存储器层次结构

徐子扬

从“运行内存”说起

在购买电脑时我们常常能见到“16G+512G”“32G+2T”这种字眼，那么其中的“16G”“32G”指的是(B)，“512G”“2T”指的是(A)

A.硬盘　　B.内存　　C.运行内存　　D.ROM

运行内存是一个错误的说法，一般人们所说的运行内存实际指的就是内存。

不可以将硬盘与内存混淆

常见存储器

- 内存：又称主存，将硬盘中的数据暂存，方便CPU使用。同时也存放CPU的临时数据。
- 硬盘：计算机的外部存储装置，硬盘中的数据必须先输入到内存中才能被CPU使用。
- RAM：随机访问存储器，分为SRAM和DRAM两大类，是内存中最重要的一部分。断电会失去保存的数据
- ROM：只读存储器，也是随机访问存储器的一种，在断电后也能保持数据。

这四者只有硬盘属于外存，其他属于内存。

（个人理解：一般语境下的内存特指DRAM，而广义的内存也可包括SRAM、DRAM、ROM等。）

随机访问存储器

（内存）

SRAM和DRAM

RAM分为静态的SRAM和动态的DRAM两类。SRAM的造价更贵，速度更快。

在用途方面，SRAM常用于制作高速缓存存储器（cache）而DRAM则用于制作主存以及图形系统的帧缓冲区。

一般来说，桌面系统的SRAM只有几MB，而DRAM则有数百乃至数千MB。

	每位晶体管数	相对访问时间	持续的?	敏感的?	相对花费	应用
SRAM	6	1×	是	否	1000×	高速缓存存储器
DRAM	1	10×	否	是	1×	主存，帧缓冲区

图 6-2　DRAM 和 SRAM 存储器的特性

SRAM

- 存储单元：一个双稳态的存储器单元。
- 特性：

只能在两个电压上保持稳定，其余状态均会迅速转移到这两个状态上，如同下图的钟摆一样。

只要有电，它可以一直保持稳态，而不受电子噪声的干扰。

图 6-1　倒转的钟摆。同 SRAM 单元一样，钟摆只有两个稳定的配置或状态

DRAM

- 存储单元：由一个电容和一个访问晶体管组成。
- 特性：

  对干扰十分敏感，电容的电压被扰乱后不会恢复。

  暴露在光线下会导致电容电压改变。（可用于照相机中的传感器）

  漏电会使单元在较短时间内失去电荷
- 可通过重写来刷新内存的每一位，有时也可通过纠错码来修正。

传统DRAM

一般用d*w来描述DRAM芯片的组织，其中d为一枚DRAM芯片中超单元的数量，而一个超单元则有w个单元，共存储d*w位信息。

超单元被组织为r行c列的长方形阵列，通过（i，j）描述第i行第j列的超单元。

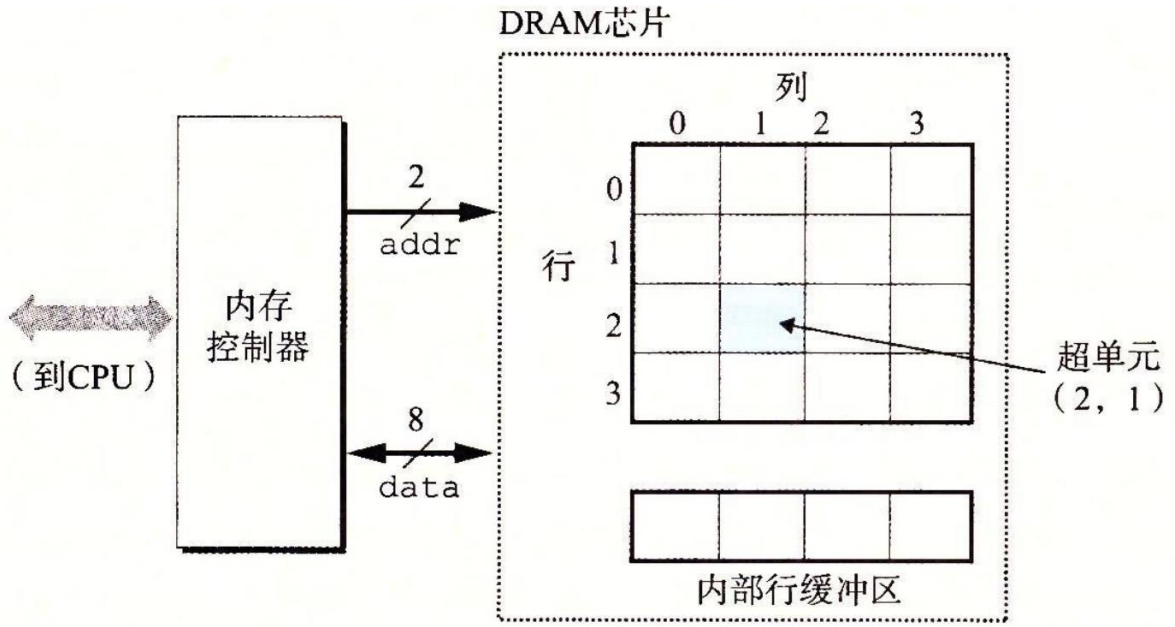


图 6-3　一个 128 位 16×8 的 DRAM 芯片的高级视图

DRAM芯片与内存控制器之间通过引脚连接，每个引脚传递1bit信息。内存控制器分用两根引脚分两次传入行列地址，DRAM取出信息后通过w根引脚将信息传出。

传统DRAM

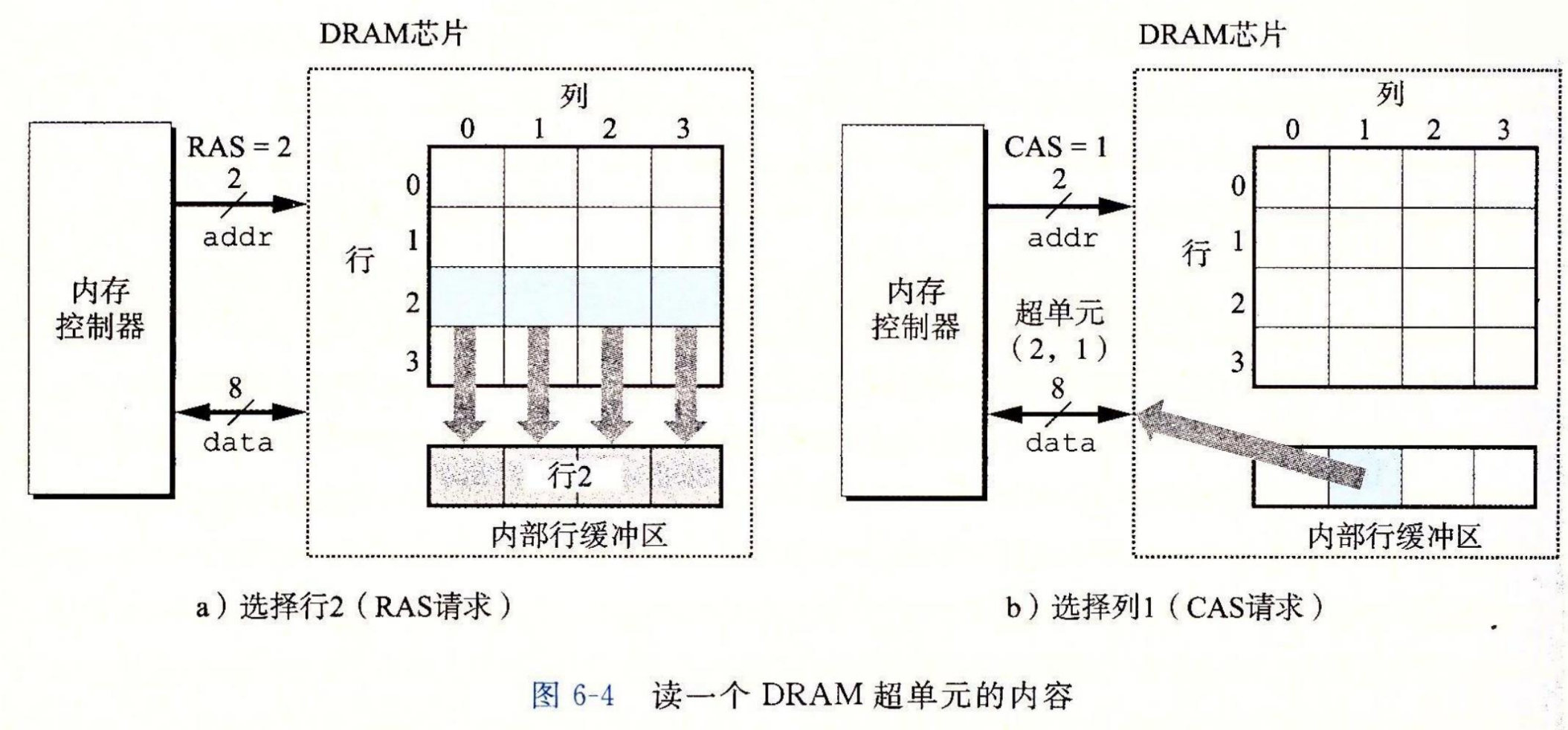


图 6-4　读一个 DRAM 超单元的内容

内存模块

一系列的DRAM芯片封装在内存模块中，插在CPU的扩展槽上，可以一次性传递更多信息。

CPU输入地址A，内存控制器将其转为（i，j）的形式，从DRAM芯片中取出（i，j）对应地址的信息，组合在一起，传回内存控制器，再传回CPU。

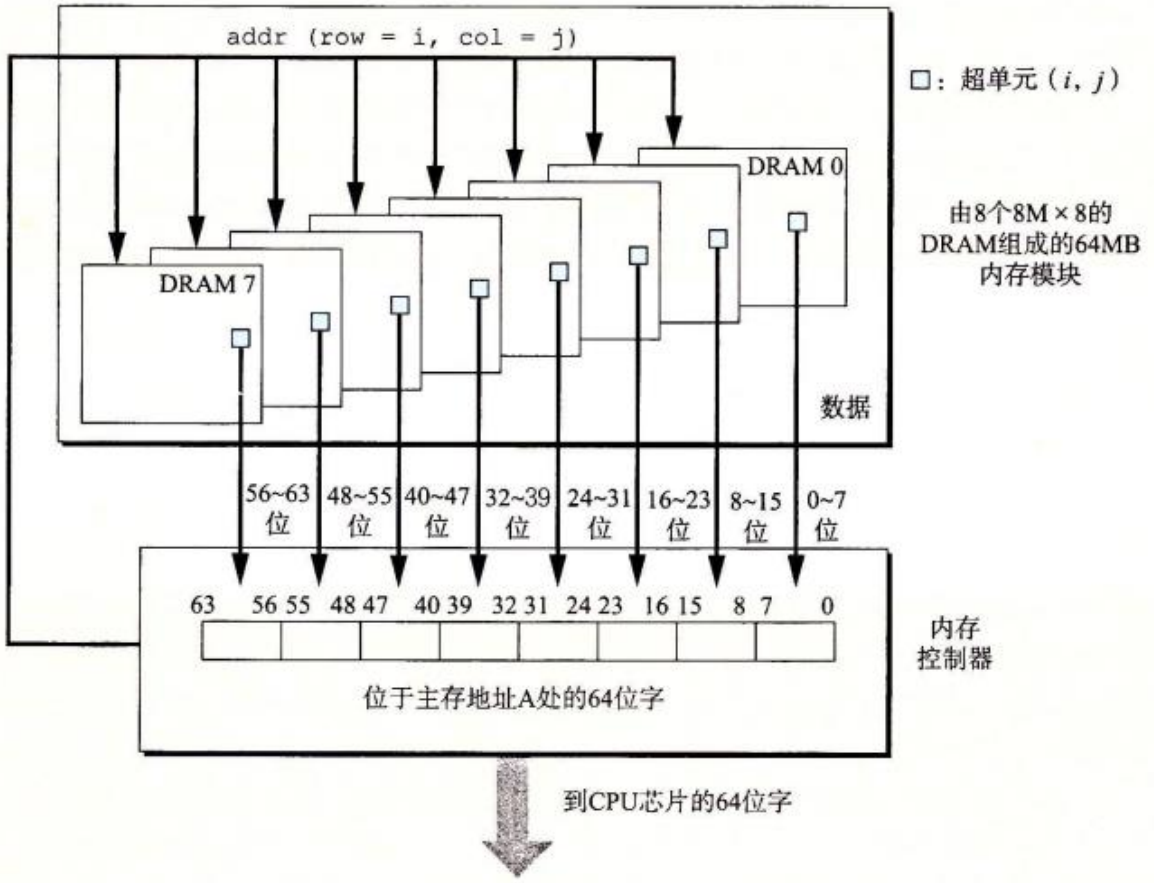


图 6-5 读一个内存模块的内容

多个内存模块连接至内存控制器，聚合为主存。内存控制器可以选择包含地址A的模块并正确地取得数据。

增强DRAM

- 快页模式DRAM（FRM DRAM）：读取同一行的超单元时，请求的次数更少，速度更快。
- 扩展数据输出DRAM（EDO DRAM）：加强版FPM DRAM。
- 同步DRAM（SDRAM）：有一个同步接口，与计算机时钟信号同步，减少读取数据流水线中的等待时间。
- 双倍速率同步DRAM（DDR SDRAM）：加强版SDRAM。
- 视频RAM（VRAM）：用于图形系统的帧缓冲区。通过位移整个内部缓冲区来输出，允许读写并行。

目前，市面上主流CPU大多数采用的都是DDR SDRAM，这项技术已经更新到DDR5，在数据传输速度方面有很大的提升。

非易失性存储器

RAM在断电后会丢失数据，而ROM不会因断电丢失数据，因此称ROM为非易失的。ROM虽然是只读存储器，但不是全都只读。

- 可编程ROM（PROM）：只能被编程1次。

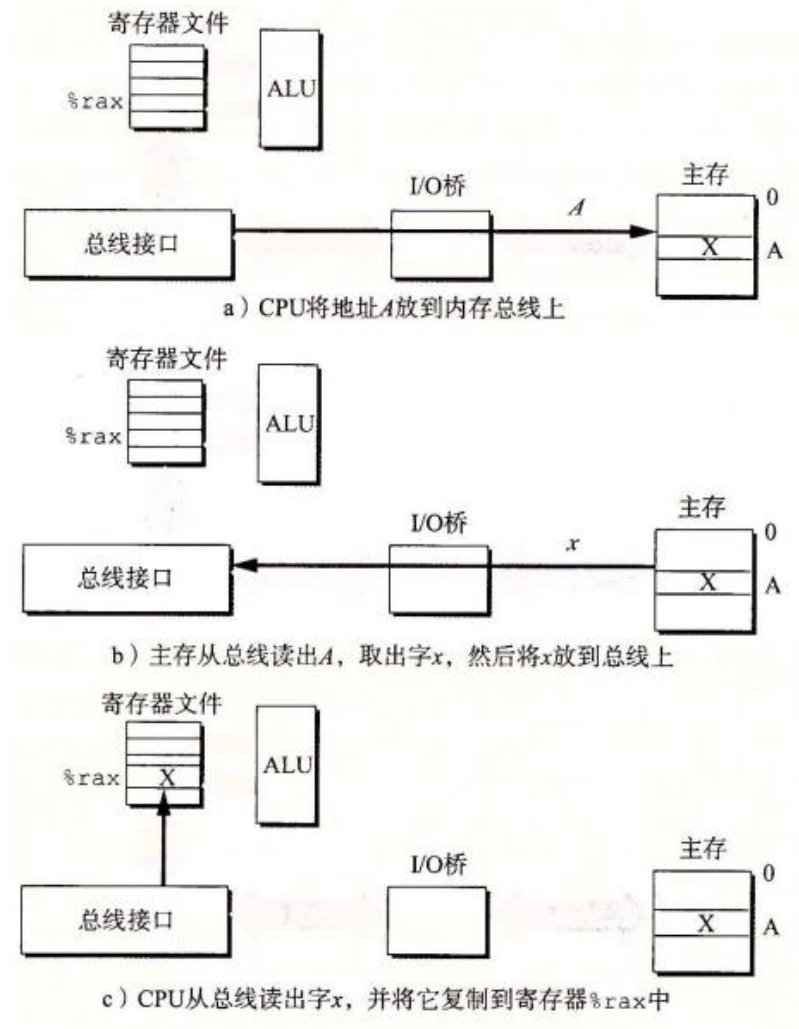
——可擦写可编程ROM（EPROM）：可重编程1000次。

——电子可擦除PROM（EEPROM）：可重编程10^5次。

- 闪存：一种重要的存储技术，基于EEPROM，被广泛使用，是SSD的基础。

ROM中保存的程序称为固件，一般包括BIOS、驱动等。

访问主存



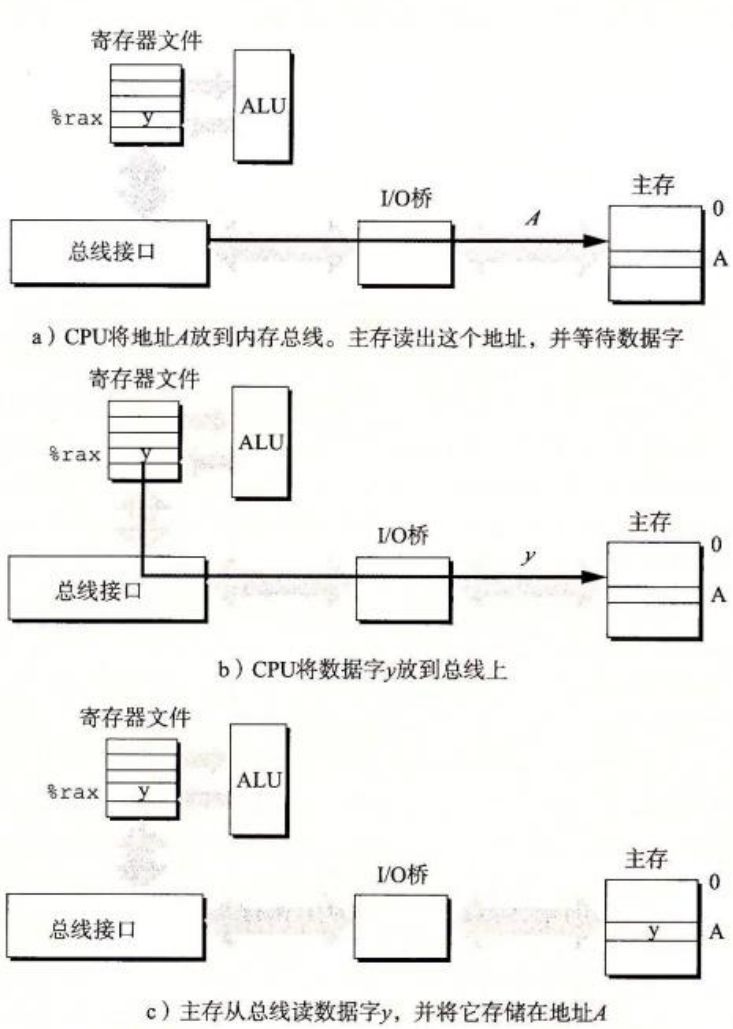
a）CPU将地址*A*放到内存总线上

b）主存从总线读出*A*，取出字*x*，然后将*x*放到总线上

c）CPU从总线读出字*x*，并将它复制到寄存器%rax中

图 6-7　加载操作 movqA,%rax 的内存读事务

读内存



a）CPU将地址*A*放到内存总线。主存读出这个地址，并等待数据字

b）CPU将数据字*y*放到总线上

c）主存从总线读数据字*y*，并将它存储在地址*A*

图 6-8　存储操作 movq % rax,A 的内存写事务

写内存

硬盘储存

(外存)

机械硬盘和固态硬盘

主流的硬盘有两大种类：机械硬盘与固态硬盘。

机械硬盘即磁盘，又称旋转磁盘，数据存储在磁盘的盘片上，通过磁头读写数据。磁盘的价格比固态硬盘便宜，可以长时间保存数据，但读写速度也远比固态硬盘慢。

固态硬盘即SSD，没有移动的部分，数据存储在闪存颗粒上。价格高，读写快，长时间不使用可能导致数据丢失。

磁盘结构

磁盘驱动器

- 主轴
  - 旋转
- 盘片
  - 表面上
  - 磁道（同心圆）
    - 组成 → 柱面
    - 划分
      - 间隙（标识格式化位）
      - 扇区（通常512Bytes）
- 读/写头
  - 读写数据
  - 连接 / 定位
- 传动臂
  - 寻道

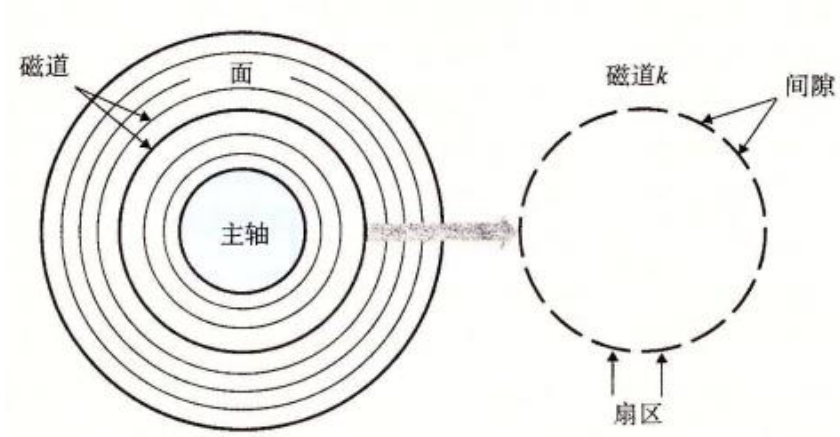


图 6-9　磁盘构造

磁盘容量

磁盘的容量由以下因素决定:

- 记录密度：一英寸磁道容纳的字节数
- 磁道密度：盘片中心出发一英寸的段内拥有磁道数
- 面密度：以上两者的乘积

磁盘容量的计算公式:

$$\text{磁盘容量}=\frac{\text{字节数}}{\text{扇区}}\times\frac{\text{平均扇区数}}{\text{磁道}}\times\frac{\text{磁道数}}{\text{表面}}\times\frac{\text{表面数}}{\text{盘片}}\times\frac{\text{盘片数}}{\text{磁盘}}$$

制造商通过增加面密度来增加容量。现代大容量磁盘使用多区记录的技术来减少间隙部分的浪费。

磁盘操作

读/写头通过传动臂定位到所需的磁道上，而盘面在读/写头下方以一定速度旋转，通过读/写头下方的位可以被读取或更改。

对于多个盘面，每个盘面都有一个独立的读/写头，但它们是一致行动、始终位于同一个柱面上的。

为了读取数据，需要以下步骤：

- 寻道：定位到所需磁道，所需时间为寻道时间。一般3~9ms，最高20ms。
- 旋转：到所需磁道后等待目标扇区转到读/写头下，所需时间为旋转时间。
- 传送：之后只需等所有扇区依次通过，所需时间为传送时间。

总耗时称为访问时间，主要是寻道和旋转费时较长。在找到第一个字节后，不需要太多时间即可访问剩余字节。

寻道时间和旋转延迟大致相等，可以用寻道时间*2来估计磁盘访问时间。

逻辑磁盘块

对于操作系统，磁盘就是一个B个扇区大小的逻辑块的序列。磁盘中的一个设备磁盘控制器，被用作实际扇区与逻辑块之间的桥梁，处理两者之间的转换。

OS发送读取某逻辑块的命令，控制器收到后进行快速表查找，将逻辑块号翻译为（盘面，磁道，扇区）的三元组，然后磁盘将对应位置的数据传输到控制器的缓冲区中，最后由控制器将数据复制到主存。

连接I/O设备

磁盘和鼠标、键盘等输入/输出设备，都要通过I/O总线连接到CPU和内存。I/O总线比系统总线和内存总线慢，但是与底层CPU无关，兼容性强。I/O总线可以连接以下几类设备：

- USB（通用串行总线）控制器：连接到USB总线。USB总线应用十分广泛，可以连接各种设备。
- 图形卡（适配器）：包含硬件和软件逻辑，负责代表CPU在显示器上画像素。可以连接监视器。
- 主机总线适配器：连接一个或多个磁盘。常用的磁盘接口有SCSI和SATA，前者更快更贵且可支持多个磁盘。
- 其他：可以通过主板扩展槽连接，如网络适配器。

连接I/O设备

I/O总线 连接
- USB控制器 连接 USB总线 连接 各种设备
- 图形卡 连接 监视器等
- 主机总线适配器 连接 磁盘
- 扩展槽 连接 其他

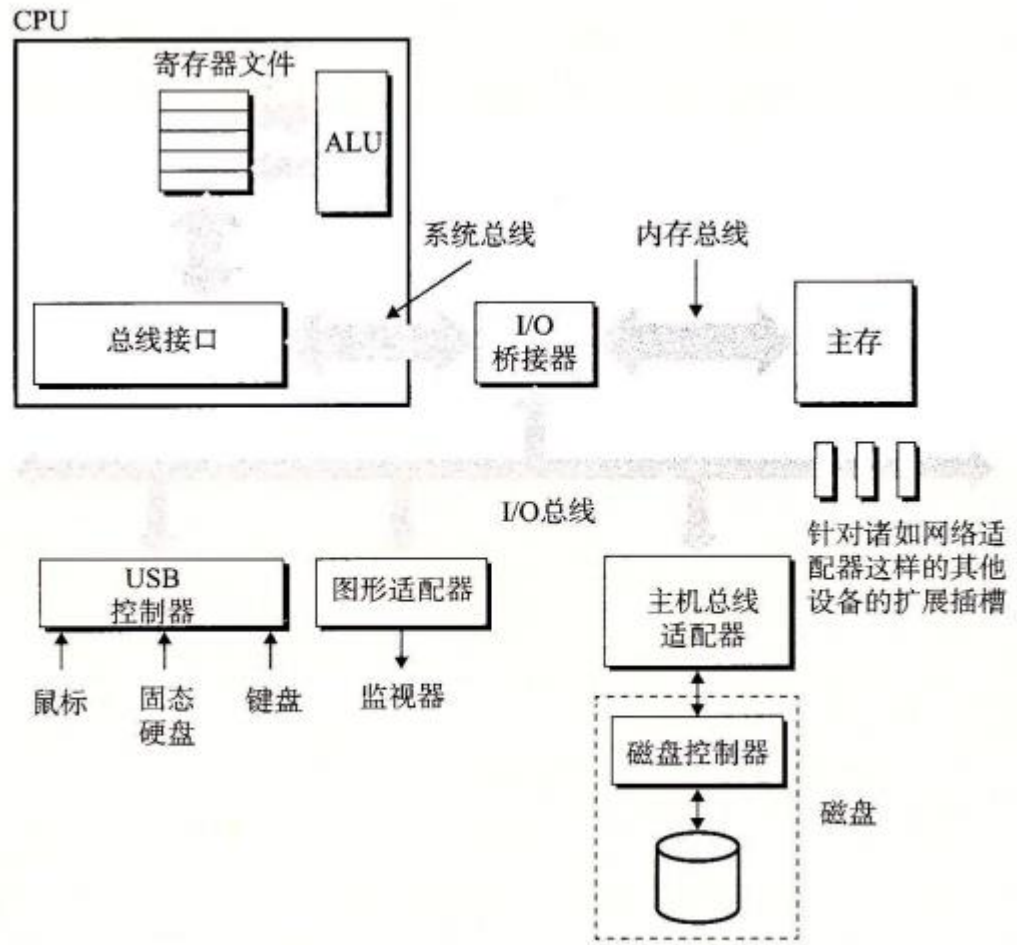
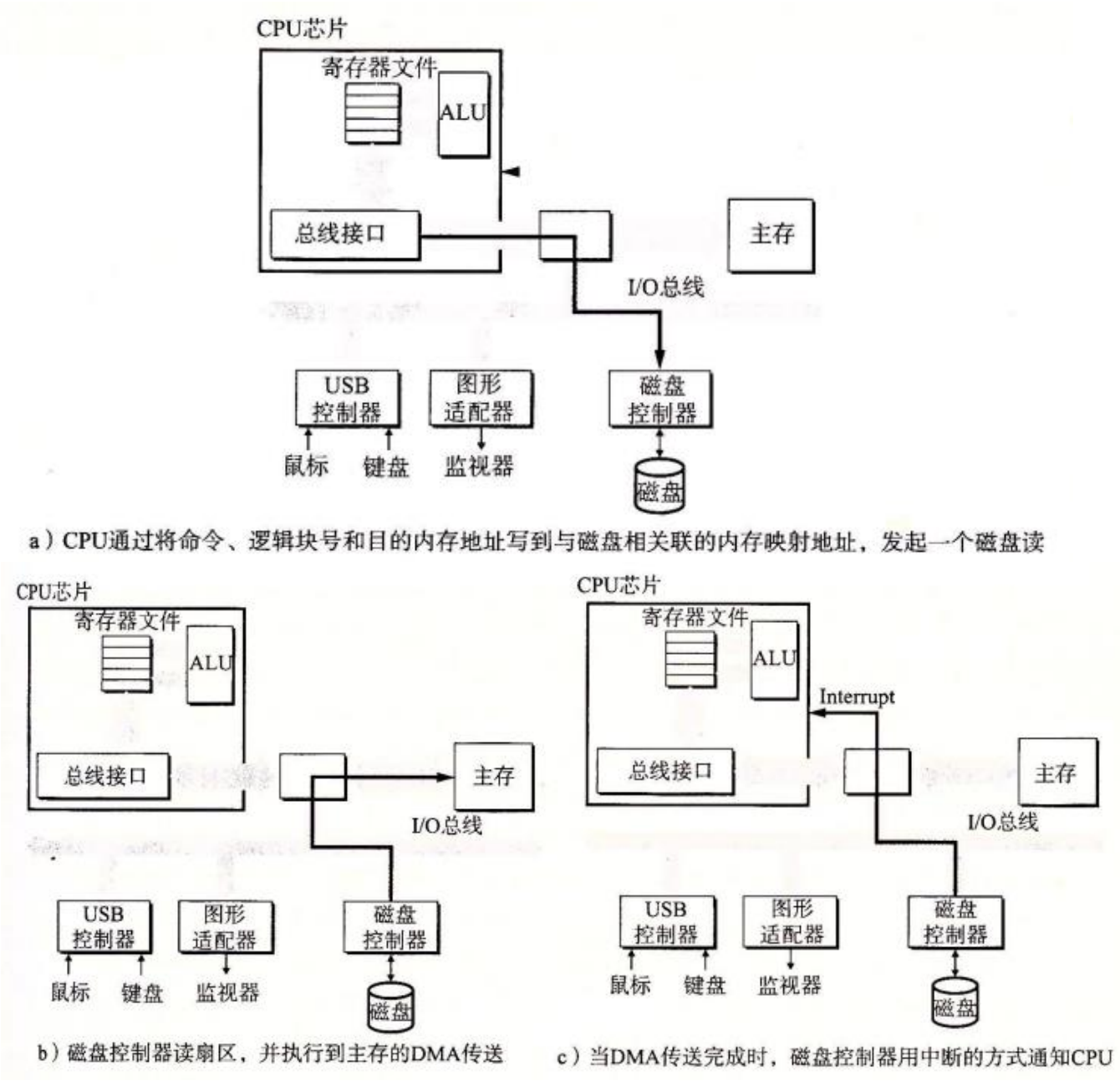


图 6-11 总线结构示例，它连接 CPU、主存和 I/O 设备

访问磁盘

内存映射：地址空间对每个I/O设备预留一个或多个地址，称为I/O端口。对I/O设备操作时只需对端口的地址操作。

DMA传送：磁盘自行将数据传入主存，而不需要CPU干涉的过程。使用DMA传送可以使CPU在漫长的数据传输时间内做别的工作，只需在传输完成时通知CPU即可。



固态硬盘

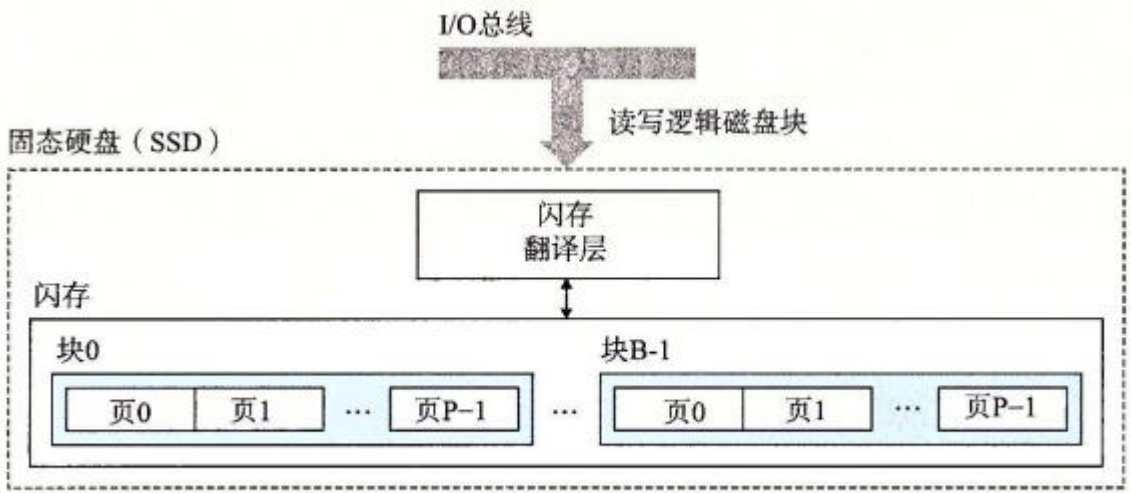


图 6-13　固态硬盘(SSD)

类比：
闪存翻译层 ~ 磁盘控制器
闪存芯片 ~ 磁盘驱动器
块 ~ 逻辑块
页 ~ 扇区

特性：
数据以页为单位读写，只有在一页所属的整个块被擦除后才能重写（一个块被擦除后其中的页可以直接重写）。
在进行大约100000次重复写后块会磨损坏，无法再次使用。
随机写速度慢。因为擦除整个块需要较长时间；且如果目标页已有数据，需要将整个块复制到其他块，再擦除目标页所在块，最后才能重写。

优点：没有移动部件，随机访问快，能耗低，结实。
缺点：会磨损，价格贵。

16. 以下关于存储器的说法中，错误的是：

A. 对于旋转磁盘(rotating disk)，可以通过提高旋转速率(rotational speed)的方式减少旋转时间(rotational latency)和传送时间(transfer time)，从而降低访问时间(access time)。

B. 在CPU向磁盘控制器(disk controller)发送读取数据请求后，磁盘控制器会读取指定扇区的内容，并将其传送到内存的指定位置，这一传送过程不需要CPU参与，称为DMA(direct memory access)传送。

C. 对于同一个SSD，读取速度通常比写入速度快。

D. SRAM是非易失性(nonvolatile)存储器，其数据在断电后不会丢失。

D

局部性和存储器层次结构

局部性原理

计算机倾向于引用之前引用过的数据，或是它附近的数据，这种倾向被称为局部性原理，前者为时间局部性，后者为空间局部性。一个具有良好局部性的程序运行速度要比局部性差的程序快。

计算机广泛运用局部性来提高运行速度。

在硬件上利用局部性，是通过引入高速缓存存储器，保存最近引用的数据和指令，可以提高对主存的访问速度。

在操作系统层面，系统使用虚拟地址空间来作为高速缓存。

在程序设计上，将网络文档保存在本地以供再次使用等。

数据引用的局部性

多次引用同一个变量，就具有时间局部性。

顺序读取内存中的变量，就具有空间局部性。

空间局部性的好坏和引用步长有关。步长越长，程序的空间局部性越差。

可以通过尽可能缩短步长来提高程序运行效率。

```
1   int sumarrayrows(int a[M][N])
2   {
3       int i, j, sum = 0;
4
5       for (i = 0; i < M; i++)
6           for (j = 0; j < N; j++)
7               sum += a[i][j];
8       return sum;
9   }
```

a）另一个具有良好局部性的程序

地址	0	4	8	12	16	20
内容	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
访问顺序	1	2	3	4	5	6

b）数组a的引用模式（$M=2$，$N=3$）

图 6-18　有良好的空间局部性，是因为数组是按照与它存储在内存中一样的行优先顺序来被访问的

```
1   int sumarraycols(int a[M][N])
2   {
3       int i, j, sum = 0;
4
5       for (j = 0; j < N; j++)
6           for (i = 0; i < M; i++)
7               sum += a[i][j];
8       return sum;
9   }
```

a）一个空间局部性很差的程序

地址	0	4	8	12	16	20
内容	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
访问顺序	1	3	5	2	4	6

b）数组a的引用模式（$M=2$，$N=3$）

图 6-19　函数的空间局部性很差，这是因为它使用步长为 N 的引用模式来扫描

取指令的局部性

指令也是放在内存中的，因此取指令也具有局部性。

按顺序读取内存中的指令，就具有良好的空间局部性。

多次执行同一段指令，就具有良好的时间局部性。

与数据引用相比，指令很少会被修改。

```
1    int sumvec(int v[N])
2    {
3        int i, sum = 0;
4
5        for (i = 0; i < N; i++)
6            sum += v[i];
7        return sum;
8    }
```

a）一个具有良好局部性的程序

存储器层次结构

不同存储技术访问时间差异较大，以及计算机软件倾向于有良好局部性。这两者相互结合，自然而然地便产生了这种金字塔形的存储器层次结构。

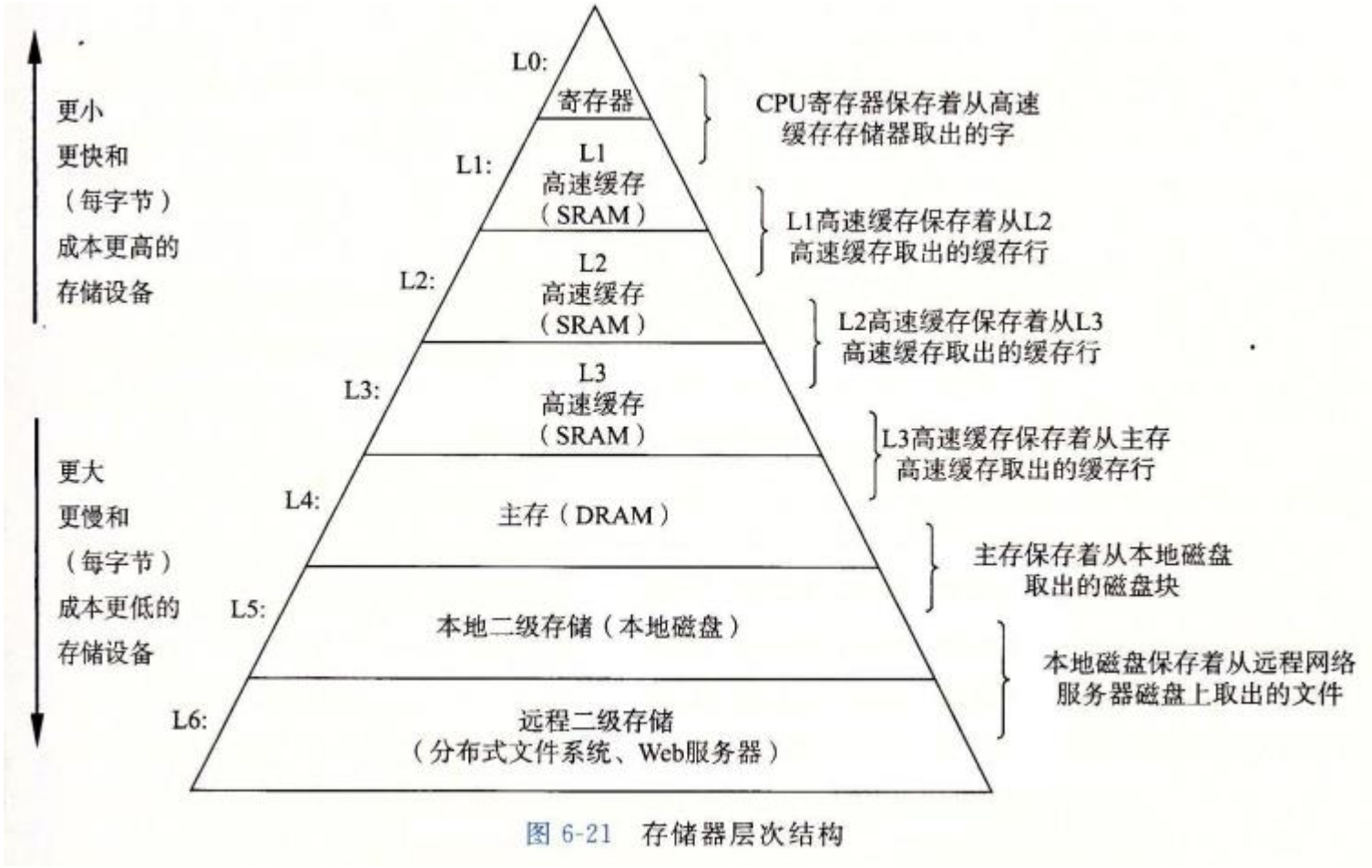


图 6-21　存储器层次结构

各层次速度对比

- L1 cache reference： 0.5ns
- L2 cache reference： 7ns
- Main memory reference：100ns
- Read 4KB randomly from SSD：150，000ns
- Read 1MB sequentially from SSD:1,000,000ns
- Read 1MB sequentially from Disk:20,000,000ns

以L1为参照，设L1使用了1s，那么按照这个标准从磁盘读取1MB数据需要花费1年多!

存储器层次结构中的缓存

存储器层次结构的中心思想是，对于每个k，位于第k层的存储设备作为第k+1层设备的缓存，每一层都缓存低一层的数据对象。

第k+1层的存储器都划分为若干的块，每个块有连续的地址或名字，大小可以固定也可以可变。第k层划分为较少的块，缓存有第k+1层块的子集的副本。

块的大小在相邻两层次是固定的，不同层次之间块的大小不同，一般离CPU越远，块越大。

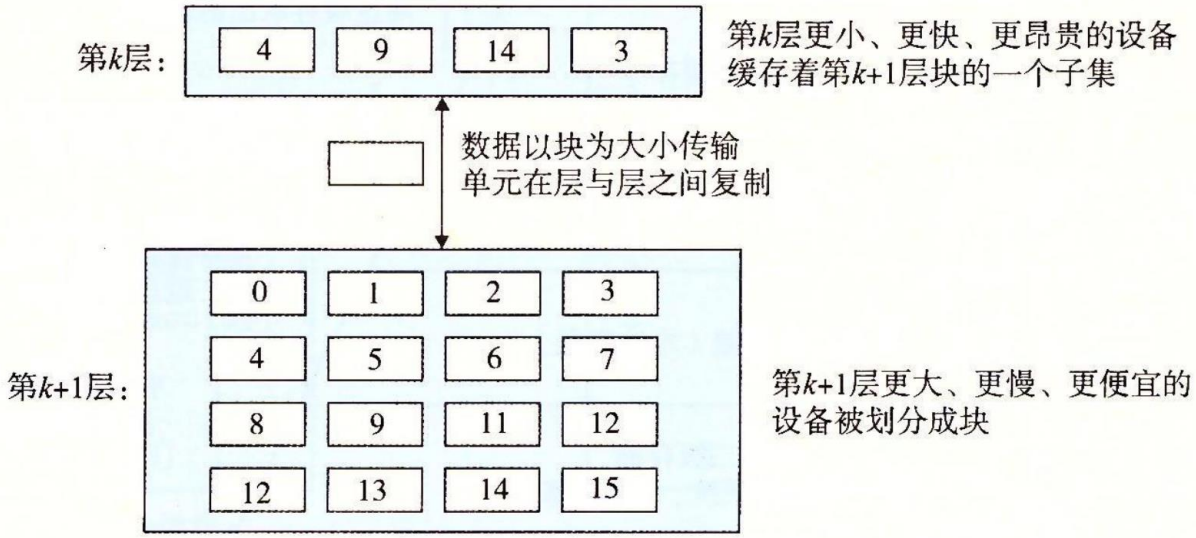


图 6-22　存储器层次结构中基本的缓存原理

- 缓存命中：程序需要第k+1层的数据d时，现在第k层查找，如果找到了，则为缓存命中，可直接从第k层读取d。局部性良好的程序更容易做到缓存命中。
- 缓存不命中：如果第k层没有找到d，则称为缓存不命中。此时第k层从第k+1层取出d，如果第k层满了，就可能覆盖当前存在的块，覆盖的块由缓存的替换策略决定（如随机替换、选择最后访问时间最远的块等）。

缓存不命中有多种

- 冷不命中（强制性不命中）：当第k层缓存为空时发生。
- 冲突不命中：发生不命中时执行某种放置策略来决定取出的块的放置位置。由限制性的放置策略会引起这种不命中。
- 容量不命中：工作集超过容量大小时发生。

管理缓存

对于每一层的存储器，都需要某种东西管理缓存，即负责将缓存划分成块，在不同层次间传递块，判定是否命中，并处理它们。管理缓存的逻辑可以是硬件、软件或两者结合

例如，编译器管理寄存器文件，缓存层次结构的最高层。它决定当发生不命中时何时发射加载，以及确定哪个寄存器来存放数据。L1、L2 和 L3 层的缓存完全是由内置在缓存中的硬件逻辑来管理的。在一个有虚拟内存的系统中，DRAM 主存作为存储在磁盘上的数据块的缓存，是由操作系统软件和 CPU 上的地址翻译硬件共同管理的。对于一个具有像 AFS 这样的分布式文件系统的机器来说，本地磁盘作为缓存，它是由运行在本地机器上的 AFS 客户端进程管理的。在大多数时候，缓存都是自动运行的，不需要程序采取特殊的或显式的行动。