unit is normally a cell (one bit per cell).
  - 多个RAM芯片构成一个内存/Multiple RAM chips form a memory.

- **RAM有两种类型/RAM comes in two varieties:**
  - SRAM 静态RAM/SRAM (Static RAM)
  - DRAM 动态RAM/DRAM (Dynamic RAM)

SRAM vs DRAM

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

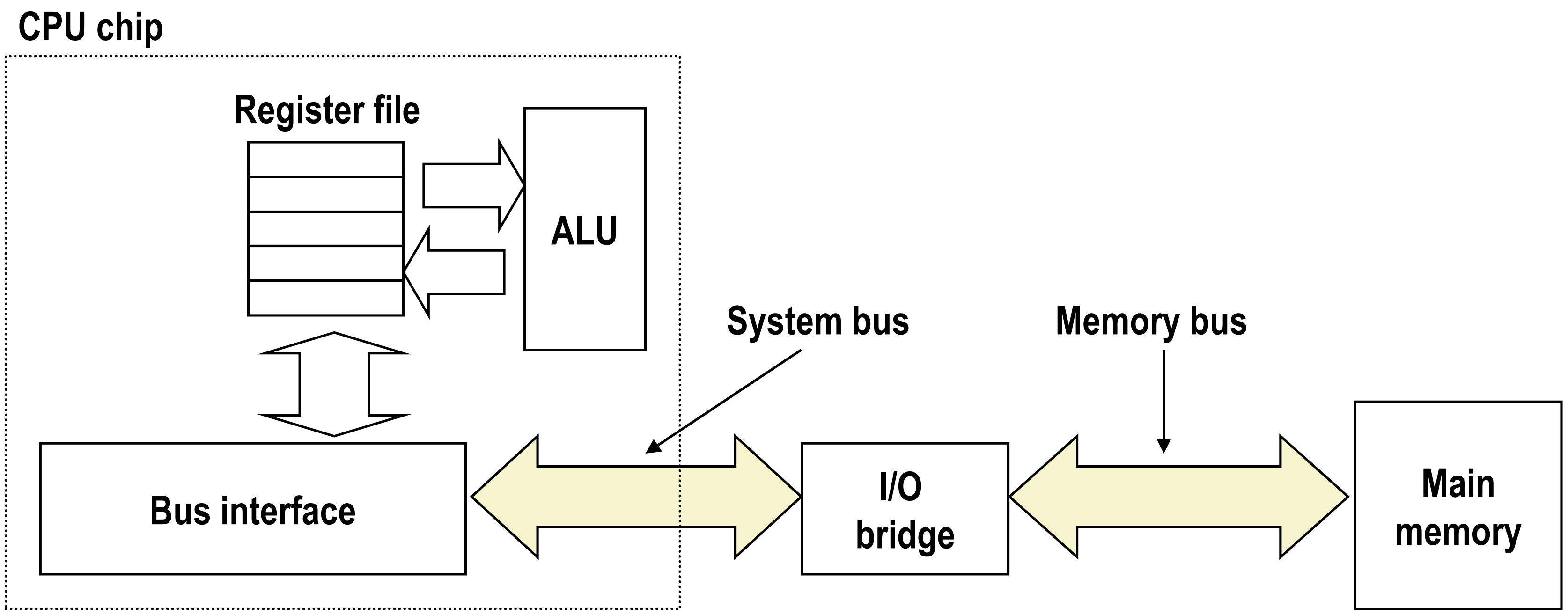
EDC： error detection and correction

非易失性内存/Nonvolatile Memories

- **DRAM和SRAM都是易失性存储/DRAM and SRAM are volatile memories**
  - 掉电信息丢失/Lose information if powered off.
- **非易失性内存掉电仍保留数据/Nonvolatile memories retain value even if powered off**
  - 只读内存/Read-only memory (ROM): programmed during production
  - 可编程ROM：只能编程一次/Programmable ROM (PROM): can be programmed once
  - 可擦除ROM：块擦除/Eraseable PROM (EPROM): can be bulk erased (UV, X-Ray)
  - 电可擦除ROM/Electrically eraseable PROM (EEPROM): electronic erase capability
  - Flash存储：EEPROM，部分（块级别）擦除能力/Flash memory: EEPROMs. with partial (block-level) erase capability
    - 10万次擦写上限/Wears out after about 100,000 erasings
- **非易失性存储的使用/Uses for Nonvolatile Memories**
  - 固件存储在ROM中(BIOS、硬盘控制器、网卡、显卡、安全子系统)/Firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,…)
  - 固态硬盘（替代智能手机、MP3播放器、平板、笔记本中的机械硬盘）/Solid state disks (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops,…)
  - 磁盘Cache/Disk caches

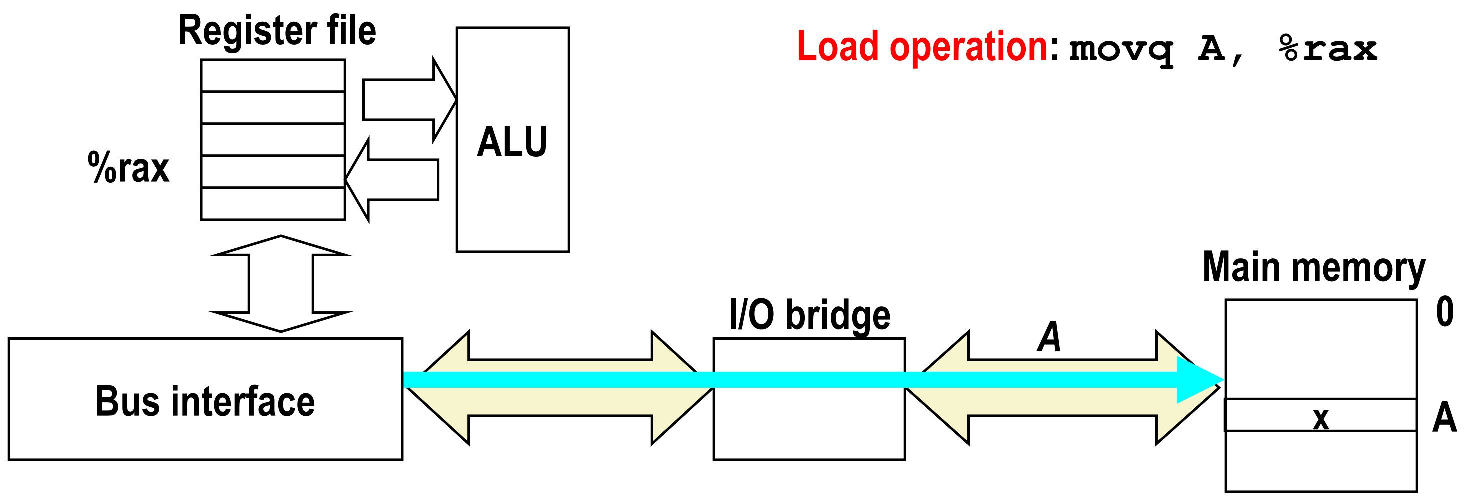
传统CPU和内存之间的互连结构/Traditional Bus Structure Connecting CPU and Memory

- 总线是一组并行的用于传输地址、数据和控制信号的导线/A bus is a collection of parallel wires that carry address, data, and control signals.
- 总线通常是多个设备共享的/Buses are typically shared by multiple devices.



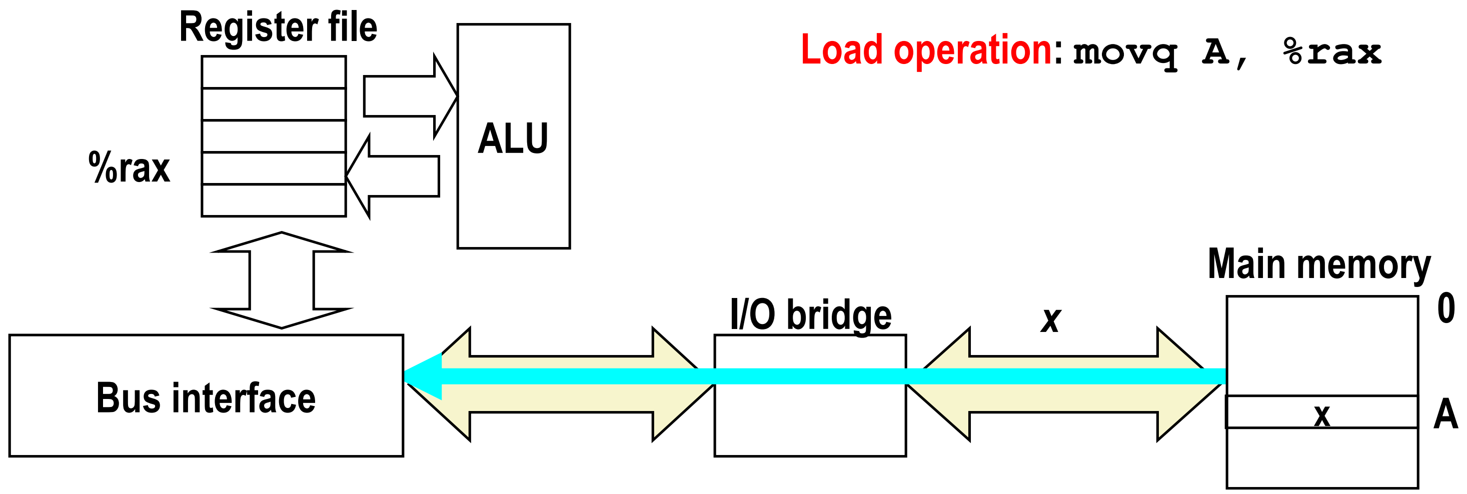
内存读事务/Memory Read Transaction (1)

- CPU将地址A放到内存总线上/CPU places address A on the memory bus.



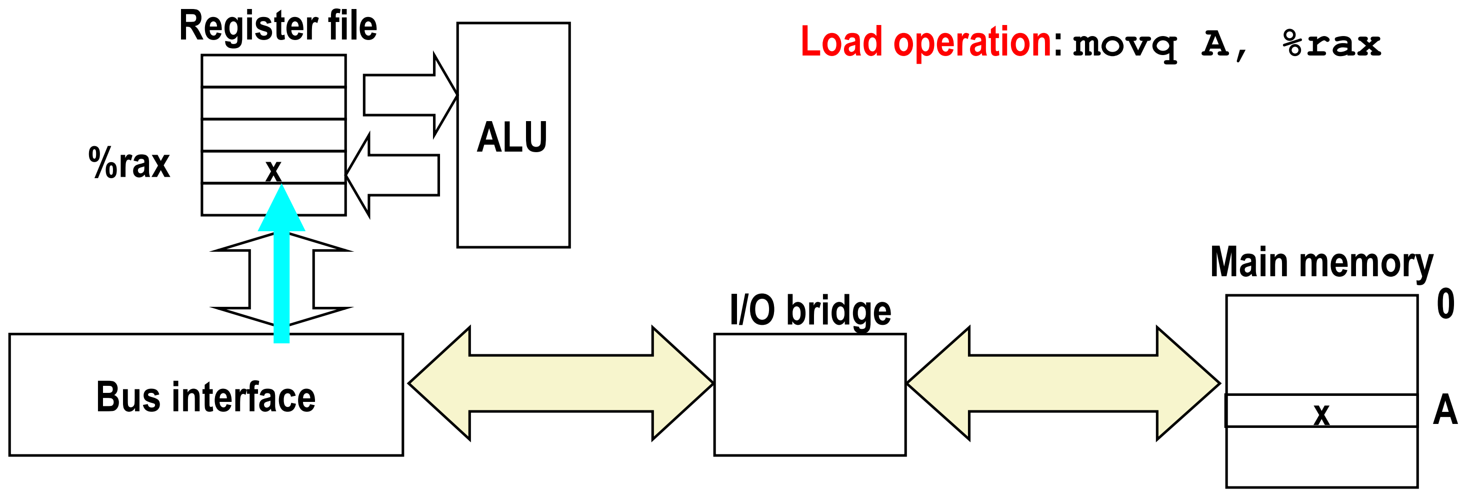
内存读事务/ Memory Read Transaction (2)

- **主存从内存总线获得地址A，读取对应的字x，并将其放置到总线上/Main memory reads A from the memory bus, retrieves word x, and places it on the bus.**



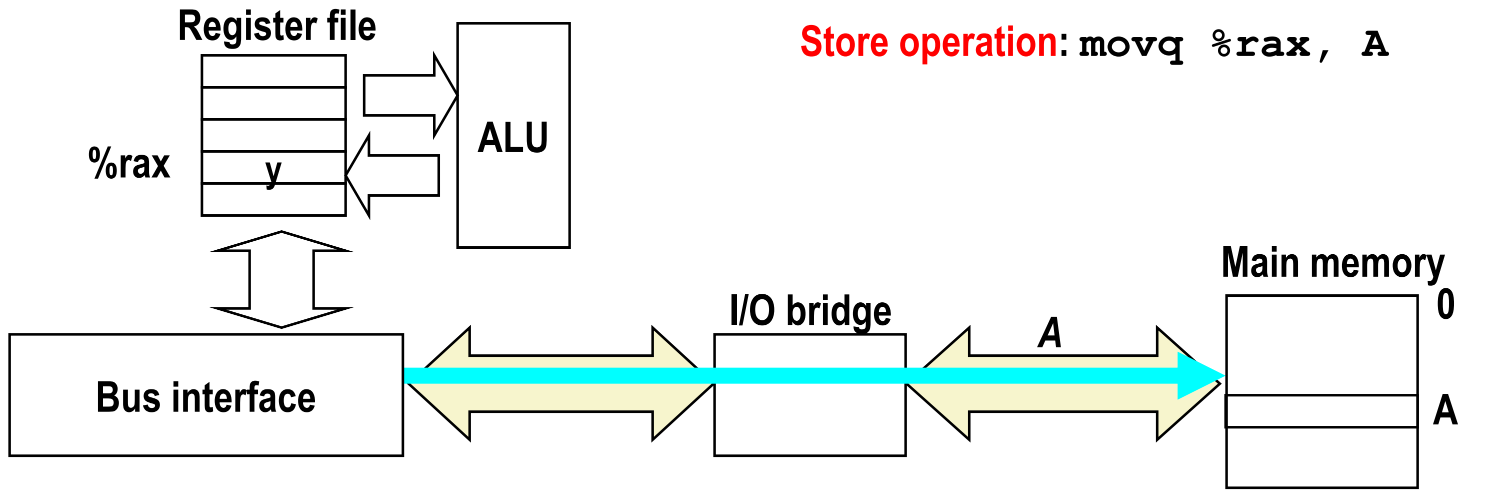
内存读事务/ Memory Read Transaction (3)

- CPU从总线读取x并拷贝到寄存器%rax中/CPU read word x from the bus and copies it into register %rax.



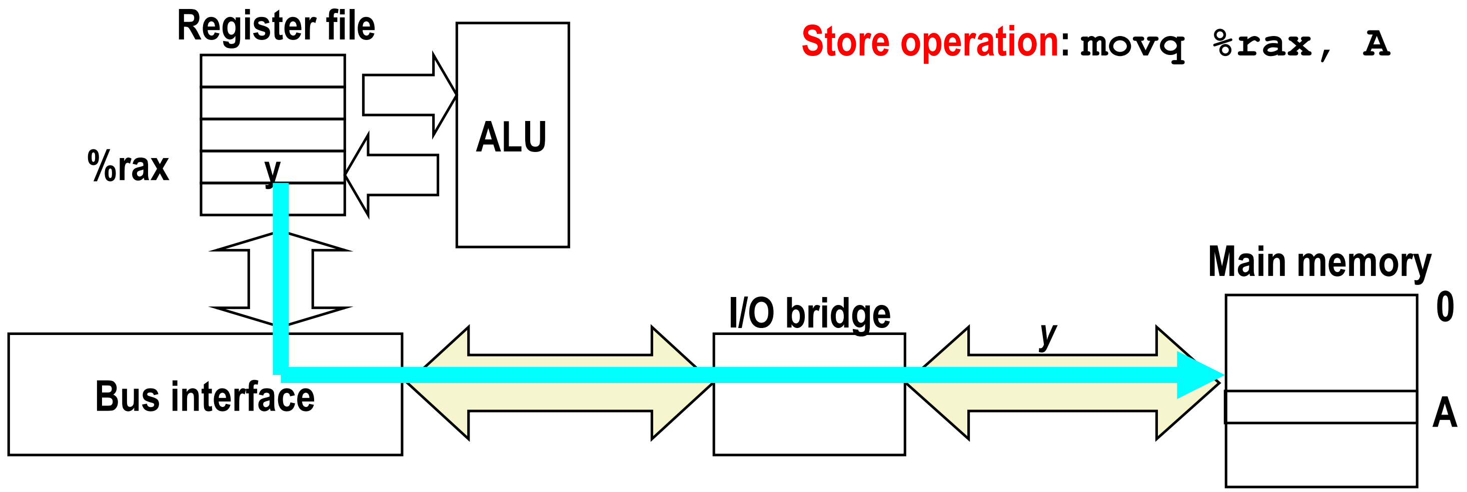
内存写事务/ Memory Write Transaction (1)

- CPU将地址A放到总线上，主存读取地址并等待数据到来/CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.



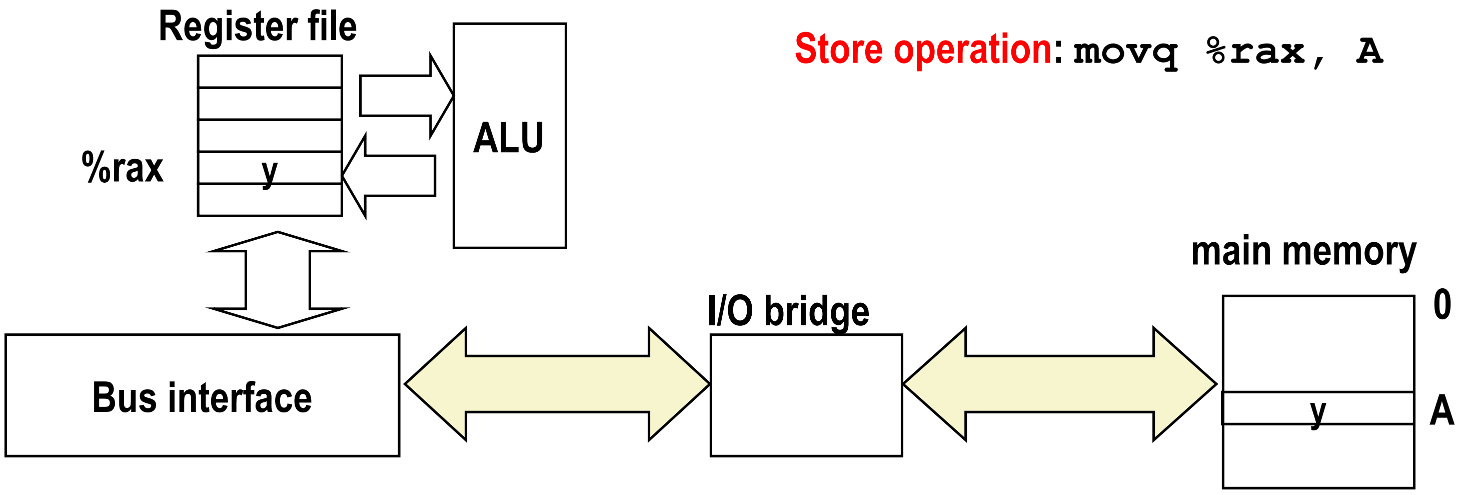
内存写事务/ Memory Write Transaction (2)

- CPU将数据字y放到总线上/CPU places data word y on the bus.



内存写事务/ Memory Write Transaction (3)

- 主存从总线读取数据字y并将其写入地址A/Main memory reads data word y from the bus and stores it at address A.



硬盘驱动内部是什么？ What's Inside A Disk Drive?

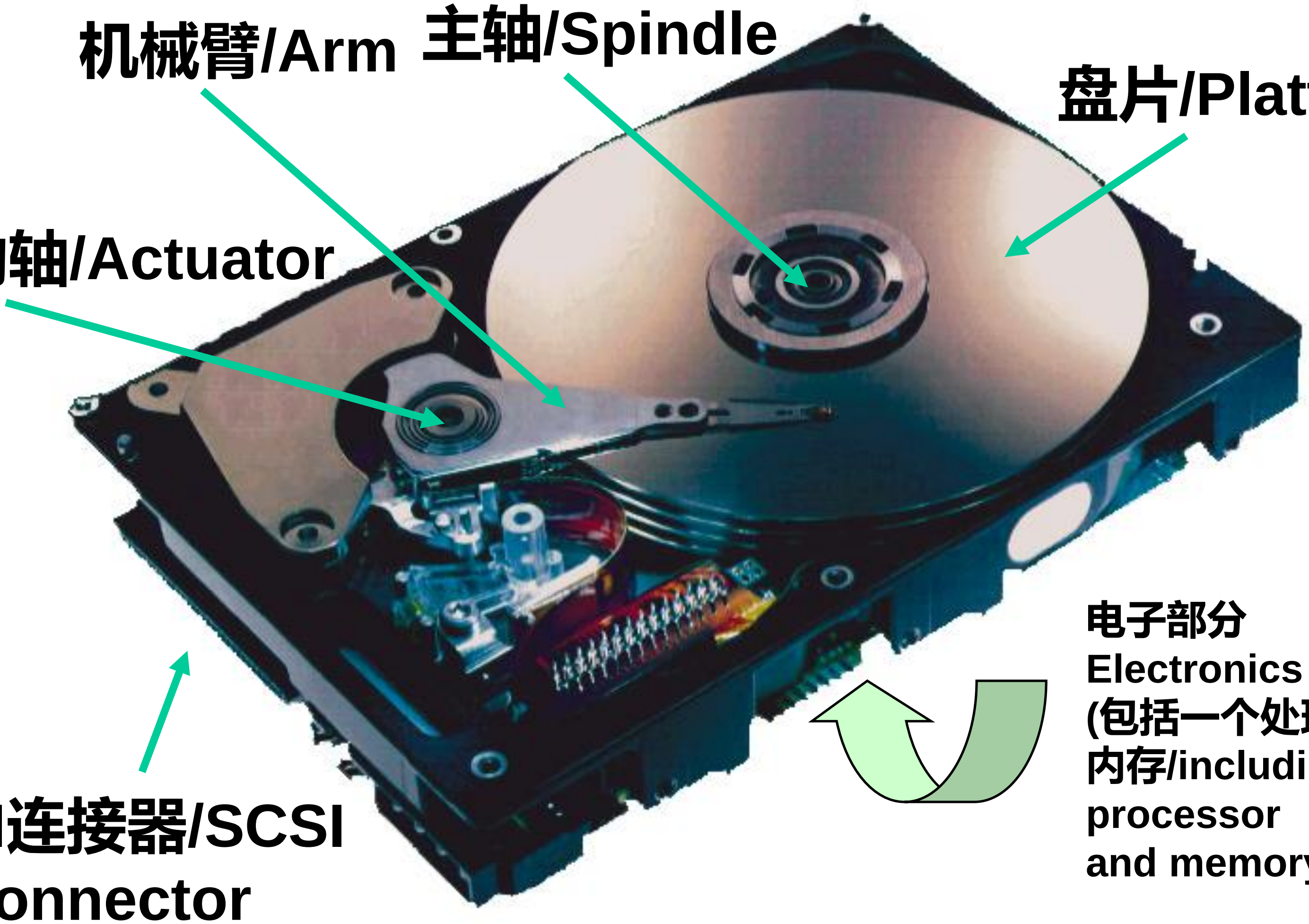
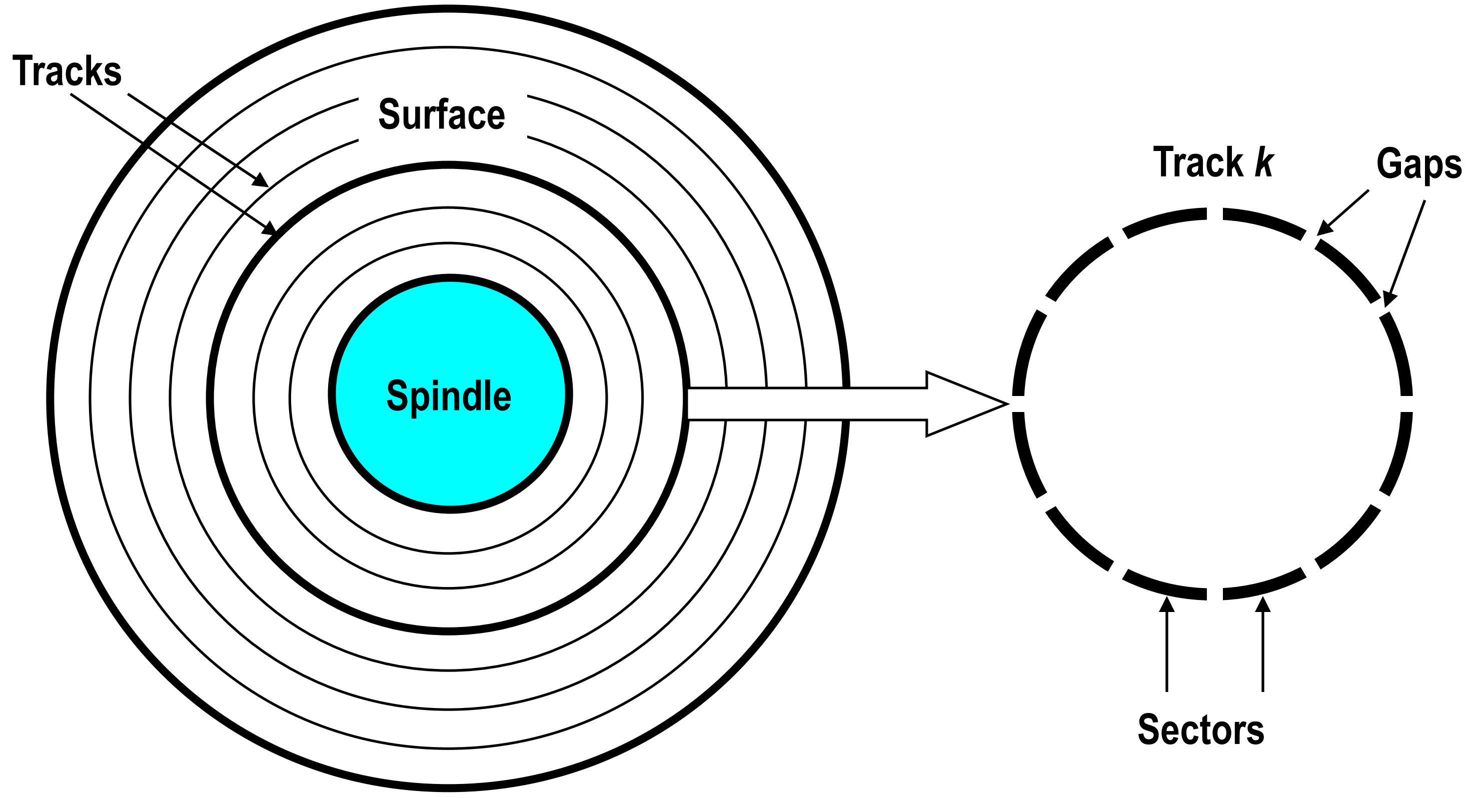

机械臂/Arm
主轴/Spindle
盘片/Platters
传动轴/Actuator
SCSI连接器/SCSI connector
电子部分 Electronics (包括一个处理器和内存/including a processor and memory!)

Image courtesy of Seagate Technology

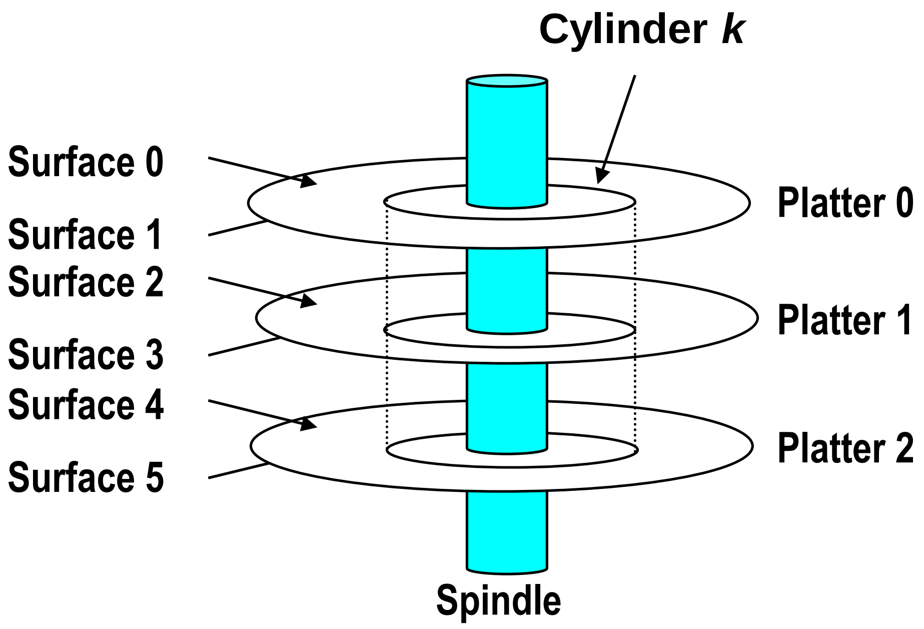
磁盘结构/Disk Geometry

- 磁盘由盘片构成，每个盘片有两面/Disks consist of platters, each with two surfaces.
- 每个盘面由多个同心圆的磁道构成/Each surface consists of concentric rings called tracks.
- 每个磁道由多个空白分隔的扇区构成Each track consists of sectors separated by gaps.



磁盘结构/ Disk Geometry (Muliple-Platter View)

- **对齐的磁道形成一个圆柱体/Aligned tracks form a cylinder.**

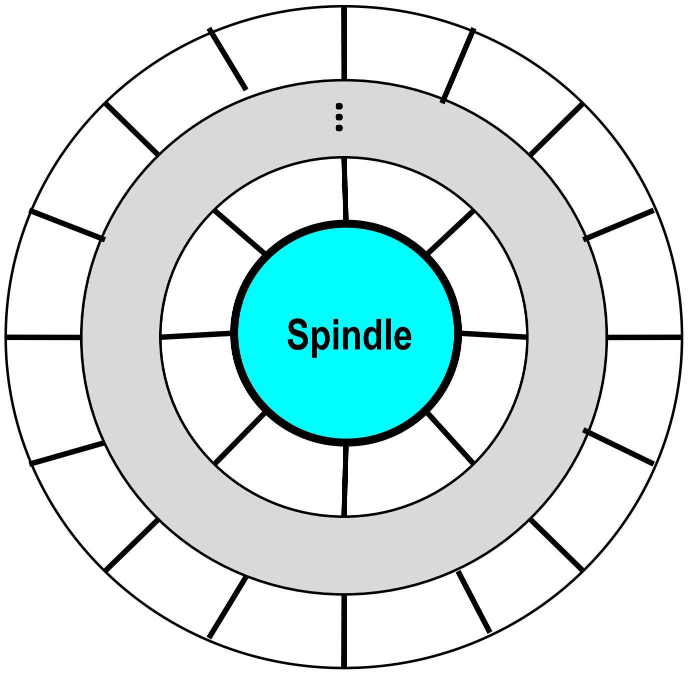


磁盘容量/Disk Capacity

- **容量：可以存储的最大bit数量/Capacity: maximum number of bits that can be stored.**
  - 供应商通常以GB为单位说明磁盘容量/Vendors express capacity in units of gigabytes (GB)，where 1 GB = 10^9 Bytes.
- **容量由以下因素决定/Capacity is determined by these technology factors:**
  - 刻录密度/Recording density (bits/in):可压缩到1英寸磁道段中的比特数/number of bits that can be squeezed into a 1 inch segment of a track.
  - 磁道密度/Track density (tracks/in):可挤压成1英寸径向段的轨道数/number of tracks that can be squeezed into a 1 inch radial segment.
  - 单位面积密度/Areal density (bits/in2):记录和磁道密度的乘积/product of recording and track density.

记录区域/Recording zones

- **现在的磁盘将磁道划分为多个区域 /Modern disks partition tracks into disjoint subsets called recording zones**
  - 区域中的每个磁道具有相同的扇区数，由最内磁道的圆周决定/Each track in a zone has the same number of sectors, determined by the circumference of innermost track.
  - 每个区域有不同数量的扇区/磁道，外部区域比内部区域有更多的扇区或磁道 /Each zone has a different number of sectors/track, outer zones have more sectors/track than inner zones.
  - 因此，我们在计算容量时使用平均扇区数/磁道数/So we use **average** number of sectors/track when computing capacity.



计算磁盘容量/Computing Disk Capacity

Capacity = (# bytes/sector) x (avg. # sectors/track) x
(# tracks/surface) x (# surfaces/platter) x
(# platters/disk)

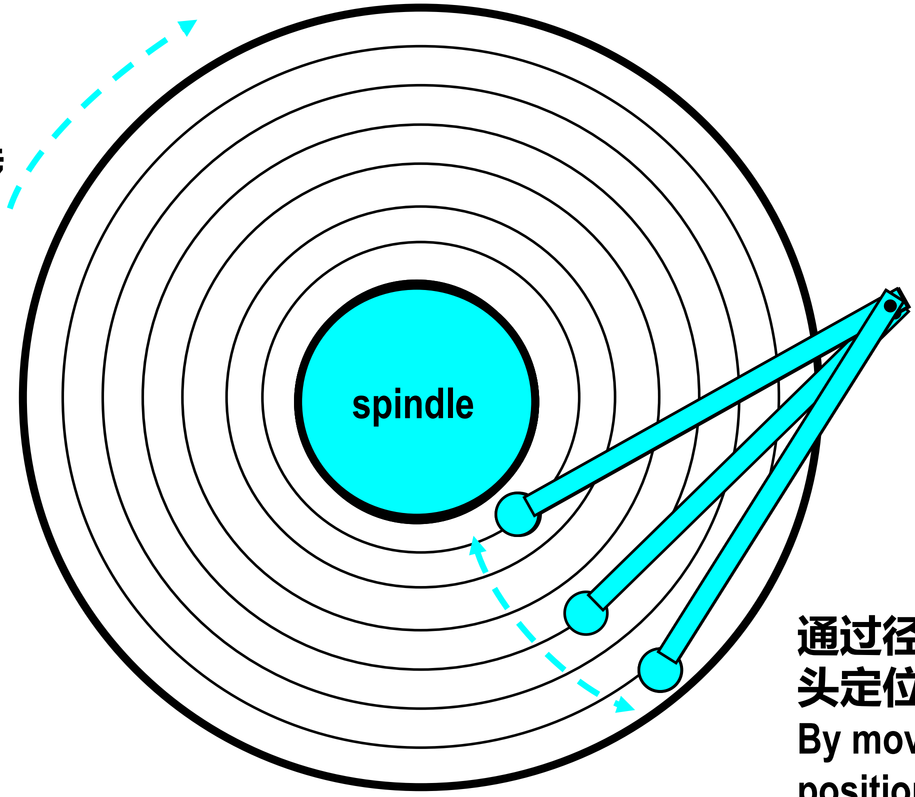
Example:

- 512 bytes/sector
- 300 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

Capacity = 512 x 300 x 20000 x 2 x 5
= 30,720,000,000
= 30.72 GB

磁盘操作/Disk Operation (Single-Platter View)

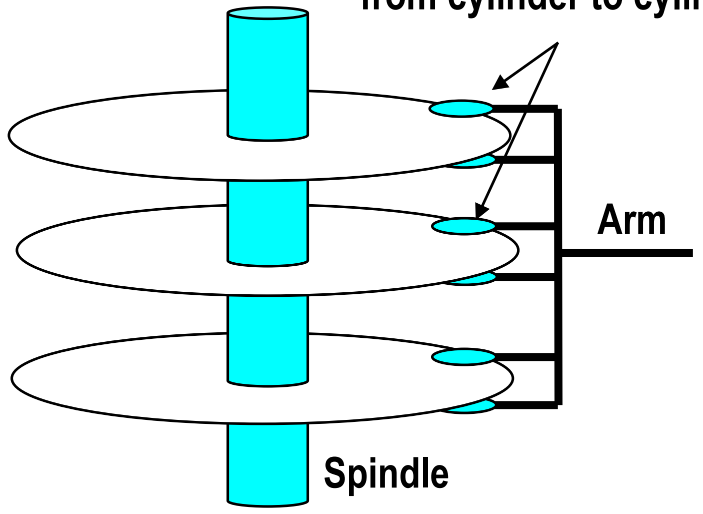
The disk surface
spins at a fixed
rotational rate
磁盘表面
以固定速度旋转



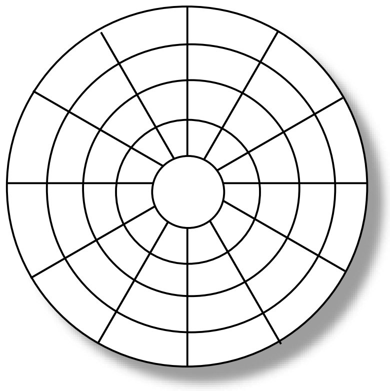
读/写头连接在臂的末端
，并在薄气垫上飞过磁盘
表面/The read/write *head*
is attached to the end
of the *arm* and flies over
the disk surface on
a thin cushion of air.

通过径向移动，臂可以将读/写
头定位在任何磁道上。
By moving radially, the arm can
position the read/write head over
any track.

磁盘操作/Disk Operation (Multi-Platter View)


读/写磁头在圆柱间一致移动
Read/write heads move in unison from cylinder to cylinder
Arm
Spindle

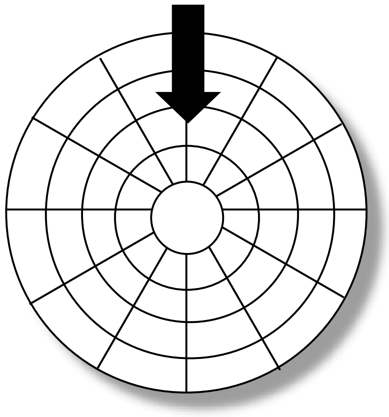
磁盘结构/Disk Structure - top view of single platter



表面按照磁道划分/Surface organized into tracks

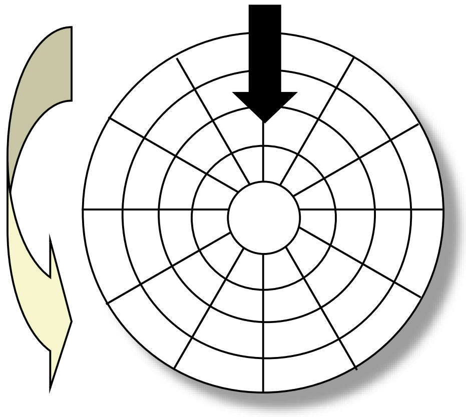
每个磁道划分为多个扇区/Tracks divided into sectors

磁盘访问/Disk Access



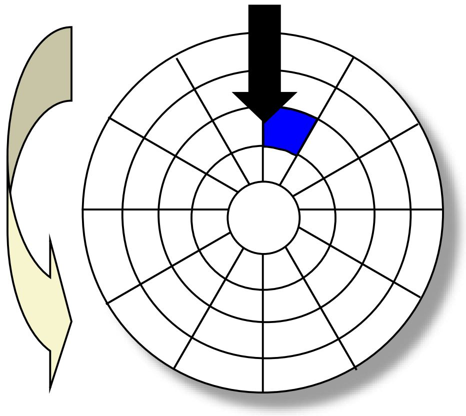
磁头在某个磁道上面/Head in position above a track

磁盘访问/Disk Access



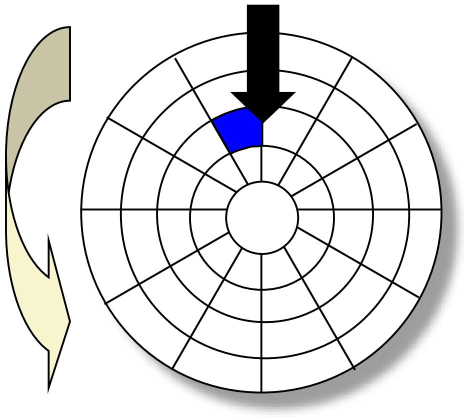
沿着逆时针方向旋转/Rotation is counter-clockwise

磁盘访问-读/Disk Access – Read



即将开始读蓝色的扇区/About to read blue sector

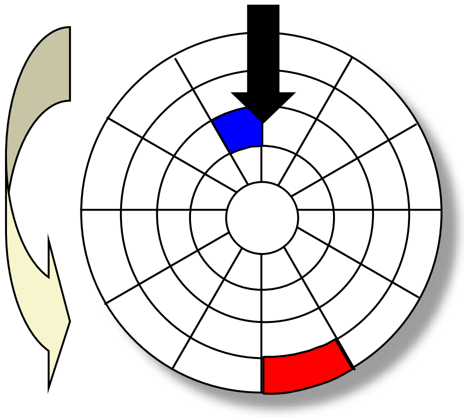
磁盘访问-读/ Disk Access – Read



After BLUE read

读写蓝色的扇区后/After reading blue sector

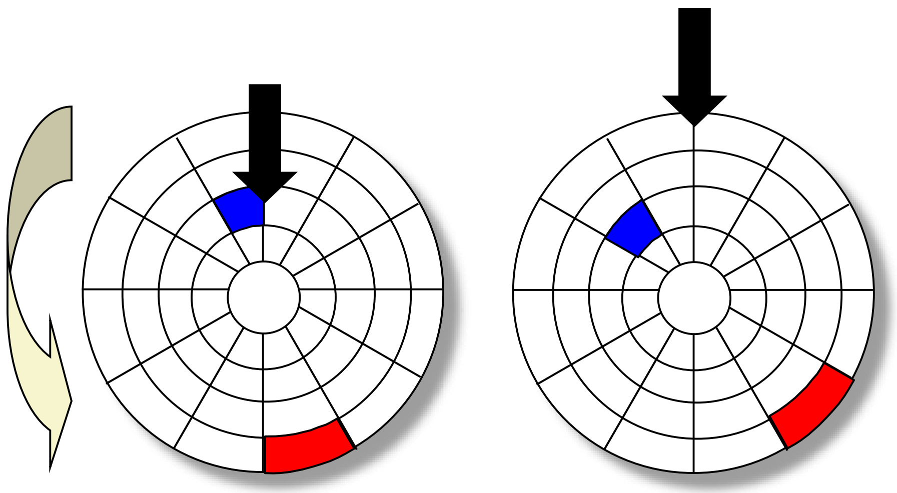
磁盘访问-读/ Disk Access – Read



After **BLUE** read

下一步是读取红色的扇区/Red request scheduled next

Disk Access – Seek

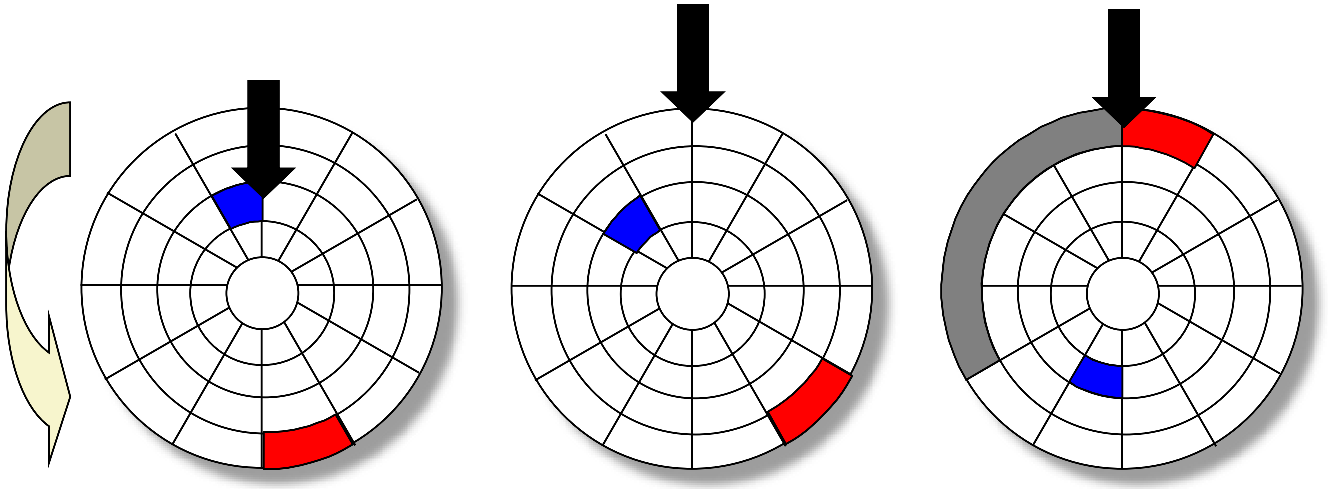


After BLUE read

Seek for RED

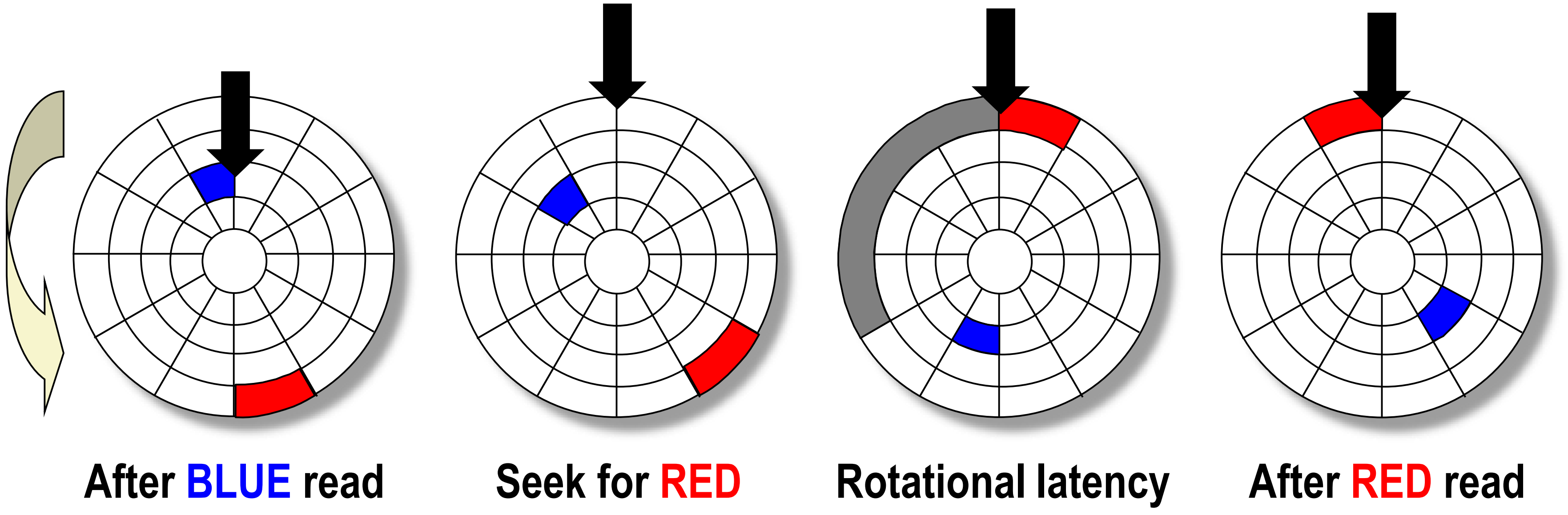
寻道红色的磁道/Seek to red’s track

磁盘访问-延时/ Disk Access – Rotational Latency



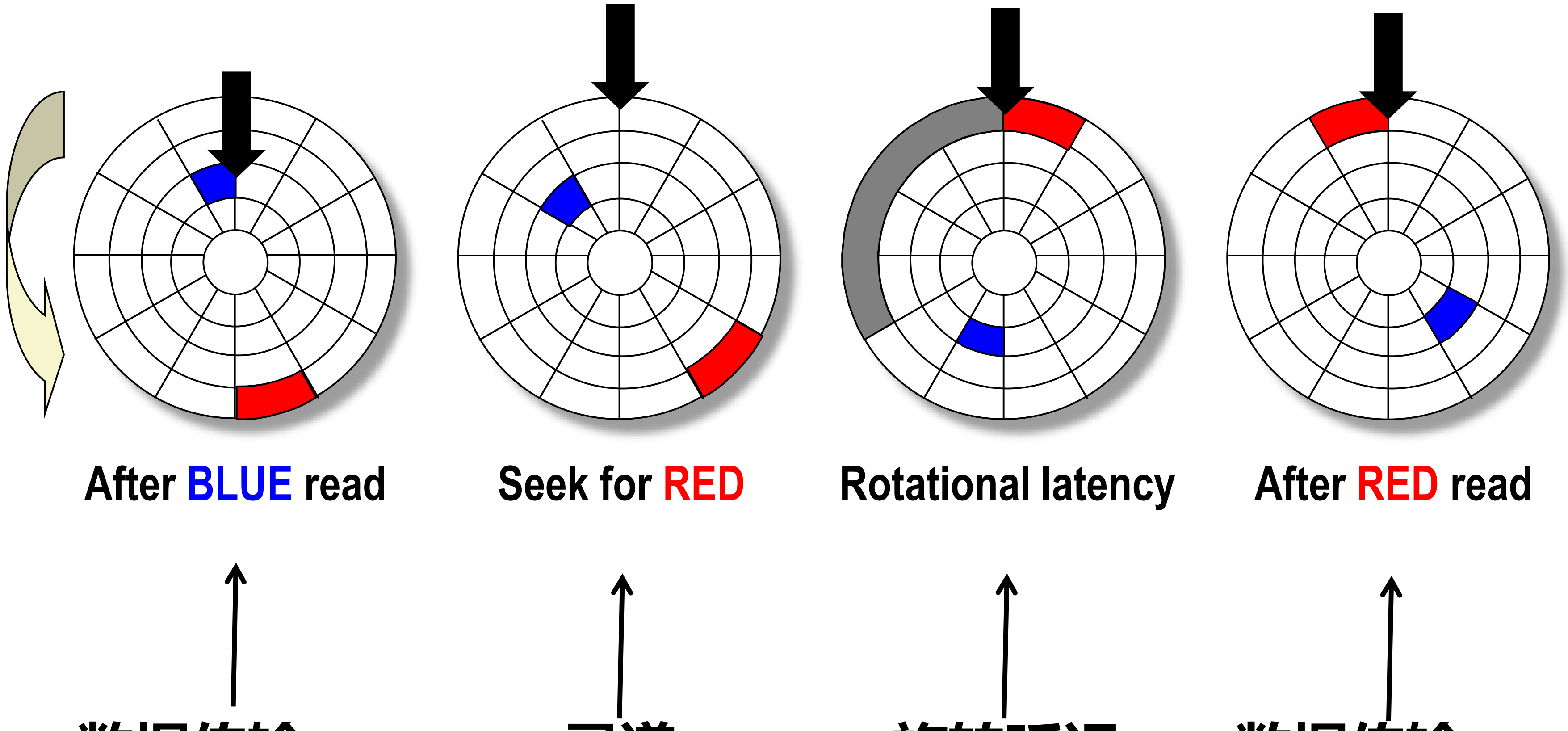
等待红色扇区旋转到位置/Wait for red sector to rotate around

磁盘访问-读/ Disk Access – Read



完成红色扇区读写Complete read of red

磁盘访问-服务时间构成/Disk Access – Service Time Components



数据传输
Data transfer

寻道
Seek

旋转延迟
Rotational latency

数据传输
Data transfer

磁盘访问时间/Disk Access Time

- **访问目标扇区的平均时间/Average time to access some target sector approximated by :**
  - Taccess = Tavg seek + Tavg rotation + Tavg transfer
- **寻道时间/Seek time (Tavg seek)**
  - 将磁头移动到目标扇区的上面/Time to position heads over cylinder containing target sector.
  - Typical Tavg seek is 3—9 ms
- **旋转延迟/Rotational latency (Tavg rotation)**
  - 将r/w头移动到目标扇区第一个bit的等待时间/Time waiting for first bit of target sector to pass under r/w head.
  - Tavg rotation = 1/2 x 1/RPMs x 60 sec/1 min
  - Typical Tavg rotation = 7200 RPMs
- **传输时间/Transfer time (Tavg transfer)**
  - 读取目标扇区比特位的时间/Time to read the bits in the target sector.
  - Tavg transfer = 1/RPM x 1/(avg # sectors/track) x 60 secs/1 min.

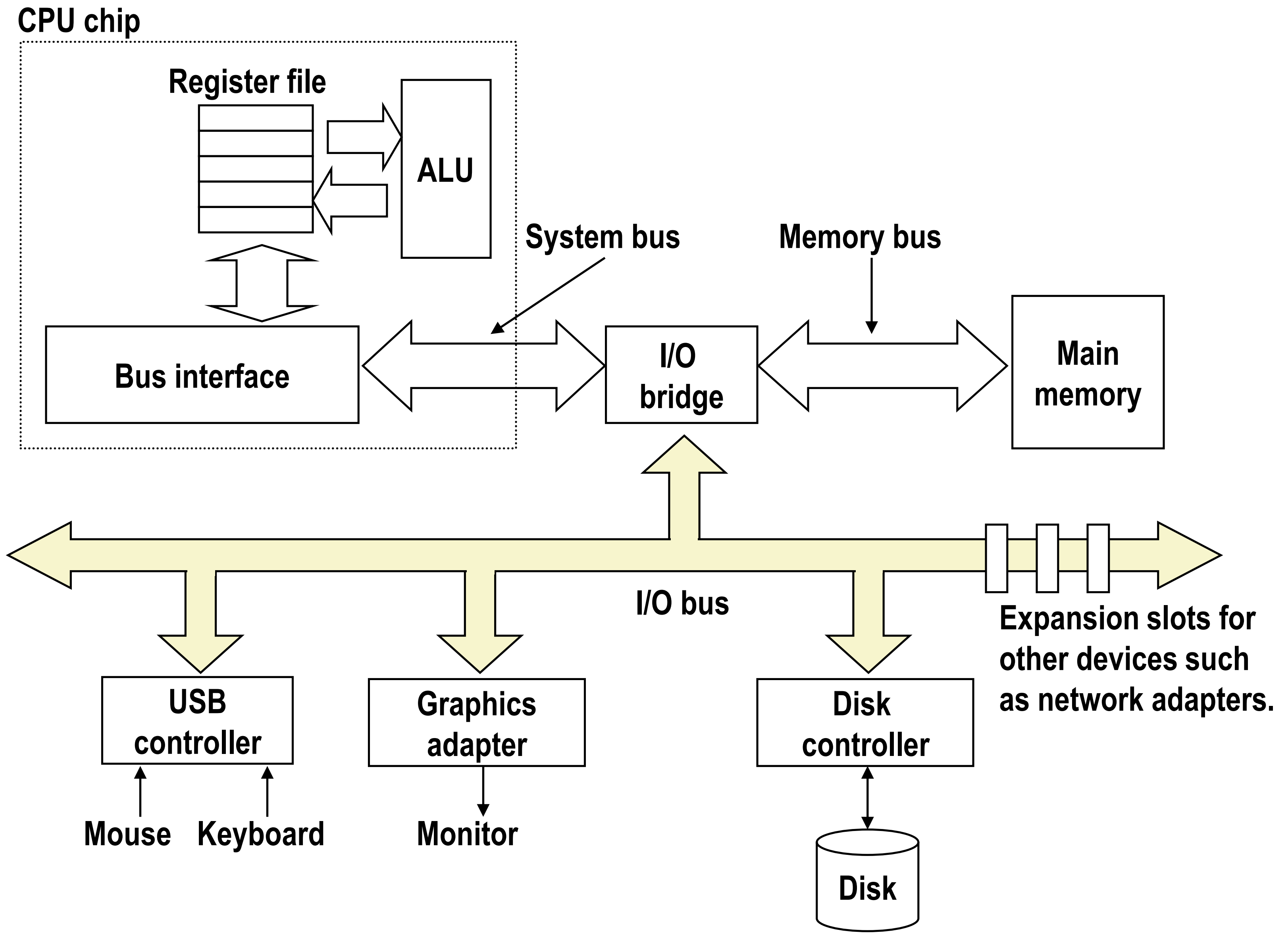
磁盘访问时间举例/Disk Access Time Example

- **假设/Given:**
  - 旋转速度/Rotational rate = 7,200 RPM
  - 平均寻道时间/Average seek time = 9 ms.
  - 平均每个track的扇区数/Avg # sectors/track = 400.
- **可知/Derived:**
  - 平均旋转时间/Tavg rotation = 1/2 x (60 secs/7200 RPM) x 1000 ms/sec = 4 ms.
  - 平均传输时间/Tavg transfer = 60/7200 RPM x 1/400 secs/track x 1000 ms/sec = 0.02 ms
  - 访问时间/Taccess = 9 ms + 4 ms + 0.02 ms
- **需要注意的点/Important points:**
  - 访问延迟主要由寻道时间和旋转延迟决定/Access time dominated by seek time and rotational latency.
  - 扇区内的第一个bit访问是最耗时的，其他的都比较快/First bit in a sector is the most expensive, the rest are free.
  - SRAM的访问时间大概是4 ns/doubleword， DRAM是60ns/SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
    - 磁盘比SRAM慢4万倍/Disk is about 40,000 times slower than SRAM,
    - 磁盘比DRAM慢2500倍/2,500 times slower then DRAM.

逻辑磁盘块/Logical Disk Blocks

- **现在的磁盘通过抽象的逻辑视图隐藏了复杂的扇区布局/Modern disks present a simpler abstract view of the complex sector geometry:**
  - 可用的扇区集合通过大小为b的逻辑块建模/The set of available sectors is modeled as a sequence of b-sized logical blocks (0, 1, 2, ...)
- **逻辑块到物理块之间的映射/Mapping between logical blocks and actual (physical) sectors**
  - 由磁盘控制器完成/Maintained by hardware/firmware device called disk controller.
  - 将逻辑块请求转换为(surface,track,sector)三元组/Converts requests for logical blocks into (surface,track,sector) triples.
- **允许控制器为每个区域留出备用主轴/Allows controller to set aside spare cylinders for each zone.**
  - 解释了格式化容量和最大容量之间的差异/Accounts for the difference in “formatted capacity” and “maximum capacity”.

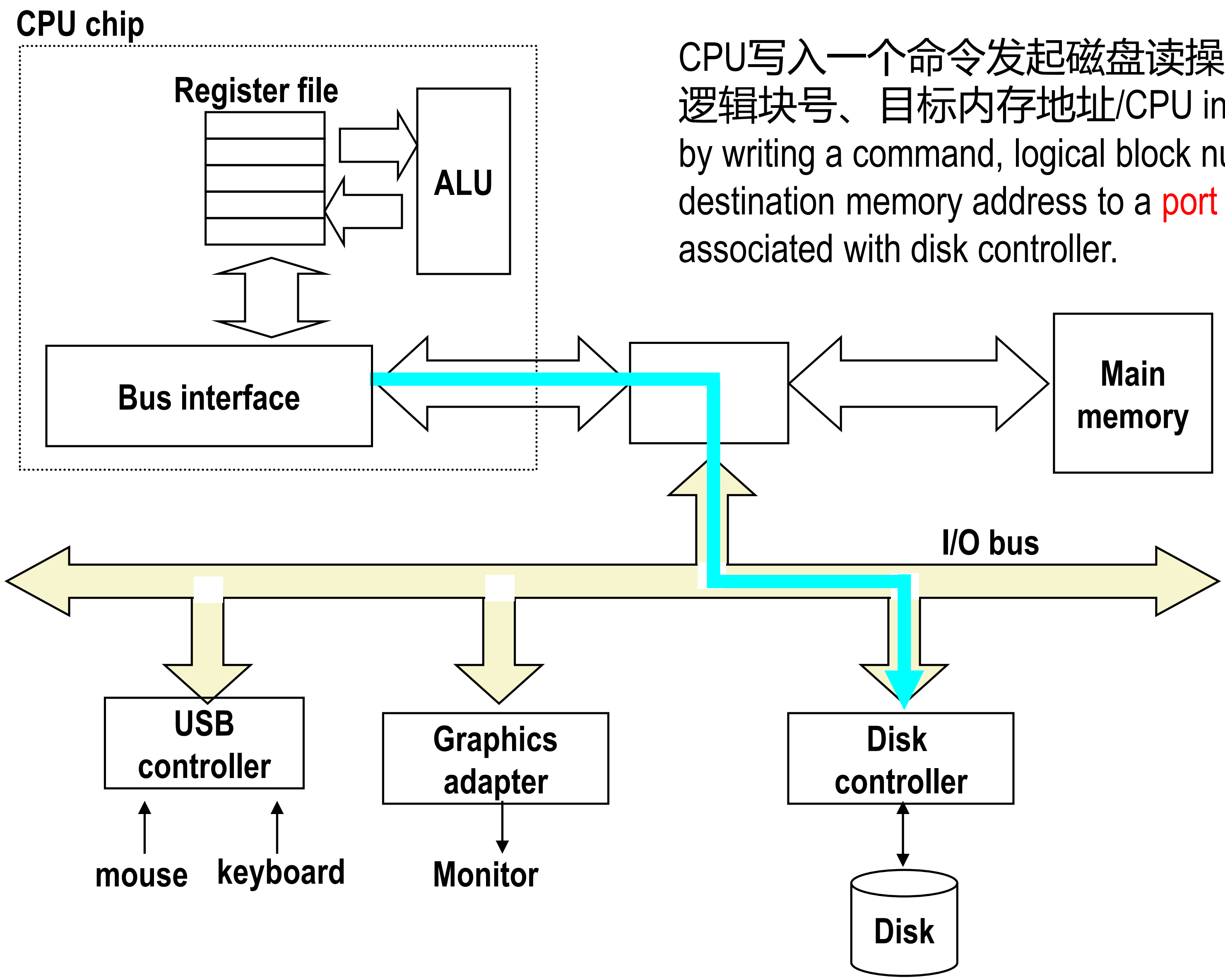
I/O总线/ I/O Bus



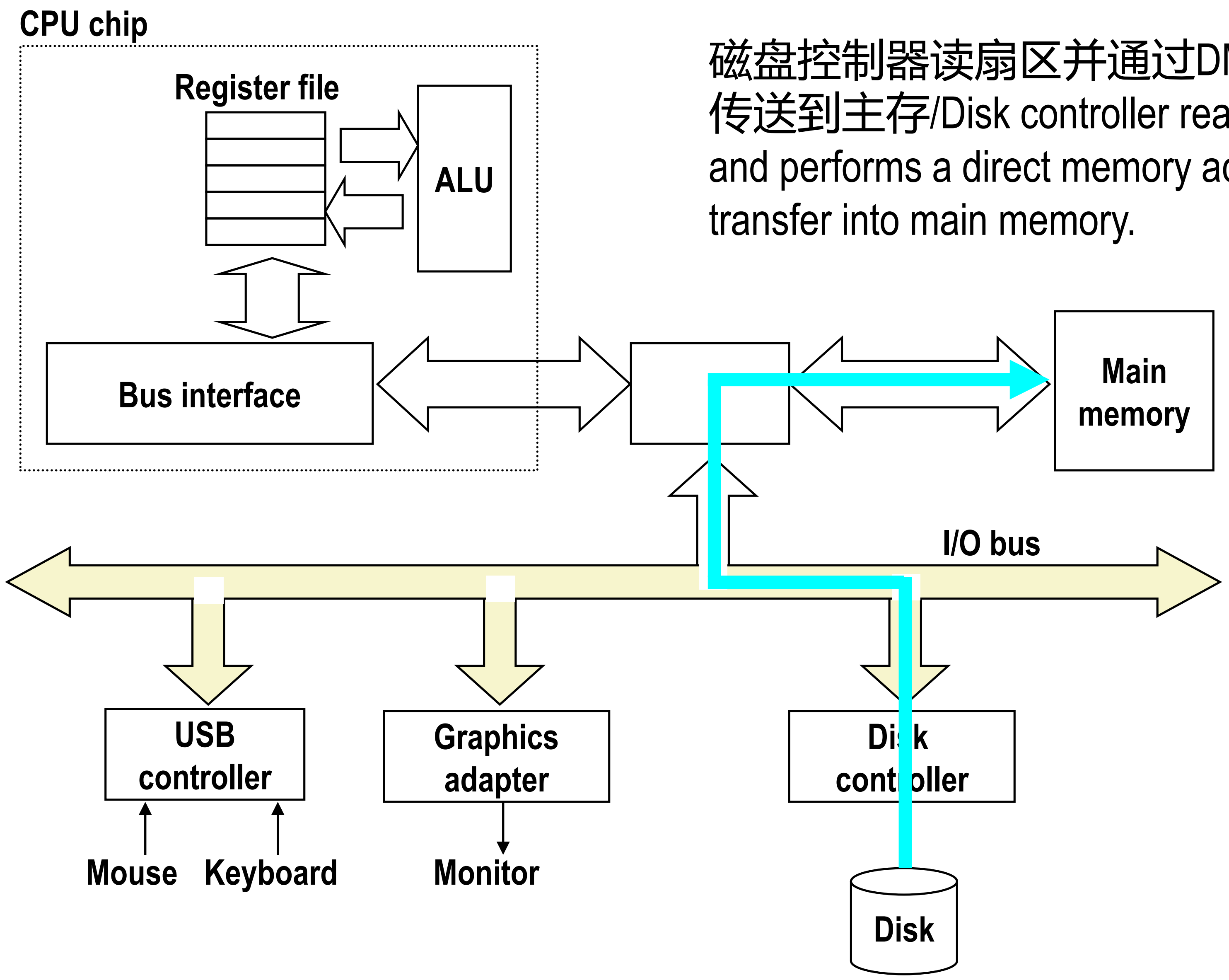
CPU chip
Register file
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory
I/O bus
Expansion slots for other devices such as network adapters.
USB controller
Graphics adapter
Disk controller
Mouse
Keyboard
Monitor
Disk

读取磁盘扇区/Reading a Disk Sector (1)

CPU写入一个命令发起磁盘读操作，命令包括逻辑块号、目标内存地址/CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.

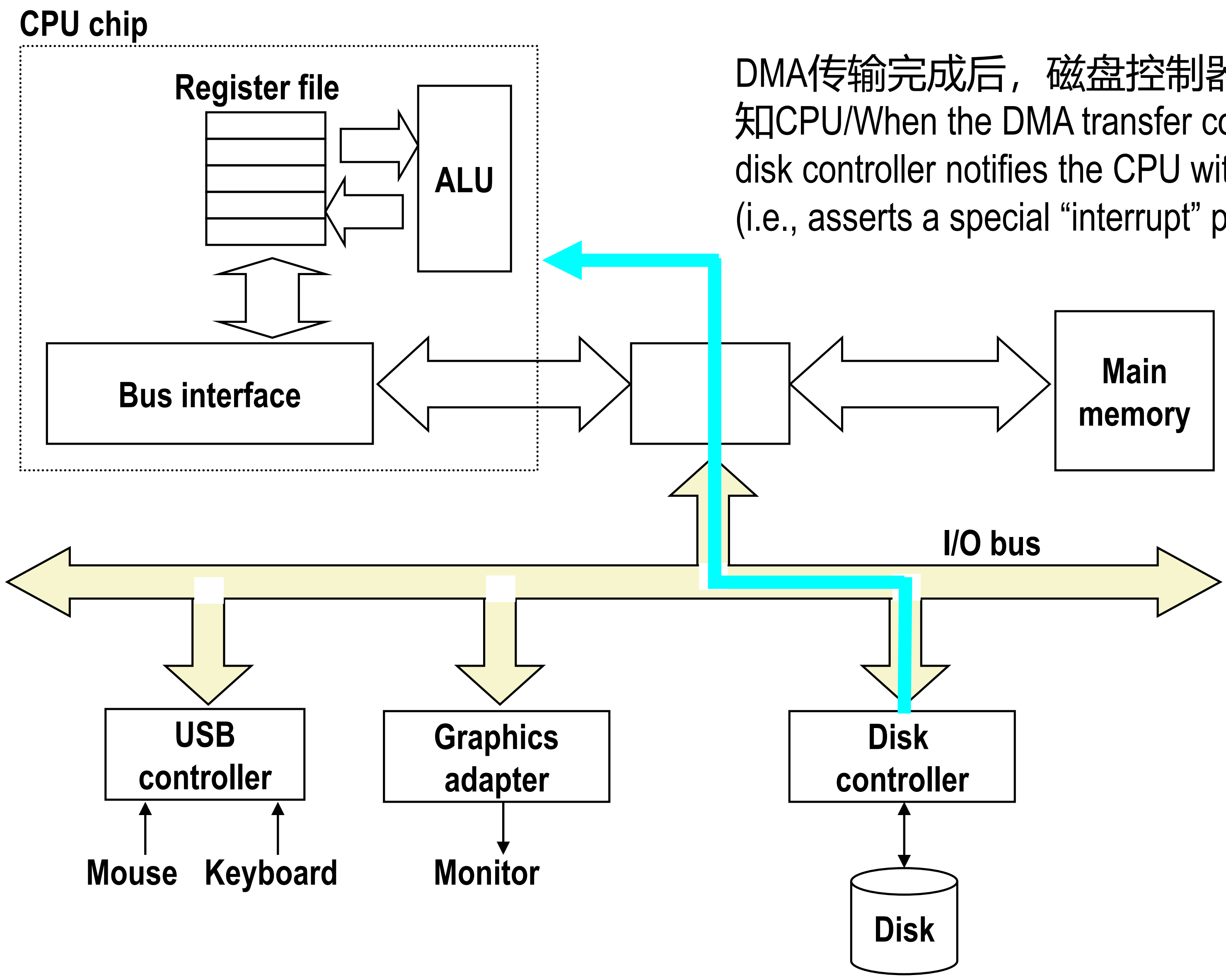


读取磁盘扇区/ Reading a Disk Sector (2)



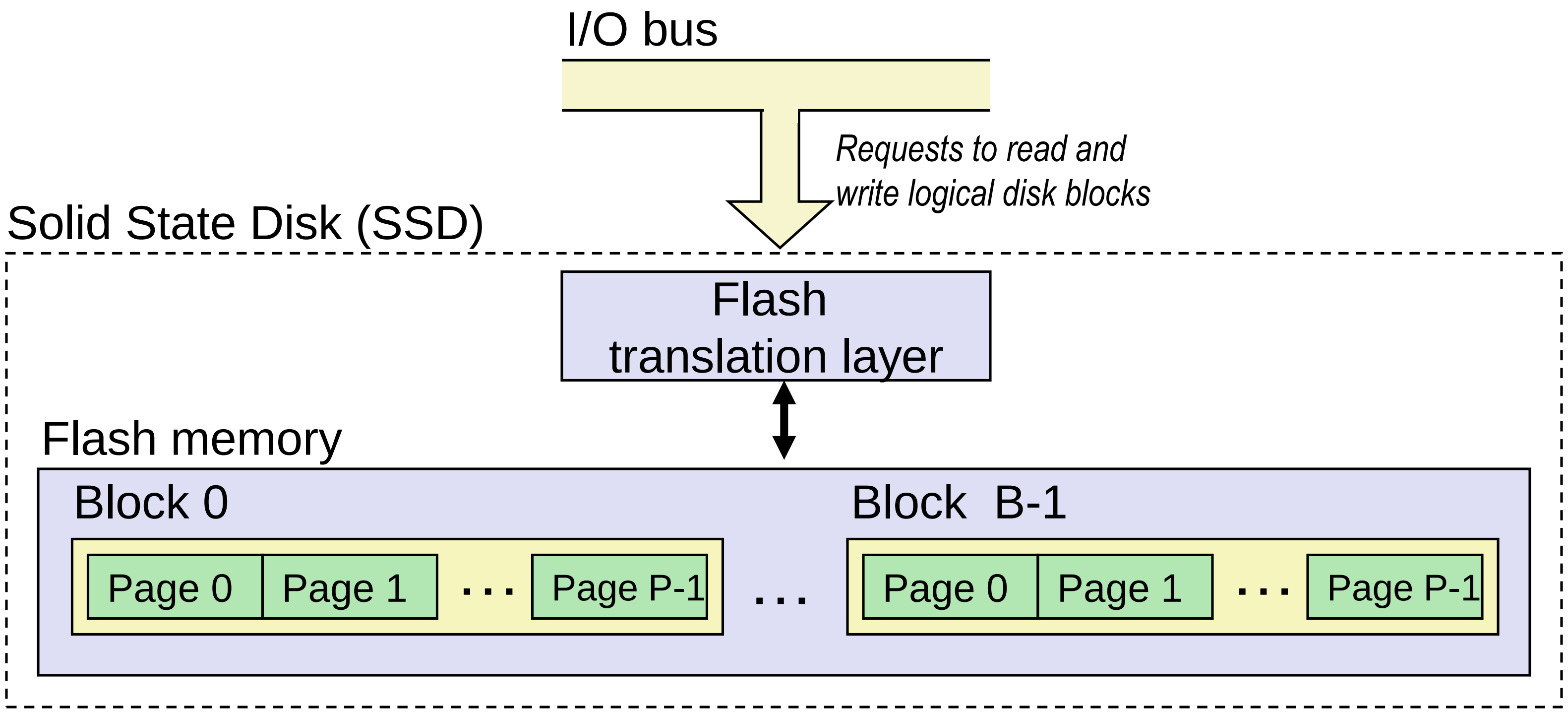
磁盘控制器读扇区并通过DMA将数据传送到主存/Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory.

读取磁盘扇区/ Reading a Disk Sector (3)



DMA传输完成后，磁盘控制器通过中断通知CPU/When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special “interrupt” pin on the CPU)

固态盘/Solid State Disks (SSDs)



- 页/Pages: 512KB to 4KB, 块/Blocks: 32 to 128 pages
- 数据以页为单位进行读/写/Data read/written in units of pages.
- 只有整个块被擦除后才能写入页/Page can be written only after its block has been erased
- 大概10万次写操作后块就会损坏/A block wears out after about 100,000 repeated writes.

SSD性能特点/SSD Performance Characteristics

Sequential read tput	550 MB/s	Sequential write tput	470 MB/s
Random read tput	365 MB/s	Random write tput	303 MB/s
Avg seq read time	50 us	Avg seq write time	60 us

- **顺序访问比随机读更快Sequential access faster than random access**
  - 存储系统的共有特性/Common theme in the memory hierarchy
- **随机写更慢一些/Random writes are somewhat slower**
  - 擦除一个块需要较长时间（~1ms）Erasing a block takes a long time (~1 ms)
  - 修改一个页块需要其他页块拷贝/Modifying a block page requires all other pages to be copied to new block
  - 早期的SSD读写的性能差异更大/In earlier SSDs, the read/write gap was much larger.

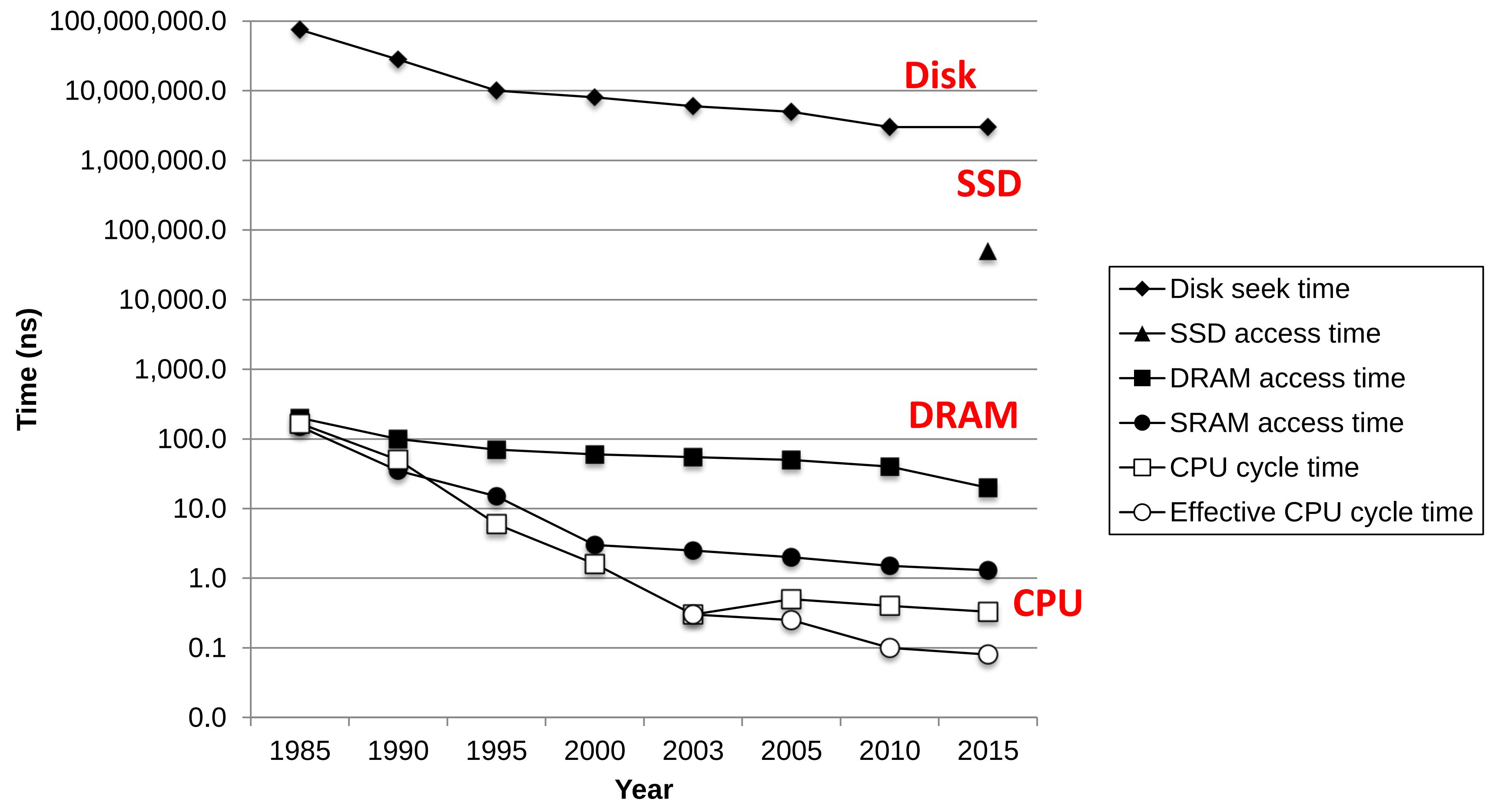
Source: Intel SSD 730 product specification.

SSD与磁盘对比/SSD Tradeoffs vs Rotating Disks

- **优点/Advantages**
  - 没有移动的部件→更快、更节能、更坚固/No moving parts → faster, less power, more rugged
- **缺点/Disadvantages**
  - 有可能损坏/Have the potential to wear out
    - 通过闪存转换层中的“磨损均衡逻辑”缓解/Mitigated by “wear leveling logic” in flash translation layer
    - Intel SSD 730保证128PB读写/E.g. Intel SSD 730 guarantees 128 petabyte (128 x 10^{15} bytes) of writes before they wear out
  - 2013年，每字节成本大概是30倍/In 2015, about 30 times more expensive per byte
- **应用/Applications**
  - MP3播放器、智能手机、笔记本/MP3 players, smart phones, laptops
  - 逐步在桌面和服务器系统中使用/Beginning to appear in desktops and servers

CPU-内存差距/The CPU-Memory Gap

DRAM、磁盘和CPU之间的速度差异不断变大/The gap widens between DRAM, disk, and CPU speeds.



局域性是关键/Locality to the Rescue!

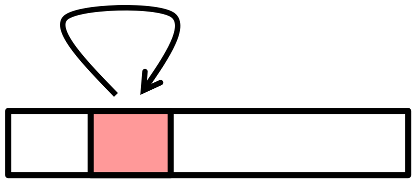
- **利用计算机程序的一个重要特性：局域性，能够消除CPU和内存之间速度差异The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality**

提纲

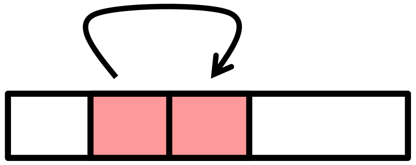
- 存储技术与趋势/Storage technologies and trends
- 访存局部性原理/Locality of reference
- 存储层次中的Cacheing/Caching in the memory hierarchy

局域性/Locality

- **局域性原理/Principle of Locality: 程序更倾向于使用临近或者最近使用过的数据和指令/Programs tend to use data and instructions with addresses near or equal to those they have used recently**



- **时间局域性/Temporal locality:**
  - 最近使用的数据在不远的将来会继续使用/Recently referenced items are likely to be referenced again in the near future



- **空间局域性/Spatial locality:**
  - 相邻的数据会在一段时间内一起访问/Items with nearby addresses tend to be referenced close together in time

局域性举例/Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- **数据访存/Data references**
  - 连续数组元素访问/Reference array elements in succession (stride-1 reference pattern).
  - 每个迭代都访问变量sum/Reference variable `sum` each iteration.
- **指令访存/Instruction references**
  - 指令连续访问/Reference instructions in sequence.
  - 按照循环迭代周期性访问/Cycle through loop repeatedly.

空间局域性/Spatial locality

时间局域性/Temporal locality

空间局域性/Spatial locality

时间局域性/Temporal locality

局域性量化评估/Qualitative Estimates of Locality

- **声明**：对于一个专业的程序员来说，根据代码来判断其数据访问的局域性程度是一项非常重要的技能/**Claim:** Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.
- **问题**：这个函数访问数组a时是否具有较好的局域性？/**Question:** Does this function have good locality with respect to array `a`?

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

局域性举例/Locality Example

- 问题：这个函数访问数组a时是否具有很好的局域性?/Question: Does this function have good locality with respect to array a?

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

局域性举例/Locality Example

- 问题：是否可以交换循环实现3-d数组a的连续访问（具有较好的空间局域性）？/Question: Can you permute the loops so that the function scans the 3-d array `a` with a stride-1 reference pattern (and thus has good spatial locality)?

```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < N; k++)
                sum += a[k][i][j];
    return sum;
}
```

存储层次/Memory Hierarchies

- **硬件和软件的一些基本和持久特性/Some fundamental and enduring properties of hardware and software:**
  - 更快的存储技术成本更高，容量更小，能耗更高/Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - CPU和主存之间的速度差异不断地在增大/The gap between CPU and main memory speed is widening.
  - 仔细考量的程序会呈现出更好地局域性/Well-written programs tend to exhibit good locality.
- **这些不同的特性之间互相弥补/These fundamental properties complement each other beautifully.**
- **启发了人们将内存组织为一个存储层次/They suggest an approach for organizing memory and storage systems known as a memory hierarchy.**

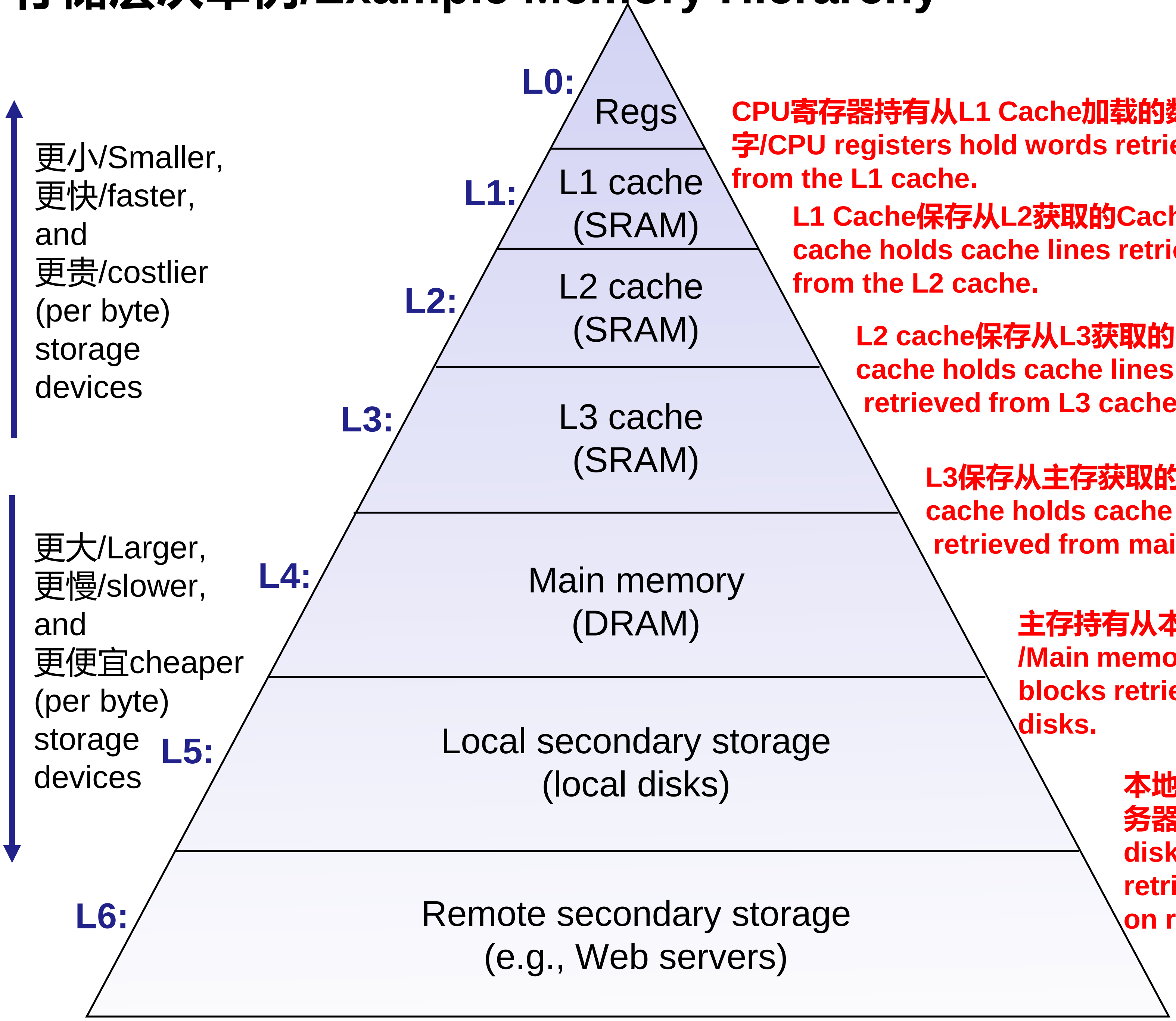
Today

- Storage technologies and trends
- Locality of reference
- **Caching in the memory hierarchy**

提纲

- 存储技术与趋势/Storage technologies and trends
- 访存局部性原理/Locality of reference
- 存储层次中的Cacheing/Caching in the memory hierarchy

存储层次举例/Example Memory Hierarchy



CPU寄存器持有从L1 Cache加载的数据字/CPU registers hold words retrieved from the L1 cache.

L1 Cache保存从L2获取的Cache行/L1 cache holds cache lines retrieved from the L2 cache.

L2 cache保存从L3获取的cache行/L2 cache holds cache lines retrieved from L3 cache

L3保存从主存获取的cache行/L3 cache holds cache lines retrieved from main memory.

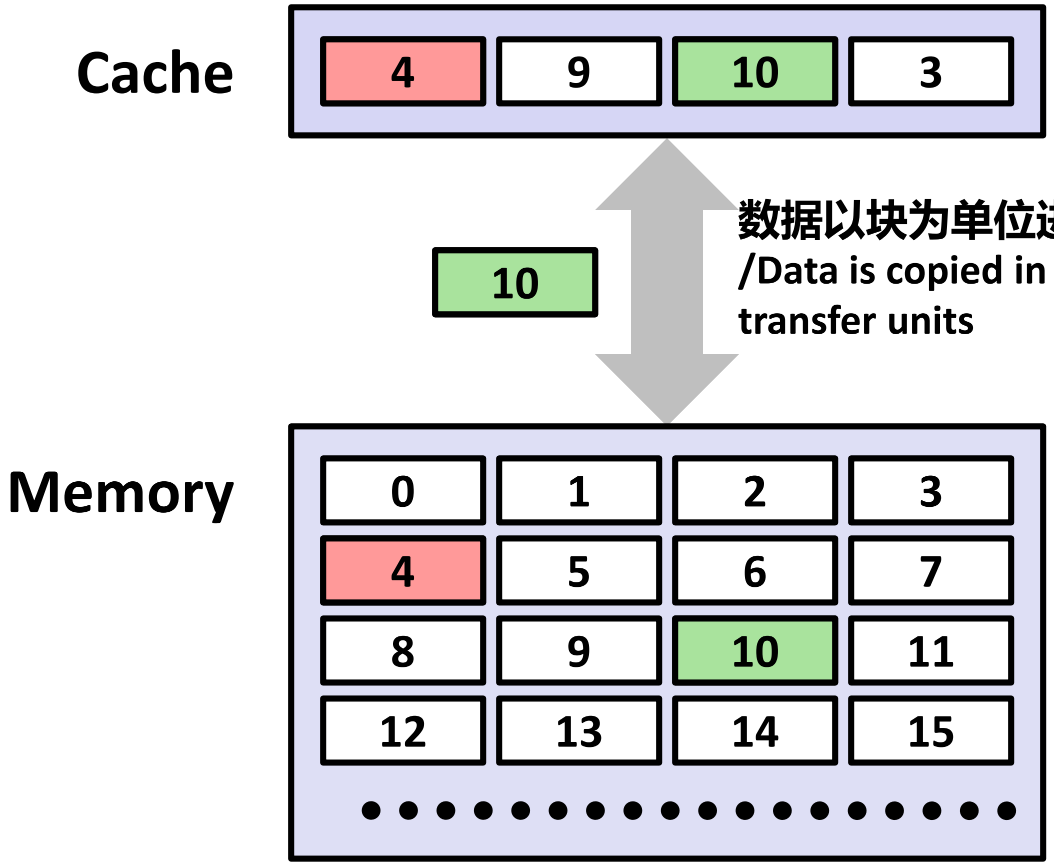
主存持有从本地磁盘读取的数据/Main memory holds disk blocks retrieved from local disks.

本地磁盘持有从远程服务器获取的数据/Local disks hold files retrieved from disks on remote servers

Caches

- ***Cache:*一个更小更快的存储模块，可以看做是较大、访问速度较低存储设备数据子集的暂存区/*Cache:* A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.**
- **存储层次的重要思路/Fundamental idea of a memory hierarchy:**
  - 将k层次更小更快的存储部件看做是k+1层次更大更慢存储部件的缓存/For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.
- **存储层次为什么能工作？/Why do memory hierarchies work?**
  - 由于局域性，程序大多数时间会方位k层次的缓存数据/Because of locality, programs tend to access the data at level k more often than they access the data at level k+1.
  - 因此，k+1层次的存储可以更慢一些，容量更大，每个比特位的成本也可以更低/Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.
- ***大思路：存储层次构造了一个成本接近于底层，但是数据访问速度接近顶层的大的存储池/Big Idea:* The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.**

Cache基本概念/General Cache Concepts

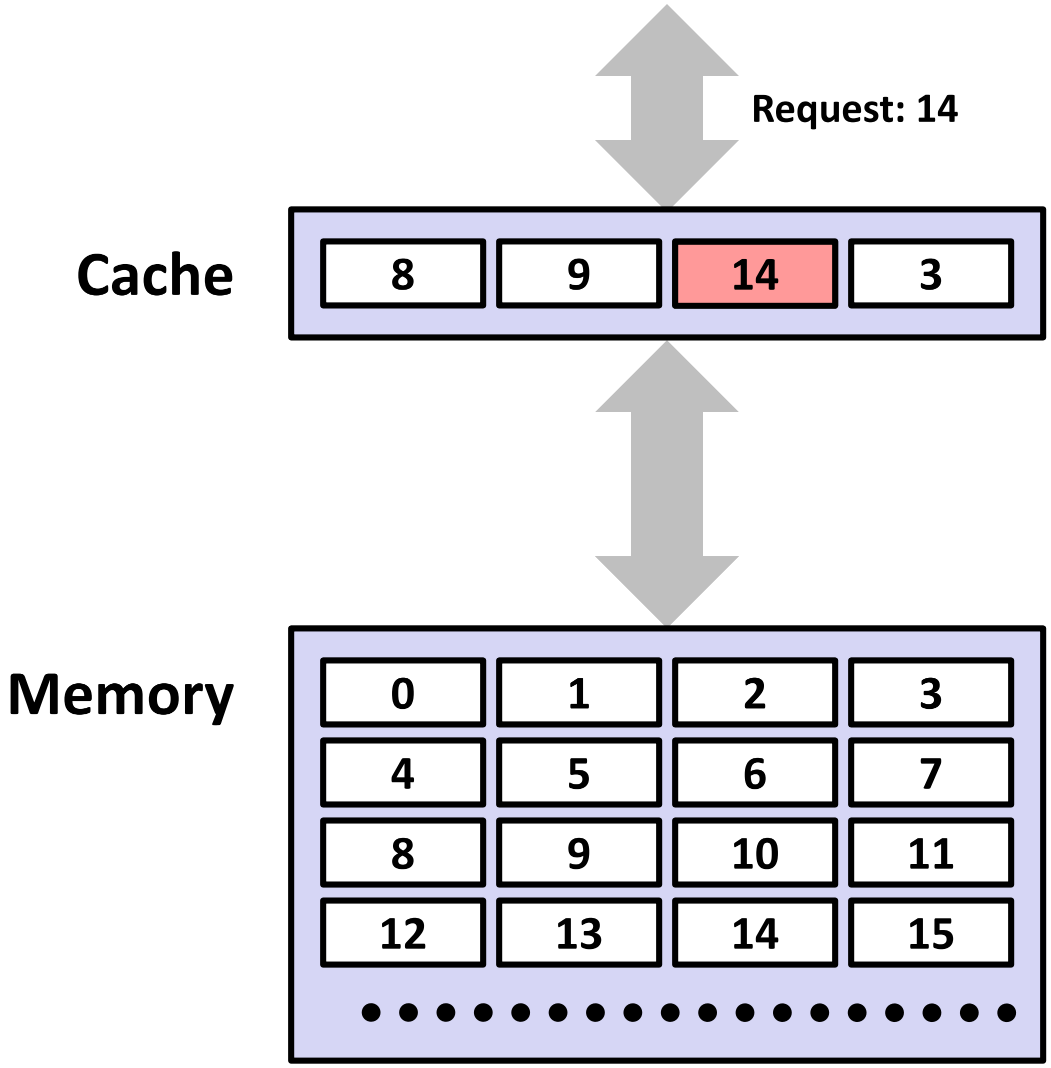


更小、更快、更贵，缓存了一部分块/
Smaller, faster, more expensive memory caches a subset of the blocks

数据以块为单位进行拷贝传输
/Data is copied in block-sized transfer units

更大、更慢、更便宜的主存被划分为多个块/
Larger, slower, cheaper memory viewed as partitioned into “blocks”

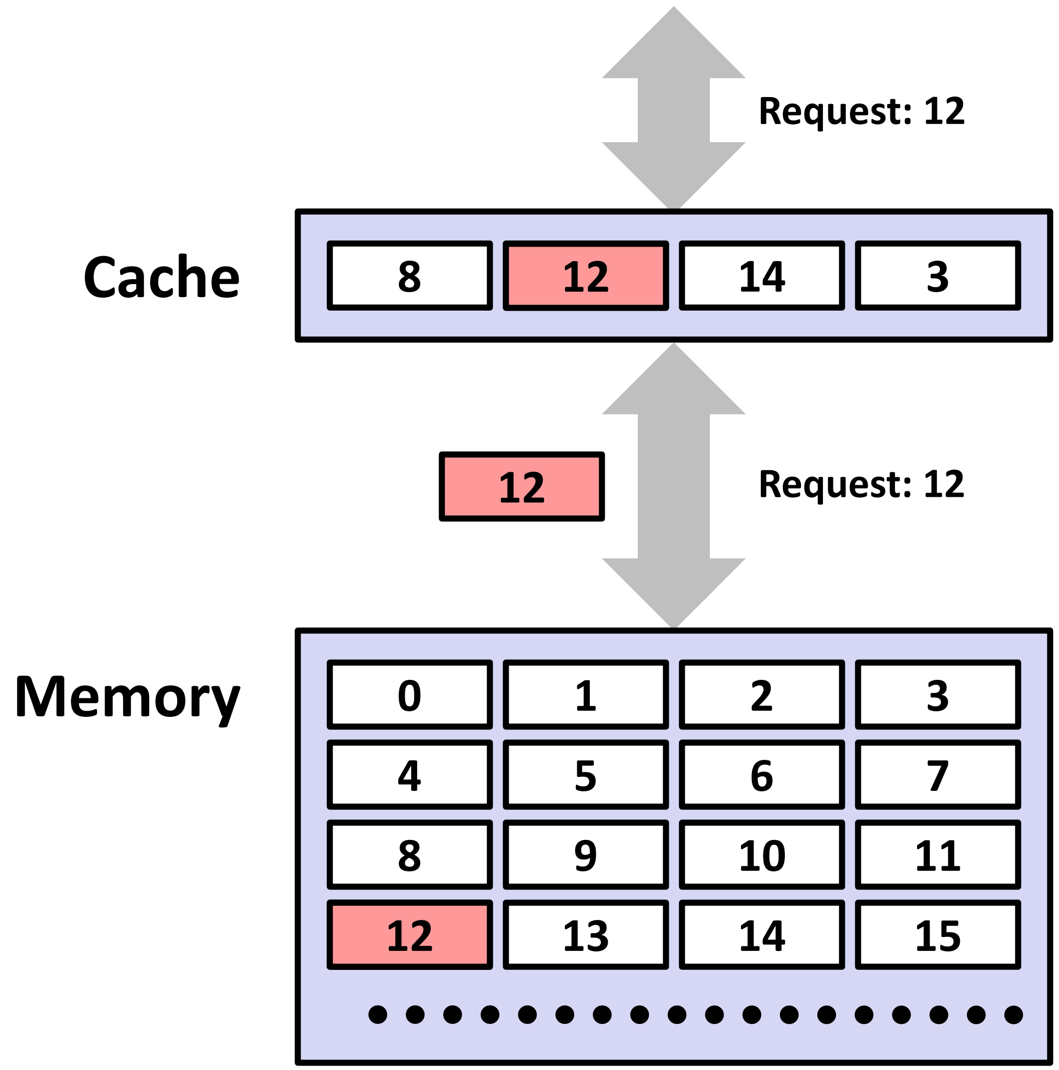
Cache命中/General Cache Concepts: Hit



需要访问块b中的数据/
Data in block b is needed

块b在cache中：命中/
Block b is in cache:
Hit!

Cache丢失/General Cache Concepts: Miss



需要访问块b中的数据/
Data in block b is needed

块b不在cache中：丢失！/
Block b is not in cache:
Miss!

从内存中加载块b/
Block b is fetched from memory

将块b存储在cache中/
Block b is stored in cache

- 放置策略Placement policy: 决定b放在哪里/determines where b goes
- 替换策略Replacement policy: 决定把那个块换出/determines which block gets evicted (victim)

Cache丢失类型/General Caching Concepts: Types of Cache Misses

- **冷启动丢失/Cold (compulsory) miss**
  - 冷启动丢失是因为刚开始Cache是空的/Cold misses occur because the cache is empty.
- **冲突丢失/Conflict miss**
  - 大多数Cache将k+1层的数据块映射到k层更小的一个或者多个块位置/Most caches limit blocks at level k+1 to a small subset (sometimes a singleton) of the block positions at level k.
    - 例如k+1层的块被映射到k层（i mod 4）的位置
    - E.g. Block i at level k+1 must be placed in block (i mod 4) at level k.
  - 当多个块被映射到k层的同一个位置时就会出现冲突/Conflict misses occur when the level k cache is large enough, but multiple data objects all map to the same level k block.
    - 例如访问0,8,0,8,0,8就会每次导致Cache丢失/E.g. Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time.
- **容量丢失/Capacity miss**
  - 当活跃使用的Cache块大小大于Cache容量时就会导致丢失/Occurs when the set of active cache blocks (working set) is larger than the cache.

存储层次中的Cache举例/Examples of Caching in the Mem. Hierarchy

Cache类型 Cache Type	Cache的内容 What is Cached?	Cache的位置 Where is it Cached?	延迟（周期） Latency (cycles)	由谁管理 Managed By
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-Chip L1	4	Hardware
L2 cache	64-byte blocks	On-Chip L2	10	Hardware
Virtual Memory	4-KB pages	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

总结/Summary

- CPU、内存和大容量存储之间的速度差异还在不断持续增大/The speed gap between CPU, memory and mass storage continues to widen.

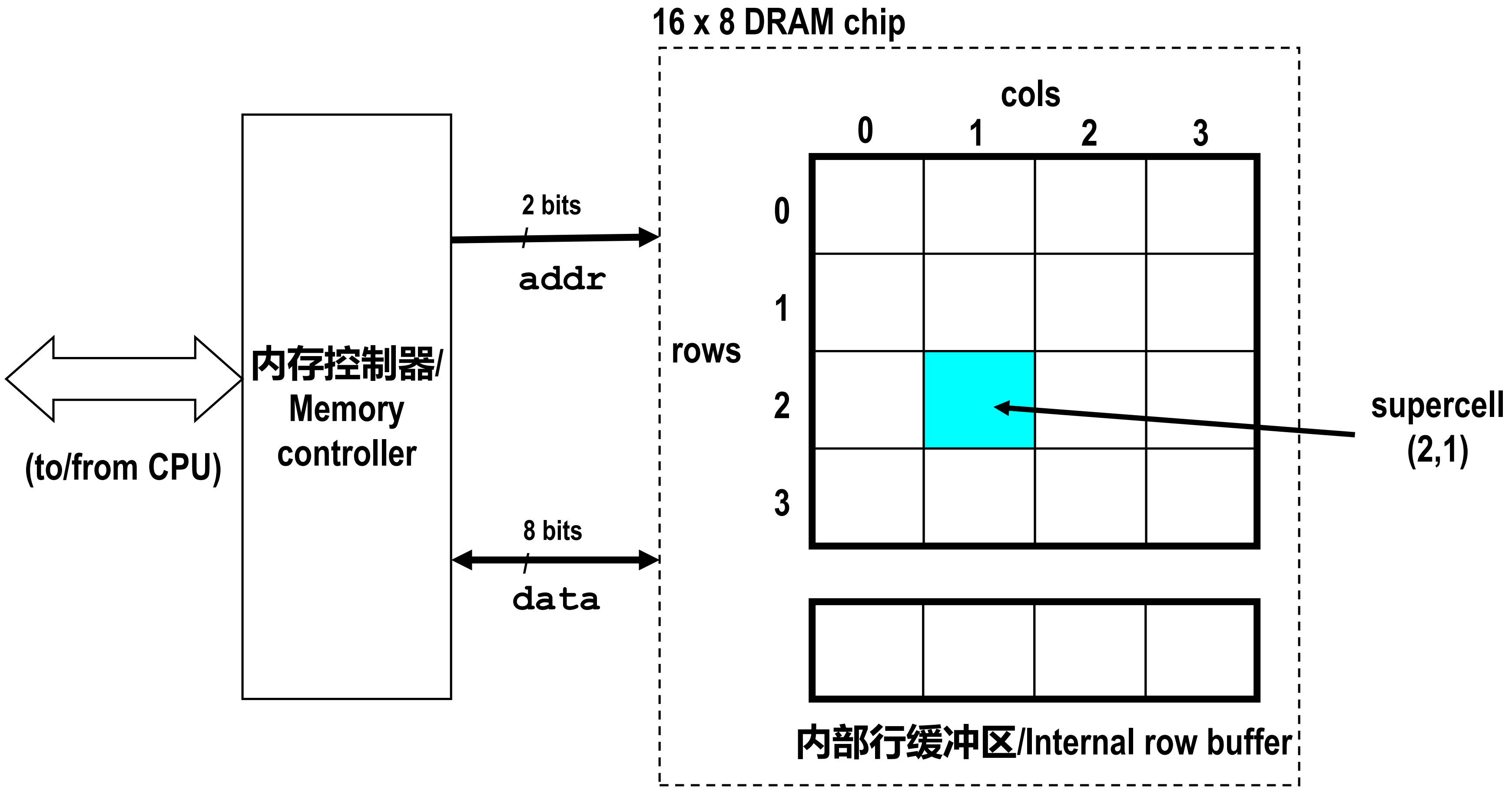
- 仔细考量的代码能能够很好地利用数据局域性/Well-written programs exhibit a property called *locality*.

- 存储层次中的Cache机制利用局部性消除了层次之间的速度差异/Memory hierarchies based on *caching* close the gap by exploiting locality.

补充材料/Supplemental slides

传统DRAM结构/Conventional DRAM Organization

- **d x w DRAM:**
  - dw位被组织为d个w位宽的supercell
  - dw total bits organized as d supercells of size w bits



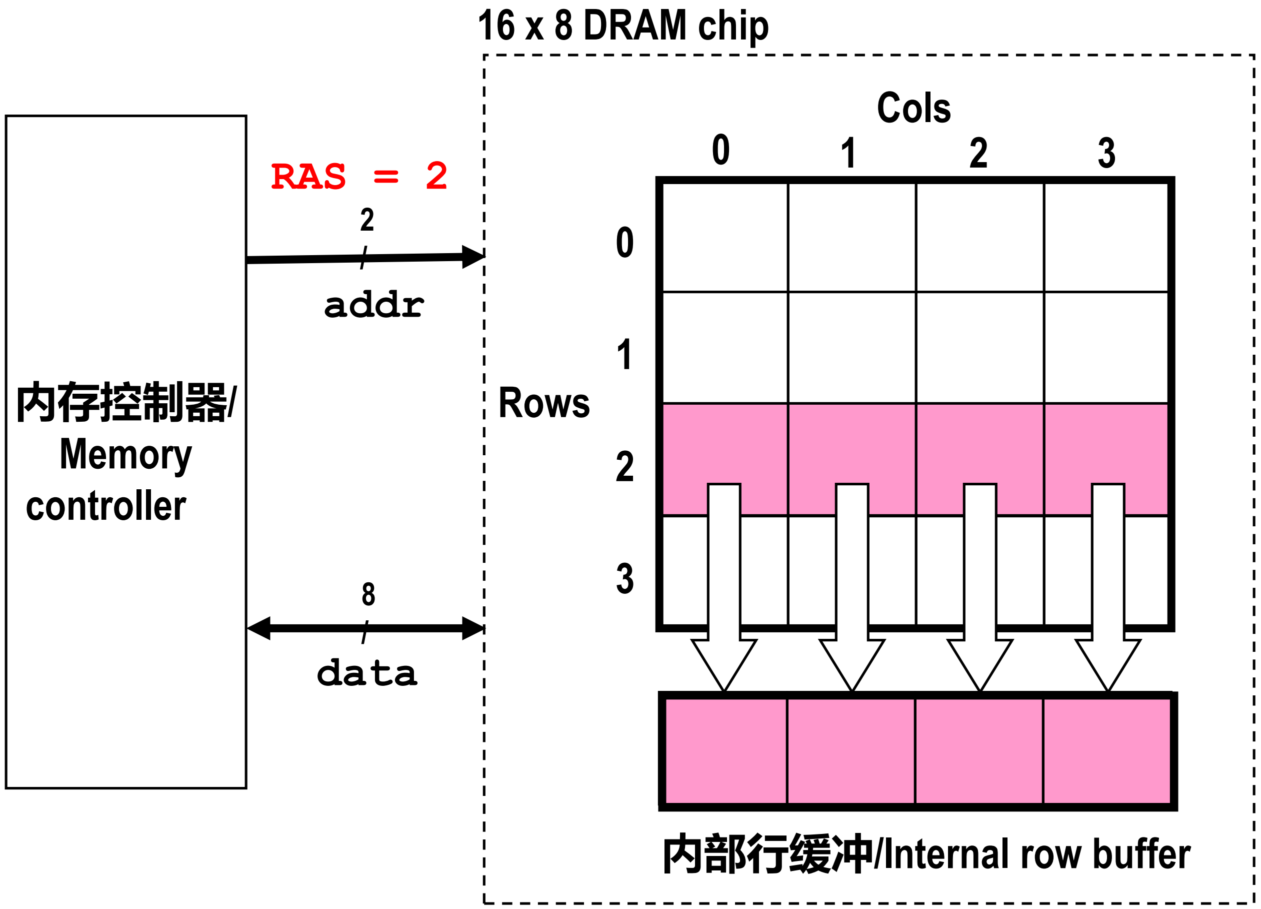
读取DRAM Supercell(2,1)/Reading DRAM Supercell (2,1)

步骤1(a)：行选通(RAS)选择行2

Step 1(a): Row access strobe (RAS) selects row 2.

步骤1(b)：将数据从DRAM阵列拷贝到缓冲区

Step 1(b): Row 2 copied from DRAM array to row buffer.



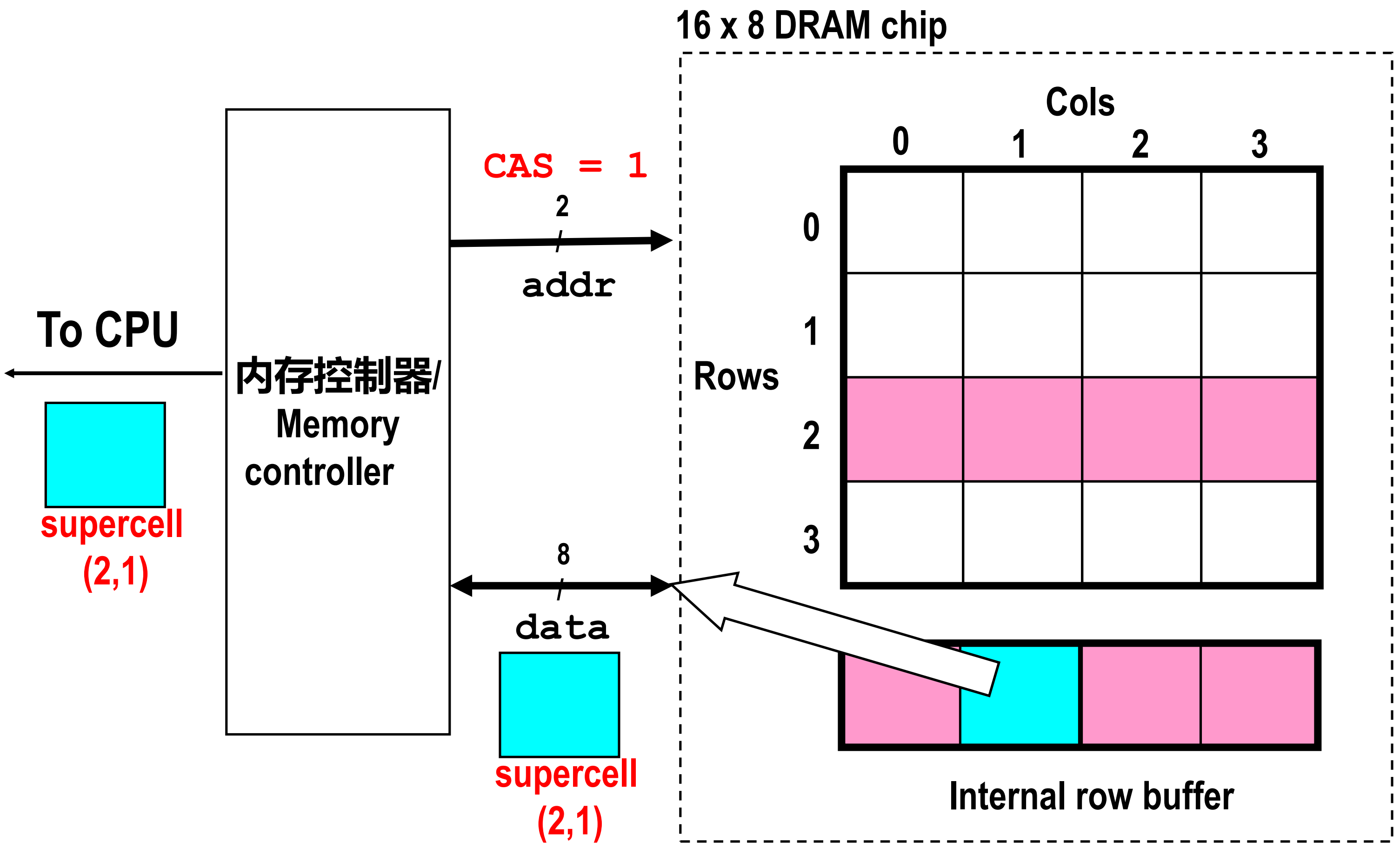
读取DRAM Supercell(2,1)/Reading DRAM Supercell (2,1)

步骤2(a):列访问选通(CAS)选择列1

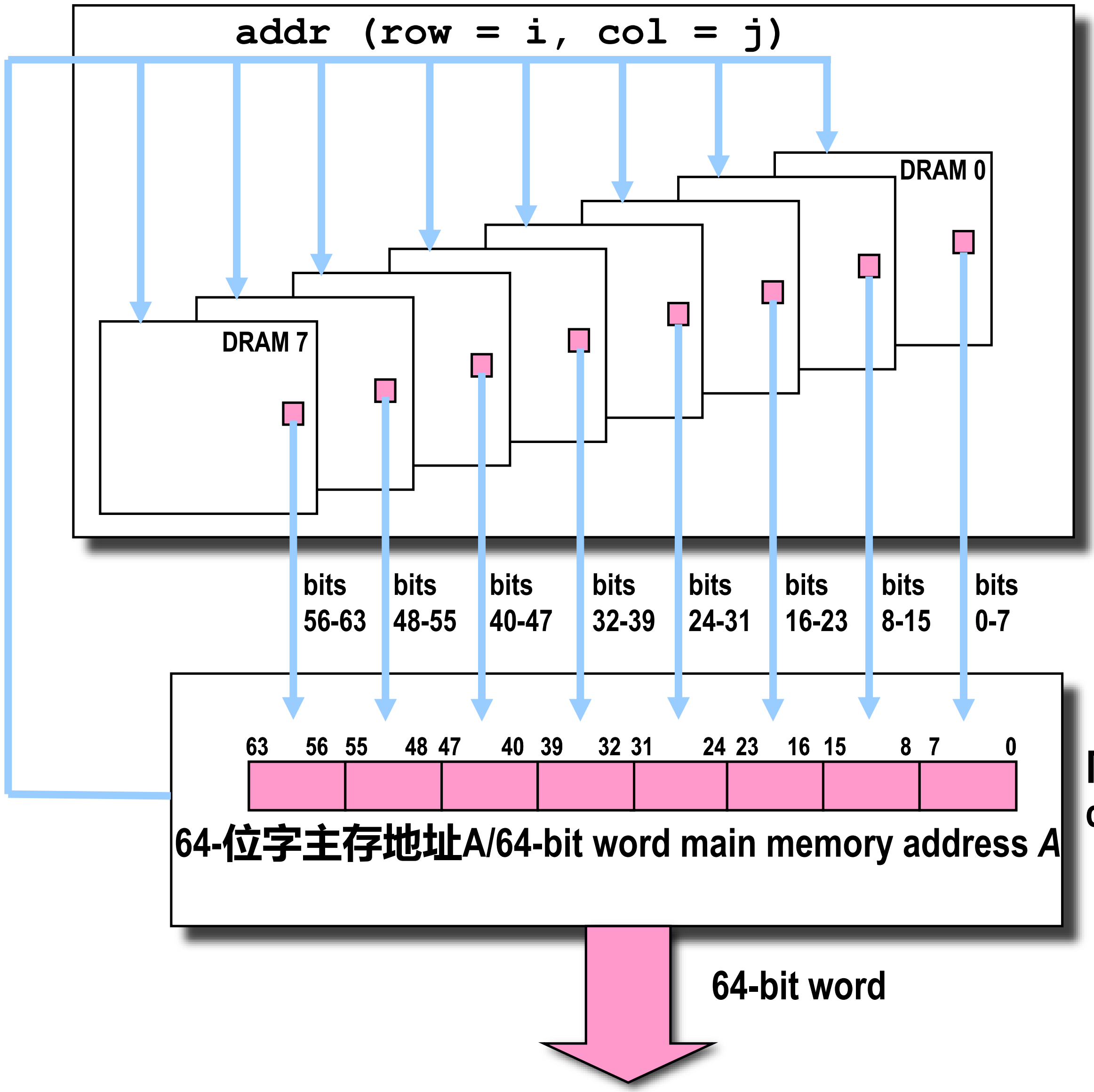
Step 2(a): Column access strobe (CAS) selects column 1.

步骤2(b):将Supercell(2,1)从缓冲区拷贝到数据线，最终返回给CPU

Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.



存储模块/Memory Modules



🟪 : supercell (i,j)

64MB的存储模块由8个8Mx8的DRAM构成/
64 MB
memory module
consisting of
eight 8Mx8 DRAMs

内存控制器/Memory controller

增强的DRAM/Enhanced DRAMs

- **基本的DRAM单元自1966年以后就没有改变过/Basic DRAM cell has not changed since its invention in 1966.**
  - 1970年由Intel商业化/Commercialized by Intel in 1970.
- **DRAM cores with better interface logic and faster I/O :**
  - 同步DRAM/Synchronous DRAM (SDRAM)
    - Uses a conventional clock signal instead of asynchronous control
    - Allows reuse of the row addresses (e.g., RAS, CAS, CAS, CAS)
  - 双倍速率DRAM/Double data-rate synchronous DRAM (DDR SDRAM)
    - 每个数据线每周期分别在时钟上升和下降沿进行数据传输/Double edge clocking sends two bits per cycle per pin
    - 根据预取缓冲区大小分为几类/Different types distinguished by size of small prefetch buffer:
      - DDR (2 bits), DDR2 (4 bits), DDR3 (8 bits)
    - 2010年以后，已经成为桌面和服务器系统的标准/By 2010, standard for most server and desktop systems
    - Intel Core i7只支持DDR3 SDRAM/Intel Core i7 supports only DDR3 SDRAM

存储发展趋势/Storage Trends

SRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	2,900	320	256	100	75	60	*320*	*116*
access (ns)	150	35	15	3	2	1.5	*200*	*115*

DRAM

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/MB	880	100	30	1	0.1	0.06	0.02	*44,000*
access (ns)	200	100	70	60	50	40	20	*10*
typical size (MB)	0.256	4	16	64	2,000	8,000	16.000	*62,500*

Disk

Metric	1985	1990	1995	2000	2005	2010	2015	*2015:1985*
$/GB	100,000	8,000	300	10	5	0.3	0.03	*3,333,333*
access (ms)	*75*	28	10	8	*5*	*3*	*3*	*25*
typical size (GB)	0.01	0.16	1	20	160	1,500	3,000	*300,000*

CPU时钟频率/CPU Clock Rates

计算机历史上当设计者遇到"能耗墙"的转折点Inflection point in computer history when designers hit the "Power Wall"

	1985	1990	1995	2003	2005	2010	2015	*2015:1985*
CPU	80286	80386	Pentium	P-4	Core 2	Core i7(n)	Core i7(h)	
Clock rate (MHz)	6	20	150	3,300	2,000	2,500	3,000	500
Cycle time (ns)	166	50	6	0.30	0.50	0.4	0.33	500
Cores	1	1	1	1	2	4	4	4
Effective cycle time (ns)	166	50	6	0.30	0.25	0.10	0.08	2,075

(n) Nehalem processor
(h) Haswell processor