第 4 章
存储体系

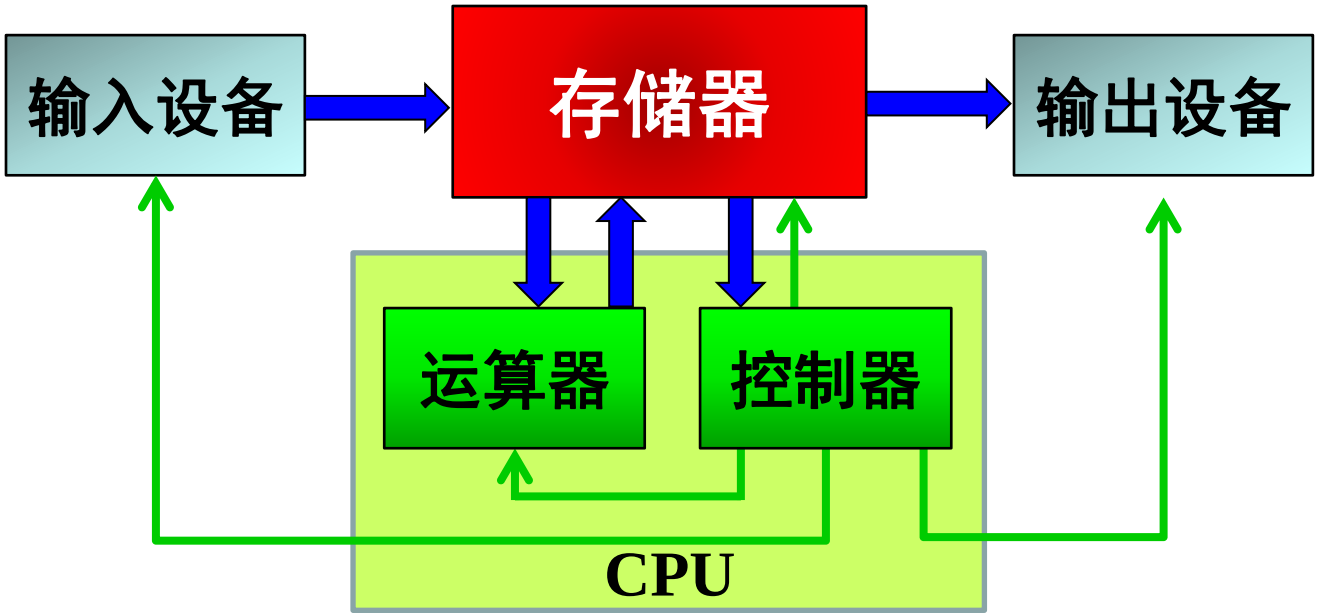
郑宏 副教授
计算机学院
北京理工大学

第四章 存储体系

学习内容：

- 4.1 存储体系概念和并行存储系统
- 4.2 虚拟存储系统
- 4.3 高速缓冲存储器（Cache）
- 4.4 Cache - 主存 - 辅存三级层次
- ARM 存储系统

4.1 存储体系概念和并行存储系统



存储器是各种信息存储和交换的中心

4.1 存储体系概念和并行存储系统

- 存储器
  - 存储数据的器件。
  - 在一台计算机中，通常有多种存储器。
  - 种类：主存储器、Cache、通用寄存器、磁盘存储器、磁带存储器、光盘存储器等。
  - 材料工艺：ECL、TTL、MOS、磁表面、激光、RAM、SRAM、DRAM等。
  - 访问方式：直接译码、先进先出、随机访问、相联访问、块传送、文件组等。

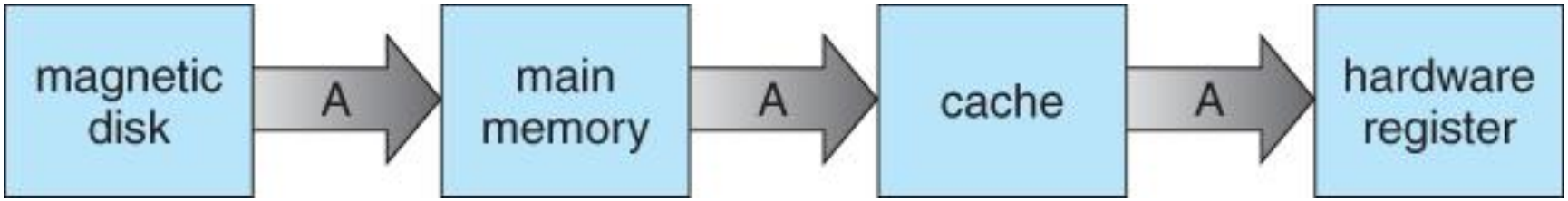
4.1 存储体系概念和并行存储系统

- 存储器（续）
  - **主存储器：** 存放正在运行的程序与数据。
  - **辅助存储器：** 存放等待运行的程序与数据。
  - **通用寄存器组：** 存放最经常用到的数据。

存储系统和存储器是两个不同的概念！

4.1 存储体系概念和并行存储系统

- 存储系统（存储体系）
  - 两个或两个以上的速度、容量、价格不同的存储器采用硬件，软件或软、硬件结合的办法联接成的一个系统。



4.1.1 存储体系的引出

- 存储器是计算机系统的核心部件之一，其容量、速度和价格是必须要考虑的因素。
- **主要目标：**

  在尽可能低的价格下，提供尽可能高的速度及尽可能大的存贮容量。

高速度、低价格、大容量！

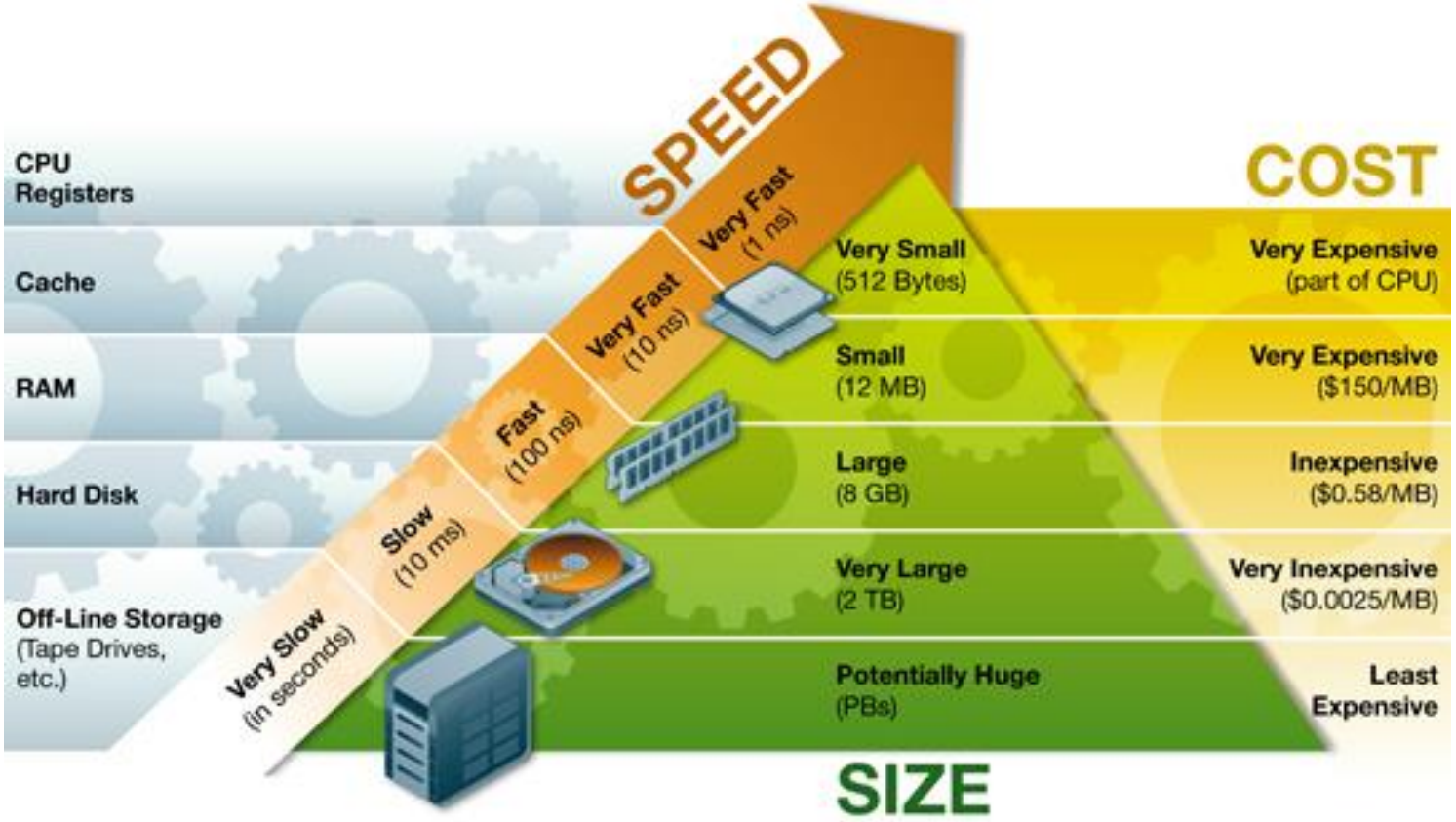
1. 容量、速度和价格的矛盾

大容量	希望能放得下所有软件
高速度	尽量和CPU的速度相匹配
低价格	系统价格中较小而合理的比例

- 但上述要求是相互矛盾的
  - 容量越大，因延迟增加而使速度降低
  - 容量越大，存储体总体价格就越高
  - 速度越高，价格将越高

1. 容量、速度和价格的矛盾

- 不同存储器的容量、速度、价格差异巨大。



- 需要考虑怎样设置才能获得高性能价格比！

2. 容量、速度和价格分析

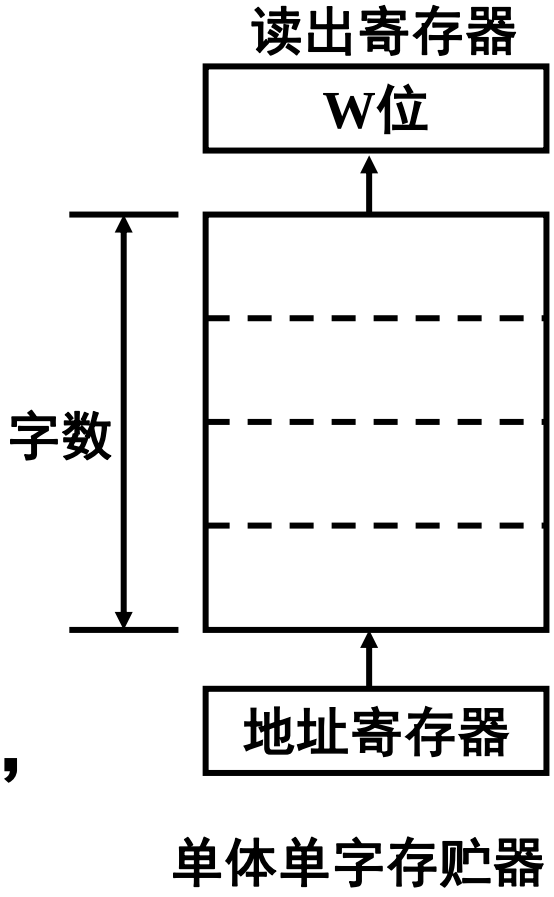
容量

- $S_M = W * L * m$，其中：

W —— 存储体的字长（位 或 字节）（设计）

L —— 每个存储体的字数（工艺）

m —— 并行工作的存储体的个数（设计）



单体单字存贮器

- 可以看出，它即与存储器器件有关，也与设计有关。

2. 容量、速度和价格分析

■速度

- 可以用访问时间T_A、存贮周期T_M和频宽B_M表示。
- T_A：存储器从接到访存读申请，到数据被读到数据总线上所需要的时间，它是启动一个访存读操作后，CPU必须等待的时间，是确定CPU与存贮器时间关系的一个重要指标。

2. 容量、速度和价格分析

■速度（续）

- T_M：连续启动一个存储体所需要的时间，即存储器进行一次存/取所需要的时间，一般它总比T_A大。
- B_M：表示存储器可以提供的数据传输率，用每秒钟传送的位数或字节数表示。分为最大(极限)频宽和实际频宽。
- 最大(极限)频宽是存储器连续访问时所能提供的频宽。

2. 容量、速度和价格分析

- **速度（续）**
  - 单体存贮器： $B_M = W / T_M$
  - 多体存贮器： $B_M = m \times W / T_M$
  - 可以看出：它即与存储器器件有关，也与设计有关。

2. 容量、速度和价格分析

- **价格**
  - 可以用总价格 C 或 位价格 c 表示
  - 价格 $c = S_M/T_M = W \times L \times m / T_M$

    与容量成正比，与速度成反比
  - 可以看出：它即与存储器器件有关，也与设计有关。

2. 容量、速度和价格分析

- 综上可见，在三个性能指标中：
  - 字数、T_A和T_M主要与器件工艺有关
  - 字长和存储体个数则可由系统设计者确定
  - 改进工艺和设计可以解决矛盾

3. 解决矛盾的措施

为满足系统对存储器的性能要求，可以采取以下措施：

- （1）改进工艺和技术，降低成本、提高速度
  - 问题：只采用一种工艺的单一存储器无法同时满足上述三个方面的要求。

Memory 优化

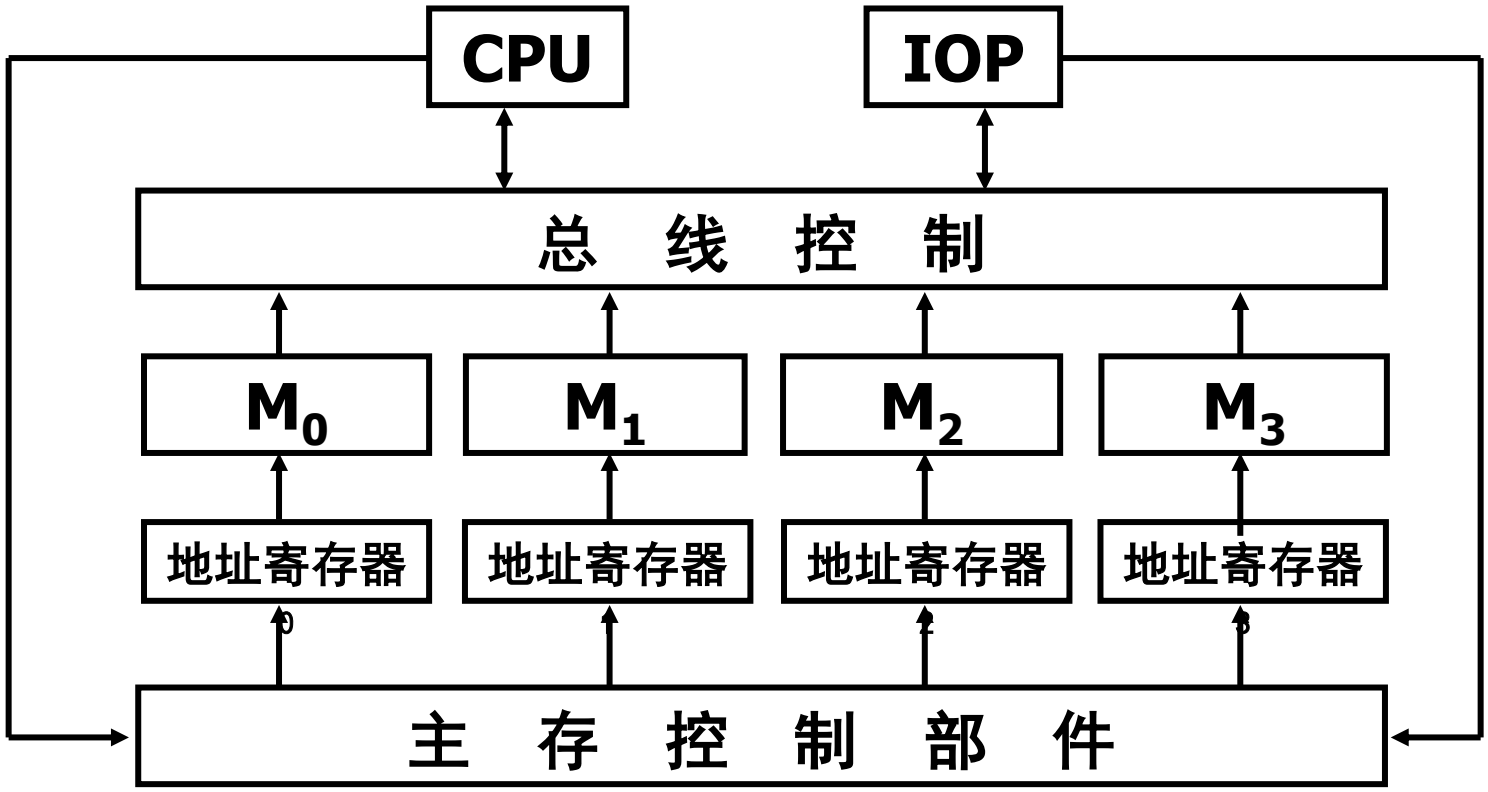
- **DDR:**
  - **DDR2**
    - **Lower power (2.5 V -> 1.8 V)**
    - **Higher clock rates (266 MHz, 333 MHz, 400 MHz)**
  - **DDR3**
    - **1.5 V**
    - **800 MHz**
  - **DDR4**
    - **1-1.2 V**
    - **1600 MHz**
- **GDDR5 is graphics memory based on DDR3**

Memory Optimizations

- **Graphics memory:**
  - **Achieve 2~5x bandwidth per DRAM vs. DDR3**
    - **Wider interfaces (32 vs. 16 bit)**
    - **Higher clock rate**
      - **Possible because they are attached via soldering instead of socketted DIMM modules**
- **Reducing power in SDRAMs:**
  - **Lower voltage**
  - **Low power mode (ignores clock, continues to refresh)**

3. 解决矛盾的措施

- （2）构成并行主存系统



多体单字交叉存贮器

3. 解决矛盾的措施

（2）构成并行主存系统（续）

- 采用单一存储器，但在组成上引入并行和重叠技术，构成并行主存系统，即多体交叉存储器，在位价格基本不变的情况下，使主存的频宽得到较大的提高。
- 问题：提高频宽的能力是有限的
  - m越多，负载变重，延时增加
  - 系统效率不是很高，因为指令的读取不是顺序的，转移指令使存贮系统效率下降。

3. 解决矛盾的措施

（3）使用存储器系统

- 用不同工艺的多种存储器组成存贮器系统，使信息以各种方式分布于不同的存储器上。
- 例如：至少有主存和辅存两种存储器
  - 主存：价格高、速度快、容量小，存放程序的活跃部分。
  - 辅存：价格低、速度慢、容量大，存放暂时不用的部分。

3. 解决矛盾的措施

- （3）使用存储器系统（续）
  - 问题：主存速度仍不能满足CPU的要求
  - 例如，在70年代，合理成本、足够容量的主存的存储周期比CPU拍宽大一个数量级
- （4）存储体系（存储层次）

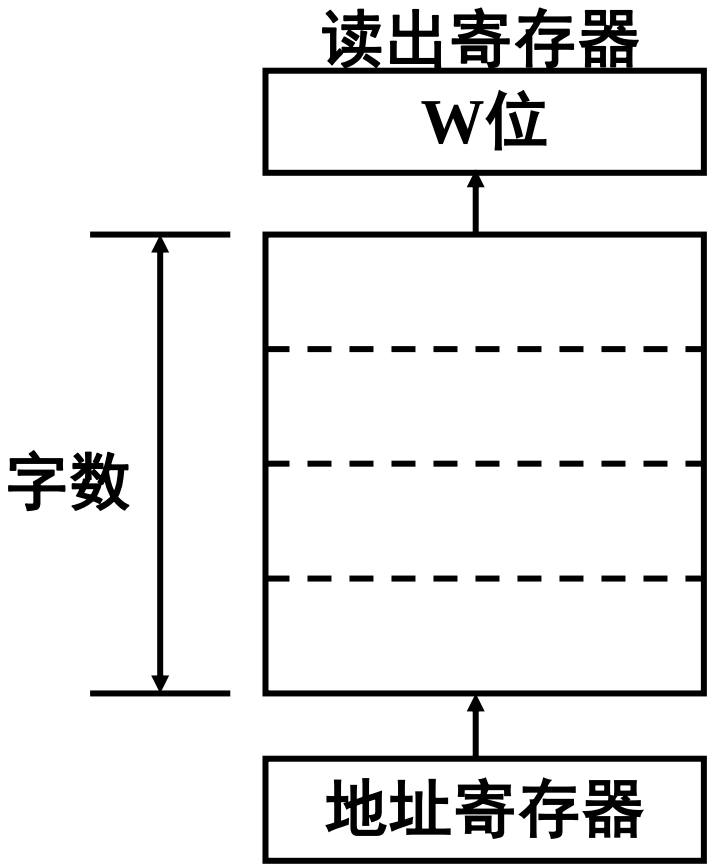
4.1.2 并行存储系统

- 特点：
  - 在一个存储周期内可以访问到多个数据，从而提高主存频宽。
- 类型：
  - 单体多字
  - 多体单字交叉存储器
  - 多体多字交叉存储器

单体单字存贮器

- 有一个字长为W位的存贮器，一次可以访问一个存贮器字。
- 若存贮器字长与CPU字长相等，其最大频宽为：

$$B_M = W / T_M$$



单体单字存贮器

提高存储系统性能的途径：并行

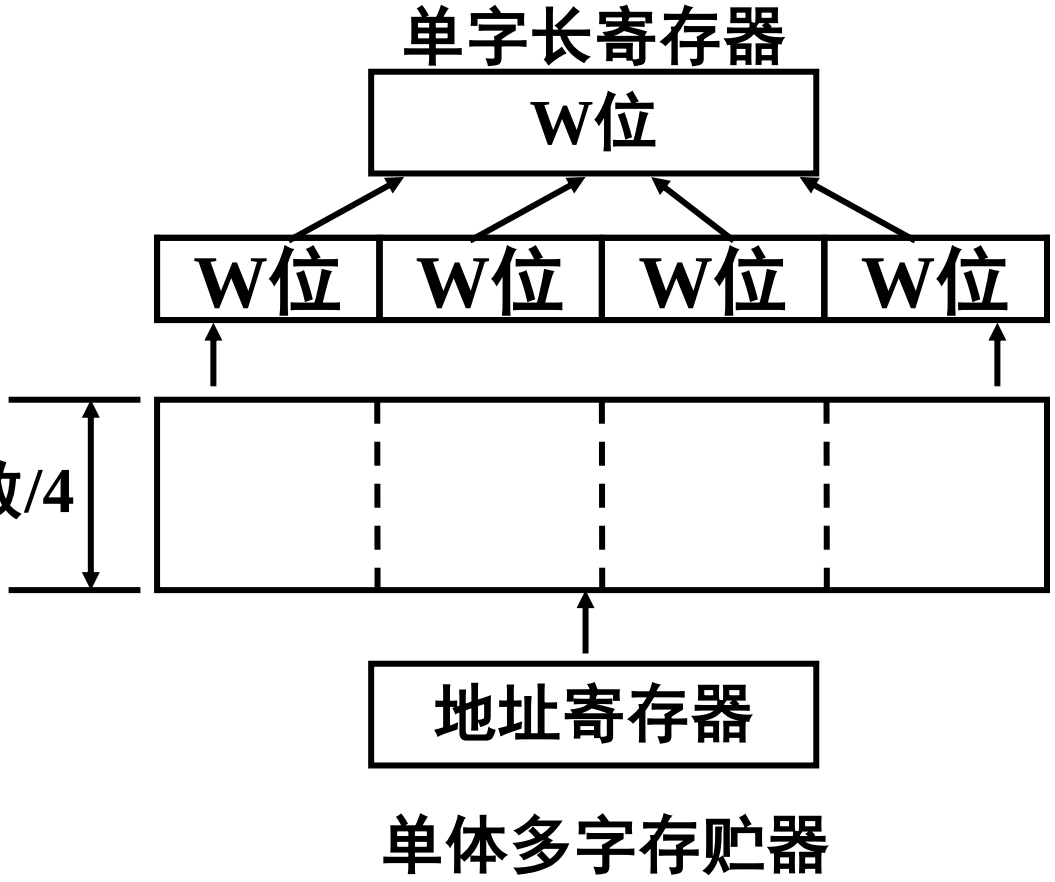
- 要提高频宽，只有设法提高存贮器字长W 才行。有三种方案：
  - 单体多字存贮器
  - 多体单字交叉存贮器
  - 多体多字交叉存贮器

单体多字存贮器

增加存贮器的字长（m倍），这样在一个主存周期内就可以读出多个CPU字。

其最大频宽为：

$$B_M = m \times W / T_M$$



单体多字存贮器

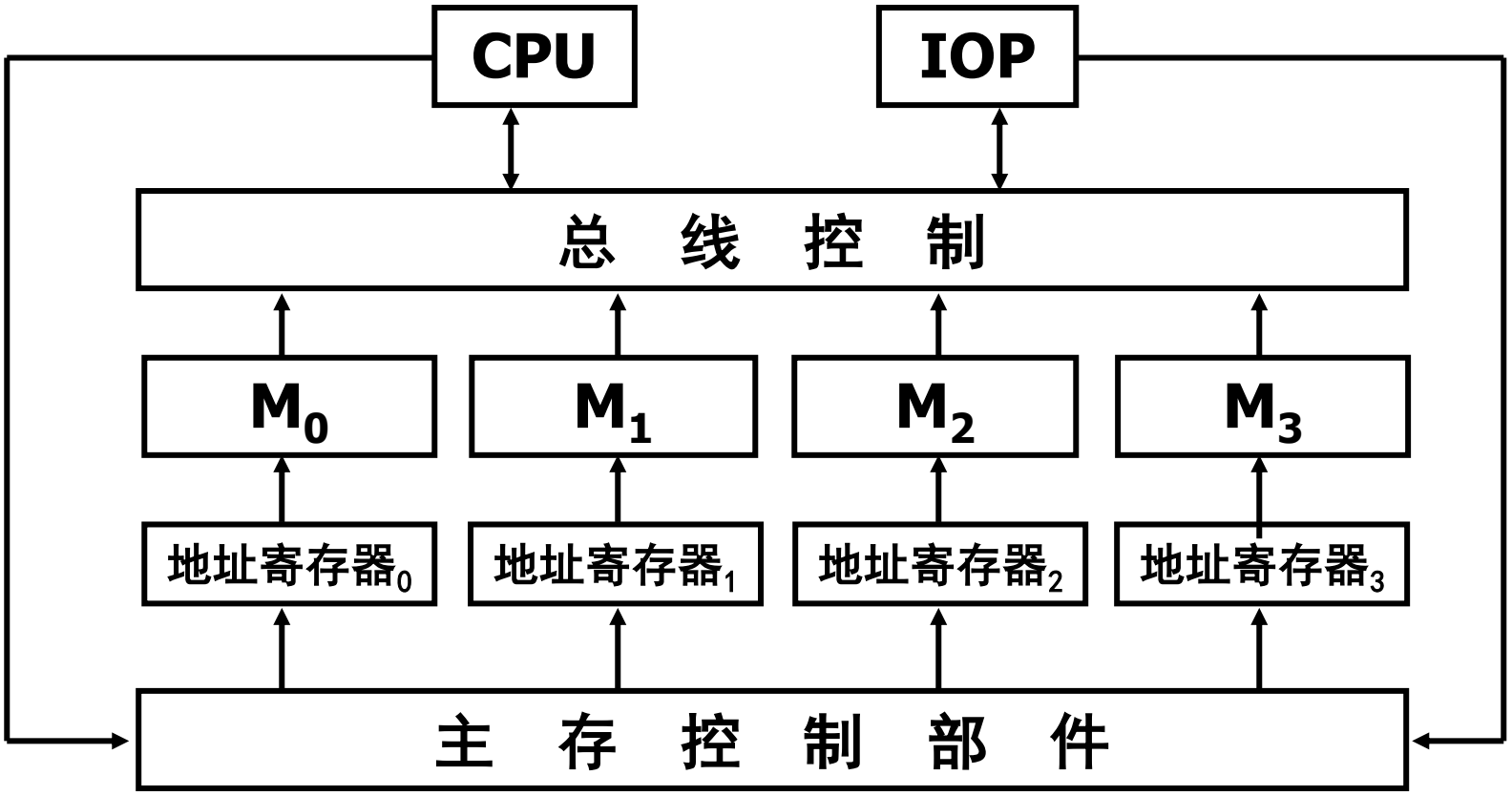
缺点：

- 需要位数足够多的寄存器；
- 多次访问总线。

多体单字存贮器

- 由多个容量较小、字长较短的相同存贮器芯片组成。
- 每个芯片都有自己的地址译码、读/写驱动等外围电路。
- 每个存贮体字长都是一个CPU字的宽度，让多个（m个）字长为W位的存贮体并行工作，一次可以访问多个存贮器字。
- 其最大频宽为：$B_M = m \times W / T_M$

多体单字存贮器



多体单字交叉存贮器

多体单字存贮器

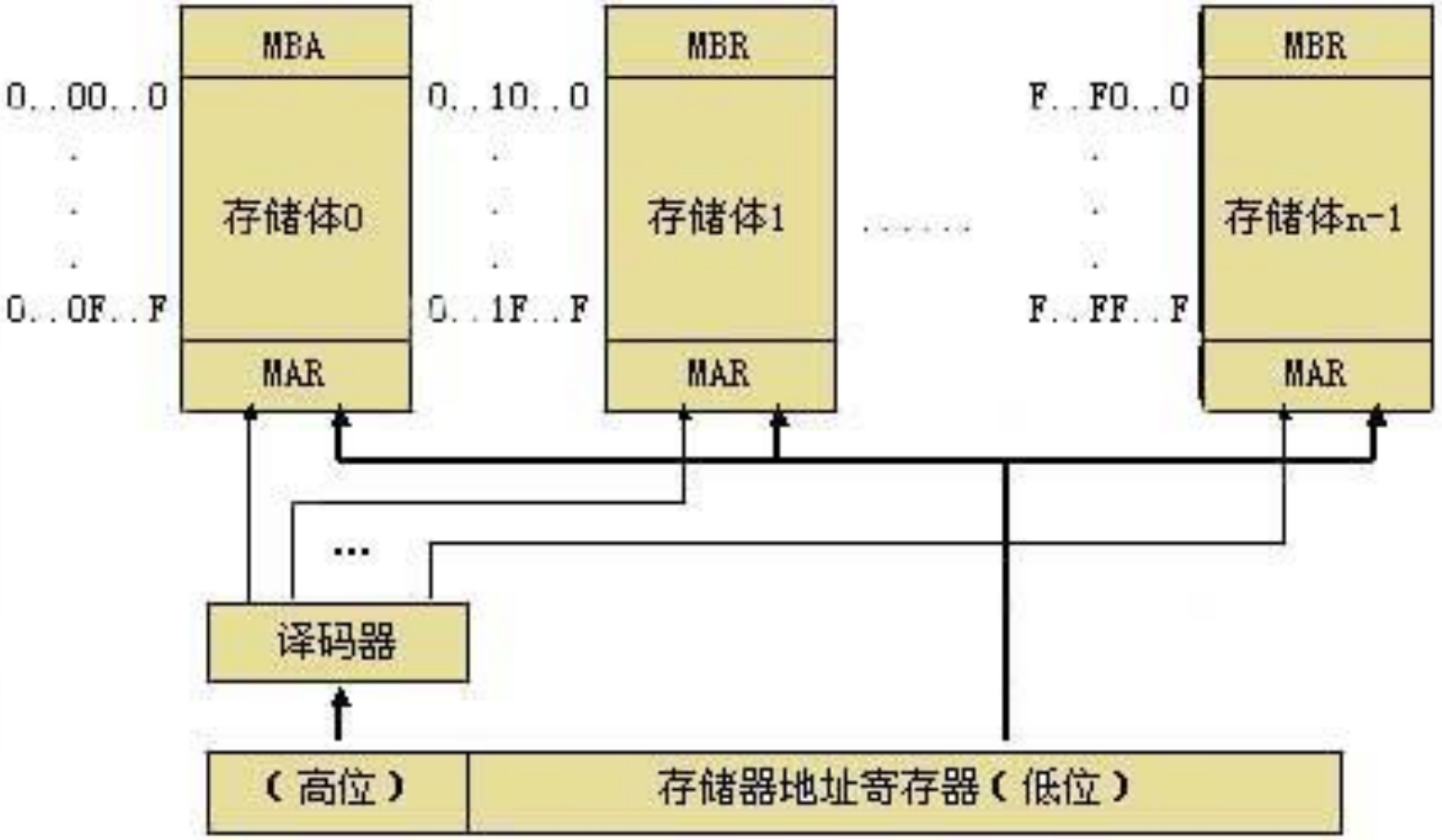
- **优点：**
  - 实际频宽比单体多字方式高，但总价格和器件的数量相差不多。
  - 并行访问不同存储体。
- **缺点：**
  - 访问冲突大

1. 取指冲突
2. 读操作数冲突
3. 写数据冲突
4. 读写冲突

多体单字存贮器

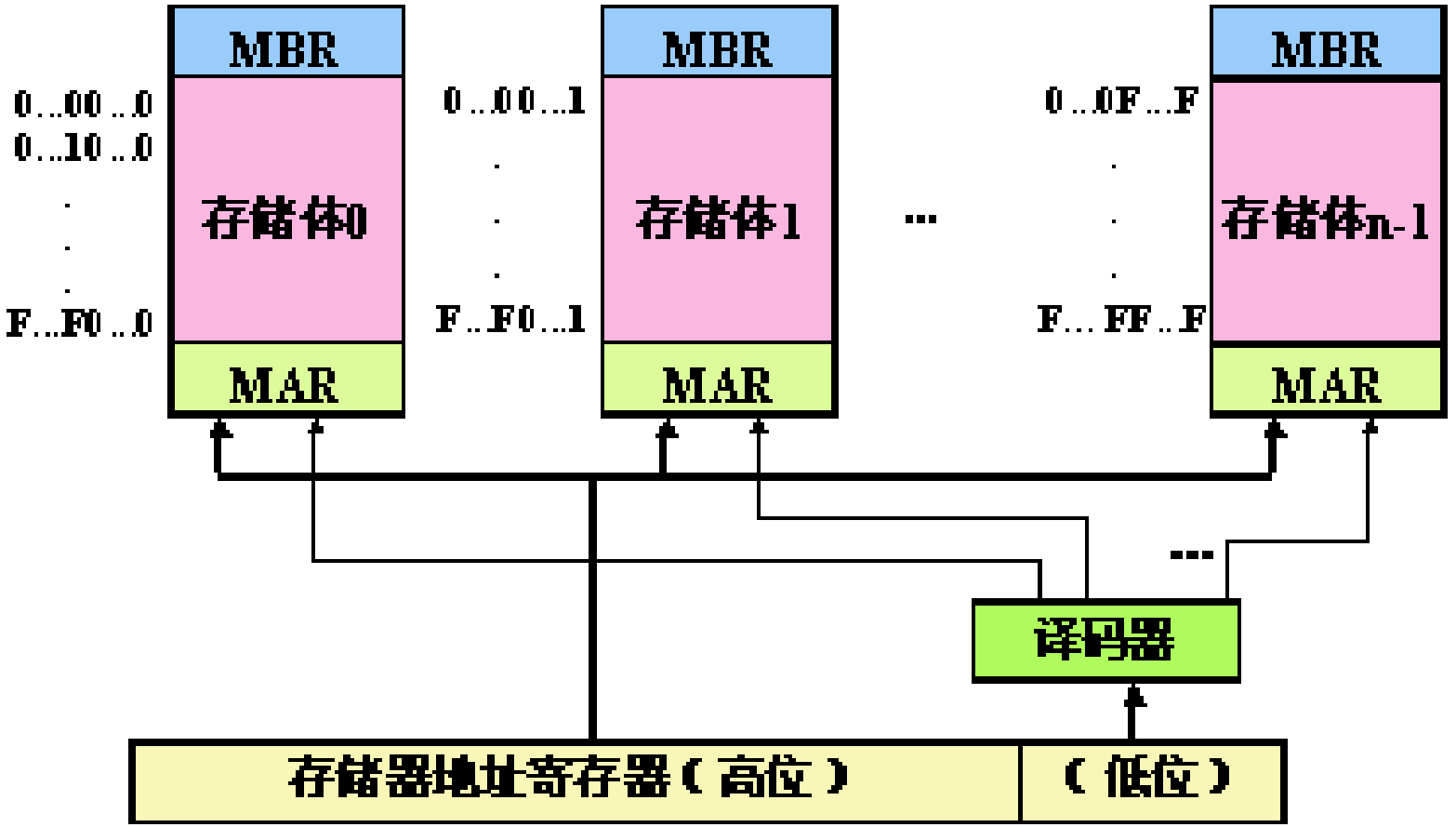
- 为减少分体冲突，CPU字在主存中可按模m交叉编址。分为：
  - 高位交叉
    - 实现方法：用地址码的高位部分区分存储体号
    - 主要目的：扩大存储器容量
  - 低位交叉
    - 实现方法：用地址码的低位部分区分存储体号
    - 主要目的：提高存储器访问速度

多体单字存贮器



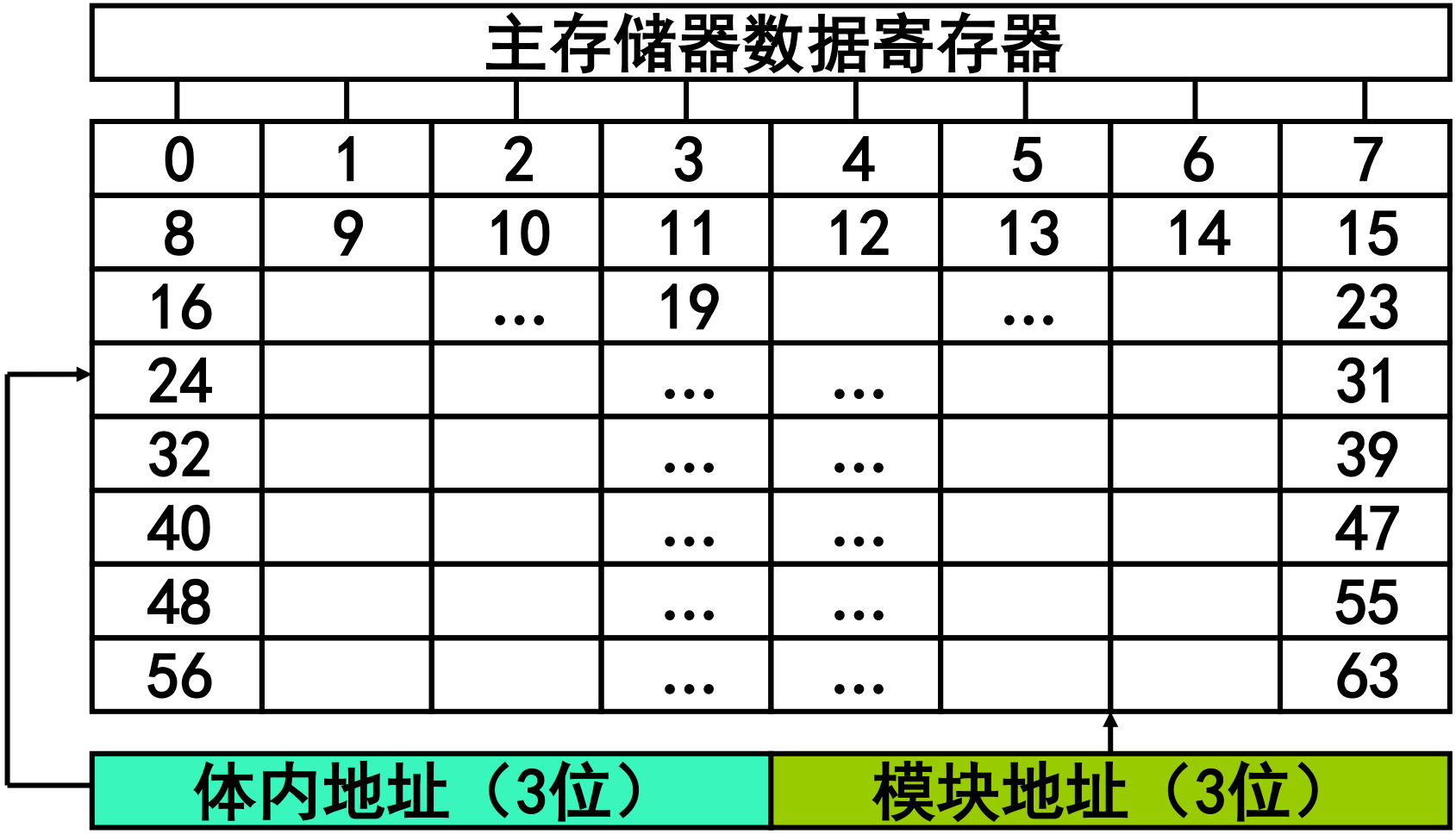
高位交叉访问存储器

多体单字存贮器



低位交叉访问存储器

多体单字存贮器



低位交叉访问存储器 - 8个存储体

多体单字存贮器

由于采用交叉存储编址，对于任何CPU读写访问或与外设DMA传送，只要是对主存连续字的成块传送，就可以实现多模块流水式并行存取，亦即使多个模块在任一时刻同时并行工作，大大提高存储器的带宽

由于CPU的速度比主存快，同时从主存取出多条指令，必然会提高机器的运行速度。

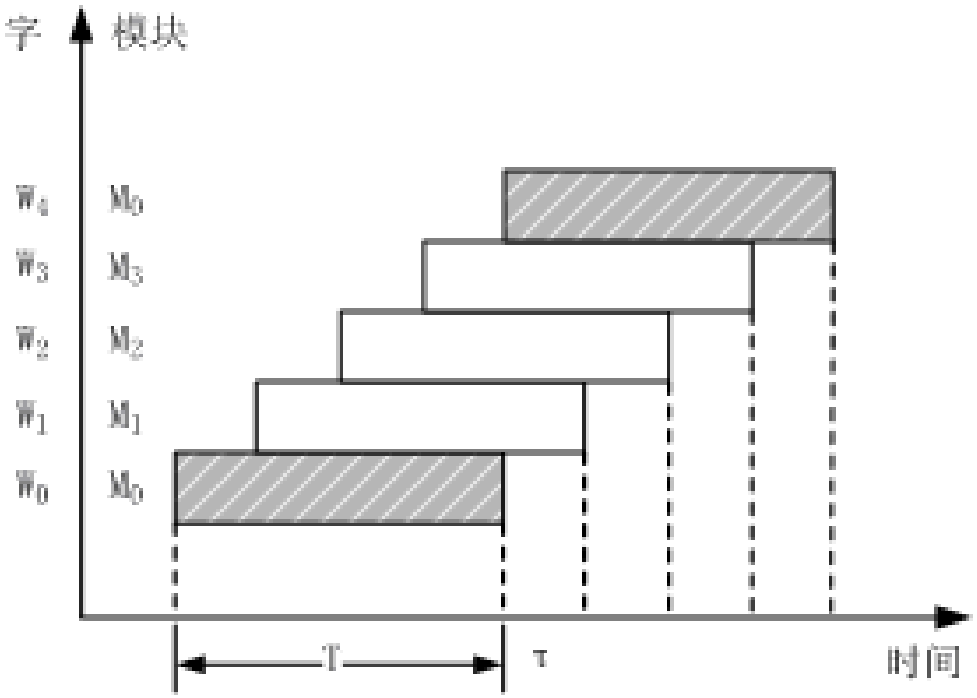


图 3-9 四模块交叉存储器的流水线方式存取示意图

低位交叉访问存储器

多体单字存贮器

模4低位交叉编址(m=4, j=0,1,2,3)

模块M_j	地址编址序列　m*i+j	对应二进制地址码最末二位的状态
M_0	0, 4, 8, 12,…, 4i+0, …	00
M_1	1,5,9,13,… 4i+1,…	01
M_2	2,6,10,14,…,4i+2,…	10
M_3	3,7,11,15,…,4i+3,…	11

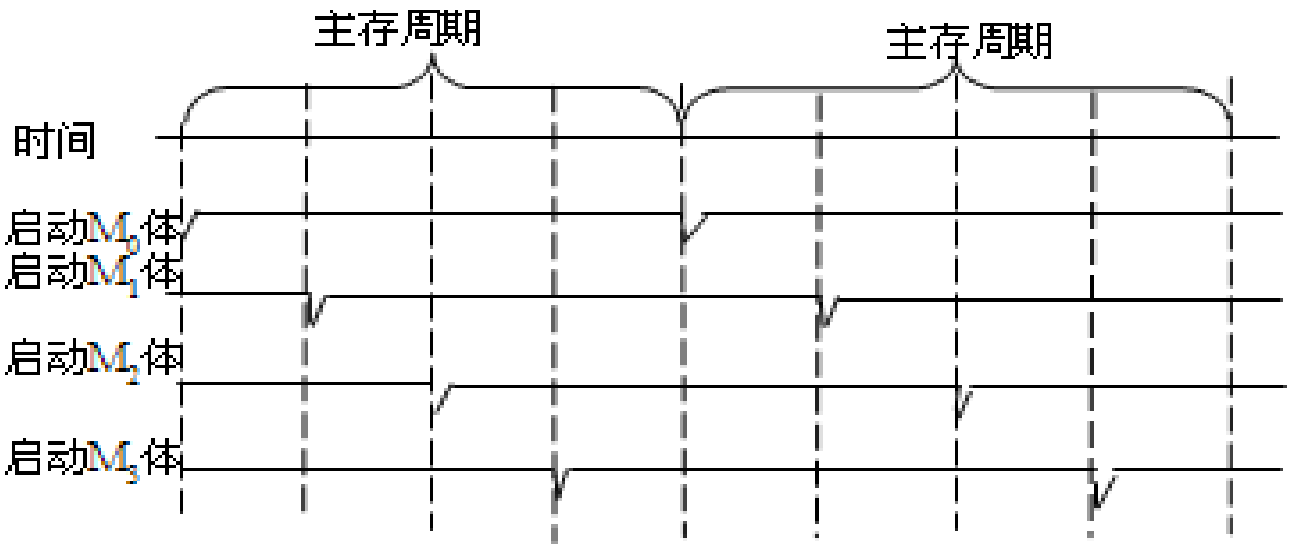


图　4个分体时启动的时间关系

同时给出多个地址，同时访问不同存储体；分时使用总线。适合流水线处理。

低位交叉访问存储器

多体多字存贮器

- 将多体单字存取与单体多字存取结合，进一步提高了频宽。

- 将上述能并行读出多个CPU字的主存系统称为并行存储系统。

4.1.2 并行存储系统

- 存在的问题
  - 提高m值，可以提高主存系统的最大频宽，但主存的实际频宽并不随着m值的增加而线性提高。
  - 原因在于：
    - m值越大，存贮器数据总线越长，总线上并联的负载越重，门的级数增加，传输延迟增加；
    - 转移指令使系统的效率下降。而且数据分布的离散性很大，因此实际的频宽可能还要低一些。

4.1.2 并行存储系统

- 可见，单纯增大m值来提高并行主存系统的频宽十分有限，而且性能价格比会随着m的增大而下降。
- 因此必须从系统结构上改进。

4.1.3 存储体系的定义与分支

- 为了解决单一种类存贮器的价格、容量及速度之间的矛盾，计算机系统的存贮系统总是利用多种不同的存贮器构成。
- 如何组织这些不同的存贮器呢？

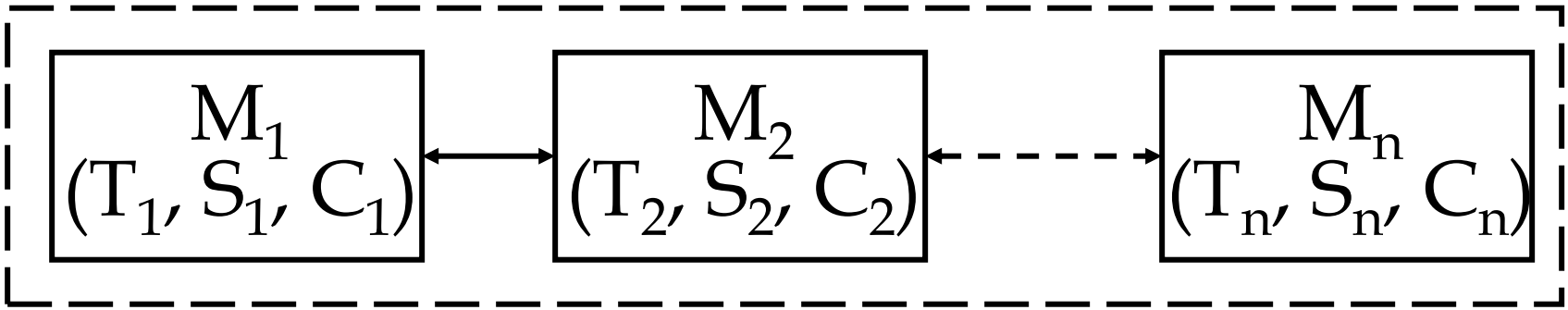
存储体系！

4.1.3 存储体系的定义与分支

- 定义：
  - 两个或两个以上速度、容量和价格各不相同的存储器用硬件、软件、或软件与硬件相结合的方法连接起来成为一个存储系统。
  - 对应用程序员透明。从应用程序员看，它是一个存储器。
  - 这个存储器的速度接近速度最快的那个存储器，存储容量与容量最大的那个存储器相等，单位容量的价格接近最便宜的那个存储器。

4.1.3 存储体系的定义与分支

从外部看：



$T \approx \min(T_1, T_2, \ldots, T_n)$，用存储周期表示

$S \approx \max(S_1, S_2, \ldots, S_n)$，用MB或GB表示

$C \approx \min(C_1, C_2, \ldots, C_n)$，用每位的价格表示

4.1.3 存储体系的定义与分支

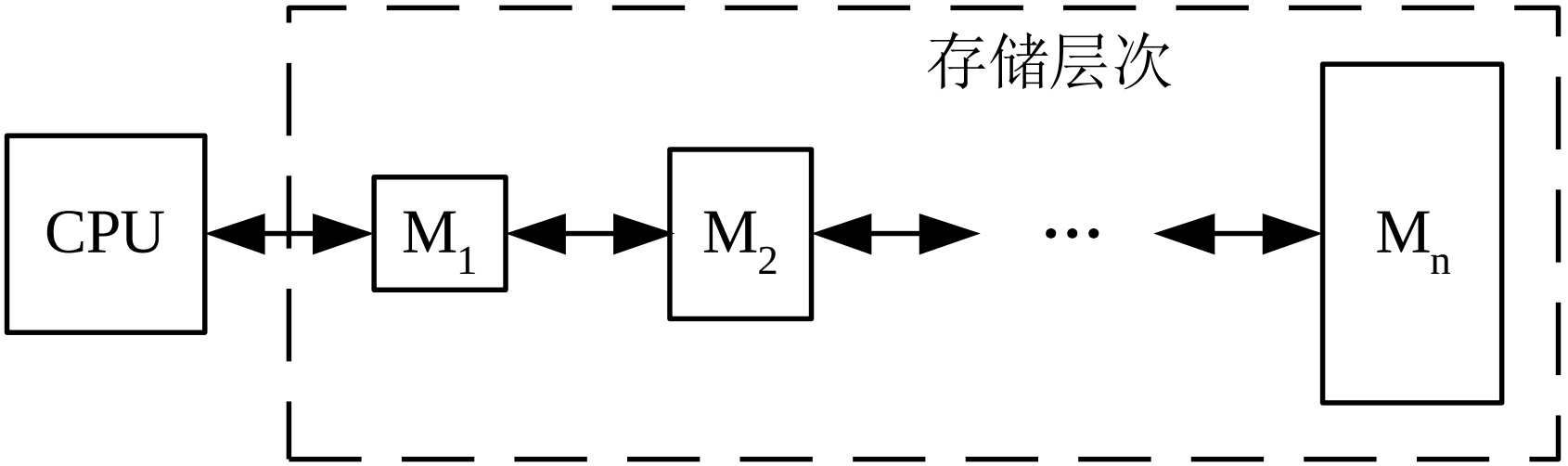


图 多级存储层次

两个典型分支：

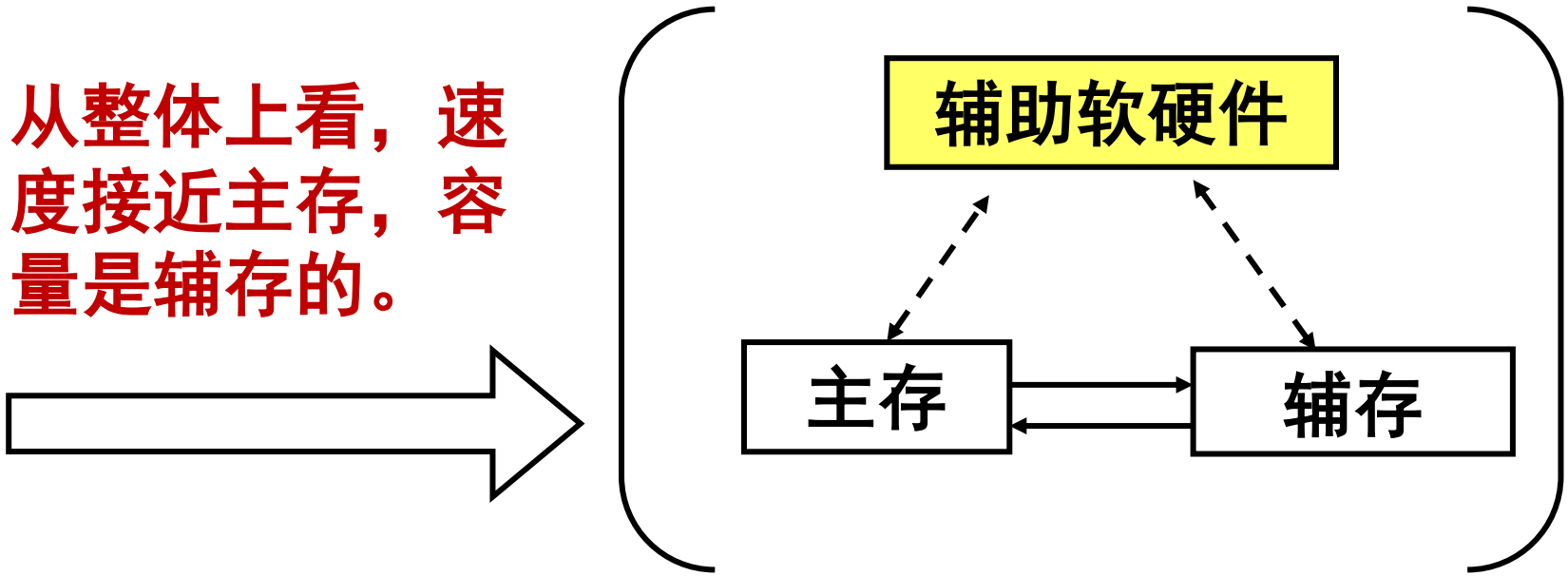
- **虚拟存储系统**
- **Cache存储系统**

虚拟存储系统

■ 解决容量问题

● 主存—辅存存储层次（二级存储层次）

从整体上看，速度接近主存，容量是辅存的。



主存—辅存存贮层次

虚拟存储系统

解决容量问题（续）

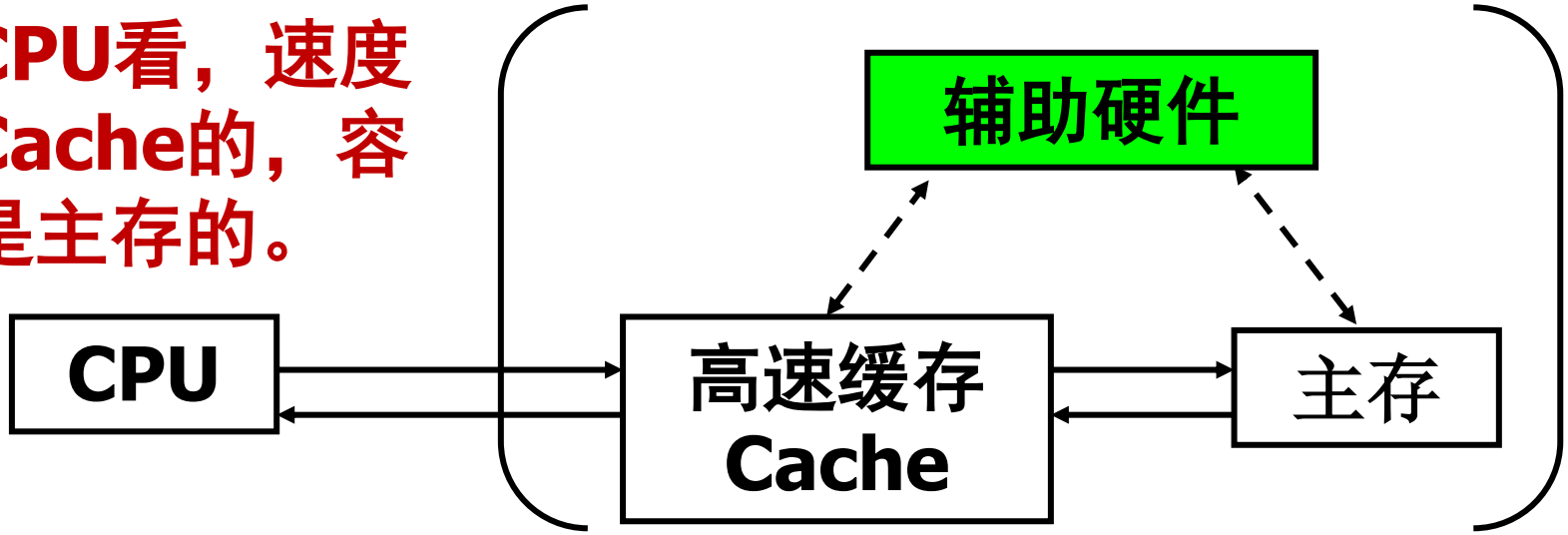
- 利用了I/O处理可与CPU并行操作的能力；
- 借助操作系统、硬件等实现地址变换和程序定位，实现主存和辅存之间的数据传送，使主存、辅存构成了一个完整的整体。
- 对应用程序员是透明；
- 主存—辅存存贮层次的不断发展和完善，就形成了虚拟存贮器。

Cache存储系统

解决速度问题

- Cache－主存存贮层次（二级存储层次）

从CPU看，速度是Cache的，容量是主存的。



Cache－辅存存贮层次

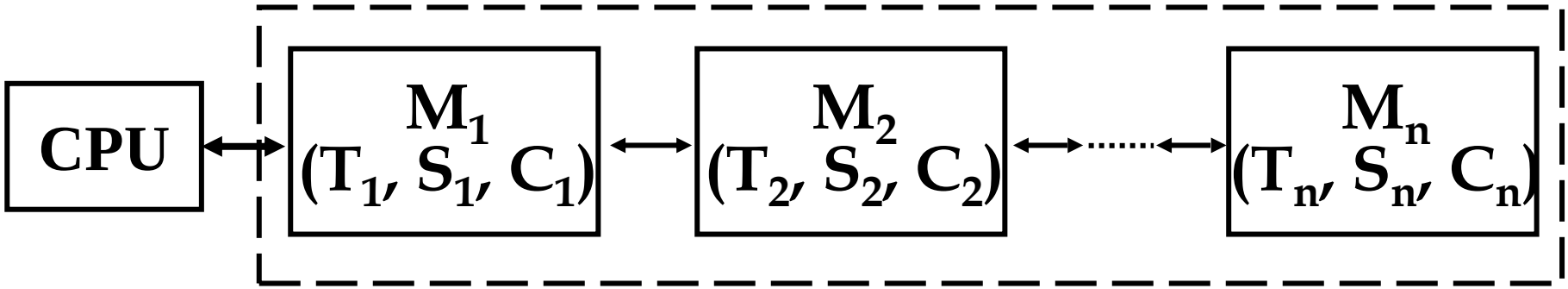
Cache存储系统

解决速度问题（续）

- 在CPU和主存之间增加一级速度快、容量小、位价格较高的高速缓冲存贮器（Cache）
- 借助于辅助硬件实现 Cache和主存之间的传送，使Cache和主存构成一个整体
- 对应用程序员和系统程序员都是透明的。

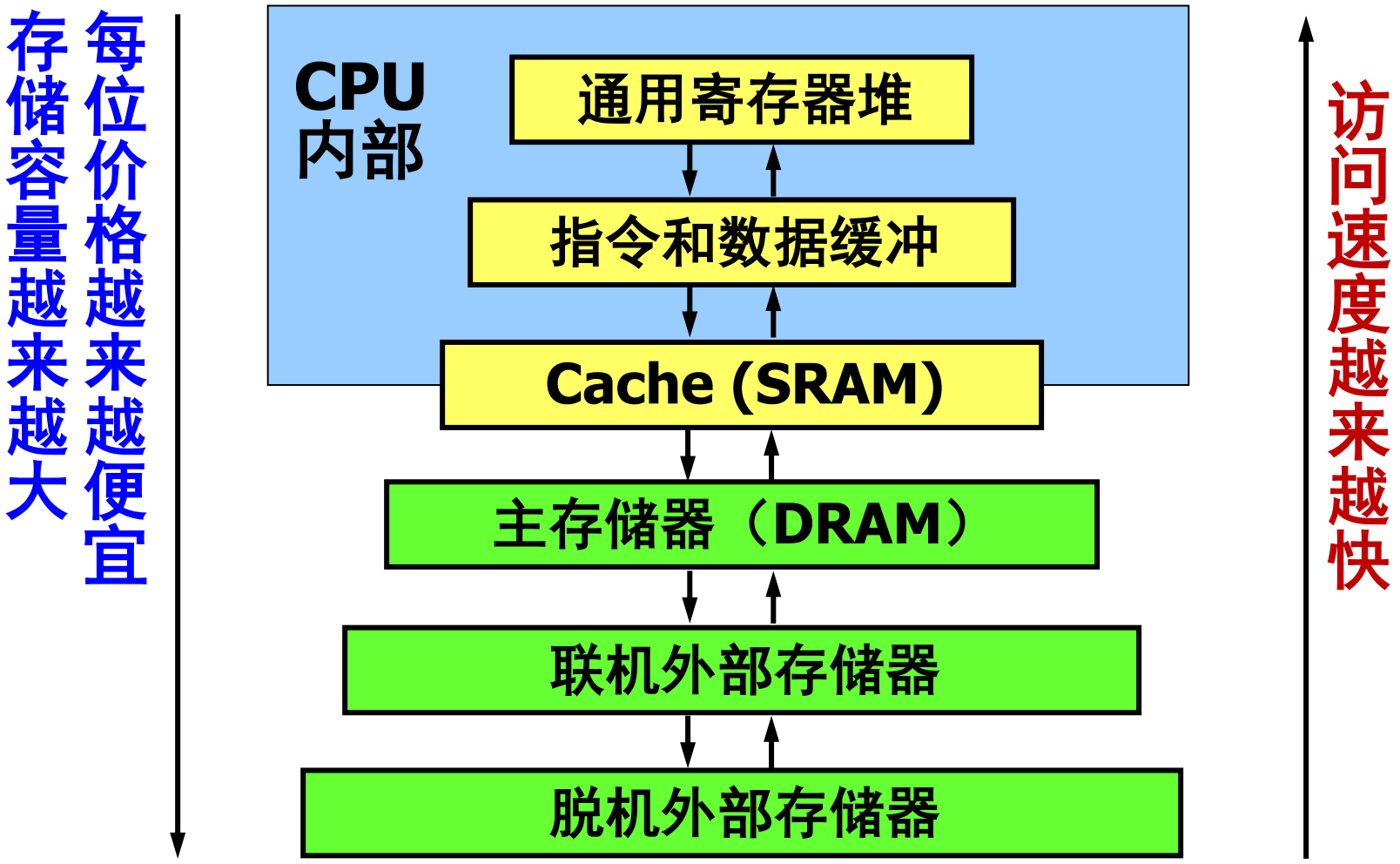
多级存贮层次

- 二级存贮层次可以扩展到多级存贮层次



多级存贮层次

4.1.3 存储体系的定义与分支



计算机中的存储层次结构

多级存贮层次

各级存储器的主要性能特性（1/2）

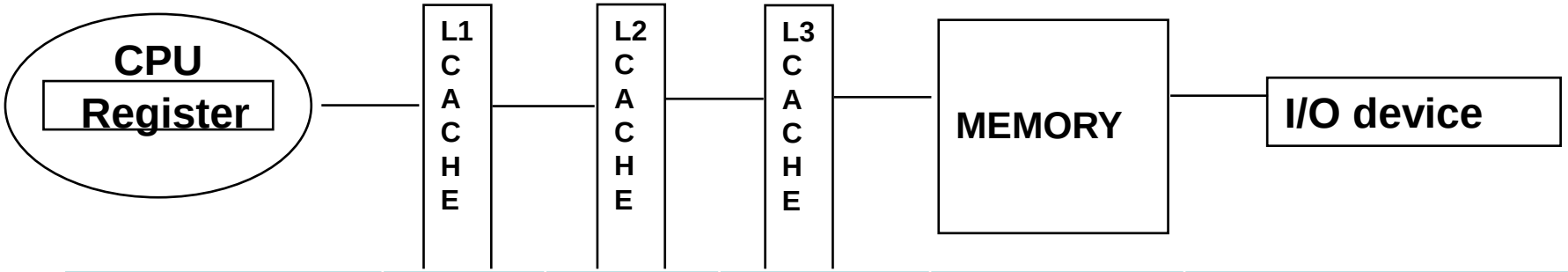
存储器层次	通用寄存器	缓冲栈	Cache
存储周期	< 10ns	< 10ns	10 - 60ns
存储容量	< 512B	< 512B	8K-2MB
价格$c/KB	1200	80	3.2
访问方式	直接译码	先进先出	相联访问
材料工艺	ECL	ECL	SRAM
分配管理	编译器分配	硬件调度	硬件调度
带宽MB/s	400-8000	400-1200	200-800

多级存贮层次

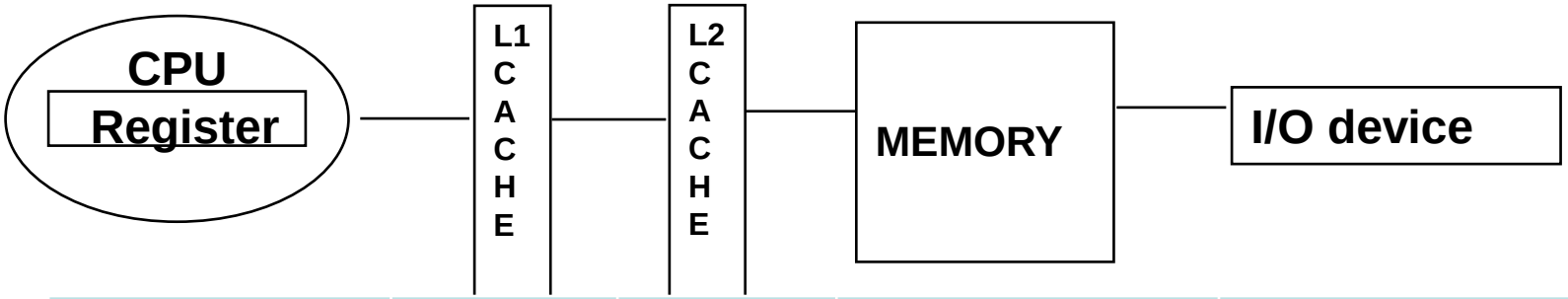
各级存储器的主要性能特性（2/2）

存储器层次	主存储器	磁盘存储器	脱机存储器
存储周期	60-300ns	10 - 30ms	2 - 20 min
存储容量	32M-1GB	1G-1TB	5G-10TB
价格$c/KB	0.36	0.01	0.0001
访问方式	随机访问	块访问	文件组
材料工艺	DRAM	磁表面	磁、光等
分配管理	操作系统	系统/用户	系统/用户
带宽MB/s	80-160	10-100	0.2 - 0.6

现代计算机系统的多级存储层次



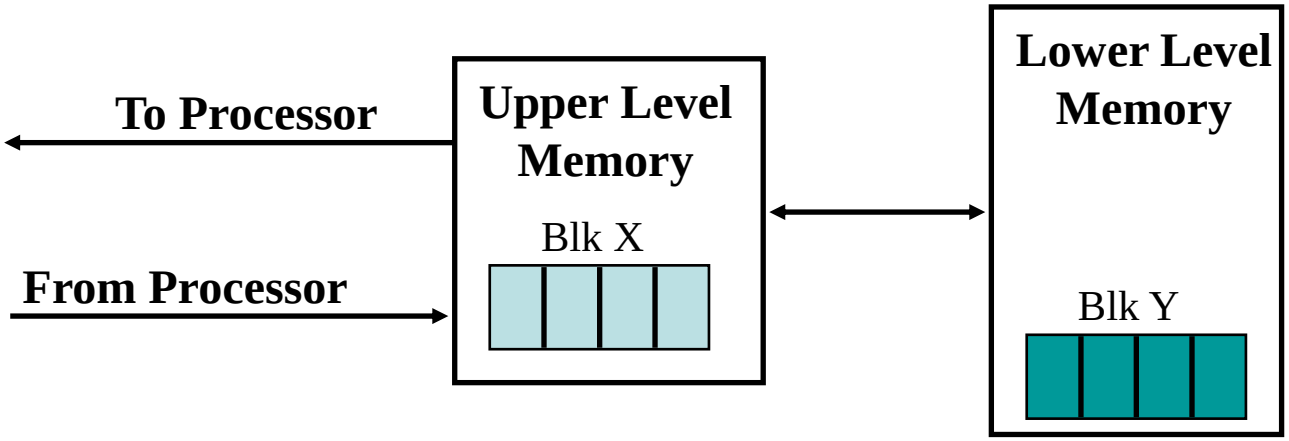
300ps	1ns	3-10ns	10-20ns	50-100ns	5-10ms
1000B	64KB	256K	2-4MB	4-16GB	4-16TB



500ps	2ns	10-20ns	50-100ns	25-50μs
500B	64KB	256K	256-512GB	4-8GB

4.1.3 存储体系的定义与分支

- 不同存贮层次之间的数据传送
  - 程序或数据分布在不同层次的存贮器上；
  - 为了使存贮体系有效工作，当CPU要用到某个地址的内容时，总是希望被访问的数据已经在M_1中；
  - 因此，需要在不同层次的存贮器传送数据。



4.1.3 存储体系的定义与分支

- 不同存贮层次之间的数据传送
  - 把哪些数据从M_n传送到M_1?
  - 把数据放在M_1的什么地方？
  - 如果能预判出下步要访问的程序块，并提前将它们取到M_1中，就可以使存贮体系有效工作。
  - 能否预判？程序局部性 = 预判的基础

程序局部性

- **定义：**

  程序在执行时所用到的指令和数据的分布不是随机的，而是相对地簇聚成块或页。它包括**时间局部性**和**空间局部性**。

- **时间局部性**
  - 最近的未来要用到的信息可能就是当前正在使用的信息——这是由程序的循环造成的。
- **空间局部性**
  - 最近的未来要用到的信息可能就是当前信息的相邻信息——这是由程序的顺序执行造成的。

程序局部性

- 基于局部性，可以得出如下结论：
  - M1 不必存放整个程序，只需存放近期使用过的块或页即可（时间局部性）。
  - 调入时，一并把数据所在的块或页一起调入（空间局部性）。
- 预判的准确性是存贮层次设计好坏的主要标志，很大程度上，取决于所使用的算法和地址映像与变换方式。

多级存储层次的两个原则

- **原则1：一致性原则**
  - 同一信息可以处于不同层次的存储器中。同一信息在不同层次存储器中的值应保持相同。
- **原则2：包含原则**
  - 高层次存储器中的信息包含在低层次存储器中的信息中，即：

$$信息_{高层} \in 信息_{低层}$$

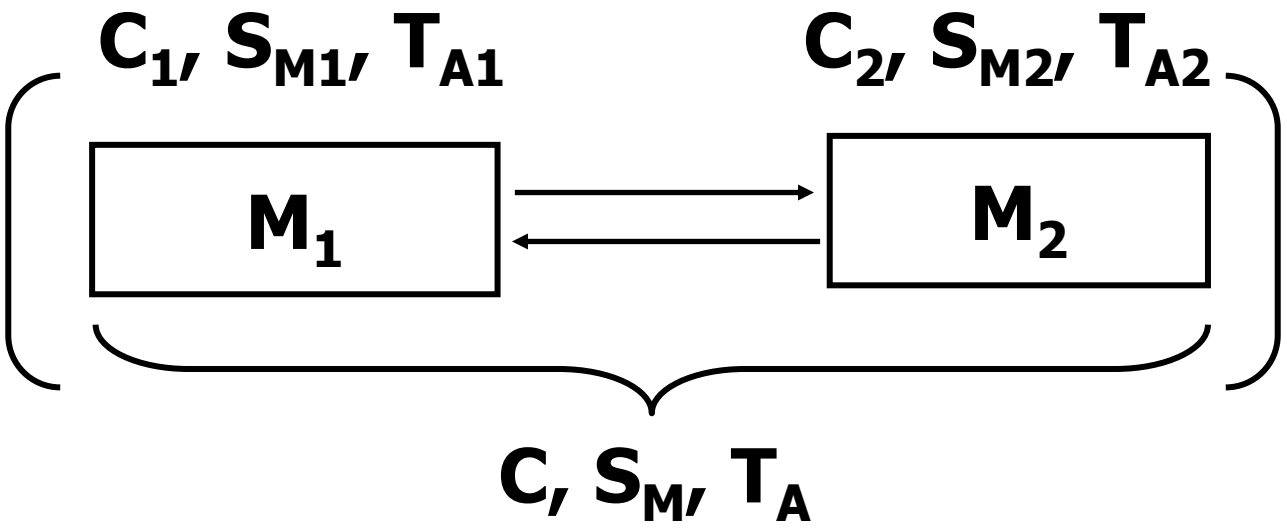
4.1.4 存储体系的性能参数

- 有以下三个：
  - 每位价格c
  - 命中率H
  - 等效访问时间T_A

- 下面以二级存贮层次为例来分析

4.1.4 存贮体系的性能参数

- 二级存贮层次



4.1.4 存贮体系的性能参数

- 每位价格c

$$C = \frac{C_1 S_{M1} + C_1 S_{M1}}{S_{M1} + S_{M2}}$$

  - 为使c接近于c_2，应使$S_{M1} << S_{M2}$

4.1.4 存贮体系的性能参数

- 命中率H
  - CPU产生的逻辑地址能在M_1中访问到的概率。
  - 命中率H可由实验或模拟方法获得。
  - 若逻辑地址流在M_1中访问到的次数为R_1，在M_2中的访问次数为R_2，则：

$$H = \frac{R_1}{R_1 + R_2}$$

  - 显然，不命中率（失效率）＝1－H

4.1.4 存贮体系的性能参数

- 等效访问时间T_A

假设M_1访问和M_2访问是同时启动

$$T_A = HT_{A1} + (1-H)T_{A2}$$

假设M_1访问和M_2访问不是同时启动

$$T_A = HT_{A1} + (1-H)(T_{A1} + T_{A2})$$

$$= T_{A1} + (1-H)T_{A2}$$

希望T_A接近T_{A1}为好。
访问效率 $e = T_{A1}/T_A$，越接近 1 越好。

4.1.4 存贮体系的性能参数

- 二级存储层次的访问效率

$$e=\frac{T_{A1}}{T_A}=\frac{T_{A1}}{HT_{A1}+(1-H)T_{A2}}=\frac{1}{H+(1-H)\times\frac{T_{A2}}{T_{A1}}}$$

  - 存储系统的访问效率主要与命中率和两级存储器的速度之比有关。

- 速度差不要太大
- 命中率越高越好

影响命中率的因素很多，例如：

- 程序的地址流
- 地址预判算法
- M_1的容量等

4.1.4 存贮体系的性能参数

- 例4-3：假设$T_2=5T_1$，在命中率H为0.9和0.99两种情况下，分别计算存储系统的访问效率。

  解：

  当H=0.9时, $e_1=1/(0.9+5(1-0.9))=0.72$

  当H=0.99时, $e_2=1/(0.99+5(1-0.99))=0.96$

4.1.4 存贮体系的性能参数

- 例4-4：在虚拟存储系统中，两级存储器的速度相差特别悬殊$T_2=10^5\ T_1$。如果要使访问效率e=0.9，问需要有多高的命中率？

  解：根据公式可得

$$0.9=\frac{1}{H+(1-H)10^5}$$

  H＝0.99999

- 能获得如此高的命中率吗？

4.1.4 存贮体系的性能参数

- **提高命中率的方法：程序局部性原理**
  - 不命中时，将M_2中相邻的几个单元中的数据一起取出送入M_1。

预取后的命中率：

$$H' = \frac{H + n - 1}{n}$$

其中：

H – 原来的命中率

n – 数据块大小×数据重复使用次数

不命中率降低 n 倍

4.1.4 存贮体系的性能参数

- 例4-5：在一个Cache存储系统中，当Cache的块大小为一个字时，命中率H=0.8；假设数据的重复利用率为5，计算块大小为４个字时，Cache存储系统的命中率是多少？假设$T_2=5T_1$，分别计算访问效率。

4.1.4 存贮体系的性能参数

- 例4-5：解

n=4×5=20, 采用预取技术后, 命中率提高到：

$$H' = \frac{H + n - 1}{n} = \frac{0.8 + 20 - 1}{20} = 0.99$$

Cache块为1个字大时, H=0.8, 访问效率为：

$$e_1 = 1/(0.8 + 5(1 - 0.8)) = 0.55$$

Cache块为4个字大时, H=0.99, 访问效率为：

$$e_2 = 1/(0.99 + 5(1 - 0.99)) = 0.96$$

4.1.4 存贮体系的性能参数

- 例4-6：在一个虚拟存储系统中，T2＝10^5 T1，原来的命中率只有0.8，如果访问磁盘存储器的数据块大小为4K字，并要求访问效率不低于0.9，计算数据在主存储器中的重复利用率至少为多少？
- 解：假设数据在主存储器中的重复利用率为m，则：

$$0.9=\frac{1}{H'+(1-H')\cdot 10^5}\quad ,\quad H'=\frac{0.8+4096m-1}{4096m}$$

解方程组得m=44，即数据在主存储器中的重复利用率至少为44次。