

计算机系统存储

[零 参考资料 2](#_Toc44857735)

[一 存储的层次结构 2](#_Toc44857736)

[1 概述 2](#_Toc44857737)

[2 Random-Access Memory(RAM) 2](#_Toc44857738)

[2.1 概述 2](#_Toc44857739)

[2.2 Static Random-Access Memory(SRAM) 2](#_Toc44857740)

[2.3 Dynamic Random-Access Memory(DRAM) 3](#_Toc44857741)

[2.4 读取指定spuercell内容 3](#_Toc44857742)

[2.5 从内存模块中读取 4](#_Toc44857743)

[2.6 访问主存 5](#_Toc44857744)

[3 Disk Storage 5](#_Toc44857745)

[3.1 概述 5](#_Toc44857746)

[3.2 访问Disk 6](#_Toc44857747)

[3.3 Solid State Disk(SSD) 7](#_Toc44857748)

[4 Cache Memories 8](#_Toc44857749)

[二 操作系统中的Memory 8](#_Toc44857750)

[三 LINUX中的Memory 8](#_Toc44857751)

[四 Virtual Memory 8](#_Toc44857752)

[五 操作系统中的Virtual Memory 8](#_Toc44857753)

[六 LINUX中的Virtual Memory 8](#_Toc44857754)

# 零 参考资料

《深入理解计算机系统 第三版》

《操作系统概念 第九版》

《深入理解Linux内核 第三版》

PS：

# 存储的层次结构

## 概述





## Random-Access Memory(RAM)

### 概述

* + - 1. 又两类：static（SRAM）、dynamic(DRAM)
         1. SRAM：速度快、价格贵、容量小、用于高速缓存，
         2. DRAM：相对SRAM，速度慢，价格便宜，容量大，用于主存和图形系统的frame buffer
      2. 整体比较

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | transistors/bit | 相对访问时间 | Persistent | sensitive | 相对成本 | 应用 |
| SRAM | 6 | 1 | Y | N | 1000 | cache memory |
| DRAM | 1 | 10 | N | Y | 1 | 主存，frame buffer |

### Static Random-Access Memory(SRAM)

* + - 1. 将每bit存储在一个bitstable存储器单元中
         1. 每个单元用一个six-transistor circuit实现
         2. 可以无限期地保持在两个不同的电压配置或状态之一
         3. 其他任何状态都是不稳定的，类似下图的inverted pendulum

当摆钟倾斜在最左或最右时，是stable的

原则上也能在垂直位置无期限地保持平衡，但此时状态是metastable



* + - 1. 只要有电，SRAM存储单元就会永远保持它地值
      2. 对干扰不是很敏感，即使有干扰，在干扰消除时，电路就会恢复稳定值

### Dynamic Random-Access Memory(DRAM)

* + - 1. 用电容器上的电荷存储每bit,每个电容只有大约30 femtofarads——30X10-15farad
      2. 对干扰很敏感，当电容电压受到干扰，就会永远丢失数据：
         1. 比如暴露在光线下
         2. 数码相机和摄像机中的传感器本质上就是DRAM单元阵列
      3. 很多原因导致DRAM单元在10-100毫秒时间内失去电荷：
         1. 计算机运行时钟周期是纳秒级，相对保持时间较长
         2. 计算机系统必须周期性通过读&写刷新每bit
         3. 有些系统还是用error-correcting codes，可以用来发现并纠正erroneous bit
      4. 一般DRAM chip：
         1. cell(bit)被分组成d个supercell
         2. 每个supercell 由w个cell(bit)组成
         3. 所有supercell被组成r行c列的矩形阵列
         4. 每个supercell 都有一个(i,j)地址，i表示行，j表示列

### 读取指定spuercell内容

* + - 1. 每个DRAM芯片连接到Memory controller，一次可以跟每个DRAM chip交互w bit数据
      2. 信息通过pin进出DRAM chip，每个pin带1bit信号
      3. 实际读取supercell流程，以从下图chip中读取supercell(2,1)为例：
         1. Memory controller 发送RAS(Row Access Strobe)=2到chip
         2. chip复制行2所有数据复制到Internal row buffer
         3. Memory controller 发送CAS(Column Access Strobe)=1到chip
         4. chip从internal row buffer复制出指定supercell中所有bit并发送给Memory controller
      4. DRAM之所以是二维阵列，而不是线性数组，是为了降低chip上的address pin数量，但因为必须分两次发送地址，增加了访问时间



### 从内存模块中读取

* + - 1. DRAM chip封装在memory modules中，插到main system board上的expansion slot中
      2. 下图展示了，内存模块的基本思路：
         1. 模块用了由8个 8\*8M的DRAM组成的64MB Memory module，8个DRAM编号0-7
         2. 每个supercell存储main memory的1byte
         3. main memory 中地址A数据由，用依序读取DRAM中supercelle（i，j）的信息组成(64 bit word)
      3. 读取地址A过程示例：
         1. 请求获取地址A数据
         2. Memory controller 将地址A转换成supercell(i,j)
         3. 将supercell(i,j)发送给memory module
         4. memory module广播supercell到每个DRAM
         5. 每个DRAM输出对应的supercell内容
         6. 合并成各个DRAM输出的内容，返回给Memory controller
         7. Memory controller返回地址A数据
      4. 通过将多个memory module 连接到memory controller 聚合成main memory，这时，当controller收到地址A时，只会将A转成supercell(i,j)发送给包含A的memory module



### 访问主存

* + - 1. 数据流通过Bus在processor和DRAM main memory之间交互
      2. 执行 movq A,%rax：
         1. CPU通过 bus insterface 发起 read transcation
         2. CPU将地址A放到System bus上，
         3. 通过I/O bridge将信号从System bus转换并发送到Memory bus
         4. Main memory 扫描到memory bus上的地址A
         5. 根据地址A，从DRAM读取数据并写到memory bus
         6. 通过I/O bridge将信号从Memory bus 转换并发送到System bus
         7. CPU从System bus获取地址A数据，并复制到register中



## Disk Storage

### 概述

* + - 1. disk 结构
         1. disk由一个或多个platter叠放在一起组成：

每个platter 有两个覆盖着magnetic recording material的surface

platter中间有个使platter以固定rotational rate旋转的spindle

* + - * 1. 一个surface由一组track组成，

每个track被划分为一组sector，

每个sector中包含相同数量的编码在magnetic recording material上的data bits

sector之间由gap隔开，这些gap不存储数据，存储的是identify sector的formatting bits

所有surface上的跟spindle距离相同的track集合称为cylinder

在multiple zone recording技术中，cylinder被分成不相交的子集，称为recording zone

recording zone 中每个cylinder的每个track都有相同数量的sector，sector数量由该zone最里面track所能包含的sector数量决定

* + - 1. disk 容量计算
         1. recording desity(bits/in):track上1英寸的segment中可以放入的bits数量
         2. track desity(tracks/in)：从platter 中心开始，半径上1英寸的segment上track数量
         3. areal density(bit/in2)：recording desity 乘以 track desity
         4. 容量公式：

 

### 访问Disk

* + - 1. disk用read/write head 读写存储在magnetic surface的bits：
         1. read/write head连接到arm末端
         2. 通过半径轴前后移动arm，可以将read/write head定位到任何track上，即seek
         3. 当read/write head定位到指定track上，可以读取/修改通过head下面的每bit
         4. 当有多个platter时：

每个surface都有独立read/write head

所有read/write head垂直排列

一致行动，任何时刻都在同一个cylinder上

* + - 1. 当read/write head遇到异物时，会停下来
      2. Seek Time:
         1. 为读取目标sector的内容，arm先将read/write head定位到sector所在的track
         2. 依赖于read/write head之前的位置和arm在surface上移动的速度
      3. Rotational latency：
         1. 当read/write head定位到目标track，等待目标sector的第一个bit旋转到read/write head下，所需时间
         2. 依赖于当read/write head到达指定track时所在surface位置以及rotational speed
         3. 最坏的情况，read/write head正好错过了目标sector，必须等待disk转一圈
         4. 最大rotational latency：
         5. 平均rotational:
      4. Transfer time:
         1. 当read/write head到达目标 sector的第一个bit时，读写目标sector内容所需时间
         2. 依赖于rotational speed和每条track上sector数量
         3. 粗略估算平均时间：
      5. 访问数据所需时间估算：
      6. 读取数据流程：
         1. 假设设备支持DMA(Direct Memory Access)
         2. 流程图



### Solid State Disk(SSD)



## Cache Memories

# 操作系统中的Memory

# LINUX中的Memory

# Virtual Memory

# 操作系统中的Virtual Memory

# LINUX中的Virtual Memory