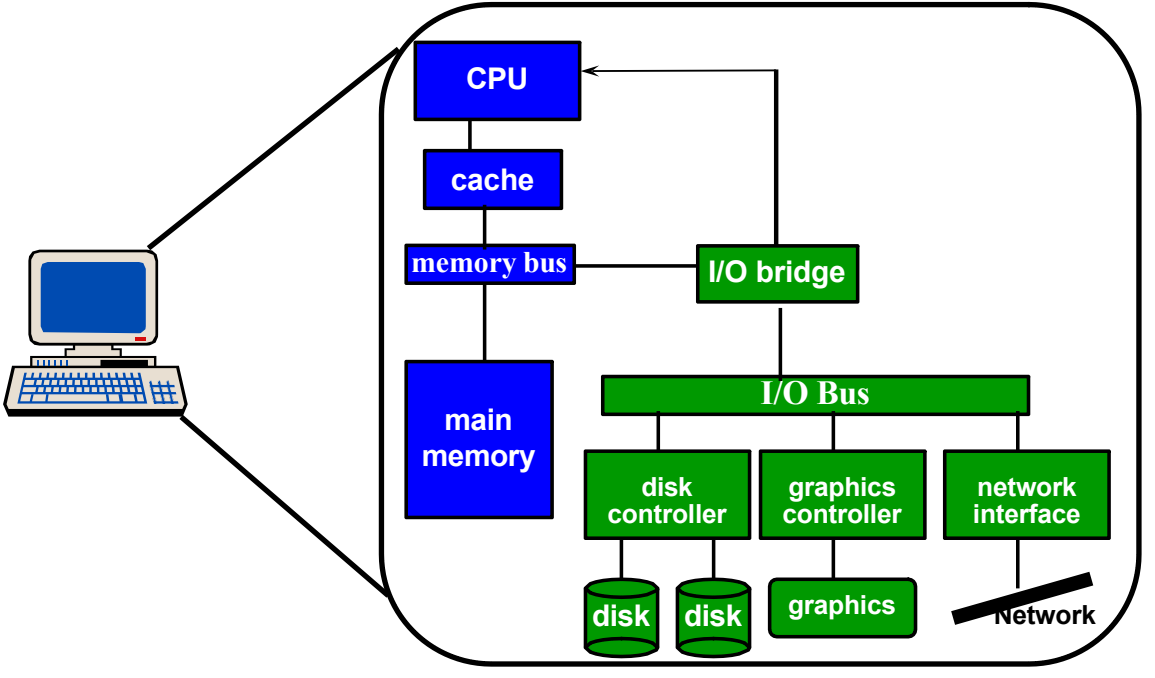
Computer Organization and Design

Memory Hierarchy
大容量与高速度：开发存储器层次结构

XM Guo

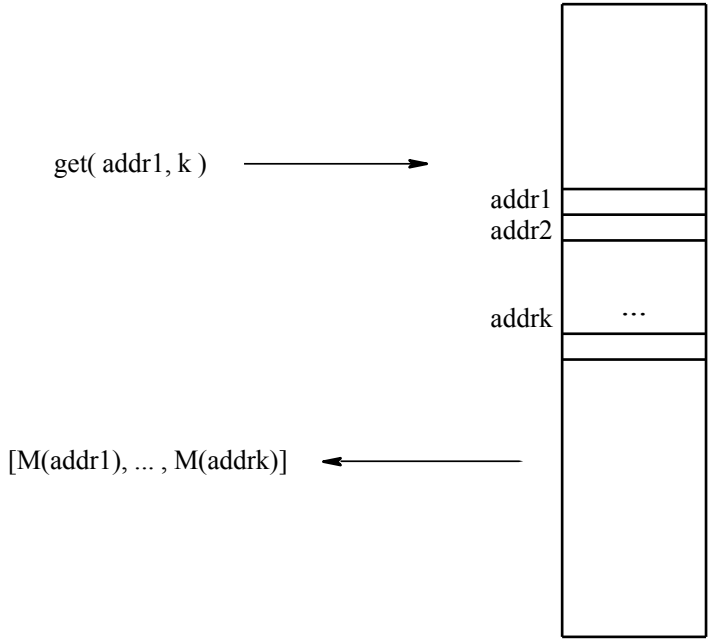
2023

System Organization



The Memory Abstraction

从概念上讲，内存是一个大的字节数组，可以通过指定起始地址和字节计数从程序中访问。



主存储器的存储单元

位是二进制数的最基本单位，也是存储器存储信息的最小单位。

当一个二进制数作为一个整体存入或取出时，这个数称为**存储字**。

存放存储字或存储字节的主存空间称为存储单元或主存单元，大量存储单元的集合构成一个**存储体**。

为了区别存储体中的各个存储单元，必须将它们逐一编号。存储单元的编号称为地址，地址和存储单元之间有一对一的对应关系。

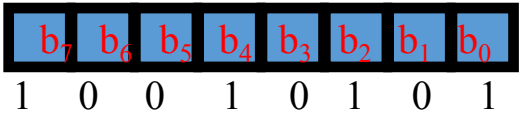
所以,存储单元是CPU访问存储器的最小单位。每个存储单元可以是一个字,也可以是一个字节。

主存储器的主要技术指标

1.存储容量：字（字节）

位（Bit）：度量数据的最小单位

字节（Byte）：最常用的基本单位



K 字节 1K = 1024 Byte

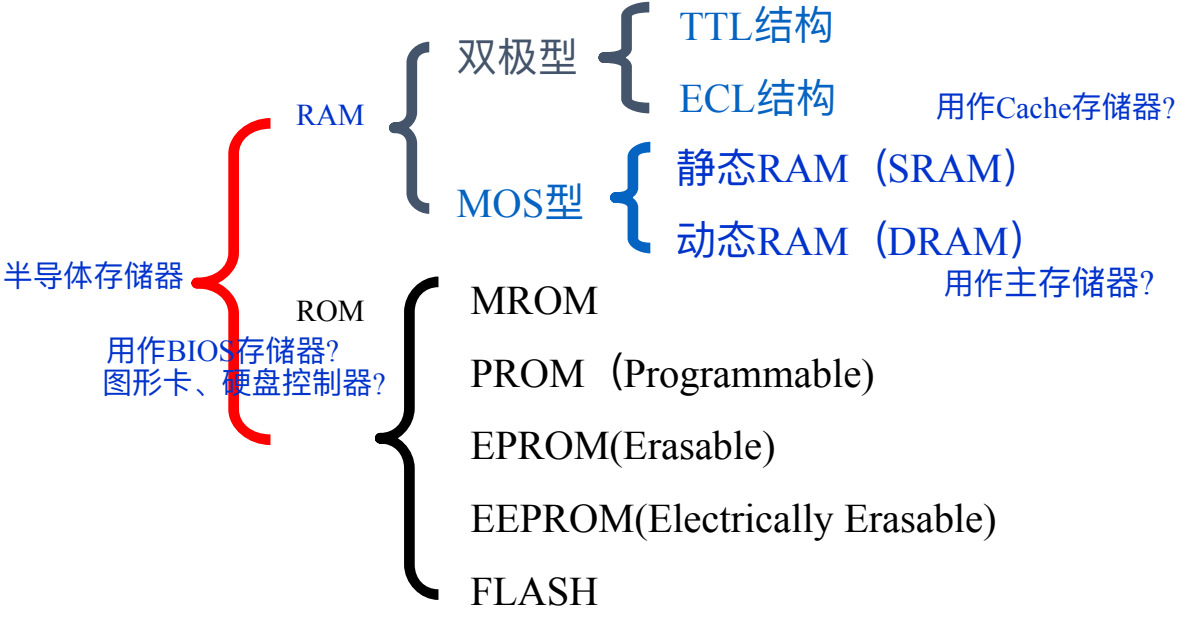
M（兆）字节 1M = 1024 KB

G（吉）字节 1G = 1024 MB

2.存储速度：有三个指标

- 访问时间T_A — 是指从向存储器发出命令，到从存储器读出信息所需的时间。
- 访问周期T_M — 是指连续两次访问存储器的最小时间间隔。 通常：$T_M > T_A$。
- 存储器带宽B_M — 是指连续访问时，存储器每秒钟所能提供的数据传送速率。其单位为：字节/秒、字/秒、位/秒。

半导体随机存取存储器(RAM)



主存储器是整个存储系统的核心，由RAM和ROM构成, 并且是二者缺一不可。

RAM用来存放供用户随机读写的用户程序和数据，也可以作为系统程序的工作区，ROM用来存放系统程序。

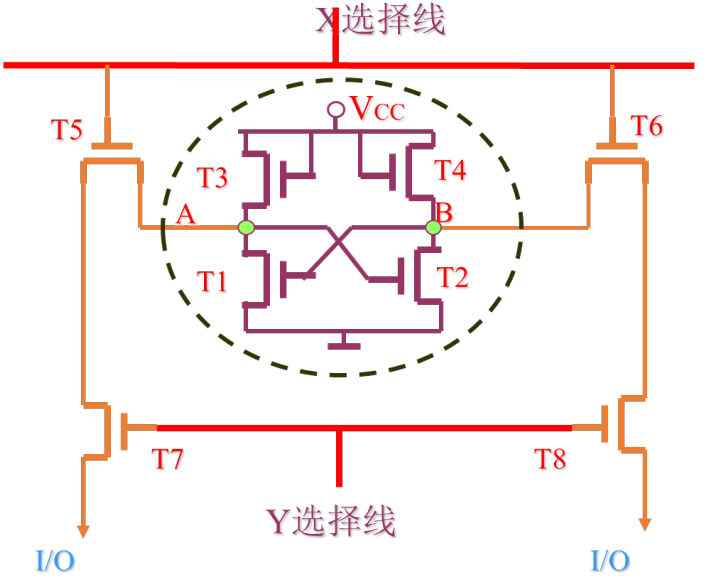
本节从最基本的存储电路开始，分别介绍RAM和ROM的不同类型存储电路、工作原理、和各自所具有的特点,以及相应存储芯片的外特性。

RAM的存储元电路

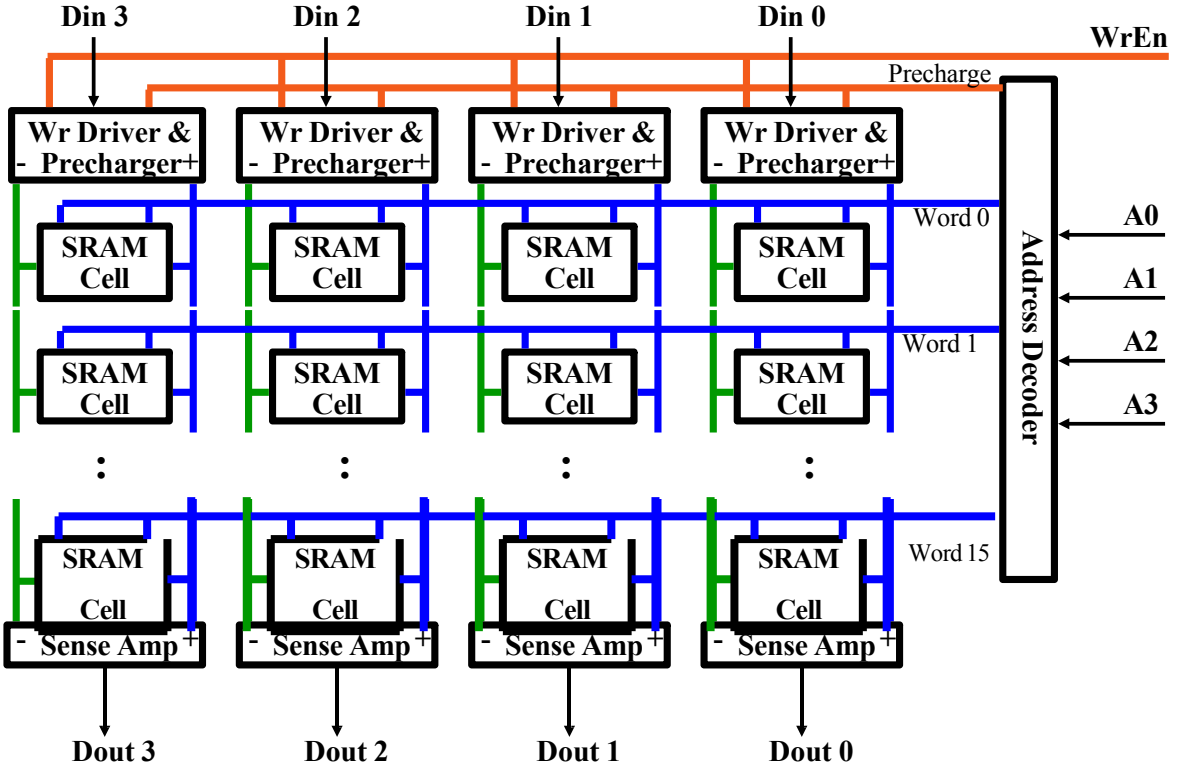
可以存放一个二进制位的电路被称为存储元，
它是构成存储器的最小信息单位。

1. SRAM存储元电路

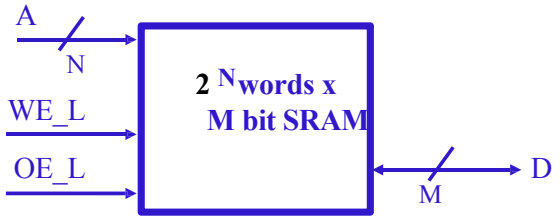
SRAM是用双稳态触发器结构记忆信息的。SRAM的存取速度快，但集成度低，功耗也较大，所以一般用来组成高速缓冲存储器和小容量主存系统。



Typical SRAM Organization: 16-word x 4-bit

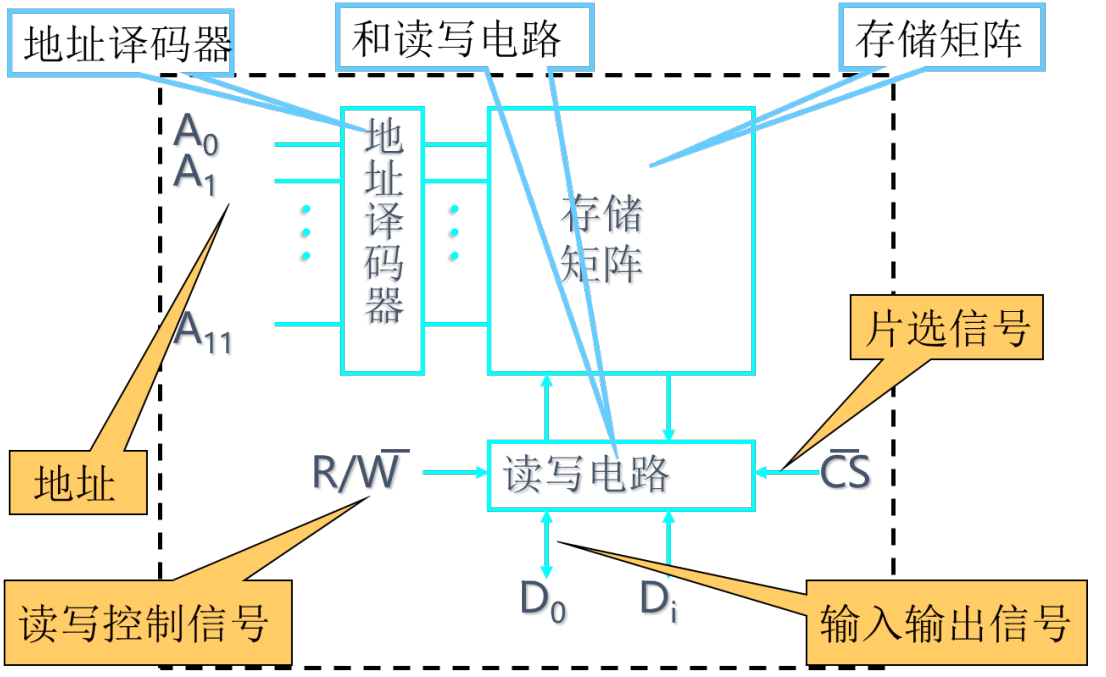


Logic Diagram of a Typical SRAM



- 写使能低有效 (WE_L)
- Din 和 Dout 用一根线 (D) 节约引脚:
  - 增加输出允许信号(OE_L)
  - WE_L 为低 (Low), OE_L 为高 (High)
    - D 作为数据输入引脚
  - WE_L 为高 (High), OE_L 为低 (Low)
    - D 作为数据输出引脚
  - 两个信号 WE_L and OE_L 都为低不允许:
    - Result is unknown. Don’t do that!!!

SRAM的基本结构

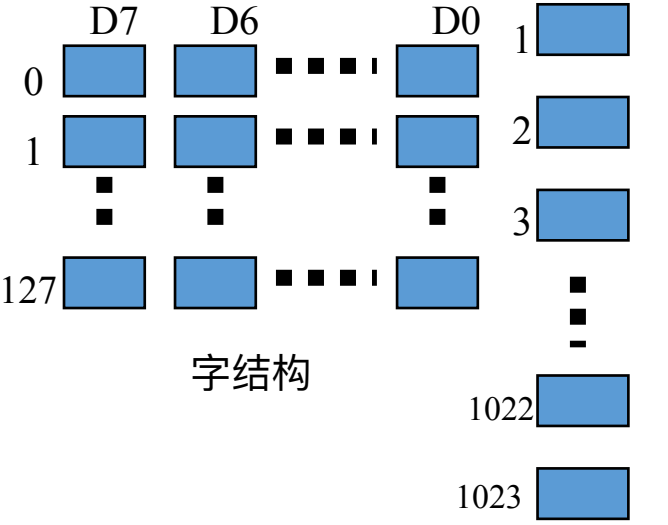

地址译码器
和读写电路
存储矩阵
A0
A1
A11
地址译码器
存储矩阵
片选信号
地址
R/W
读写电路
CS
D0
Di
读写控制信号
输入输出信号

存储矩阵

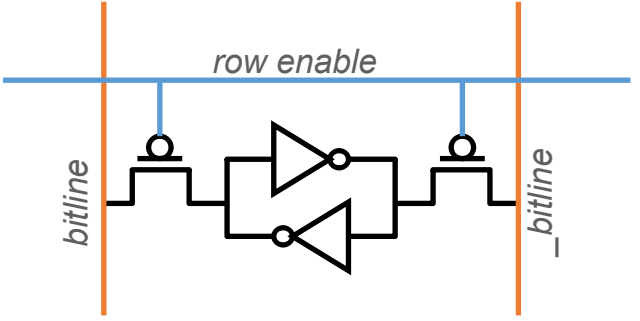
存储器中需要将许多基本单元电路按一定的顺序排列成阵列形式，这样的阵列称为存储矩阵.排列方式:**字结构**和**位结构**.

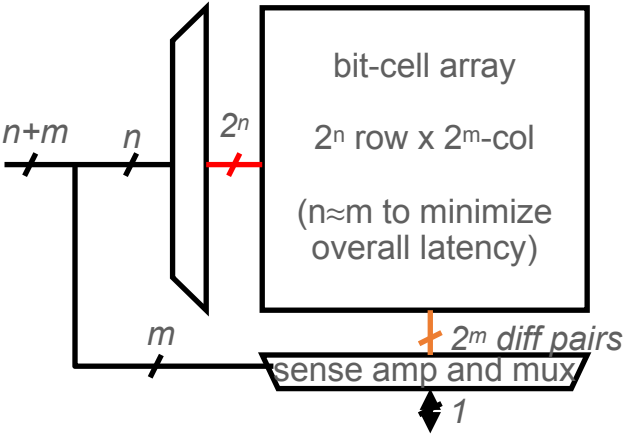
字结构:同一芯片存放一个字的多位，如8位。

- 位结构:同一芯片存放多个字的同一位.



SRAM延时





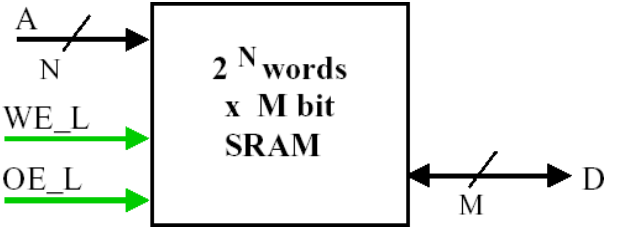
读过程

1. 地址译码
2. 驱动行选
3. 选中的位线单元驱动位线
   (整个行一起读取)
4. 列选
   (数据准备就绪)
5. 位线准备下次读或写

访问延迟主要在第2、3和5步

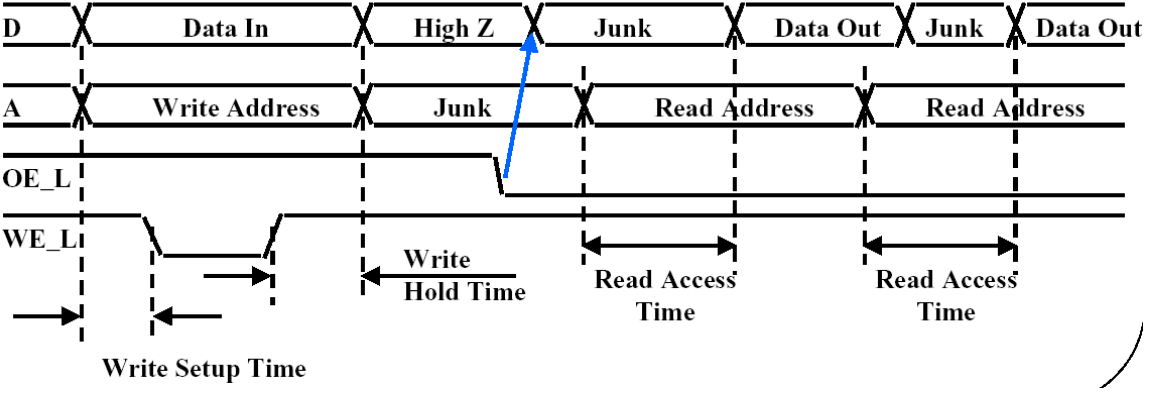
- step 2 与 2^m成正比
- step 3 and 5 与 2^n成正比

Typical SRAM Timing



Write Timing:

Read Timing:

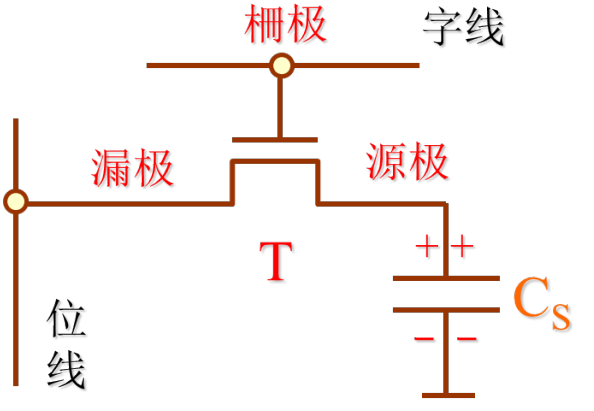


Dynamic RAM Cell DRAM

慢, 便宜,比SRAM集成度高

通过电容 C_S 有无存电荷区分信号 0、1

DRAM存储元电路



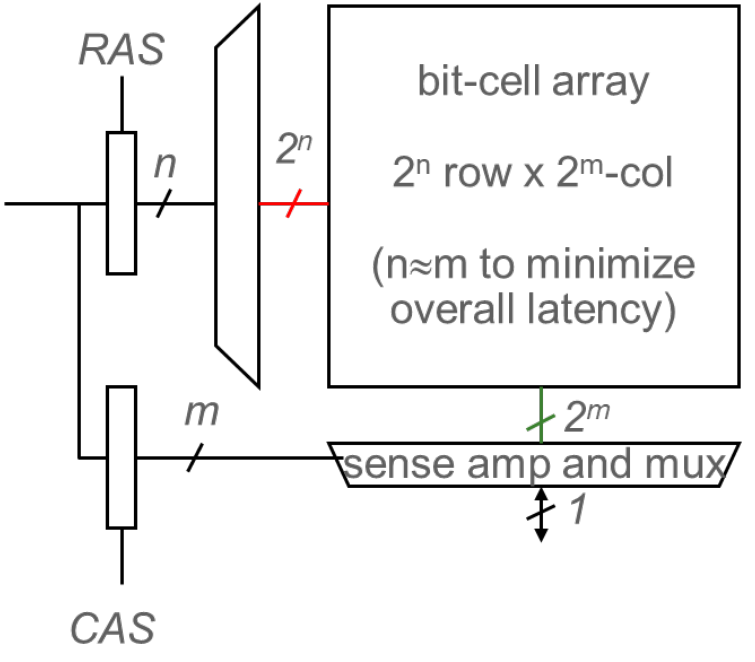
读出时数据线有电流为 “1”

读出时数据线无电流为 “0”

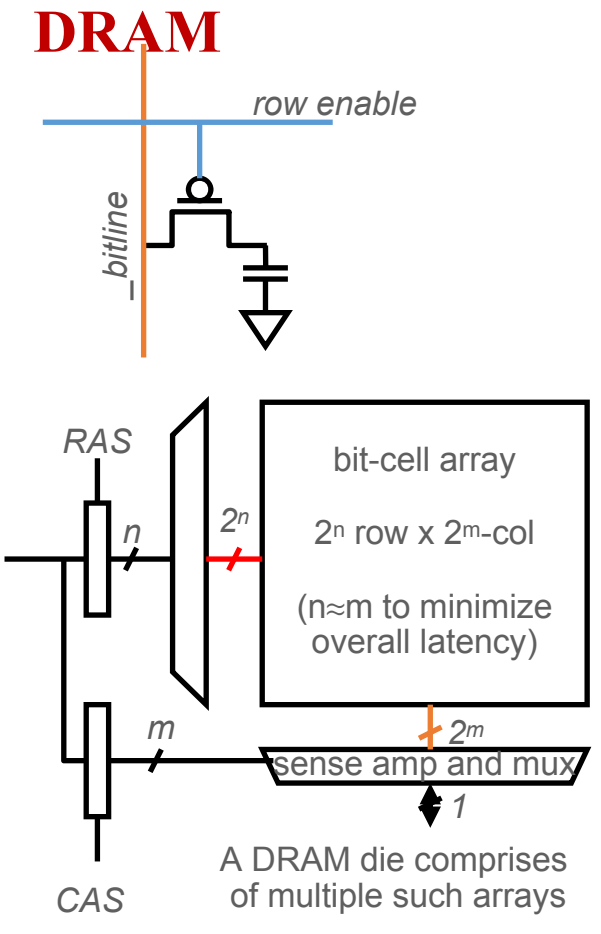
动态 RAM 芯片结构

由于DRAM芯片集成度高，容量大，为了减少芯片引脚数量，行地址和列地址共用地址线，行地址由行地址选通信号 $\overline{\text{RAS}}$ 送入存储芯片，列地址由列地址选通信号 $\overline{\text{CAS}}$ 送入存储芯片。

由于采用了地址复用技术，因此，DRAM芯片每增加一条地址线，实际上是增加了两位地址，也即增加了4倍的容量。



DRAM芯片把地址线分成两部分，行地址、列地址



电容存电荷（1）或失去电荷（0）表示数据位（非恢复性）

位单元在读取时会失去电荷

位单元随时间也会失去电荷

读取过程

1~3 与静态随机存储器一样

4. 触发器放大电路重新产生位线，数据通过选择器输出

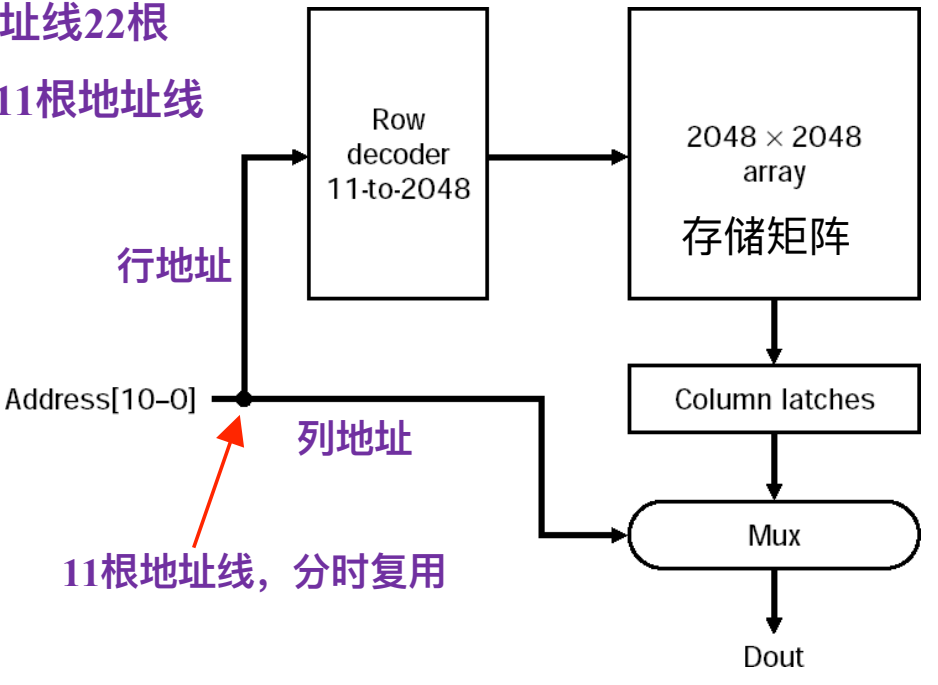
5. 预充电所有位线

刷新: DRAM 控制器通过读取行再写入进行定期刷新 (10s of ms)

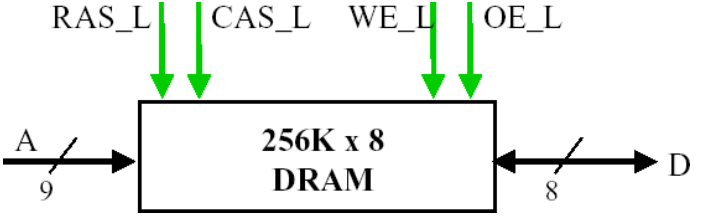
4Mx1 DRAM Organization

4M=2^22,需地址线22根

芯片上，22/2=11根地址线

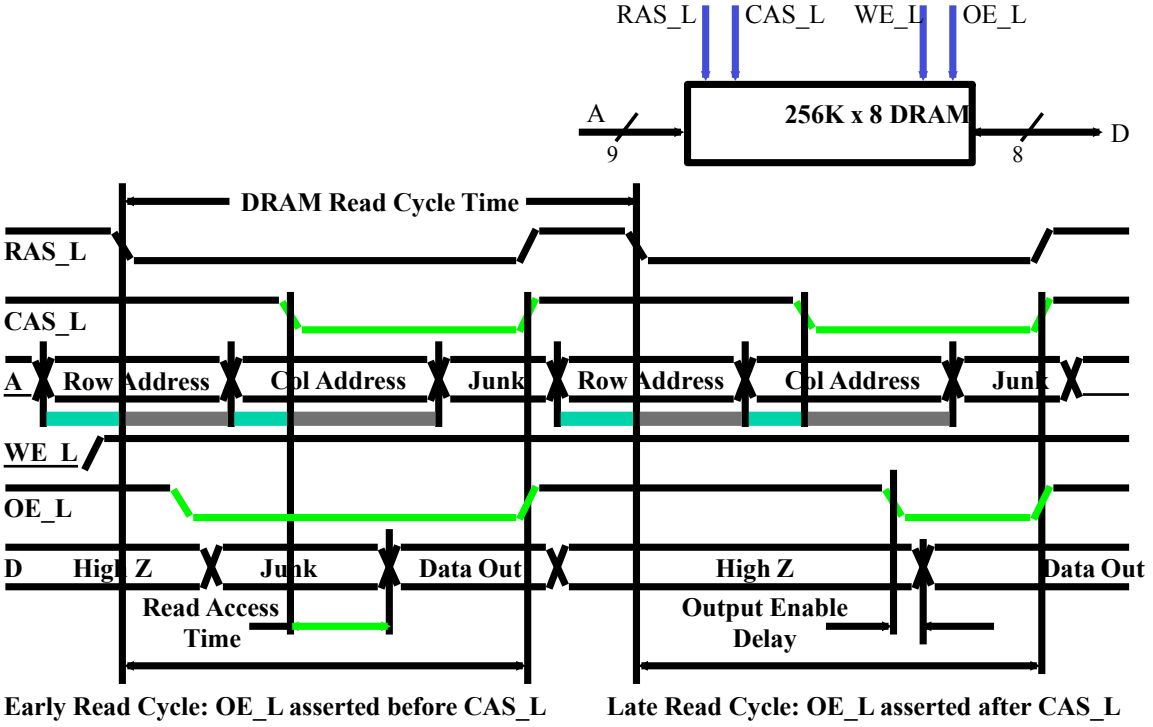


Logic Diagram of a Typical DRAM

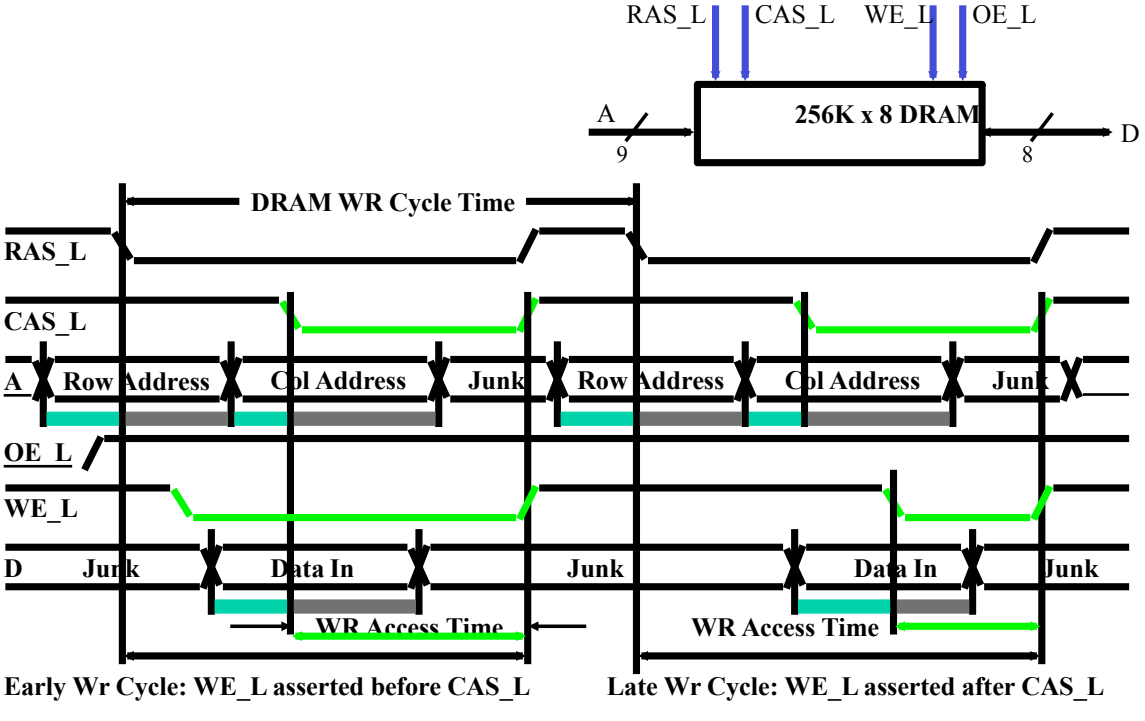


- 行和列地址共享地址线引脚 (A)
  - RAS_L 行选变低: 引脚 A 锁存行地址
  - CAS_L goes low列选变低: 引脚 A 锁存列地址
  - RAS/CAS edge-sensitive边沿触发
- Din 和 Dout 数据输入和输出用同一引脚(D)
- 控制信号 (RAS_L, CAS_L, WE_L, OE_L) 低有效

DRAM Read Timing DRAM读时序图



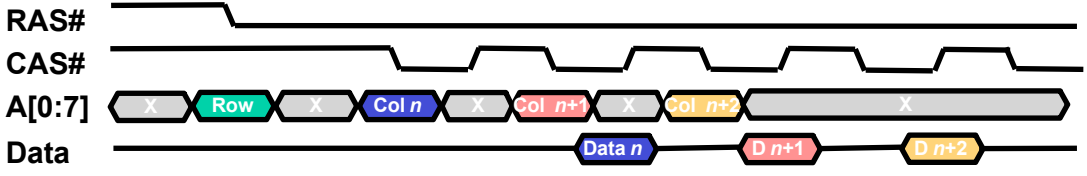
DRAM Write Timing DRAM写时序图



DRAM的类型

- 除前述的基本DRAM外，
- 为了提高DRAM的访问速度，出现了**快速页模式**DRAM(**FPM DRAM**—Fast Page Mode DRAM)、
- 扩展数据输出DRAM (**EDO DRAM**--Extended Data Output DRAM)、
- 爆发式扩展数据输出DRAM(**BEDO** DRAM--Burst Extended Data Output DRAM)和
- 同步DRAM (**SDRAM**--Synchronous DRAM)。

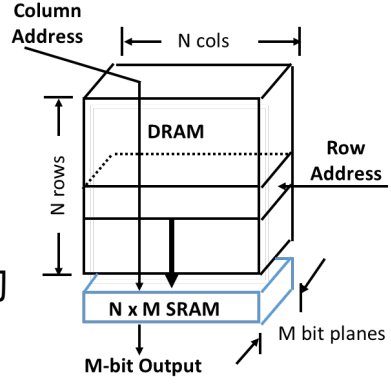
快速页面模式- FPM DRAM



一次行选，整个行被存到一个静态寄存器中，

其后可输入多个列地址，它们和行地址分别组

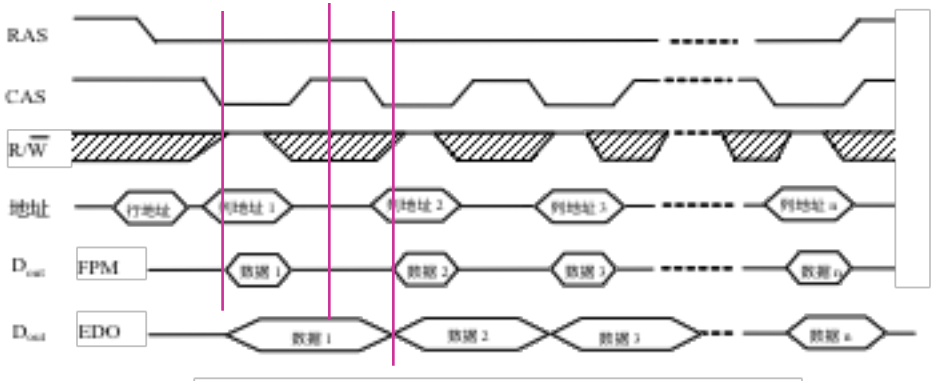
成全地址，选中字存储单元并进行读或写操作



- 当一行被读到 **SRAM “register”静态寄存器中后**
  - 只有列选 CAS 被需要，用来访问其它的在那行的N个字（M-bit）
  - 在 列选CAS 切换时， 行选RAS一直 保持有效

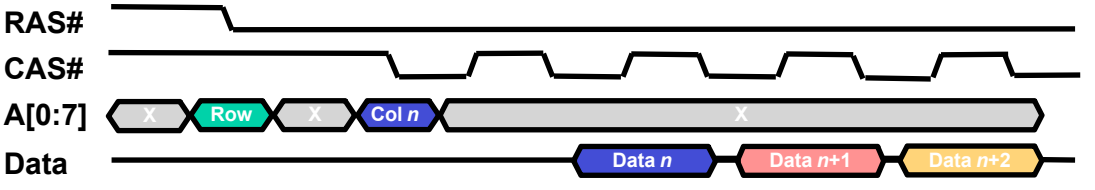
扩展数据输出DRAM（EDO DRAM）

可以扩展输出数据的有效时间，直到CAS再次有效为止，如图的最后一行波形



以读操作为例，操作时序如图。注意，在FPM　DRAM中，当列地址选通信号CAS无效时，没有输出数据，见图的倒数第二行波形。

Burst DRAM 爆发式动态随机存储器

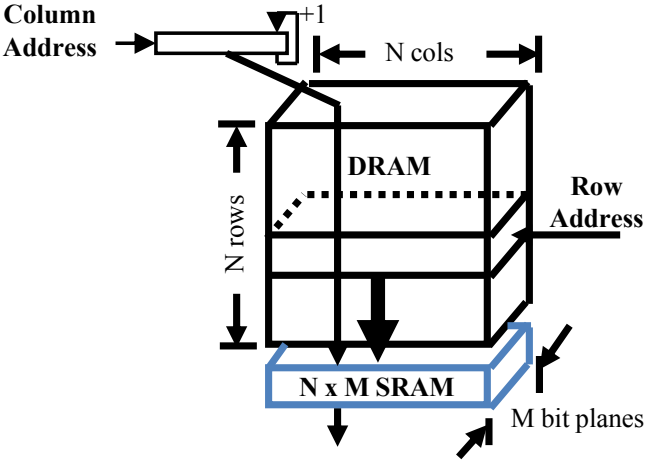


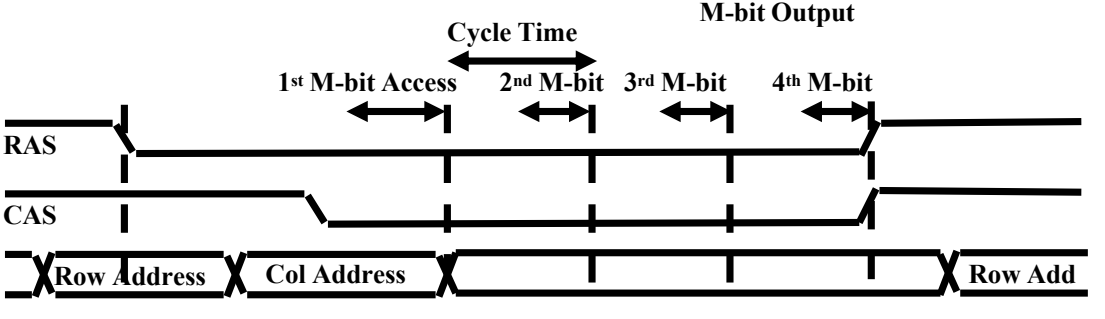
自己生成一个连续的地址

地址突发功能允许从一个外部地址内部最多生成四个连续地址，节省了内存访问时间。

同步动态随机存储器(SDRAM) 的操作

- 当一行被读到静态随机寄存器（SRAM register）后
  - 输入列选 CAS作为起始“爆发式”地址以及爆发长度
  - 从该行中的一系列顺序地址爆发传输数据
    - 时钟控制爆发中连续字的传输– 300MHz in 2004





内存技术的意义

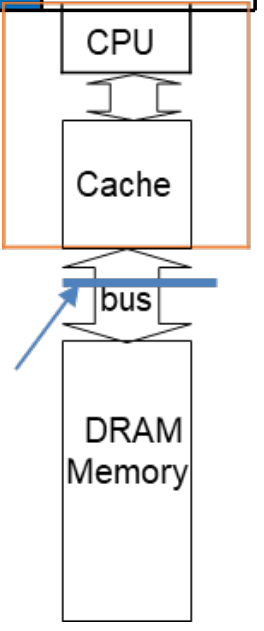
- 使用当前 DRAM 技术的单字访问内存延迟会很慢
- 我们必须改进带宽(bandwidth)/吞吐率
  - Idea 1: 一次访问多个字 (to exploit spatial locality)
  - 技术: Fast Page Mode, DDR SDRAM, etc.
  - Idea 2: 增加并行服务的访问数量
  - Technology:多个内存模块交叉访问（interleaving memory accesses）

Simple Main Memory

Send address to MM	1 clock
MM (DRAM) Access Time for first word	2 clocks
Transfer time for one word	1 clock

4-word cycle = 16 cycles
How to improve?
Lower latency?
A,B,T are fixed

- **Higher bandwidth?**



Cycle	Addr	Mem	steady
1	12	S	*
2		A	*
3		A	*
4		T	*
5	13	S	*
6		A	*
7		A	*
8		T	*
9	14	S	*
10		A	*
11		A	*
12		T	*
13	15	S	*
14		A	*
15		A	*
16		T	*

S-送内存地址

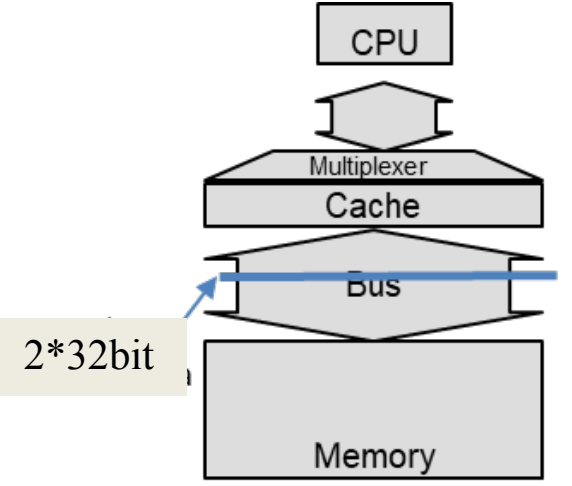
A-内存字访问时间（读或写）

T-传送字时间

Bandwidth: Wider DRAMs

Cycle	Addr	Mem	steady
1	12	S	*
2		A	*
3		A	*
4		T	*
5	14	S	*
6		A	*
7		A	*
8		T	*

- 64-bit DRAM instead
- 4-word cycle = (1+2+1)*2=8 cycles
- 64-bit buses are more expensive (Pentium vs. 486)

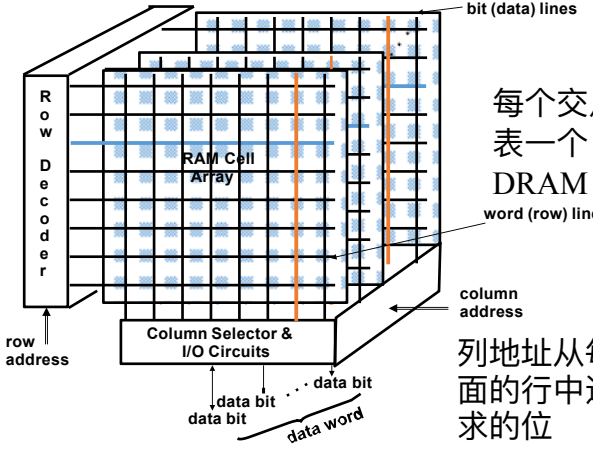


S-送内存地址一个时钟， A-内存访问（4个字）时间2个时钟周期

T-传送时间一个时钟周

Memory Module Interleaving

- 使用了两个或更多兼容（最好相同）的内存模块.
- 在一个内存模块中，多个芯片“并行”使用”.
- 例如。 8个模块，每个模块内“并行”使用8个芯片。 达到一个 $8 \times 8 = 64$ 位内存总线.
- 内存交错可以通过“双通道内存架构”等技术实现



每个交点代表一个 1-T DRAM 单元

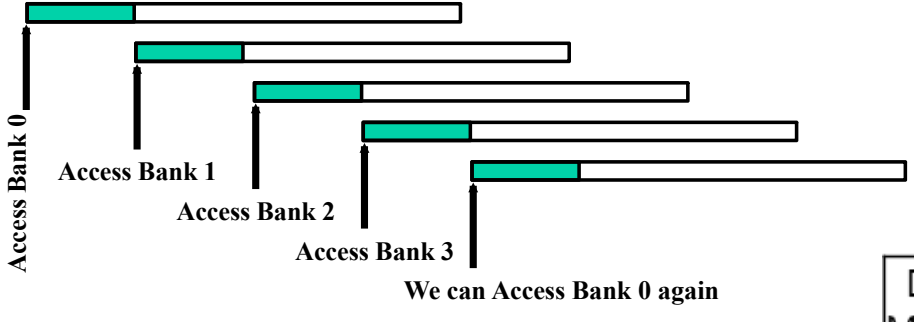
列地址从每个平面的行中选择请求的位

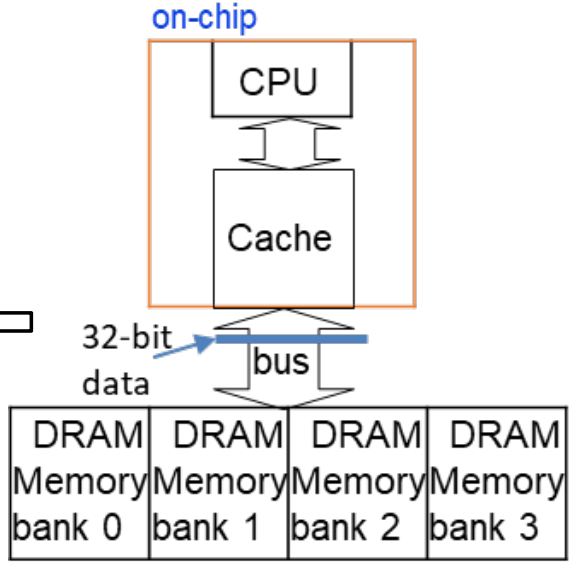
Bandwidth: Interleaving/Banking

使用多个DRAMs模块, 扩展它们的总带宽

- 每个 DRAM 模块叫做一个 bank
- M个 32-bit 的 banks（每个模块32字）
- 简单的交错(Simple interleaving): banks 共享地址线，连续字在不同模块。

Access Pattern with 4-way Interleaving:





4个模块“并行”使用。 达到一个 4 × 32 = 128 位内存总线

Simple Interleaving

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12	S				
2		A	S			
3		A	A	S		*
4		T	A	A	S	*
5			T	A	A	*
6				T	A	*
7					T	*

Parallel access

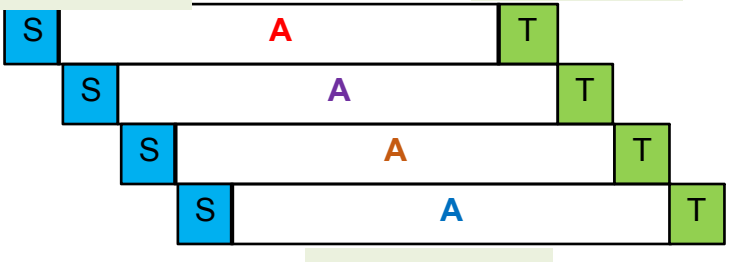
Access time: 2 cycles (A)

Transfer time: 1 cycle (T)

– 仍然使用 32 位总线!

向4个BANK 依次送内存地址

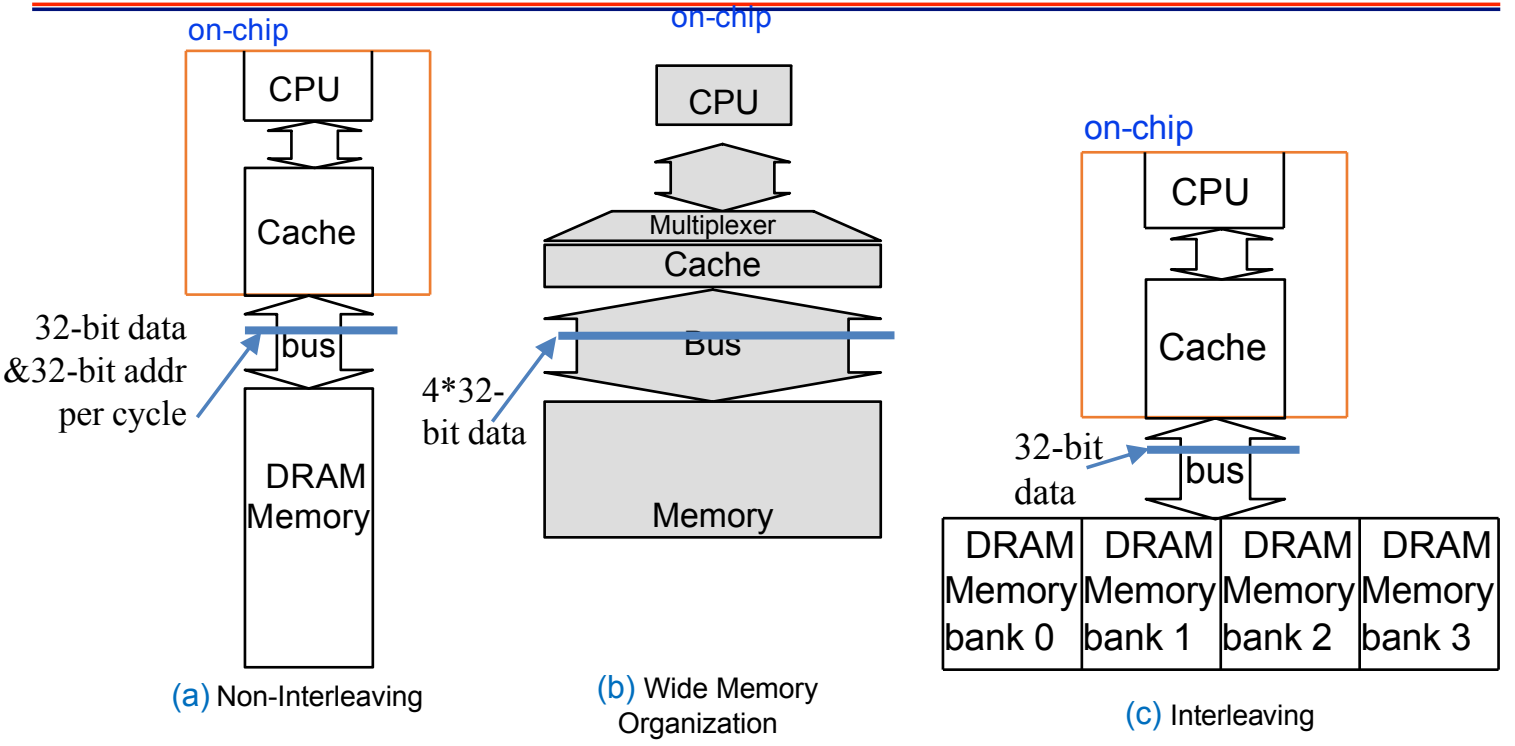
依次传送



内存访问可以并行进行

- 4-word access
- = (1+2+4*1)
- = 7-cycles

Memory Systems that Support Caches



Organization Comparison

- Assume following latencies

Send address to MM	1 clock
MM (DRAM) Access Time for first word	15 clocks
MM (DRAM) Access Time for each of subsequent words in same row(fast page)	8 clocks
Transfer time for one word	1 clock

- Find time to access a cache line of 4-words

MM Organization	Total clock cycles miss penalty	Bandwidth(bytes per clock)
a. Narrow Memory	1 + 4*15 + 4*1 = 65 clocks	(4 x 4)/65=0.246
b. Wide Memory	1 + 15 + 1 = 17 clocks	(4 x 4)/17=0.941
c. fast page mode	1+15+3*8+1=51 clocks	(4 x 4)/51 = 0.314
d. Interleaved Memory	1 + 15 + 4*1 = 20 clocks	(4 x 4)/20=0.8

静态和动态存储器芯片特性

静态RAM（SRAM）

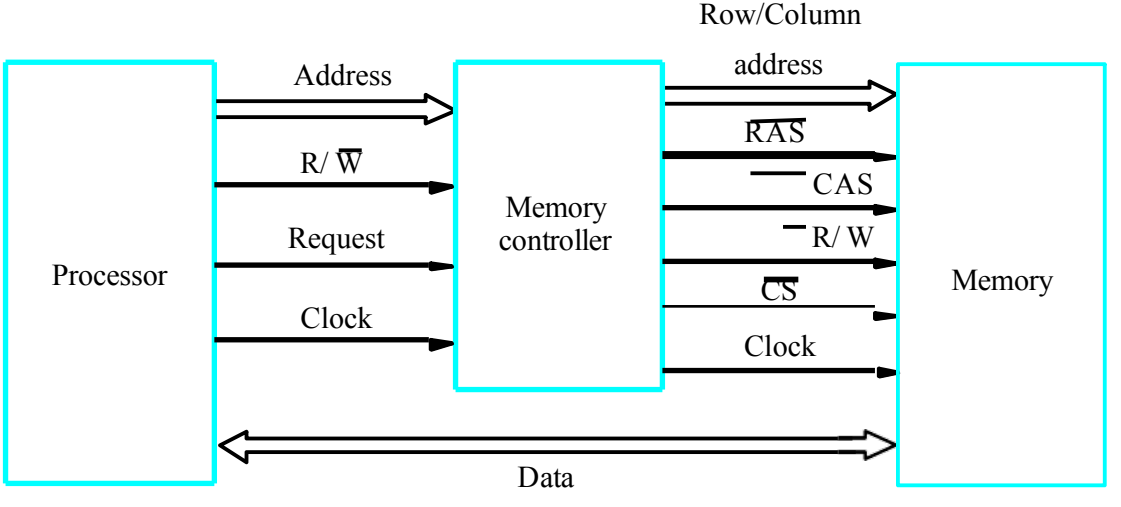
每个单元用6晶体管电路存储一个位。只要保持通电（易失性），就会无限期保留值。对电噪声等干扰相对不敏感。比DRAM更快但更昂贵。

动态RAM（DRAM）

每个单元用电容器和晶体管存储一个比特。值必须每10–100毫秒刷新一次（易失性）。对干扰敏感，比SRAM慢且便宜。

Memory Controller 内存控制器

- 内存控制器通常用于存储器和处理器之间的接口。
- DRAMs有一个稍微复杂的接口，因为它们需要刷新，并且它们通常具有分时复用信号以减少引脚数。 .
- SRAM 接口比较简单，不需要内存控制器.



RAS (CAS) = Row (Column) Address Strobe; CS = Chip Select

只读存储器（ROM）

ROM： 工作时只能快速地读取已存储的数据、而不能快速地随时写入新数据。

优点：最突出的特征是掉电后数据不丢失，用于存储数字系统中固定不变的数据和程序 。

ROM分为可编程PROM （Programmable ROM）和掩模ROM(Mask ROM)。

Mask ROM：数据是制造过程中写入的，可永久保存，但使用者不能改写。

PROM：数据是由使用者通过编程工具写入的。

ROM的寻址方式与RAM相同，采用随机寻址，即用地址译码器选择字存储单元。

ROM可以用双极型或单极型（MOS）元件实现。

EEPROM

电路特点：

1.是用电脉冲编程。

2.可利用电脉冲擦除存储的数据。

3.对E^2PROM编程存储字节要用+21v 10ms脉冲，E^2PROM适合大容量（几千个字节）的编程，可大量减少编程时间。

突出优点：

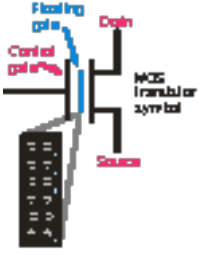
对存储的数据可以单字节擦除或改变。

即单字节擦除，单字节改变。

闪存（Flash Memory）

闪存是一种高密度可读写的的非易失存储器。它可以在掉电情况下将数据保存很长时间。

Flash存储使用具有栅极的MOS管进行存储。当正电压施加在门控端时浮动栅极使可以存储电荷(logic 0)，如果浮动栅极没有电荷，则存储 1。



logic 0 is stored

logic 1 is stored

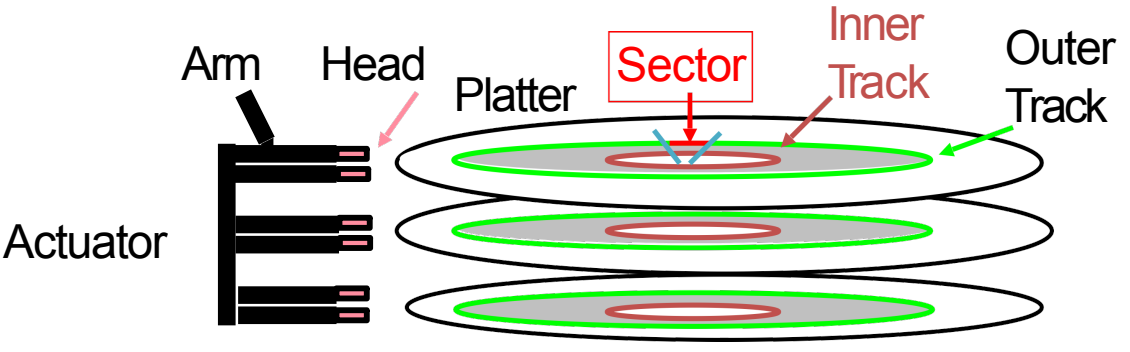
闪存的特点和应用

- 理想的存储器具有大容量、非易失、在系统读写能力、较高的操作速度和低成本等特点。
- ROM、PROM、UV EPROM、EEPROM、SRAM和DRAM，在前述的某些方面各具有一定优势。只有闪存综合具有理想存储器的特点，闪存具有RAM（随机访问）和ROM（非易失性）的特性，它通常被认为是两者的混合体。只是在写入速度方面比SRAM和DRAM差。

存储器	非易失	高密度	单管存储单元	在系统写入	写速度*
闪存	YES	YES	YES	YES	较快
SRAM	NO	NO	NO	YES	最快
DRAM	NO	YES	YES	YES	快
MASK ROM	YES	YES	YES	NO	
PROM	YES	YES	YES	NO	
UV EPROM	YES	YES	YES	NO	
EEPROM	YES	NO	NO	YES	最慢

* 写入速度是与SRAM比较的。

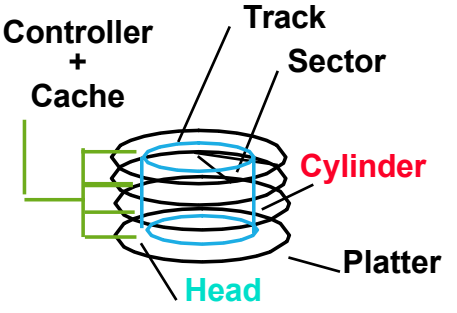
硬盘设备术语



- 几个盘片(Platter)，在两个表面上都以磁性方式记录信息（通常）
- 磁道(Track)中的位记录，这些磁道又分为扇区(Sector)（例如，512 字节）
- 执行器(Actuator)将磁头(Head)（臂端Arm）移动到磁道上方（“搜索”），等待磁头下方的扇区旋转，然后读取或写入

Disk Device Performance

- 长期、非易失性存储,最低级内存：
- 慢； 大的; 便宜
- 涂有磁性表面的旋转盘片
- 用于访问信息的可移动读/写头



- **Disk Access Time = Seek Time + Rotation Time + Transfer Time + Controller Overhead**
  - **Seek Time** = 将磁头组件定位在正确柱体上的时间
  - **Rotation Time**=磁盘旋转到要访问的块的第一个扇区到达磁头的时间
  - **Transfer Time**=块的扇区所花费的时间，以及扇区之间旋转经过磁头所花费的时间

闪存与硬盘

- 微型驱动器和闪存（例如 CompactFlash）并驾齐驱
- 均为非易失性（无需电源即可保留内容）
- 闪存的好处：功耗更低，不会死机 （没有移动部件，需要向上/向下旋转 μdrives）
- 磁盘成本 = 电机固定成本 + 机械臂，但实际磁介质成本非常低
- 闪存成本=闪存芯片的最高成本/比特
- 随着时间的推移，闪存的每比特成本下降，密集存储的竞争力增加

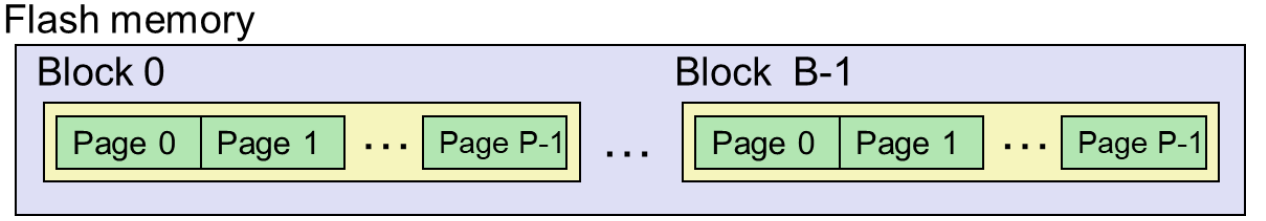
Solid State Disks (SSDs)

页：512KB 到 4KB，块：32 到 128 页，以页为单位读取/写入数据。

擦除一个块需要很长时间（~1 ms），修改一个块页面需要将所有其他页面复制到新块中

优点： 没有移动部件，速度更快, 功耗更低, 更坚固

缺点：会磨损



Flash and Latency...

闪存带宽接近旋转硬盘

- 旋转硬盘仍然用作更大的存储，最便宜

Flash 的优势：没有寻道时间！

- 随机访问与顺序访问相比没有额外的延迟

这是巨大的： 操作系统引导应尽可能线性

- 苹果在 Mac 上花费了大量精力来优化这一点……
- 当将旋转硬盘换成固态硬盘SSD 时，启动时间缩短了一半……

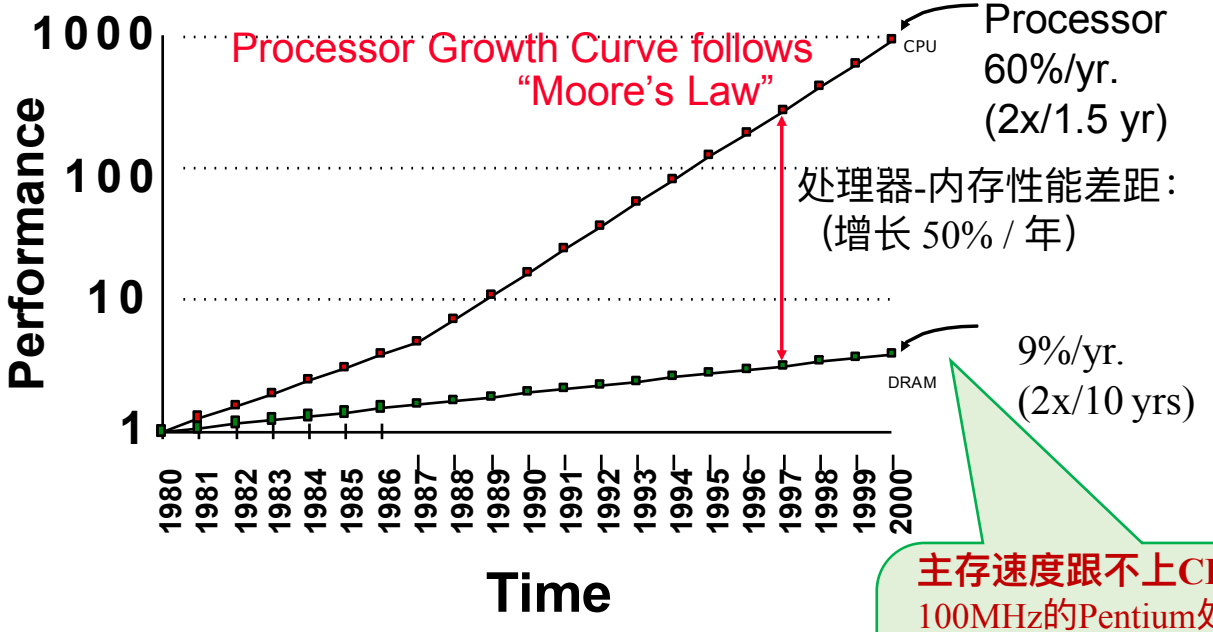
计算机需要什么样的存储器?

已经学过的存储器：SRAM、DRAM、ROM
Hard Disk、Optical Disk、Flash Memory

	Capacity	Latency	Cost
Register	<1KB	1ns	$$$$
SRAM	1MB	2ns	$$$
DRAM	1GB	10ns	$
Hard disk*	100GB	10ms	¢
Want	100GB	1ns	cheap

单独用某一种存储器不能满足我们的需要!
结合各种存储器的特点，采用层次式存储器结构来构建计算机的存储系统!

谁在乎内存层次结构?



处理器-内存(DRAM)性能差距

主存速度跟不上CPU性能

100MHz的Pentium处理器平均10ns执行一条指令，而DRAM典型访问时间60～120ns。

"处理器性能提升对系统的贡献被DRAM性能所屏蔽"

理想的存储器

- 零访问时间 (latency)
- 无限大的存储空间
- 零造价
- 无限大的带宽 (支持并行访问)

但是

- 越大越慢

  要用更多的时间决定位置

- 越快越贵

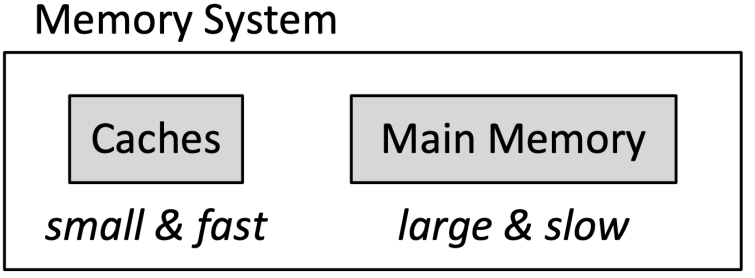
  内存技术: SRAM vs. DRAM vs. SSD vs. Disk vs. Tape

- 高带宽意味着更贵

  需要更多 banks, 更多端口, 更多通道, 更高的频率或更快的技术

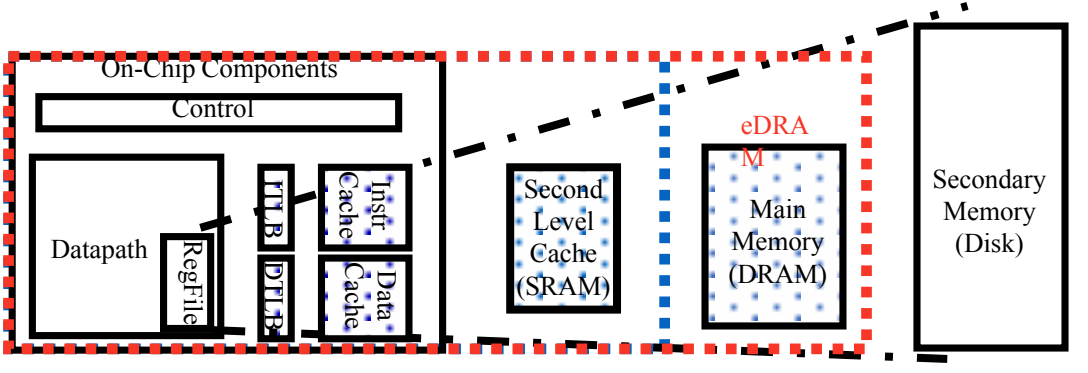
为什么存储器要用层次结构?

- 我们要 又大又快
- 单层结构实现不了
- 想法: 用多级存储 (随着级别远离处理器，逐渐变大和变慢)，确保处理器所需的大部分数据保持在快速级别



Big Idea: Memory Hierarchy
计算机设计另一个伟大思想-存储层次结构

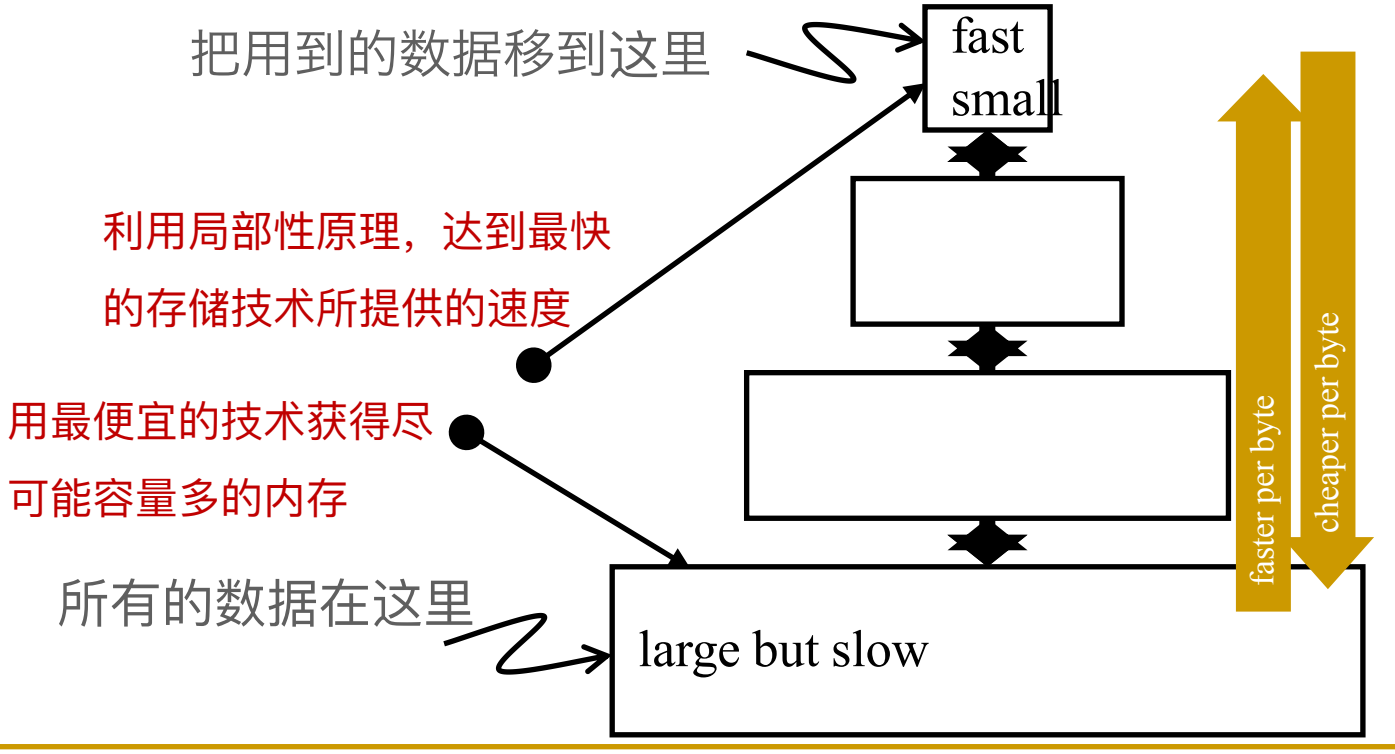
- **利用局部性原则**
  - 用最便宜的技术获得尽可能容量多的内存
  - 达到最快的存储技术所提供的速度



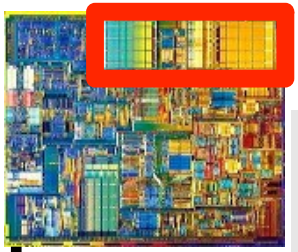
Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

现代计算机的内存系统由一系列从最快到最慢的黑匣子组成,为提高性能/价格，将各存储器组成一个金字塔式的层次结构，取长补短协调工作.

存储器层次结构


把用到的数据移到这里
fast
small
利用局部性原理，达到最快
的存储技术所提供的速度
用最便宜的技术获得尽
可能容量多的内存
所有的数据在这里
large but slow
faster per byte
cheaper per byte

A typical memory hierarchy



on-chip cache
KBs **L1**

L1, L2 缓存（cache）通常用静态随机存储器SRAM, 提供最快速度的技术

off-chip cache
MBs **L2**

main memory
GBs

Disk
TBs

	Cost	Access time
on-chip cache		< 1ns
off-chip cache	2.5 $/MB	5ns
main memory	0.07 $/MB	60ns
Disk	0.0004 $/MB	10,000,000ns

Increasing cost per bit

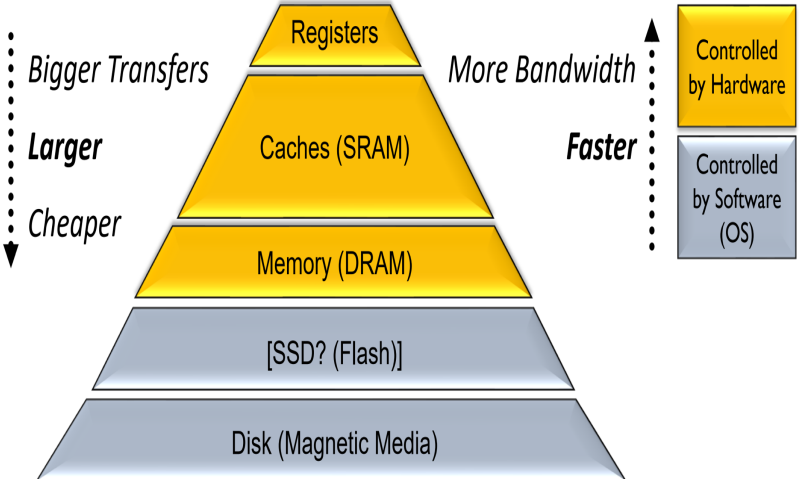
Increasing speed

Increasing latency

为解决主存容量不足的问题，借助于软硬件在高速硬盘上开辟了——**虚拟存储空间**。最便宜的技术

程序访问局部性

计算机存储层次结构



工作过程

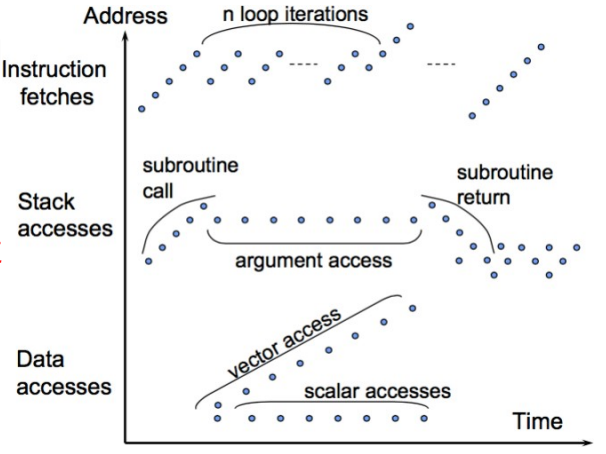
- CPU运行时，需要的操作数大部分来自寄存器
- 需从(向)存储器中取(存) 数时，先访问cache；若在，取自cache
- 若不在cache，则访问RAM；若在，则取自RAM
- 若不在RAM，则访问磁盘，操作数从磁盘中读出→RAM →cache

构成了一个统一管理、统一调度，并且对用户来说透明的一体化的存储器系统.

程序的访存特性

为什么存储层次化结构是有效的?

- **时间局部性**temporal locality
  - 最近的访问项（指令/数据）很可能在不久的将来再次被访问
  - 往往会引起对最近使用区域的集中访问
  - 策略：保留data，复用
    - 内存地址不一定集中!
- **空间局部性**spatial locality
  - 一个进程访问的访问项其地址彼此很近
  - 往往会访问在存储器空间的同一区域
  - 策略：保留data及其相邻者，预取
    - 内存地址连续!
- 例
  - for i := 0 to 10000 do
    A[i] := 0;
- 局部性实现：内存分块



Typical Access Address Pattern

“程序访问局部性”特点!