第3章 存储系统

3.1 存储器概述

存储器是计算机系统中的记忆设备，用来存放程序和数据。现代计算机系统都是以存储器为中心，计算机若要开始工作，必须先把有关程序和数据装到存储器中，程序才能开始运行。在程序执行过程中，CPU所需的指令要从存储器中取出，运算器所需的原始数据要从存储器中取出，运算结果必须在程序执行完毕之前全部写到存储器中，各种输入输出设备也直接与存储器交换数据。因此，在计算机运行过程中，存储器是各种信息存储和交换的中心。

3.1.1 基本概念

构成存储器的存储介质，目前主要采用半导体器件和磁性材料。存储器中最小的存储单位可以是一个双稳态半导体电路或一个CMOS晶体管或磁性材料的存储元，它可存储一个二进制代码。这个二进制代码位是存储器中最小的存储单位，称为一个存储位或存储元，若干个存储位可以组成一个存储单元，许多存储单元可以组成一个存储器，这些存储单元的集合也称为存储体。

我们知道，存储器是用来存储程序和数据的，而程序和数据都是用二进制来表示的。二进制数中每一位（bit）“0”或“1”就是由存储器的一个存储位来存储的。存储器的容量以字节（Byte，简称为B）为单位表示，比如640KB、1MB、32MB、1GB等，其相互关系如下：

1KB=1024B，1MB=1024KB，1GB=1024MB，1TB=1024GB

CPU向存储器送入和从存储器取出数据信息时，通常采用较大的信息单位“字”（Word）来工作。

总而言之：

⑴ 位（bit）是二进制数的最小单位，也是数字计算机的最小信息单位，通常用“b”表示。

⑵ 字节（Byte）包含8个bit，通常用“B”表示。存储器容量一般都是以字节为单位的。

⑶ 字（Word）由若干个字节组成。至于一个字到底等于多少个字节，则取决于计算机的字长，即计算机一次所能处理的数据的最大位数。例如，对于32位机，1 Word = 4 Bytes = 32 bits。

3.1.2 存储器的分类

根据存储材料的性能及使用方法的不同，存储器可以有各种不同的分类方法。

1．按存储介质分

存储介质必须满足两个基本要求：

⑴ 必须有两个明显区别的状态，分别表示二进制代码0和1；

⑵ 两个物理状态的改变速度要快，它直接影响存储器的读写速度。

目前使用的存储介质主要是半导体器件、磁存储介质和光存储介质。用半导体器件组成的存储器称为半导体存储器，如计算机主存；用磁性材料做成的存储器称为磁表面存储器，它通过磁头和磁记录介质的相对运动完成读出和写入，如磁盘、磁带；利用激光技术在光存储介质上写入和读出信息的存储器称为光盘存储器，如只读型光盘（CD-ROM、DVD-ROM）、可读写光盘（CD-RW，DVD±RW）、一次性光盘等。

2．按存取方式分

如果任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关，则这种存储器称为随机存储器，如半导体存储器；如果存储单元的内容只能按某种顺序来存取，存取时间与存储单元的物理位置有关，取决于访问存储单元的地址顺序，则这类存储器称为顺序存储器，如磁带存储器。与顺序存储器相比，随机存储器的存取速度快得多，但每一位的价格也要高很多。

3．按存储器的读写功能分

有些半导体存储器中存储的内容是固定不变的，只能读出而不能写入，通常用来存放固定不变的程序、汉字字型库等，在制造芯片时由厂家预先写入，这类半导体存储器称为只读存储器（ROM）；既能读出内容又能写入新内容的半导体存储器称为随机读写存储器（RAM），用来存放正在执行的程序和正在访问的数据。

4．按信息的可保存性分

断电后信息就消失的存储器称为非永久记忆存储器，如半导体存储器 RAM；断电后仍能保存信息的存储器称为永久记忆存储器，如磁介质存储器、光盘存储器。

5．按在计算机系统中的作用分

根据在计算机系统中所起的作用，存储器可分为主存储器、辅助存储器、高速缓冲存储器、控制存储器等。

3.1.3 存储器的分级结构

一个存储器的性能通常用速度、容量、价格三个主要指标来衡量。计算机对存储器的要求是容量大、速度快、成本低，需要尽可能地同时兼顾这三方面的要求。但是一般来讲，存储器速度越快，价格也越高，因而也越难满足大容量的要求。目前通常采用多级存储器体系结构，使用高速缓冲存储器、主存储器和外存储器，如图 3-1 所示。

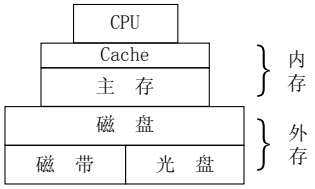

图 3-1 存储系统的分级结构

CPU 能直接访问的存储器称为内存储器（简称内存），包括高速缓冲存储器和主存储器。CPU 不能直接访问的存储器称为外存储器（简称外存，也叫辅助存储器），外存的信息必须调入内存才能被 CPU 使用。

高速缓冲存储器（Cache）是计算机系统中的一个高速、小容量的半导体存储器，它位于高速的 CPU 和低速的主存之间，用于匹配两者的速度，达到高速存取指令和数据的目的。和主存相比，Cache 的存取速度快，但存储容量小。

主存储器，简称主存，是计算机系统的主要存储器，用来存放计算机正在执行的大量程序和数据，主要由 MOS 半导体存储器组成。

外存储器，简称外存，是计算机系统的大容量辅助存储器，用于存放系统中的程序、数据文件及数据库。与主存相比，外存的特点是存储容量大，位成本低，但访问速度慢。目前，外存储器主要有磁盘存储器、磁带存储器和光盘存储器。

由 Cache 和主存储器构成的 Cache－主存系统，其主要目标是利用与 CPU 速度接近的 Cache 来高速存取指令和数据以提高存储器的整体速度，从 CPU 角度看，这个层次的速度接近 Cache，而容量和每一位的价格则接近主存；由主存和外存构成的虚拟存储器系统，其主要目的是增加存储器的容量，从整体上看，其速度接近于主存的速度，其容量则接近于外存的容量。计算机存储系统的这种多层次结构，很好地解决了容量、速度、成本三者之间的矛盾。这些不同速度、不同容量、不同价格的存储器，用硬件、软件或软硬件结合的方式连接起来，形成一个系统。这个存储系统对应用程序员而言是透明的，在应用程序员看来它是一个存储器，其速度接近于最快的那个存储器，存储容量接近于容量最大的那个存储器，单位价格则接近最便宜的那个存储器。

3.1.4 半导体存储器芯片

半导体存储器芯片按照读写功能可分为随机读写存储器（Random Access Memory，RAM）和只读存储器（Read Only Memory，ROM）两大类。RAM 可读可写，断电时信息会丢失；ROM 中的内容只能读出，不能写入，信息可永久保存，不会因为断电而丢失。

1．随机读写存储器

目前广泛使用的半导体随机读写存储器是 MOS 半导体存储器，按保存数据的机理分为静态存储器（Static RAM，SRAM）和动态存储器（Dynamic RAM，DRAM）。

1）静态存储器（SRAM）

利用双稳态触发器来保存信息，只要不断电信息就不会丢失。

静态存储器的集成度低，成本高，功耗较大，通常作为 Cache 的存储体。

2）动态存储器（DRAM）

利用 MOS 电容存储电荷来保存信息，使用时需要不断给电容充电才能保持信息。

动态存储器电路简单，集成度高，成本低，功耗小，但需要反复进行刷新（Refresh）操作，工作速度较慢，适合作为主存储器的主体部分。

刷新操作：为防止存储的信息电荷泄漏而丢失信息，由外界按一定规律不断地给栅极进行充电，补足栅极的信息电荷。

DRAM 工作时必须要有刷新控制电路，操作比较复杂。由于要不间断地进行刷新，故称这种存储器为动态存储器。动态 MOS 存储器主要采用“读出”的方式进行刷新，依次读出存储器的每一行，就可完成对整个 DRAM 的刷新。

DRAM 存储器的刷新需要有硬件线路的支持，这些控制线路可以集成在一个半导体芯片上，形成 DRAM 控制器。借助于 DRAM 控制器，可以把 DRAM 当作 SRAM 一样使用，从而为系统设计带来很大的方便。

3）增强型 DRAM（EDRAM）

EDRAM 芯片是在 DRAM 芯片上集成一个高速小容量的 SRAM 芯片而构成的，这个小容量的 SRAM 芯片起到高速缓存的作用，从而使 DRAM 芯片的性能得到显著改进。

当 CPU 从主存 DRAM 中读取数据时，会将包含此数据的整个数据块都写入高速缓存 SRAM 内，下次读取连续地址数据时，CPU 就可以从这个 SRAM 中直接取用，而不必到较慢的 DRAM 中读取，如此即可加快 CPU 的存取速度。

将由若干 EDRAM 芯片组成的存储模块做成小电路插件板形式，就是目前普遍使用的内存条。

2．只读存储器

只读存储器 ROM 是一种存储固定信息的存储器，其特点是在正常工作状态下只能读取数据，不能即时修改或重新写入数据。

只读存储器电路结构简单，且存放的数据在断电后不会丢失，特别适合于存储永久性的、不变的程序代码或数据（如常数表、函数、表格和字符等），计算机中的自检程序就是固化在 ROM 中的。

ROM 的最大优点是具有不易失性。

只读存储器有不可重写只读存储器（MROM、PROM）和可重写只读存储器（EPROM、EEPROM、闪速存储器等）两大类。

1）不可重写只读存储器

⑴掩模只读存储器（MROM）

掩模只读存储器，又称固定 ROM。这种 ROM 在制造时，生产厂家利用掩模（Mask）技术把信息写入存储器中，使用时用户无法更改，适宜大批量生产。

掩模只读存储器可分为二极管 ROM、双极型三极管 ROM 和 MOS 管 ROM 三种类型。

⑵可编程只读存储器（PROM）

可编程只读存储器（Programmable ROM，简称 PROM），是可由用户一次性写入信息的只读存储器，是在 MROM 的基础上发展而来的。

PROM 的缺点是用户只能写入一次数据，一经写入就不能再更改。

2）可重写只读存储器

这类 ROM 由用户写入数据（程序），当需要变动时还可以进行修改，使用起来比较方便。可重写 ROM 有紫外线擦除 EPROM、电擦除 EEPROM 和闪速存储器 Flash ROM 三种类型。

⑴光擦可编程只读存储器（EPROM）

EPROM 的特点是其中的内容可以用特殊的装置进行擦除和重写。EPROM 出厂时，其存储内容为全“1”，用户可根据需要改写为“0”，当需要更新存储内容时，可将原存储内容擦除（恢复为全“1”），以便写入新的内容。

EPROM 一般是将芯片置于紫外线下照射 15~20 分钟左右，以擦除其中的内容，然后用专用的设备（EPROM 写入器）将信息重新写入，一旦写入则相对固定。

在闪速存储器大量应用之前，EPROM 常用于软件开发过程中。

⑵电擦可编程只读存储器（EEPROM 或 E^2PROM）

用紫外线擦除 EPROM 的操作复杂，速度很慢。EEPROM 可以用电气方法将芯片中的存储内容擦除，擦除时间较快，甚至可以在联机状态下操作。

EEPROM 既可使用字擦除方式又可使用块擦除方式，使用字擦除方式可擦除一个存储单元，使用块擦除方式可擦除数据块中所有存储单元。

⑶闪速存储器（Flash ROM）

闪速存储器 Flash ROM 是 20 世纪 80 年代中期出现的一种块擦写型存储器，是一种高密度、非易失性的读/写半导体存储器，它突破了传统的存储器体系，改善了现有存储器的特性。

Flash ROM 中的内容或数据不像 RAM 一样需要电源支持才能保存，但又像 RAM 一样具有可重写性。在某种低电压下，其内部信息可读不可写，类似于 ROM，而在较高的电压下，其内部信息可以更改和删除，类似于 RAM。

Flash ROM 可以用软件在 PC 机中改写或在线写入，信息一旦写入即相对固定。因此，在 PC 机中可用于存储主板的 BIOS 程序。由于能进行改写，便于用户自行升级 BIOS，但这也给病毒以可乘之机，著名的 CIH 病毒正是利用这个特点来破坏 BIOS，从而导致整个系统瘫痪的。

另外，由于单片存储容量大，易于修改，Flash ROM 也常用于数码相机和 U 盘中，因其具有低功耗、高密度等特点，且没有机电移动装置，特别适合于便携式设备，成为替代磁盘的一种理想工具。

3.2 主存储器

3.2.1 主存储器的技术指标

主存储器是 CPU 能直接访问的存储器，由随机读写存储器 RAM 和只读存储器 ROM 组成，能快速地进行读或写操作。衡量一个主存储器性能的技术指标主要有存储容量、存取时间、存储周期和存储器带宽。

1．存储容量

在一个存储器中可以容纳的存储单元的总数称为存储容量（Memory Capacity）。

存储单元可分为字存储单元和字节存储单元。所谓字存储单元，是指存放一个机器字的存储单元，相应的单元地址称为字地址；而字节存储单元，是指存放 1 个字节（8 位二进制数）的存储单元，相应的地址称为字节地址。如果一台计算机中可编址的最小单位是字存储单元，则该计算机称为按字编址的计算机；如果一台计算机中可编址的最小单位是字节存储单元，则该计算机称为按字节编址的计算机。一个机器字可以包含数个字节，所以一个字存储单元也可包含数个字节存储单元。

为了描述方便和统一，目前大多数计算机采用字节为单位来表征存储容量。在按字节寻址的计算机中，存储容量的最大字节数可由地址码的位数来确定。例如，一台计算机的地址码为 n 位，则可产生 2^n 个不同的地址码，如果地址码被全部利用，则其最大容量为 2^n 个字节。一台计算机设计定型以后，其地址总线、地址译码范围也已确定，因此其最大存储容量是确定的，而实际配置存储容量时，只能在这个范围内进行选择，通常情况下主存储器的实际存储容量远远小于理论上的最大容量。一般而言，存储器的容量越大，所能存放的程序和数据就越多，计算机的解题能力就越强。

存储容量的单位通常用 KB、MB、GB 来表示，K 代表 2^{10}，M 代表 2^{20}，G 代表 2^{30}。

1KB=1024B，1MB=1024KB，1GB=1024MB

2．存取时间

存取时间即存储器访问时间（Memory Access Time），是指启动一次存储器操作到完成该操作所需的时间。

具体地说，读出时为取数时间，写入时为存数时间。取数时间就是指存储器从接受读命令到信息被读出并稳定在存储器数据寄存器中所需的时间；存数时间就是指存储器从接受写命令到把数据从存储器数据

寄存器的输出端传送到存储单元所需的时间。

3．存储周期

存储周期又称为访问周期，是指连续启动两次独立的存储器操作所需间隔的最小时间，它是衡量主存储器工作性能的重要指标。存储周期通常略大于存取时间。

4．存储器带宽

存储器带宽是指单位时间里存储器所存取的信息量，是衡量数据传输速率的重要指标，通常以位/秒（bps，bit per second）或字节/秒（Byte/s）为单位。

例如，总线宽度为32位，存储周期为250ns，则

存储器带宽 = 32b/250ns = 128Mb/s = 128Mbps

存取时间、存储周期、存储器带宽都反映了主存的速度指标。

3.2.2 主存储器的基本组成

主存储器由存储体、寻址系统、存储器数据寄存器、读写系统及控制线路等组成，如图3-3所示。

EMBED Visio.Drawing.11

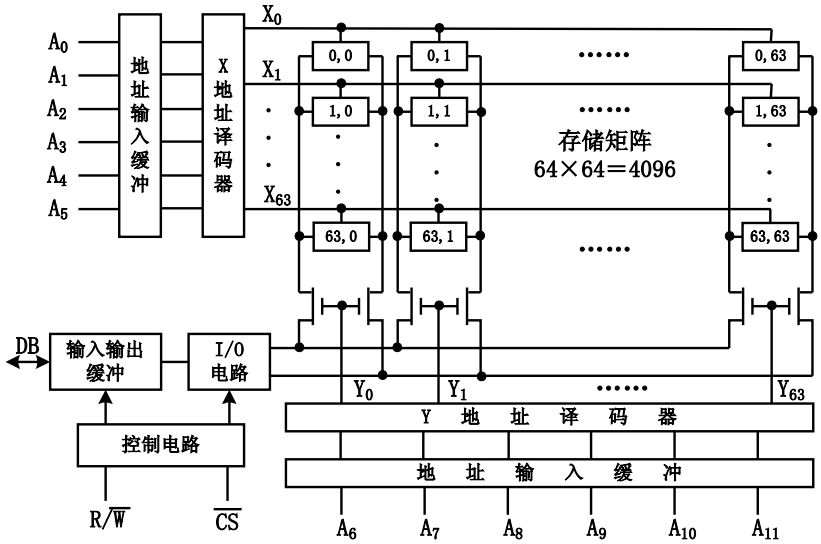

图3-3 主存储器的基本组成

1．存储体

存储体是一个由存储单元按照一定规则排列起来的存储阵列。存储体是存储器的核心，是存储信息的实体。

2．寻址系统

寻址系统就是读出和写入信息的地址选择机构，包括存储器地址寄存器（MAR）和地址译码器。

地址译码器接收来自地址寄存器的n位地址，经译码后产生2^n个地址选择信号，并从2^n个单元中选出一个单元。通常用X选择线（行线）和Y选择线（列线）的交叉来选择所需要的单元。

存储器地址寄存器MAR具有地址缓冲功能，可使CPU和主存的速度都得到充分发挥和提高。MAR从功能上看属于主存，但在一些微型机中常被放在CPU内，并可兼作别用，在速度要求较高的计算机中，CPU与主存中都设有地址寄存器。

3．存储器数据寄存器（MDR）

一般把存储器数据寄存器MDR作为存储器接收输入数据和发出输出数据用的数据缓冲器件。在数据传送中，它可以起到数据缓冲作用，使CPU与主存速度相匹配，从而使两者的速度都能得到发挥和提高。

4．读写系统

读写系统包括写入信息和读出信息所需线路。写入信息所需线路包括写入线路、写驱动器等；读出信息所需线路包括读出线路、读驱动器和读出放大器等。

5．控制线路

无论是读或写操作，都需要由一系列明确规定的连续操作步骤来完成，这就需要主存时序线路、时钟脉冲线路、读逻辑控制线路、写或重写逻辑控制线路以及动态存储器的定时刷新线路等，这些线路总称为存储器控制线路。存储器控制线路控制逻辑电路接收片选信号 CS（Chip Select）及来自 CPU 的读/写控制信号，形成芯片内部控制信号，并控制数据的读出和写入。

主存储器的工作原理：由 CPU 发来的地址送到存储器地址寄存器中，在读写控制线路的作用下，经过地址译码后，选中存储体中某一存储单元，对该存储单元进行读/写操作，读出或写入的信息都暂存于存储器数据寄存器中。

3.2.3 主存储器的扩展

CPU 对存储器进行读/写操作，首先由地址总线给出地址信号，然后发出读操作或写操作的控制信号，最后在数据总线上进行信息交流。因此，存储器同 CPU 连接时，要完成地址线、数据线和控制线的连接。

目前生产的存储器芯片的容量是有限的，在字数或字长方面与存储器的实际要求都有差距，所以需要在字向和位向两方面进行扩充才能满足实际存储器的容量要求，通常采用位扩展法、字扩展法、字位同时扩展法。

1．位扩展法

假定使用 8K×1 位的 RAM 芯片，那么组成 8K×8 位的存储器，可采用图 3-4 所示的位扩展法，此时只需把字长由 1 位加大到 8 位，而存储器的字数（8K）则与存储器芯片的字数一致。图中，每一片 RAM 的字数是 8K（2^{13}），故其地址线为 13 条（A_0~A_{12}），可满足整个存储体容量的要求；每一片 RAM 对应数据的 1 位（只有 1 条数据线），故只需将它们分别接到数据总线上的相应位即可。在这种方式中，对芯片没有选片要求，就是说芯片均按已被选中来考虑。在这种连接中，每一条地址总线接有 8 个负载，每一条数据线接有 1 个负载。

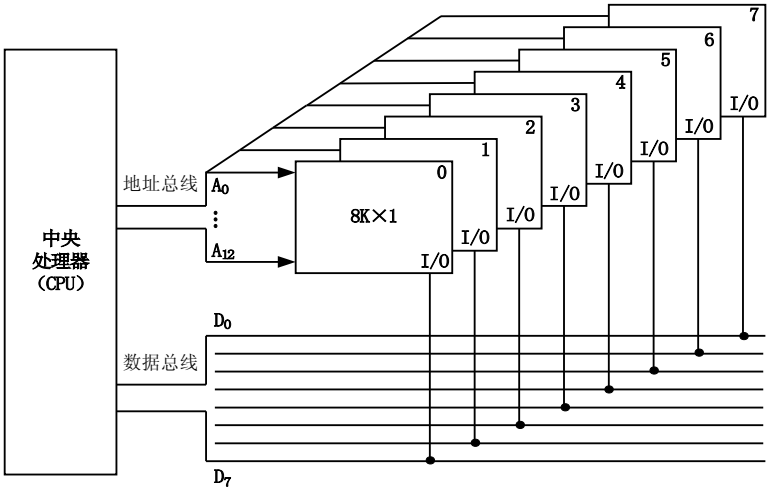

图 3-4 位扩展法组成 8K×8 位存储器

2．字扩展法

字扩展法仅在字向扩充，而位数不变，因此可以将芯片的地址线、数据线、读/写控制线并联，而由片选信号来区分芯片的具体地址，故片选信号端连接到选片译码器的输出端。

使用 16K×8 位的 RAM 芯片，采用字扩展法组成 64K×8 位存储器的连接如图 3-5 所示，其中每一片 RAM 的字数是 16K（2^{14}），故其地址线为 14 条（A_0~A_{13}）。图中，4 片芯片的数据线与数据总线的 D_0~D_7 相连，对应 8 位数据，地址总线低位地址 A_0~A_{13} 与各片芯片的 14 位地址端相连，两位高位地址 A_{14}、A_{15} 经 2:4 译码器与 4 片芯片的片选端相连，其地址空间分配见表 3-1。

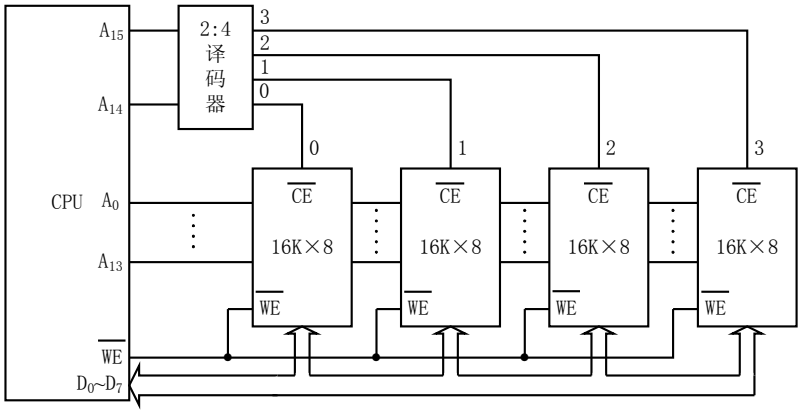

图 3-5 字扩展法组成 64K×8 位存储器

表 3-1 图 3-5 的地址空间分配表

地址 片号	片外	A_{15}	A_{14}	片内	A_{13}	A_{12}	A_{11}	…	A_1	A_0	说明
0		0	0		0	0	0	…	0	0	最低地址
		0	0		1	1	1	…	1	1	最高地址
1		0	1		0	0	0	…	0	0	最低地址
		0	1		1	1	1	…	1	1	最高地址
2		1	0		0	0	0	…	0	0	最低地址
		1	0		1	1	1	…	1	1	最高地址
3		1	1		0	0	0	…	0	0	最低地址
		1	1		1	1	1	…	1	1	最高地址

3．字位同时扩展法

一个存储器的容量假定为 M×N 位，若使用 L×K 位的芯片（L<M, K<N），则需要在字向和位向同时进行扩展，此时共需要（M/L）×（N/K）个存储器芯片。

3.3 高速存储器

高速 CPU 与主存储器在速度上不相匹配，而且在一个 CPU 周期中有可能需要用到几个存储器字，这成为了限制高速计算的主要问题。为了使 CPU 不至于因为等待存储器读写操作的完成而无事可做，可以采取一些特殊措施，以加速 CPU 和存储器之间的有效传输：

⑴主存储器采用更高速的技术来缩短存储器的读出时间，或加长存储器的字长；

⑵采用并行操作的双端口存储器；

⑶在 CPU 和存储器之间插入一个高速缓冲存储器（Cache），以缩短读出时间

⑷在每个存储器周期中存取几个字。

3.3.1 双端口存储器

常规存储器是单端口存储器，每次只接收一个地址，访问一个存储单元，从中读取或写入一个字节或字。主存储器是信息交换的中心，一方面 CPU 频繁地与主存交换信息，另一方面外设也较频繁地与主存交换信息，而单端口存储器每次只能接受一个访存者，或是读或是写，这就影响到存储器的整体工作速度。为此，在某些系统中采用了双端口存储器。

图 3-6 所示的双端口存储器具有两个彼此独立的读写口，每个读写口都有一套自己的地址寄存器和译码电路，可以并行地独立工作。两个读写口可以按各自接收的地址同时读出或写入，或一个写入而另一个读出。与两个独立的存储器不同，两个读写口的访存空间相同，可以访问同一个存储单元。通常使双端口存储器的一个读写口面向 CPU，另一个读写口则面向外设或输入输出处理机。

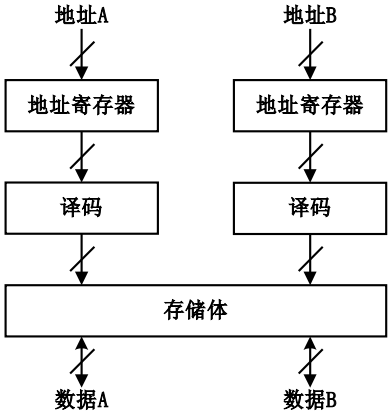

图 3-6 双端口存储器

由此可见，双端口存储器由于具有两组相互独立的读写控制线路，可以对存储器中任何位置上的数据进行并行、独立的存取操作，因而是一种高速工作的存储器。

如果两个端口同时访问存储器的同一个存储单元，便会发生读写冲突。为解决此问题，可以设置一个“忙”标志。在发生读写冲突时，片上判断逻辑决定对哪个端口优先进行读写操作，而对另一个被延迟的端口置“忙”标志，即暂时关闭此端口。等到优先端口完成读写操作，才将被延迟端口的“忙”标志复位，重新开放此端口，允许延迟端口进行存取。

3.3.2 多模块交叉存储器

1.存储器的模块化组织

速度和容量是主存储器设计的两大主要课题，计算机的发展对主存储器提出了更高速度和更大容量的要求。如果主存储体由物理上互相分隔的若干个模块构成，并给每个模块配置自己的存储器地址寄存器（MAR）、存储器数据寄存器（MDR）和读写电路，使每个模块都成为一个能独立进行读写操作的存储器，那么就有可能在任一给定时刻对几个模块同时执行读或写操作，从而提高整个主存的平均存取时间。由多个能独立操作的模块所组成的存储器，称为多模块存储器（多体存储器），其地址分配方案有两种：

1）顺序方式

在常规的主存储器设计中，访问地址采用顺序方式，如图 3-7 所示。CPU 送来的主存地址被分成高 n 位和低 m 位。主存地址的高 n 位表示模块号，其模块号为 0,1,2,…,2^n-1，共 2^n 个，译码后从 2^n 个模块中选中一个模块；主存地址的低 m 位表示块内地址，m 位译码后，选定模块中的一个具体的存储字单元。在一个模块内，程序从低位地址连续存放。这样，当 CPU 执行对主存连续单元的读写请求时，只有一个模块和 CPU 进行数据存取操作，其他模块则可停止工作或与外部设备进行直接存储器存取（DMA）操作。

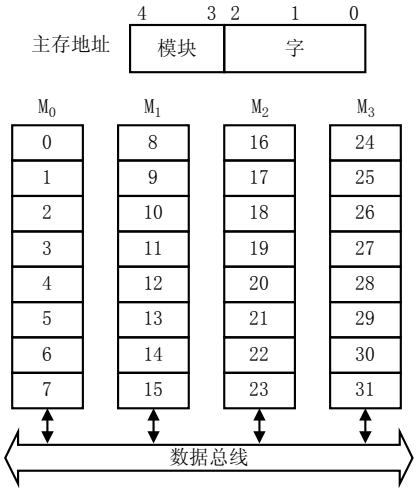

图 3-7 存储器模块的顺序组织方式

顺序方式的优点是可以通过简单地增加模块而方便地扩展系统存储容量，并且一旦发生故障，出现故障的模块仅仅影响存储空间的一个局部区域，其他模块可以照常工作。但在这种结构中，各模块彼此串行工作，存储器带宽受到很大限制，难以有效提高主存速度。

2）交叉方式

多模块存储器的另一种地址分配方案是交叉方式，如图 3-8 所示。图中的存储器容量为 32 个字，分成 4 个模块，每个模块 8 个字。将 4 个线性地址 0,1,2,3 依次分配给 M_0,M_1,M_2,M_3 模块，再将线性地址 4,5,6,7 依次分给 M_0,M_1,M_2,M_3 模块，……。当存储器寻址时，用地址寄存器的低 2 位选择 4 个模块中的 1 个，而用高 3 位选择模块中的 8 个字。

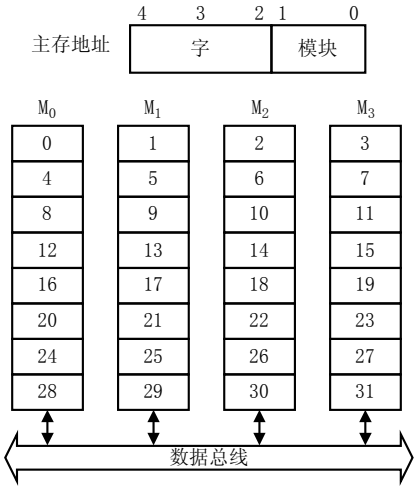

图 3-8 存储器模块的交叉组织方式

主存地址的低 n 位表示模块号，高 m 位表示块内地址，使得连续地址分布在相邻的不同模块内，而同一个模块内的地址都是不连续的，容量相同的不同模块各自以等同的方式与 CPU 交换信息。图中的模块 M_0,M_1,M_2,M_3 采用地址交叉编址方法，即将单元地址依次排在各个模块中，称为模 4 交叉编址。若有 m 个模块，则称为模 m 交叉编址。

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（如图 3-9 所示）。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。

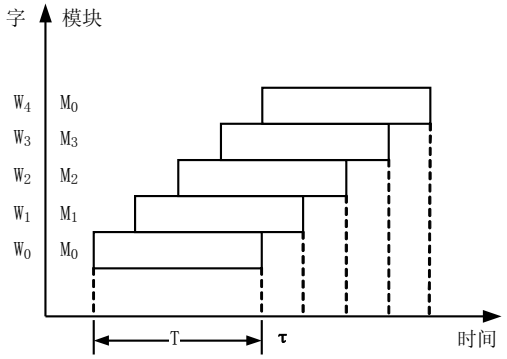

图 3-9 四模块交叉存储器的流水线方式存取示意图

CPU 访问 4 个存储体，可以分时启动、分时控制，即每经过 $\frac{1}{4}$ 个存储周期 T 就访问一个模块。在 0 时刻启动模块 M_0，在 $\frac{1}{4}T,\frac{2}{4}T,\frac{3}{4}T$ 时刻分别启动模块 M_1,M_2,M_3，经 T 周期后，四个模块就都进入了并行工作状态，各模块在存储周期内互相重叠访问。这样，对每个存储体来说，存储周期是 T，而对 CPU 来说，存储周期为

$\frac{T}{4}$。在理想情况下，主存的周期缩短为$\frac{T}{n}$，n 为模块数，T 为每个存储体的存储周期。程序转移及操作数寻址通常会导致对存储器的访问不是按顺序地址，甚至有可能多个地址访问同一个模块，这些都使得交叉存储器的实际有效周期略低于理论上的有效周期，但交叉存储器能大大提高存取速度则是显而易见的。

这种方案也有明显的不足，即任一模块一旦出现故障，都将影响到整个存储器。

2. 多模块交叉存储器的基本结构

图 3-10 所示为四模块交叉存储器的结构框图。

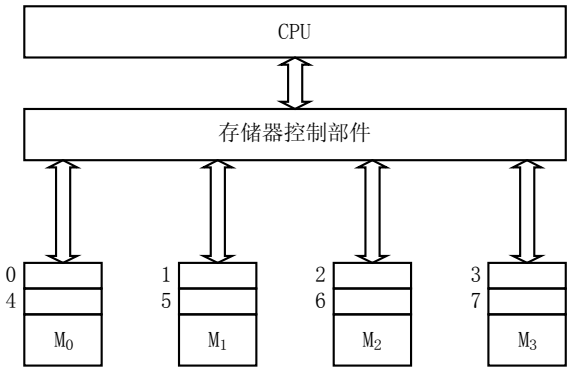

图 3-10 四模块交叉存储器结构框图

主存被分为 4 个相互独立、容量相同的模块，每个模块有自己的读写控制电路、地址寄存器和数据寄存器，各自以同等的方式与 CPU 交换信息。地址码的低位字段经过译码选择不同的模块，而高位字段则指向相应模块内的存储字。连续地址分布在相邻的不同模块内，同一个模块内的地址都是不连续的。

因此，借由交叉存储方式，可以实现对连续字成块传送的多模块流水式并行存取。CPU 同时访问 4 个模块，由存储器控制部件控制它们分时使用数据总线进行信息传递。对每一个存储器模块而言，从 CPU 给出访存命令直到读出信息仍然使用一个存取周期时间，但对 CPU 而言，它可以在一个存取周期内连续访问 4 个模块，各模块的读写过程重叠进行。所以多模块交叉存储器是一种并行存储器结构，可以大大提高存储器的带宽。

3.3.3 相联存储器

1. 相联存储器的基本原理

前面介绍的存储器都是按地址访问的存储器，而相联存储器则是按内容访问的存储器。

相联存储器是选择记录中的一个字段内容作为地址来存取的存储器。选用来寻址存储器的字段叫做关键字。例如，存储器中存放学生信息，如果选学号作为关键字，就用所给的学号作为地址来访问存储器，当要查找某个学号学生的其他信息时，就可以通过学号直接访问存储器，得到相关信息。

存放在相联存储器中的项可以看成具有下列格式：

KEY，DATA

其中 KEY 是地址，DATA 是被读写的信息。

相联存储器的基本原理是：把存储单元所存内容的某一部分作为检索项（即关键字项），用来检索存储器，并读出或写入存储器中与该检索项相符的存储单元的内容。

2. 相联存储器的组成

相联存储器由存储体、检索寄存器、屏蔽寄存器、符合寄存器、比较线路、代码寄存器、控制线路等组成，如图 3-11 所示。

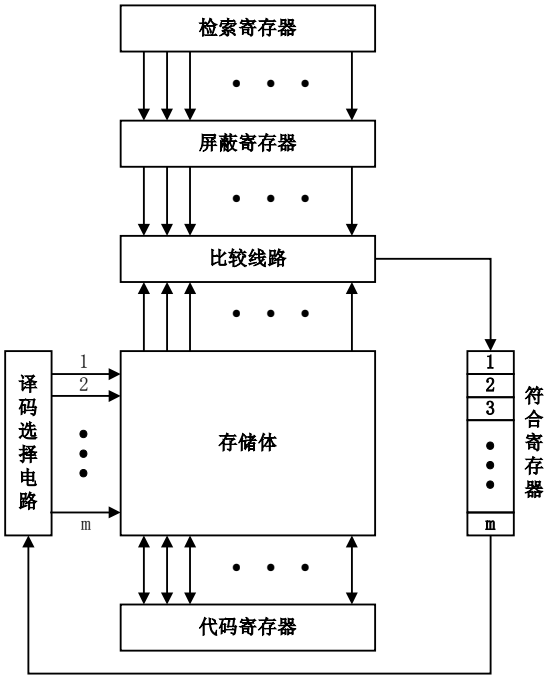

图 3-11 相联存储器的组成框图

存储体：由高速半导体存储器构成，以求快速存取。

检索寄存器：用来存放检索字，其位数与相联存储器的存储单元的位数相等，每次检索时，取其中若干位作为检索项（即关键字项）。◆

屏蔽寄存器：用来存放屏蔽码，其位数与检索寄存器位数相同，检索项所对应的位值为“1”，其他位值均为“0”。屏蔽寄存器用来将检索寄存器中除检索项以外的位置“0”。

符合寄存器：用来存放按检索项内容进行检索的存储体中与之符合的单元地址，其位数等于相联存储器的存储单元数，每一位对应一个存储单元，位的序数即为相联存储器的单元地址。

比较线路：把检索项和从存储体中读出的所有单元内容的相应位进行比较，如果有某个存储单元与检索项符合，就把符合寄存器的相应位置“1”，表示该字已被检索。◆

代码寄存器：用来存放存储体中读出的数据，或者存放向存储体中写入的数据。

在计算机系统中，相联存储器主要用于虚拟存储器中存放段表、页表和快表，以及高速缓冲存储器 Cache 中存放块地址。这是因为，在这两种应用中，都需要快速查找。

3.4 高速缓冲存储器（Cache）

在计算机系统中，CPU 的运行速度与主存的访问速度极不匹配。例如，800MHz Pentium III CPU 一条指令的执行时间约为 1.25ns，而 133MHz SDRAM 的存取时间为 7.5ns，即在 83%的时间内 CPU 都处于等待状态，运行效率极低。为了提高 CPU 利用率，现代计算机体系结构中广泛采用高速缓冲存储器（Cache）技术。

3.4.1 Cache基本原理

在设计和开发系统程序和应用程序时，程序员通常采用模块化的程序设计方法。某一模块的程序，往往集中在存储器逻辑地址空间中很小的一块范围内，且程序地址分布是连续的。也就是说，CPU 在一段较短的时间内，是对连续地址的一段很小的主存空间频繁地进行访问，而对此范围以外地址的访问甚少，这种现象称为程序访问的局部性。

高速缓冲存储器（Cache）技术就是利用程序访问的局部性原理，把程序中正在使用的部分（活跃块）存放在一个小容量的高速 Cache 中，使 CPU 的访存操作大多针对 Cache 进行，从而解决高速 CPU 和低速主存之间速度不匹配的问题，使程序的执行速度大大提高。

1. Cache 的功能

Cache 是介于 CPU 和主存之间的小容量存储器，存取速度比主存快，接近 CPU。它能高速地向 CPU 提供指令和数据，提高程序的执行速度。Cache 技术是为了解决 CPU 和主存之间速度不匹配而采用的一项重要技术。

Cache 是主存的缓冲存储器，由高速的 SRAM 组成，所有控制逻辑全部由硬件实现，对程序员而言是透明的。随着半导体器件集成度的不断提高，当前有些 CPU 已内置 Cache，并且出现了两级以上的多级 Cache 系统。Cache 系统与 CPU 和主存的关系如图 3-12 所示。

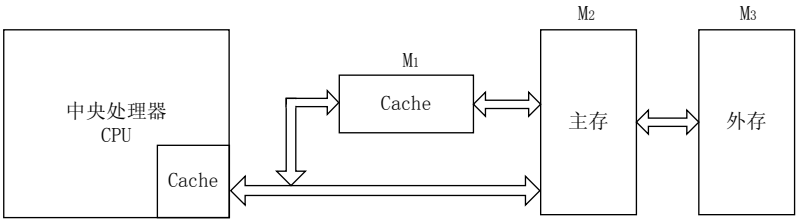

图 3-12 Cache 与 CPU 和主存的关系

2. Cache 的基本原理

CPU 与 Cache 之间的数据交换是以字为单位的，而 Cache 与主存之间的数据交换则是以块为单位的。一个块由若干个定长字组成。

当 CPU 读取主存中的一个字时，该字的主存地址被发给 Cache 和主存，此时，Cache 控制逻辑依据地址判断该字当前是否存在于 Cache 中：若在，该字立即被从 Cache 传送给 CPU；若不在，则用主存读周期把该字从主存读出送到 CPU，同时把含有这个字的整个数据块从主存读出送到 Cache 中，并采用一定的替换策略将 Cache 中的某一块替换掉，替换算法由 Cache 管理逻辑电路来实现。

Cache 原理图如图 3-13 所示。图中，按内容寻址的相联存储器（表），用于存放与 Cache 中数据相对应的主存地址，可以快速检索、判断 CPU 读取的某个字当前是否存在于 Cache 中。

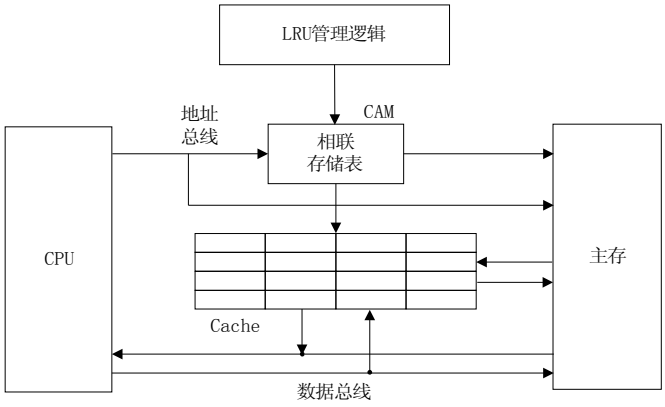

图 3-13 Cache 原理图

3. Cache 的命中率◆

基于程序访问的局部性原理，增加 Cache 使得要访问的数据绝大多数都可以在 Cache 中找到，这样才能在性能上使主存的平均读出时间尽可能接近 Cache 的读出时间。Cache 的工作效率通常用“命中率”来表示。

命中率指的是 CPU 要访问的信息在 Cache 中的概率，Cache 的命中率越高，CPU 访问主存的速度就越接近访问 Cache 的速度。通常 Cache 的容量越大，存储的块也越多，CPU 的命中率就越高。但是，当 Cache 的容量达到一定值时，命中率并不会随着容量的增大而增加，而且 Cache 容量的增大将导致成本的增加，所以，Cache 的容量一般是命中率与成本价格的折中。

在一个程序执行期间，设 Nc 表示 Cache 完成存取的总次数，Nm 表示主存完成存取的总次数，h 定义为命中率，则有

$$h = \frac{N_c}{N_c + N_m} \qquad \text{(式 3-1)}$$

若 t_c 表示命中时的 Cache 访问时间，t_m 表示未命中时的主存访问时间，1-h 表示未命中率，则 Cache－主存系统的平均访问时间 t_a 为：

$$t_a = ht_c + (1 - h)t_m \qquad \text{(式 3-2)}$$

设 e 表示访问效率，则有

$$e = \frac{t_c}{t_a} \qquad \text{(式 3-3)}$$

为提高访问效率 e，命中率 h 越接近 1 越好。命中率 h 与程序的行为、Cache 的容量、组织方式、块的大小有关。

3.4.2 地址映射

Cache 的容量很小，它保存的内容只是主存内容的一个子集，且 Cache 与主存的数据交换是以块为单位的。为了把信息放到 Cache 中，必须应用某种函数把主存地址定位到 Cache 中，这称为地址映射。在信息按这种映射关系装入 Cache 后，CPU 执行程序时，会将程序中的主存地址变换成 Cache 地址，这个变换过程叫做地址变换。

Cache 的地址映射方式有直接映射、全相联映射和组相联映射。假设某台计算机主存容量为 1 MB，被分为 2048 块，每块 512B；Cache 容量为 8KB，被分为 16 块，每块也是 512B。下面以此为例介绍三种基本的地址映射方法。

1. 直接映射

直接映射的 Cache 组织如图 3-14 所示。主存中的一个块只能映射到 Cache 的某一特定块中去。例如，主存的第 0 块、第 16 块、……、第 2032 块，只能映射到 Cache 的第 0 块；而主存的第 1 块、第 17 块、……、第 2033 块，只能映射到 Cache 的第 1 块……。

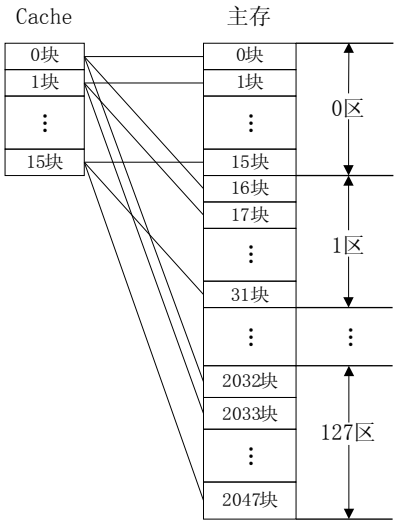

图 3-14 直接映射示意图

直接映射是最简单的地址映射方式，它的硬件简单，成本低，地址变换速度快，而且不涉及替换算法问题。但是这种方式不够灵活，Cache 的存储空间得不到充分利用，每个主存块只有一个固定位置可存放，容易产生冲突，使 Cache 效率下降，因此只适合大容量 Cache 采用。例如，如果一个程序需要重复引用主存中第 0 块与第 16 块，最好将主存第 0 块与第 16 块同时复制到 Cache 中，但由于它们都只能复制到 Cache 的第 0 块中去，即使 Cache 中别的存储空间空着也不能占用，因此这两个块会不断地交替装入 Cache 中，导致命中率降低。

2. 全相联映射

图 3-15 是全相联映射的 Cache 组织，主存中任何一块都可以映射到 Cache 中的任何一块位置上。

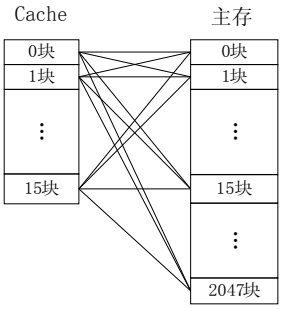

图 3-15 全相联映射示意图

全相联映射方式比较灵活，主存的各块可以映射到 Cache 的任一块中，Cache 的利用率高，块冲突概率低，只要淘汰 Cache 中的某一块，即可调入主存的任一块。但是，由于 Cache 比较电路的设计和实现比较困难，这种方式只适合于小容量 Cache 采用。

3. 组相联映射

组相联映射实际上是直接映射和全相联映射的折中方案，其组织结构如图 3-16 所示。主存和 Cache 都分组，主存中一个组内的块数与 Cache 中的分组数相同，组间采用直接映射，组内采用全相联映射。也就是说，将 Cache 分成 u 组，每组 v 块，主存块存放到哪个组是固定的，至于存到该组哪一块则是灵活的。例如，主存分为 256 组，每组 8 块，Cache 分为 8 组，每组 2 块。

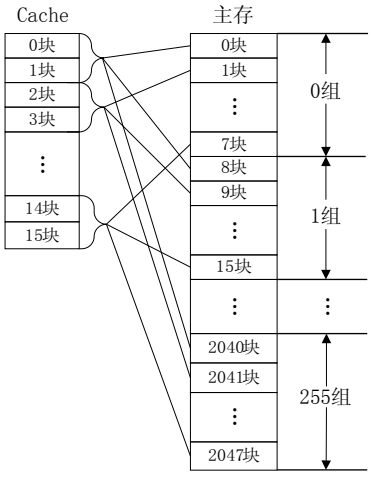

图 3-16 组相联映射示意图

主存中的各块与 Cache 的组号之间有固定的映射关系，但可自由映射到对应 Cache 组中的任何一块。例如，主存中的第 0 块、第 8 块……均映射于 Cache 的第 0 组，但可映射到 Cache 第 0 组中的第 0 块或第 1 块；主存的第 1 块、第 9 块……均映射于 Cache 的第 1 组，但可映射到 Cache 第 1 组中的第 2 块或第 3 块。

常采用的组相联结构 Cache，每组内有 2、4、8、16 块，称为 2 路、4 路、8 路、16 路组相联 Cache。组相联结构 Cache 是前两种方法的折中方案，适度兼顾二者的优点，尽量避免二者的缺点，因而得到普遍采用。

3.4.3 替换策略

Cache 工作原理要求它尽量保存最新数据，当从主存向 Cache 传送一个新块，而 Cache 中可用位置已被占满时，就会产生 Cache 替换的问题。替换问题与 Cache 的组织方式紧密相关：对直接映射 Cache 来说，只要把此可用位置上的主存块换出 Cache 即可；对全相联和组相联 Cache 来说，要从若干个可用位置中选取一个位置，把其中的主存块换出 Cache。

常用的替换算法有下面三种。

1. 最不经常使用（LFU）算法◆

LFU（Least Frequently Used，最不经常使用）算法将一段时间内被访问次数最少的那个块替换出去。每块设置一个计数器，从0开始计数，每访问一次，被访块的计数器就增1。当需要替换时，将计数值最小的块换出，同时将所有块的计数器都清零。

这种算法将计数周期限定在对这些特定块两次替换之间的间隔时间内，不能严格反映近期访问情况，新调入的块很容易被替换出去。

2. 近期最少使用（LRU）算法

LRU（Least Recently Used，近期最少使用）算法是把CPU近期最少使用的块替换出去。这种替换方法需要随时记录Cache中各块的使用情况，以便确定哪个块是近期最少使用的块。每块也设置一个计数器，Cache每命中一次，命中块计数器清零，其他各块计数器增1。当需要替换时，将计数值最大的块换出。

LRU算法相对合理，但实现起来比较复杂，系统开销较大。这种算法保护了刚调入Cache的新数据块，具有较高的命中率。LRU算法不能肯定调出去的块近期不会再被使用，所以这种替换算法不能算作最合理、最优秀的算法。但是研究表明，采用这种算法可使Cache的命中率达到90%左右。

3. 随机替换

最简单的替换算法是随机替换。随机替换算法完全不管Cache的情况，简单地根据一个随机数选择一块替换出去。随机替换算法在硬件上容易实现，且速度也比前两种算法快。缺点则是降低了命中率和Cache工作效率。

Cache命中率除了和替换算法有关外，还与Cache的容量及块的大小有关。

3.4.4 写操作策略

由于Cache的内容只是主存内容的一个子集，应当与主存内容保持一致，而CPU对Cache的写入更改了Cache的内容。为此，可选用写操作策略使Cache内容与主存内容保持一致。

1、写回法（Write-Back）

当CPU写Cache命中时，只修改Cache的内容，而不是立即写入主存；只有当此块被换出时才写回主存。

使用这种方法写Cache和写主存异步进行，显著减少了访问主存的次数，但是存在数据不一致的隐患。实现这种方法时，每个Cache块必须配置一个修改位，以反映此块是否被CPU修改过。

2、全写法（Write-Through）

当写 Cache命中时，Cache与主存同时发生写修改。

使用这种方法写Cache和写主存同步进行，因而较好地维护了 Cache与主存的内容一致性。实现这种方法时，Cache中的每个块无需设置修改位以及相应的判断逻辑，但由于Cache对CPU向主存的写操作没有高速缓冲功能，从而降低了Cache的功效。

3、写一次法（Write-Once）

写一次法是基于写回法并结合全写法的写操作策略，写命中与写未命中的处理方法与写回法基本相同，只是第一次写命中时要同时写入主存，以便于维护系统全部Cache的一致性。

3.5 虚拟存储器

3.5.1 虚拟存储器基本原理

1. 什么是虚拟存储器

当代计算机系统的主存主要由半导体存储器组成，由于工艺和成本的原因，主存的容量受到限制。然而，计算机系统软件和应用软件的功能不断增强，程序规模迅速扩大，要求主存的容量越大越好，这就产生了矛盾。为了给大的程序提供方便，使它们摆脱主存容量的限制，可以由操作系统把主存和辅存这两级存储系统管理起来，实现自动覆盖。也就是说，一个大作业在执行时，其一部分地址空间在主存，另一部分在辅存，当所访问的信息不在主存时，则由操作系统而不是程序员来安排I/O指令，把信息从辅存调入主存。从效果上来看，好像为用户提供了一个存储容量比实际主存大得多的存储器，用户无需考虑所编程序在主存中是否放得下或放在什么位置等问题。我们称这种存储器为虚拟存储器。

虚拟存储器只是一个容量非常大的存储器的逻辑模型，不是任何实际的物理存储器。它借助于磁盘等辅助存储器来扩大主存容量，使之为更大或更多的程序所使用。虚拟存储器指的是主存－外存层次，它以透明的方式为用户提供了一个比实际主存空间大得多的程序地址空间。

物理地址是实际的主存单元地址，由CPU地址引脚送出，是用于访问主存的。设CPU地址总线的宽度为m位，则物理地址空间的大小就是2^m。

虚拟地址是用户编程时使用的地址，由编译程序生成，是程序的逻辑地址，其地址空间的大小受到辅助存储器容量的限制。显然，虚拟地址要比实际地址大得多。程序的逻辑地址空间称为虚拟地址空间。

程序运行时，CPU以虚拟地址来访问主存，由辅助硬件找出虚拟地址和实际地址之间的对应关系，并判断这个虚拟地址指示的存储单元内容是否已装入主存。如果已在主存中，则通过地址变换，CPU可直接访问主存的实际单元；如果不在主存中，则把包含这个字的一个存储块调入主存后再由CPU访问。如果主存已满，则由替换算法从主存中将暂不运行的一块调回外存，再从外存调入新的一块到主存。

从原理角度看，虚拟存储器和Cache－主存层次有不少相同之处。事实上，前面提到的各种控制方法是先应用于虚拟存储器中，后来才发展到Cache-主存层次中去的。不过，Cache-主存层次的控制完全由硬件实现，所以对各类程序员是透明的；而虚拟存储器的控制是软硬件相结合的，对于设计存储管理软件的系统程序员来说是不透明的，对于应用程序员来说是透明的。

主存－外存层次和Cache－主存层次所使用的地址变换及映射方法和替换策略，从原理上看是相同的，都基于程序局部性原理。它们遵循的原则是：

⑴把程序中最近常用的部分驻留在高速的存储器中；

⑵一旦这部分变得不常用了，把它们送回到低速的存储器中；

⑶这种换入换出是由硬件或操作系统完成的，对用户是透明的；

⑷力图使存储系统的性能接近高速存储器，价格接近低速存储器。

两种存储系统的主要区别在于：在虚拟存储器中未命中的性能损失，要远大于Cache系统中未命中的损失。

2. 主存－外存层次的基本信息传送单位◆

主存－外存层次的基本信息传送单位可采用页、段和段页3种不同的方案。根据地址格式的不同，虚拟存储器可分成页式虚拟存储器、段式虚拟存储器和段页式虚拟存储器3种。

页式和段式存储结构采用二维地址格式，它们把整个存储器空间（包括主存、辅存和虚拟存储器）分成若干个页或段，每个页或段又包含若干个存储单元。段页式存储结构采用三维地址格式，它把整个存储器分成若干个段，每段又分成若干页，每页包含若干个存储单元。

页式管理系统以定长的页为基本信息传送单位，主存的物理空间也被分成等长的页，每一页等长的区域称为页面，页面在主存中的位置是固定的。因此，页面的起始地址和结束地址都是固定的，这给页表的制作带来很大的方便。新页调入主存也很容易，只要有空闲的页面就可容纳。

把主存按段分配的存储管理方式称为段式管理。段是利用程序的模块化性质，按照程序的逻辑结构划分成多个相对独立的部分，如过程、数据表、阵列等。段作为独立的逻辑单位可以被其他程序段调用，这样就形成了段间连接，产生规模较大的程序。因此，把段作为基本信息单位在主存－外存之间传送和定位是比较合理的。一般用段表来指明各段在主存中的位置，每段都有它的名称（用户名称或数据结构名称或段号）、段起点、段长等。段表也是主存的一个可再定位的段。段式管理的优点是段的分界与程序的自然分界相对应，段的逻辑独立性使它易于编译、管理、修改和保护，也便于多道程序共享。某些类型的段（例如堆栈、队列）具有动态可变长度，允许自由调度以便有效利用主存空间。但是，正因为段的长度各不相同，段的起始地址和结束地址不定，这给主存空间分配带来麻烦，而且容易在段间留下许多碎片不好利用，造成浪费，这种浪费比页式管理系统要大。

页式存储管理和段式存储管理各有优缺点，段页式存储管理则是结合两者优点的一种方案。程序按模块分段，段内再分页，进入主存仍以页为基本信息传送单位，用段表和页表（每段一个页表）进行两级定位管理。

3.5.2 页式虚拟存储器

以页为基本单位的虚拟存储器叫做页式虚拟存储器。主存空间和虚存空间都划分成若干个大小相等的页，主存（即实存）的页称为实页，虚存的页称为虚页。

虚存地址分为高低两个字段：高位字段为逻辑页号，低位字段为页内地址。实存地址也分为高低两个字段：高位字段为物理页号，低位字段为页内地址。虚存地址到实存地址的变换是通过存放在主存中的页表来实现的。在页表中，对应每一个虚存逻辑页号有一个表项，表项内容包含该逻辑页所在的主存页面地址（物理页号），用它作为实存地址的高字段，与虚存地址的页内地址字段相拼接，产生完整的实存地址，据此来访问主存。地址变换如图 3-17 所示。

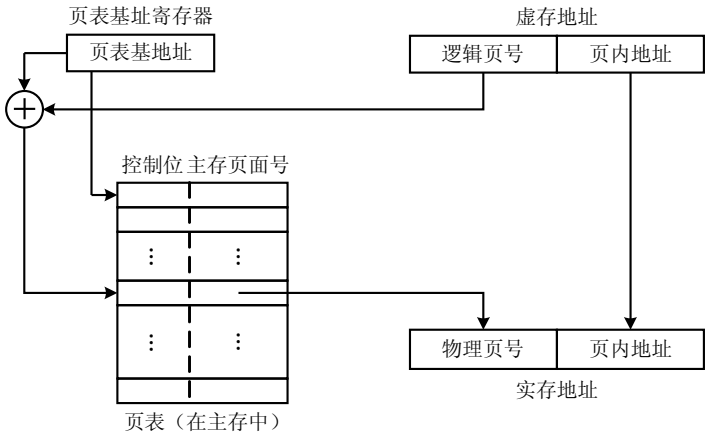

图 3-17 页式虚拟存储器结构

若计算机采用多道程序工作方式，则可为每个用户作业建立一个页表，硬件中设置一个页表基址寄存器，存放当前所运行程序的页表的起始地址。

页表中的表项除包含虚页号对应的实页号之外，还包括装入位、修改位、替换控制位等控制字段。若装入位为“1”，表示该页面已在主存中，将对应的实页号与虚地址中的页内地址相拼接就得到了完整的实地址；若装入位为“0”，表示该页面不在主存中，于是要启动 I/O 系统，把该页从外存中调入主存后再供 CPU 使用。修改位指出主存页面中的内容是否被修改过，替换时是否要写回外存。替换控制位指出需替换的页，与替换策略有关。

CPU 访存时首先要查页表，为此需要访问一次主存，若不命中，还要进行页面替换和页表修改，则访问主存的次数就更多了。为了将访问页表的时间降低到最低限度，许多计算机将页表分为快表和慢表两种。将当前最常用的页表信息存放在快表中，作为慢表部分内容的副本。快表很小，存储在一个小容量的快速存储器中，该存储器是按内容查找的相联存储器，可按虚页号名字进行查询，迅速找到对应的实页号。使用快表与慢表进行地址变换如图 3-18 所示。

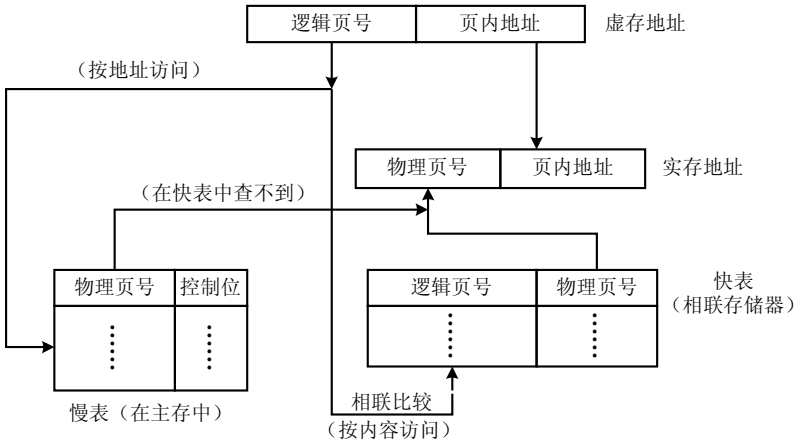

图 3-18 经快表和慢表实现内部地址变换

快表由硬件组成，比页表小得多，查表时，由逻辑页号同时去查快表和慢表。当在快表中有此逻辑页号时，就能很快地找到对应的物理页号送入实主存地址寄存器，从而做到虽采用虚拟存储器但访主存速度几乎没有下降；如果在快表中查不到，那就要花费一个访主存时间去查慢表，从中查到物理页号送入实存地址寄存器，并将此逻辑页号和对应的物理页号送入快表，替换快表中应该移掉的内容，这也要用到替换算法。

页式虚拟存储器的每页长度是固定的，页表的建立很方便，新页的调入也容易实现。但是由于程序不可能正好是页面的整数倍，最后一页的零碎空间将无法利用而造成浪费。同时，页不是逻辑上独立的实体，这使得程序的处理、保护和共享都比较麻烦。

3.5.3 段式虚拟存储器

段式虚拟存储器中的段是按照程序的逻辑结构划分的，各个段的长度因程序而异。在段式虚拟存储系统中，虚拟地址由段号和段内地址组成，为了把程序虚地址变换成主存实地址，需要一个段表。段表一般驻留在主存中，其中每一行记录了某个段对应的若干信息，包括段号、装入位、段起点和段长等。这里的段号指的是虚拟段号。装入位为“1”，表示该段已调入主存；装入位为“0”，表示该段不在主存中。由于段的大小可变，所以在段表中要给出各段的起始地址与段的长度。段表实际上是程序的逻辑结构段与其在主存中的存放位置之间的关系对照表，如图 3-19 所示。段表也是一个段，可以保存在外存中，但一般驻留在主存中。

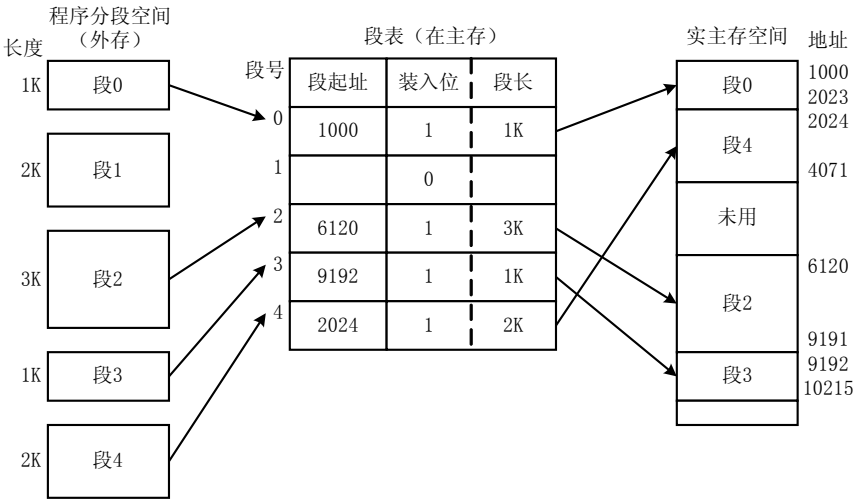

图 3-19 段式虚拟存储器中的段表

段式虚拟存储器的虚－实地址变换如图 3-20 所示。CPU 根据虚地址访存时，首先将段号与段表的起始地址相拼接，形成访问段表对应行的地址，然后根据段表的装入位判断该段是否已调入主存。若已调入主存，则从段表读出该段在主存中的起始地址，与段内地址（偏移量）相加，得到对应的主存实地址。

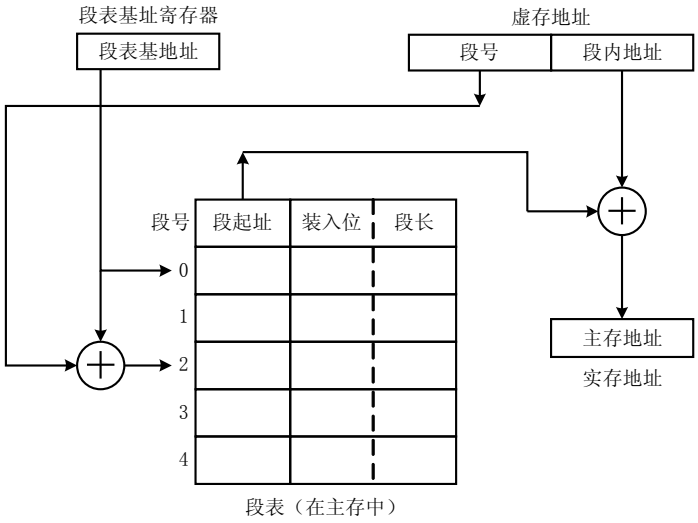

图 3-20 段式虚拟存储器地址变换

由于段的分界与程序的自然分界相对应，具有逻辑独立性，所以易于实现程序的编译、管理、修改和保护，也便于多道程序共享。但是，因为段的长度参差不齐，起点和终点不定，给主存空间分配带来了麻烦，容易在段间留下不能利用的零碎空间，造成浪费。

3.5.4 段页式虚拟存储器

段页式虚拟存储器是段式虚拟存储器和页式虚拟存储器的结合。它把程序按逻辑单位分段以后，再把每段分成固定大小的页。主存空间也划分为若干个同样大小的页。虚存和实存之间以页为基本传送单位，每个程序对应一个段表，每段对应一个页表。虚地址包含段号、段内页号、页内地址三部分。CPU 访问时，首先将段表起始地址与段号合成，得到段表地址，然后从段表中取出该段的页表起始地址，与段内页号合成，得到页表地址，最后从页表中取出实页号，与页内地址拼接形成主存实地址。

段页式存储器综合了前两种结构的优点，但要经过两级查表才能完成地址转换，要多花费一些时间。

3.5.5 替换算法

当 CPU 要访问的数据或指令不在主存中时，产生页面失效（缺页），此时就要从外存调进包含此数据或指令的页，若主存页面已全部占满，就需要采用某种替换算法将主存中的某一页替换出去。虚拟存储器中的页面替换策略与 Cache 中的块替换策略有很多相似之处，但有三点显著不同：

⑴缺页至少要涉及一次磁盘外存存取，以读取所缺的页，因缺页使系统蒙受的损失要比 Cache 未命中大得多。

⑵页面替换是由操作系统软件实现的，而 Cache 的块替换则是由硬件实现的。

⑶页面替换的选择余地很大，属于一个进程的页面都可替换。

虚拟存储器中的替换策略一般采用 LRU 算法、LFU 算法、FIFO 算法，或将两种算法结合起来使用。对于将被替换出去的页面，假如该页调入主存后没有被修改，就不必进行处理，否则就要把该页重新写入外存，以保证外存中数据的正确性。为此，在页表的每一行应设置一个修改位。