第3章

存储层次与系统

存储系统是计算机必不可少的部件之一，是计算机信息存储的核心，用以存放程序和数据。计算机就是按存放在存储器中的程序自动连续地进行工作的。由于存储器系统的速度和容量直接影响着计算机系统的工作速度和效率，因此如何设计容量大、速度快、价格低的存储器，一直是计算机发展的一个重要问题。本章主要讨论的是存储系统的概念和设计与构造原理，包括存储芯片的结构和主存储器的组织方式，以及高速缓冲存储器和虚拟存储器的工作原理。

3.1 存储系统概述

3.1.1 存储器的分类

随着计算机及其器件的发展，存储器也有了很大的发展，存储器的类型日益繁多，因而存储器的分类方法也有多种。

1. 按与 CPU 的连接和功能分类

1）主存储器

CPU 能够直接访问的存储器为主存储器，用以存放当前运行的程序和数据。由于它设在主机内部，又称内存储器，简称内存或主存。

2）辅助存储器

辅助存储器是为解决主存容量不足而设置的存储器，用以存放当前不参加运行的程序和数据，当需要运行时，成批调入内存供 CPU 使用，CPU 不能直接访问它。由于它是外部设备的一种，因此又称为外存储器，简称外存。

3）高速缓冲存储器

高速缓冲存储器是一种介于主存与 CPU 之间用于解决 CPU 与主存间速度匹配问题的高速小容量的存储器。它被用于存放 CPU 立即要运行或刚使用过的程序和数据。

2. 按存取方式分类

1）随机存取存储器

存储器任何单元的内容均可按其地址随机地读取或写入，而且存取时间与单元的物理位置无关。一般主存储器主要由随机存取存储器（Random Access Memory，RAM）组成。

2) 只读存储器

存储器任何单元的内容只能随机地读出信息,而不能写入新信息,称为只读存储器(Read Only Memory,ROM)。只读存储器可以作为主存储器的一部分,用以存放不变的程序和数据。只读存储器可以用作其他固定存储器,例如存放微程序的控制存储器、存放字符点阵图案的字符发生器等。

3) 顺序存取存储器

存储器所存信息的排列、寻址和读写操作均是按顺序进行的,并且存取时间与信息在存储器中的物理位置有关。这种存储器称为顺序存取存储器(Sequential Access Memory,SAM)。

在这种存储器中,如磁带存储器,信息通常是以文件或数据块形式按顺序存放的。信息在载体上没有唯一对应的地址,完全按顺序存放或读取。

4) 直接存取存储器

这种存储器既不像 RAM 那样能随机地访问任何存储单元,也不像 SAM 那样完全按顺序存取,而是介于 RAM 与 SAM 之间的一种存储器。目前广泛使用的磁盘就属于直接存取存储器(Direct Access Memory,DAM)。当要存取所需的信息时,它要进行两个逻辑动作,第一步为寻道,使磁头指向被选磁道,第二步在被选磁道上顺序存取。

5) 联想存储器

联想存储器也称为关联存储器(Associated Memory,AM),它支持先按照信息的关键词(如学生记录中的学号)并行查找所有的存储器单元,匹配后返回信息所在地址,然后访问所需要的全部信息。关联存储器主要用于快速比较和查找,例如页式虚拟存储器中的快表、高速缓冲存储器中的标识 Cache 都是由关联存储器构成的。

3. 按存储介质分类

凡具有两个稳定物理状态,可用来记忆二进制代码的物质或物理器件均称为存储介质。按存储介质对存储器分类,有下面几种。

1) 半导体存储器

半导体存储器是指用半导体器件组成的存储器,根据工艺不同,可分为双极型和 MOS 型。

2) 磁表面存储器

磁表面存储器利用涂在基体表面上的一层磁性材料存放二进制代码,例如磁盘、磁带等。

3) 光存储器

光存储器是利用光学原理制成的存储器,它通过能量高度集中的激光束照在基体表面引起物理的或化学的变化,记忆二进制信息。

此外还有其他一些分类方法,如按信息的可保存性可分为易失性存储器和非易失性存储器等,在此不再详述。

3.1.2 存储器系统的层次结构

无论主存储器的容量有多大,它总是无法满足人们的期望。其主要原因是,随着技术的进步,人们开始希望存放以前完全属于科学幻想领域的信息,存储器存储能力的扩大永远无法赶上需要它存放的信息的膨胀。

1. 存储系统的结构层次

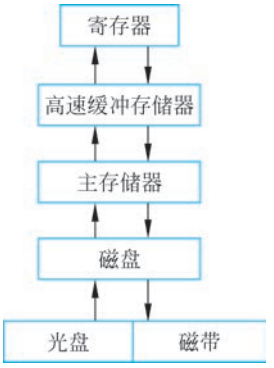

图 3-1　存储器层次结构图

存储大量数据的传统办法是采用如图 3-1 所示的层次存储结构。最上层是 CPU 中的寄存器，其存取速度可以满足 CPU 的要求。下面一层是高速缓冲存储器(Cache)，再往下是主存储器，然后是磁盘存储器，这是当前用于永久存放数据的主要存储介质。最后，还有用于后备存储的磁带、光盘存储器以及基于网络的各种文件系统。

按层次结构自上而下，有 3 个关键参数逐渐增大。第一，访问时间逐渐增长。寄存器的访问时间是几个纳秒，高速缓冲存储器的访问时间是寄存器访问时间的几倍，主存储器的访问时间是几十纳秒。再往后是访问时间的突然增大，磁盘的访问时间最少要 10ms 以上。如果加上介质的取出和插入驱动器的时间，磁带和光盘的访问时间就得以秒来计量了。

第二，存储容量逐渐增大。在现今的个人计算机中，寄存器的容量以字节为单位衡量，而高速缓存达到了几百兆字节(MB)到若干千兆字节(GB)，主存储器容量一般为若干千兆字节(GB)，磁盘的容量应该是几百千兆字节(GB)到若干太字节(TB)。磁带和光盘一般脱机存放，其容量只受限于用户的预算。

第三，用相同的钱能购买到的存储容量逐渐加大，即存储每位的价格逐渐减小。显然，主存每位的价格要高于磁盘，而磁盘每位的价格要高于磁带或光盘。

存储器系统按照层次结构组织，相应的数据也组织成层次结构。靠近处理器那一层的数据是处理器刚刚使用或即将使用的，是较低层次中数据的副本。而所有的数据被存储在较慢的硬盘中。这意味着除了程序执行过程中的中间结果外，除非数据在第 $i+1$ 层存在，否则绝不可能在第 i 层存在。

若处理器访问的数据在层次结构中的最高层找到(即命中)，则能很快被处理。若访问的数据高层没有(即缺失)，则需要访问容量大但速度慢的低层存储器，并由低层向高层逐层复制。如果高层的命中率足够高，存储器层次结构就会拥有接近高层次存储器的访问速度和接近低层次存储器的容量。

2. 传统的三级存储结构

依据图 3-1 的存储层次结构图，高速缓冲存储器-主存储器-辅助存储器(硬盘)构成的存储体系是该存储系统的核心结构，即为传统的三级存储结构。这种结构从两个方面解决存储系统的不同用户需求。

1) 高速缓冲存储器-主存储器层次结构

这一级的存储结构主要用于解决 CPU 和主存之间的速度差匹配问题。因为 CPU 的速度远远快于访存的速度，所以在 CPU 和主存之间增加一级容量小但速度很快的缓冲存储器(Cache)，以减少 CPU 的等待时间，提高 CPU 的工作效率。

2) 主存储器-辅助存储器层次结构

由于存储于主存的程序和数据是当前正在执行的程序和处理的数据，毕竟数量有限，而大量的程序和数据则保存在某个较大空间的存储器中，以备随时调用，因此这一级的存储结构考虑的是存储空间不足的问题。所以，采用大空间的辅助存储器(磁盘)解决主存储器空间不足

的问题,即磁盘是主存储器的后援存储器。

存储器层次结构的概念影响着计算机的许多其他方面,包括操作系统如何管理存储器和I/O,编译器如何产生代码,甚至对应用程序如何使用计算机也产生一定的影响。

3.1.3 主存储器的组成和基本操作

主存储器是整个存储器系统的核心,用来存放处理器当前运行的程序和数据,是CPU可直接访问的存储器。本节重点讨论主存储器的组成和基本操作。

1. 主存储器的组成

图3-2是主存储器的基本组成框图。其中存储阵列是主存储器的核心部分,它是存储二进制信息的主体,也称为存储体。存储体是由大量存储单元构成的,为了区分各个存储单元,把它们进行统一编号,这个编号称为地址,因为是用二进制进行编码的,所以又称地址码。地址码与存储单元是一一对应的,每个存储单元都有自己唯一的地址,因此要对某一存储单元进行存取操作,必须首先给出被访问的存储单元的地址。

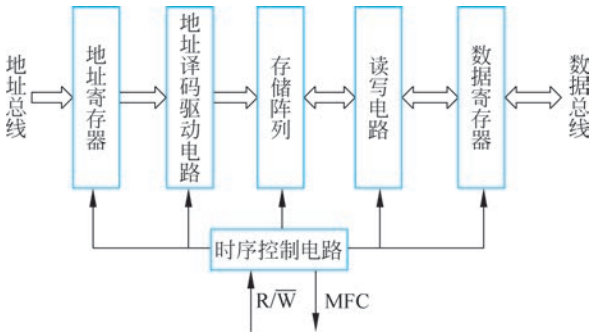

图3-2 主存储器的基本组成

主存可寻址的最小单位称为编址单位。有些计算机是按字编址的,最小可寻址信息单元是一个机器字,连续的存储器地址对应连续的机器字。目前多数计算机是按字节编址的,最小可寻址单位是1字节。一个32位字长的按字节寻址的计算机,一个存储器字包含4个可单独寻址的字节单元,由地址的低两位来区分。

地址寄存器用于存放所要访问的存储单元的地址。要对某一单元进行存取操作,首先应通过地址总线将被访问单元地址存放到地址寄存器中。

地址译码与驱动电路的作用是对地址寄存器中的地址进行译码,通过对应的地址选择线到存储阵列中找到所要访问的存储单元并驱动其完成指定的存取操作。

读写电路与数据寄存器的作用是根据CPU的读写命令,把数据寄存器的内容写入被访问的存储单元,或者从被访问单元中读出信息送入数据寄存器中,以供CPU或I/O系统使用。所以数据寄存器是存储器与计算机其他功能部件联系的桥梁。从存储器中读出的信息是经数据寄存器通过数据总线传送给CPU与I/O系统的;向存储器中写入信息,也必须先将要写入的信息经数据总线送入数据寄存器,再经读写电路写入被访问的存储单元。

时序控制电路用于接收来自CPU的读写控制信号,产生存储器操作所需的各种时序控制信号,控制存储器完成指定的操作。如果存储器采用异步控制方式,当一个存取操作完成后,该控制电路还应给出存储器操作完成(MFC)信号。

主存储器用于存放 CPU 正在运行的程序和数据，它和 CPU 的关系最为密切。主存与 CPU 间的连接是由总线支持的，其连接形式如图 3-3 所示。

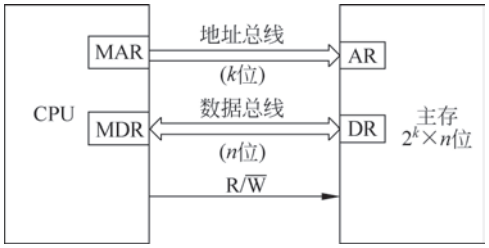

图 3-3　主存与 CPU 的连接

2. 主存储器的基本操作

存储器的基本操作是读(取)和写(存)。当 CPU 要从存储器中读取一个信息字时，CPU 首先把被访单元的地址送到存储器地址寄存器(MAR)，经地址总线送给主存，同时发出“读”命令。存储器接到“读”命令，根据地址从被选单元读出信息，经数据总线送入存储器数据寄存器(MDR)。为了存一个字到主存，CPU 把要存入的存储单元地址经 MAR 送入主存，并把要存入的信息字送入 MDR，此时发出“写”命令，在此命令的控制下经数据总线把 MDR 中的内容写入主存。

CPU 与主存之间的数据传送可采用同步控制方式，也可采用异步控制方式。目前多数计算机采用同步方式，即主存储器的操作与处理器保持同步，数据传送在固定的时间间隔内完成，此时间间隔构成了存储器的一个存储周期，异步传送方式允许选用具有不同存取速度的存储器作为主存。

3.1.4　存储器的主要技术指标

衡量一个存储器性能的主要技术指标有以下几方面。

1. 存储容量

存储容量是指半导体存储芯片能够存储的二进制信息的位数。其单位是 K 位(Kilobits)、M 位(Megabits)、G 位(Gigabits)等。需注意的是，要将其与计算机系统的存储容量区分开，当我们讨论存储芯片的容量时，采用的单位是位；当我们讨论计算机存储器的容量时，其单位是字节。因此，当存储芯片资料中提到 4M 存储芯片时，是指 4M 位的存储容量，用 4Mb 表示；若宣传资料中提到计算机存储器有 4G 时，是指 4G 字节的存储器容量，用 4GB 表示。

2. 速度

速度是存储芯片的一项重要技术指标，它影响着 CPU 的工作效率。存储芯片速度通常用访问时间和存取周期表示。

访问时间(Memory Access Time)又称取数时间，是指从启动一次存储器存取操作到完成该操作所经历的时间。对存储器的某一个单元进行一次读操作，例如 CPU 取指令或取数据，从把要访问的存储单元的地址加载到存储器芯片的地址引脚上开始，直到读取的数据或指令

在存储器芯片的数据引脚上可以使用为止,两者之间的时间间隔即为访问时间(取数时间),存储器芯片数据手册(Data Sheet)把它记为 t_A。访问时间(t_A)是一个最为常见的参数。对于某些只读存储器(ROM)芯片(如 EEPROM),t_{OE} 是指从 OE(读)信号有效开始,直到读取的数据或指令在存储器芯片的数据引脚上可以使用为止的这段时间间隔。

存取周期(Memory Cycle Time)又称存储周期或读写周期,被记为 T_M。它是指对存储器进行连续两次存取操作所需要的最小时间间隔。由于有些存储器在一次存取操作后需要有一定的恢复时间,因此通常存取周期大于或等于取数时间。

3. 存储器总线带宽

存储器总线宽度除以存取周期就是存储器带宽或频宽,它是指存储器在单位时间内所存取的二进制信息的位数,也称为数据传输率。带宽的单位一般是兆字节每秒(MB/s)。

4. 价格

半导体存储器的价格常用每位价格来衡量。设存储器容量为 S 位,总价格为 C,则每位价格可表示为 $c=C/S$。

半导体存储器的总价格正比于存储容量,而反比于存取时间。容量、速度、价格 3 个指标是相互矛盾、相互制约的。高速存储器往往价格也高,因而容量不可能很大。

除了上述几个指标外,影响半导体存储器性能的还有功耗、可靠性等因素。

3.2 半导体随机存取存储器

在现代计算机中,半导体存储器已广泛用于实现主存储器。由于主存储器直接为 CPU 提供服务,对主存的要求是能够迅速响应 CPU 的读写请求,半导体存储器在这方面能做得很好,因此半导体存储器是实现主存的首选器件。通常使用的半导体存储器分为随机存取存储器(Random Access Memory,RAM)和只读存储器(Read-Only Memory,ROM)。它们各自又有许多不同的类型。

3.2.1 半导体随机存取存储器的分类

由于大多数随机存取存储器在断电后会丢失其中存储的内容,故这类随机存取存储器又被称为易失性存储器。由于随机存取存储器可读可写,有时它们又被称为可读写存储器。随机存取存储器分为静态 RAM 和动态 RAM。

1. 静态 RAM

静态 RAM(Static RAM,SRAM)中的每个存储单位都由一个触发器构成,因此可用于存储一个二进制位,只要不断电就可以保持其中存储的二进制数据不丢失。使用触发器作为存储单位的问题是,每个存储单位至少需要 6 个 MOS 管来构造一个触发器,所以 SRAM 存储芯片的存储密度较低,即每块芯片的存储容量不会太大。近年来,人们发明了用 4 个 MOS 管构成一个存储单位的 SRAM 技术,利用该技术再加上 CMOS 技术,人们制造出了大容量的 SRAM。尽管如此,SRAM 的容量仍然远远低于同类型的动态 RAM。SRAM 通常当 Cache 使用。

2. 动态 RAM

在 1970 年，Intel 公司推出了世界上第一块动态 RAM(Dynamic RAM，DRAM)芯片，其容量为 1024 位，它使用一个 MOS 管和一个电容来存储一位二进制信息。用电容来存储信息减少了构成一个存储单位所需要的晶体管的数目。但由于电容本身不可避免地会产生漏电，因此 DRAM 存储器芯片需要频繁的刷新操作，但 DRAM 的存储密度大大提高了。

为了进一步优化与处理器的接口，DRAM 增加了时钟，使用时钟使主存储器和处理器同步，因此称为同步 DRAM(Synchronous DRAM，SDRAM)。SDRAM 支持突发(Burst Mode)数据传送方式，又称为连续数据传送，即主存储器接收到第一个数据所在存储单元的地址，就可以读/写连续的多个数据单元。这意味着若 CPU 访问连续的存储单元，则只需要传送第一个数据所在的存储单元地址，主存储器可以自动修改后续访问单元地址，大大提高了数据传输效率。

更快的 SDRAM 称为双数据速率(Dual Data Rate，DDR)SDRAM，即在一个时钟的上升沿和下降沿都可以传送数据，因此可以获得双倍的数据带宽。目前该技术在不断革新中。例如标称为 DDR4-3200 的 SDRM 芯片，其中 4 指的是第 4 代 DDR 技术，而 3200 指每秒可传输 3200 兆次，即其时钟频率是 1600MHz。

主存储器主要是由 DRAM 构成的。

3.2.2　半导体随机存取存储器的单元电路

1. 静态 RAM 单元电路

图 3-4 所示的存储单元电路是一种比较常见的六管静态 MOS 存储单元电路。图中 T_1、T_2 两个 MOS 管构成触发器，用于存储一位二进制信息位。MOS 管 T_3、T_4 是触发器的两个负载管。MOS 管 T_5、T_6 称为门控管，通过连接在这两个 MOS 管栅极上的字线 W，可以控制触发器电路与位线 b 和 b'的联系。

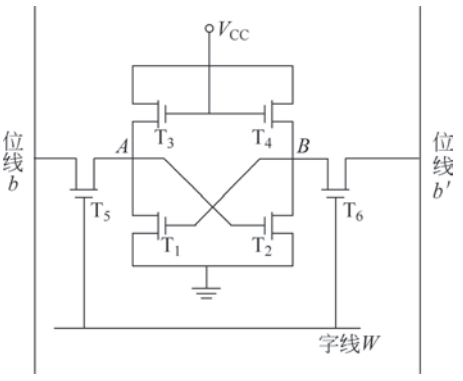

图 3-4　六管静态 MOS 存储单元电路

当加载在字线 W 上的电平为低电平时，T_5、T_6 栅极为低电平，使 T_5、T_6 两个 MOS 管呈现截止状态，从而使触发器电路与位线隔离，表示存储单元未被选中。在这种情况下，触发器的状态不可能发生改变，意味着原来存储的信息不发生变化。

当要向该存储单元写入信息时，首先在字线 W 上加载一个表示选中了这个存储单元的高电平，使 T_5、T_6 两个 MOS 管呈现导通状态，而位线上的电平状态则要由写入的信息控制。假设在图 3-4 所示的电路中，触发器 A 端为高电平状态、B 端为低电平状态，表示存储单元存储的信息是 1；触发器 A 端为低电平状态、B 端为高电平状态，则表示存储的信息是 0。假设要写入的信息是 0，则应在位线 b 上加载低电平，同时在位线 b'上加载高电平。在位线 b'上加载的高电平通过 T_6 加到 T_1 管的栅极，致使 T_1 导通，同时在位线 b 上加载的低电平通过 T_5 加载到 T_2 管的栅极，致使 T_2 截止，从而使 A 端为低电平状态，B 端为高电平状态，即写入了信息 0。类似地，若要写入的信息是 1，则在位线 b 上加载高电平，在位线 b'上加载低电平，从而使 T_2 导通、T_1 截止，使 A 端为高电平状态、B 端为低电平状态，即写入了信息 1。写入操

作结束后,字线 W 恢复到低电平状态,使 T_5、T_6 截止,从而保证了写入的信息不会发生变化。

当读出信息时,同样首先在字线 W 上加载一个表示选中了这个存储单元的高电平,使 T_5、T_6 导通。此时若原存储的信息为 0,即原先 T_1 导通、T_2 截止,因而在位线 b 上呈现低电平状态,在位线 b' 上呈现高电平状态,表示输出信息 0。同样,若原存储的信息为 1,则 T_1 截止、T_2 导通,因而在位线 b 上呈现高电平状态,在位线 b' 上呈现低电平状态,表示输出信息 1。

通过上面的分析,我们可以得知静态 MOS 存储器是利用触发器的两个稳定状态来存储二进制信息的,而且通过对读出过程的了解,我们可以看出读出时触发器的状态没有被破坏,原来存储的信息依然存在。因此,从静态 MOS 存储电路中读取其中存放的信息的过程,对原来存放的信息而言,是非破坏性的读出过程。

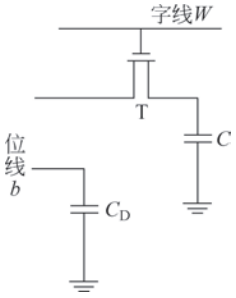

图 3-5 单管动态 MOS 存储单元电路

2. 动态 RAM 单元电路

目前最常用的动态 MOS 存储单元电路是单管动态存储单元电路,如图 3-5 所示。该电路用电容 C 存储二进制信息,若 C 上存有电荷,则表示存储的信息为 1,若 C 上无电荷,则表示存储的信息为 0。当加载在字线 W 上的电平为低电平时,MOS 管 T 截止,表示电路不被选中,保持原存储的信息不变。

当要向存储单元写入信息时,首先要在字线 W 上加载一个表示选中了这个存储单元的高电平,使 MOS 管 T 导通。若要写入的信息为 1,则要在位线 b 上加载高电平,对电容 C 充电,使其中存有电荷,实现写入了信息 1;若要写入的信息为 0,则要在位线 b 上加载低电平,使电容 C 能够通过 T 管和位线 b 放掉其中的电荷,实现写入了信息 0。

当读出信息时,同样首先要在字线 W 加载一个表示选中了这个存储单元的高电平,使 MOS 管 T 导通。若原来存储的信息为 1,即 C 中有电荷存储,在 T 导通后,C 中原来存储的电荷经过 T 管向位线 b 上泄放,致使位线 b 上有微弱电流流动,表示有输出信号,该信号经过读出再生放大器放大后,输出信息 1;若原来存储的信息为 0,即电容 C 中无电荷存储,则在位线上不会产生电流的流动,表示无输出信号,这样读出再生放大器输出信息 0。由于在读取信息 1 时,位线 b 上电流流动很微弱,这就要求读出再生放大器具有较高的灵敏度。

由于单管存储单元电路是靠存储在电容中的电荷泄放检测信息 1 的,原来存放的信息被读出后,存储单元电路的状态被破坏掉(电荷释放),因此从动态 MOS 存储单元电路中读取存放信息的过程,对原来存放的信息而言,是破坏性的读出过程,在信息被读出后,必须采取再生措施,即读出信息后要立即重写该信息。读出再生放大器具有这种再生功能。

因为单管动态 MOS 存储器单元电路中电容电荷的泄放会引起信息的丢失,所以每隔一定时间要对电路进行一次刷新操作,刷新方式将在下一节中讨论。

3.2.3 半导体随机存取存储器的芯片结构及实例

一个存储单元电路存储一位二进制信息。把大量存储单元电路按一定的形式排列起来,即构成存储体。存储体一般都排列成阵列形式,所以又称作存储阵列。把存储体及其外围电路(包括地址译码与驱动电路、读写放大电路及时序控制电路等)集成在一块硅片上,称为存储器组件。存储器组件经过各种形式的封装后,通过引脚引出地址线、数据线、控制线及电源与地线等,就制成了半导体存储器芯片。半导体存储器芯片的内部组织一般有两种结构:字片

式结构和位片式结构。

1. 字片式结构的半导体存储器芯片

图 3-6 是 64 字×8 位的字片式结构的存储器芯片的内部组织图。图中每一个小方块表示一个存储单元电路，这里略去了每个单元电路的内部结构及电源部分，仅仅画出了与每个存储单元电路相连的一根字线和两根位线。存储阵列的每一行组成一个存储单元，也是一个编址单位，存放一个 8 位的二进制字。一行中所有存储单元电路的字线连在一起，接到地址译码器对应的输出端。存储器芯片接收到的 6 位存储单元的地址，经地址译码器译码选中某一输出端有效时，与该输出端相连的一行中的每个单元电路同时进行读/写操作，从而实现了对一个存储单元中的所有位同时读/写。这种对接收到的存储单元地址仅进行一个方向译码的方式，称为单译码方式或一维译码方式。在这种结构的存储器芯片中，所有存储单元的相同的位组成一列，一列中所有存储单元电路的两根位线分别连在一起，并使用同一个读/写放大电路。读/写放大电路与双向数据线相连接。

图 3-6 所示的芯片有两根控制线，即读/写控制信号线 R/$\overline{W}$ 和片选控制信号线 $\overline{CS}$。当 $\overline{CS}$ 为低电平时，选中芯片工作；而当 $\overline{CS}$ 为高电平时，芯片不被选中。每当存储器芯片接收到某个存储单元的地址并译码后，此时若 $\overline{CS}$ 为低电平，R/$\overline{W}$ 为高电平，则要对选中芯片中的某个存储单元进行读出操作；同样，若 $\overline{CS}$ 为低电平，R/$\overline{W}$ 也为低电平，则要对选中芯片中的某个存储单元进行写入操作。

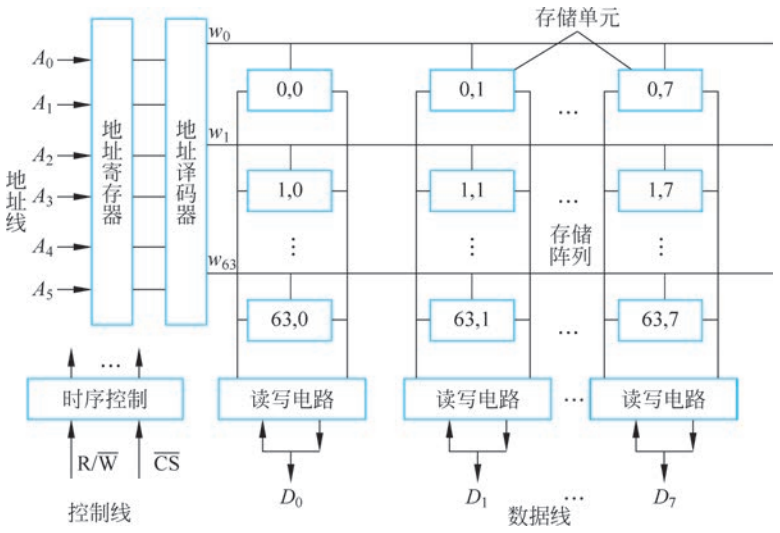

图 3-6　64 字×8 位的字片式结构 RAM 芯片

在上述字片式结构存储器芯片中，由于采用单译码方案，因此有多少个存储单元，就有多少个译码驱动电路，所需译码驱动电路较多。为减少译码驱动电路的数量，多数存储器芯片都采用双译码(也称二维译码)方案，即采用位片式结构。

2. 位片式结构的半导体存储器芯片

图 3-7 展示的是 4K×1 位的位片式结构存储器芯片的内部组织。它共有 4096 个存储单元电路，排列成 64×64 的阵列。对 4096 个存储单元进行寻址，需要 12 位地址，在此将其分为 6 位行地址和 6 位列地址。对于一个给定的访问某个存储单元电路的地址，分别经过行、列地

址译码器的译码后,致使一根行地址选择线和一根列地址选择线有效。行地址选择线选中的某一行中的64个存储单元电路可以同时进行读写操作。列地址选择线用于选择控制64个多路转接开关中的一个,即表示选中一列,每个多路转接开关由两个MOS管组成,分别控制两条位线。选中的那一个多路转接开关的两个MOS管呈现"开"状态,使这一列的位线与读/写电路接通;其余63个没被选中的多路转接开关的两个MOS管则呈现"关"状态,使其余63列的位线与读/写电路断开。

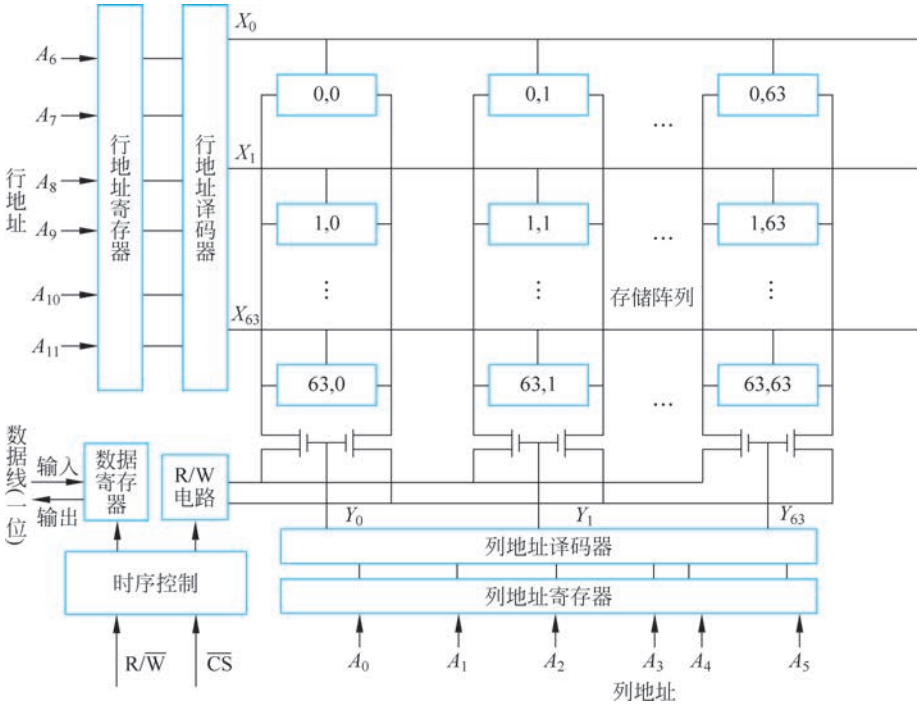

图3-7 4K×1位的双译码方式的RAM芯片结构

当选中该芯片工作时,首先给定要访问的存储单元的地址,并给出有效的片选信号$\overline{CS}$和读写信号R/$\overline{W}$,通过对行列地址的译码,找到被选中的行和被选中的列两者交叉处的唯一一个存储单元电路,读出或写入一位二进制信息。

从图3-7可以看出,这种双译码方案,对于4096个字只需128个译码驱动电路(针对行有64个,针对列也有64个),若采用单译码方案,4096个字将需要4096个译码驱动电路。

3. 半导体RAM芯片实例

为了加深对芯片结构的理解,下面以动态MOS存储器芯片TMS 4116为例,进一步说明MOS型存储器的结构及工作原理。

TMS 4116是由单管动态MOS存储单元电路构成的随机存取存储器芯片,其容量为16K×1位,图3-8所示为TMS 4116芯片的逻辑结构框图和引脚分配图,地址码有14位,为了节省引脚,该芯片只用了A_0~A_6七根地址线,采用分时复用技术,分两次把14位地址送入芯片。首先送入低7位地址A_6~A_0,由行地址选通信号$\overline{RAS}$把这7位地址送到行地址缓冲器锁存,高7位地址A_{14}~A_8,由列地址选通信号$\overline{CAS}$打入列地址缓冲器锁存。

D_{IN}、D_{OUT}分别为数据输入线和数据输出线,它们各有自己的数据缓冲寄存器。$\overline{WE}$为

写允许控制线，$\overline{WE}$ 为高电平时为读出，$\overline{WE}$ 为低电平时为写入。该芯片没有专门设置选片信号，一般用 $\overline{RAS}$ 信号兼做选片控制信号，只有 $\overline{RAS}$ 有效(低电平)时，芯片才工作。

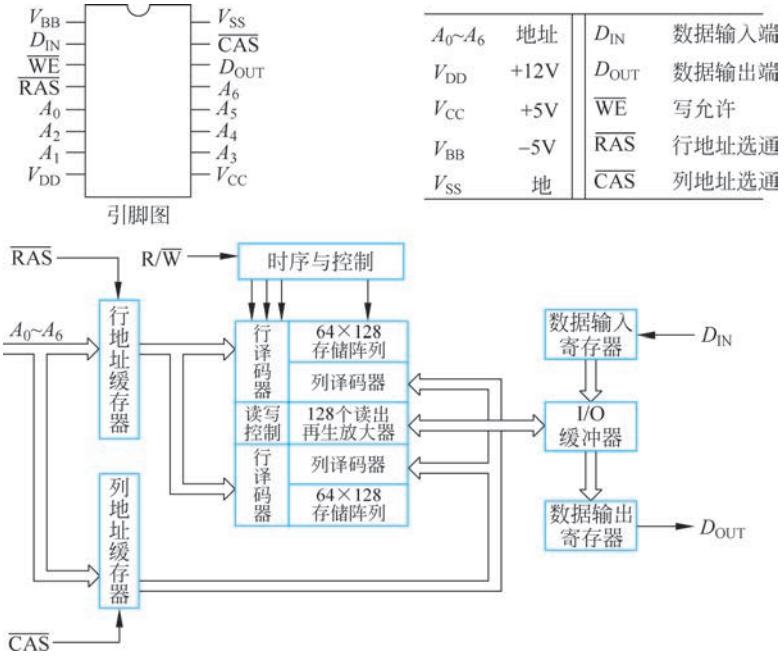

图 3-8　TMS 4116 动态存储器逻辑结构框图与引脚

图 3-9 是 TMS 4116 芯片的存储阵列结构图。16K×1 位共 16 384 个单管 MOS 存储单元电路，排列成 128×128 的阵列，并将其分为两组，每组为 64 行×128 列。每根行选择线控制 128 个存储电路的字线。每根列选择线接到列控制门的删极，控制读出再生放大器与 I/O 缓冲器的接通，控制数据的读出或写入。每根列选择线控制一个读出再生放大器，128 列共有 128 个读出再生放大器，一列中的 128 个存储电路分为两组，每 64 个存储电路为一组，两组存储电路的位线分别接入读出再生放大器的两端。

读出时，行地址经行地址译码选中某一根行线有效，接通此行上的 128 个存储电路中的 MOS 管，使电容所存储的信息分别送到 128 个读出再生放大器。由于是破坏性读出，经放大后的信息又送回到原电路进行重写，使信息再生。当列地址经列译码选中某根列线有效时，接通相应的列控制门，将该列上读出放大器输出的信息送入 I/O 缓冲器，经数据输出寄存器输出到数据总线上。

写入时，首先将要写入的信息由数据输入寄存器经 I/O 缓冲器送入被选列的读出再生放大器中，然后写入行、列同时被选中的存储单元。

综上可知，当某个存储单元被选中进行读/写操作时，该单元所在行的其余 127 个存储电路也将自动进行一次读出再生操作，这实质上是完成一次刷新操作。故这种存储器的刷新是按行进行的，每次只加行地址，不加列地址，即可实现被选行上的所有存储电路的刷新。

读出再生放大器的结构形式如图 3-10 所示。图中 T_1、T_2、T_3、T_4 组成放大器，位于两侧的行选择线仅画出了行选 64 和行选 65，T_6、T_7 与 C_s 是两个预选单元，由 XW_1 与 XW_2 控制。在读写之前，先使两个预选单元中的电容 C_s 预充电到 0 与 1 电平中间值(预充电路略)，并使 $\Phi_1=0$、$\Phi_2=1$，使 T_3、T_4 截止，T_5 导通，使读出放大器两端 W_1、W_2 处于相同电位。

读出时，先使 $\Phi_2=0$，T_5 截止。放大器处于不稳定平衡状态。这时使 $\Phi_1=1$，T_3、T_4 导

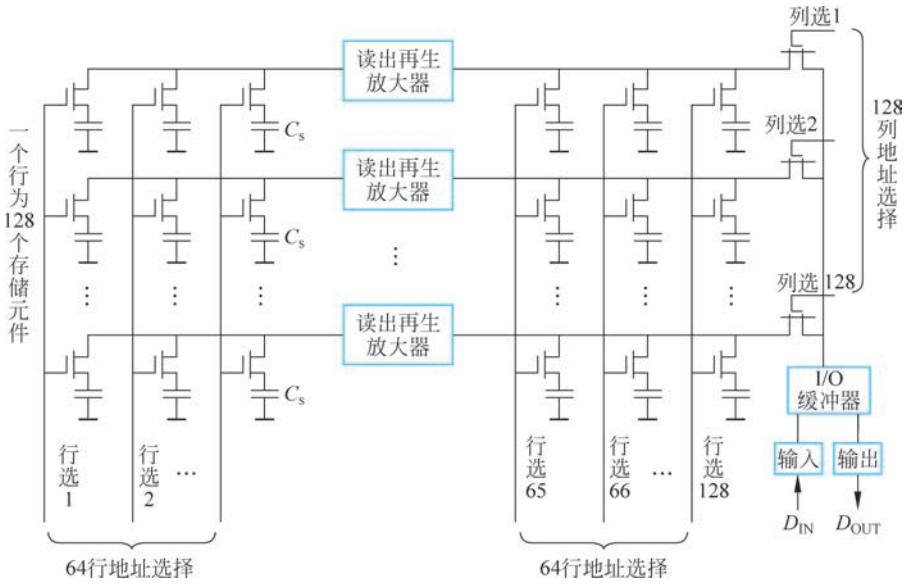

图 3-9 TMS 4116 动态存储器存储阵列图

通，T_1、T_2、T_3、T_4 构成双稳态触发器，其稳定状态取决于 W_1、W_2 两点的电位。设选中的行选择线处于读出放大器右侧(如行 65)，同时使另一侧的预选单元选择线有效。这样，在放大器两侧的位线 W_1 和 W_2 上将有不同电位：预选单元侧具有 0 与 1 电平的中间值，被选行侧则具有所存信息的电平值 0 或 1。若选中存储电路原存 1，则 W_2 电位高于 W_1 的电位，使 T_1 导通，T_2 截止，因而 W_2 端输出高电平，经 I/O 缓冲器输出 1 信息，并且 W_2 的高电平使被选存储电路的电容充电，实现信息再生。若被选存储电路原存 0，则 W_2 的电位低于 W_1 的电位，从而使 T_1 截止，T_2 导通，W_2 端输出低电平，经 I/O 缓冲器输出 0 信息，并回送到原电路，使信息再生。

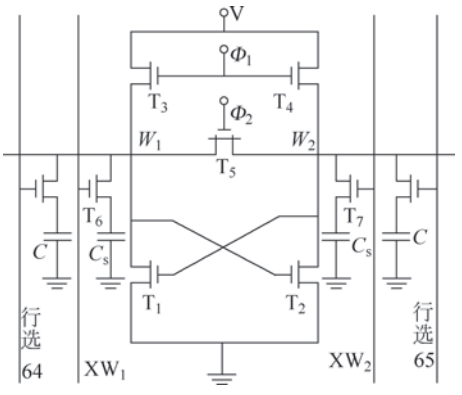

图 3-10 读出再生放大器电路

写入时，在 T_3、T_4 开始导通的同时，将待写信息加到 W_2 上。若写 1，则 W_2 加高电平，将被选电路的存储电容充电为有电荷，实现写 1。若写 0，则 W_2 为低电平，使被选电路的存储电容放电为无电荷，实现写 0。

4. 动态 RAM 的刷新方式

目前常用的动态 RAM 存储单元电路是如图 3-5 所示的电路，由于电容电荷的泄放会引起信息的丢失，因此 DRAM 需要定时刷新。隔多少时间进行一次刷新操作，主要根据电容电荷的泄放速度决定。设存储电容为 C，其两端电压为 u，电荷 $Q=C\cdot u$，则泄漏电流为：$I=\dfrac{\Delta Q}{\Delta t}=C\dfrac{\Delta u}{\Delta t}$，因而泄漏时间 $\Delta t=C\dfrac{\Delta u}{I}$，若 $C=0.2\text{pF}$，允许电压变化 $\Delta u=1\text{V}$，泄漏电流 $I=0.1\text{nA}$，所以 $\Delta t=0.2\times10^{-12}\times\dfrac{1}{0.1\times10^{-9}}=2\text{ms}$。

由此得出，上面示例的动态 MOS 存储器每隔 2ms 必须刷新一次，称作刷新最大周期。随

着半导体芯片技术的进步，刷新周期可达到 2ms、4ms、8ms，甚至更长。

由于 DRAM 存储电路的读操作是破坏性的，读完操作要立即再生，因此对 DRAM 芯片的刷新实质上是一次读操作。刷新是按行进行的，每次只加行地址，不加列地址，即可实现被选行上的所有存储电路的刷新。控制电路中有专门的刷新地址计数器指明刷新行地址，每刷新一行，刷新地址计数器加 1。动态存储器的刷新方式通常有下面几种。

1）集中式刷新方式

这种刷新方式是按照存储器芯片容量大小集中安排刷新操作的时间段，在此时间段内对芯片内所有的存储单元电路执行刷新操作，在此期间禁止 CPU 对存储器进行正常的访问，称它为 CPU 的“死区”。例如，某动态存储器芯片的容量为 16K×1 位，存储矩阵为 128×128。一次刷新操作可同时刷新存储阵列中位于同一行的 128 个存储元，因此对芯片内的所有存储单元电路全部刷新一遍需要 128 个存取周期。刷新操作要求在 2ms 内留出 128 个存取周期专门用于刷新，假设该存储器的存取周期为 500ns，则在 2ms 内有 64μs 专门用于刷新操作，其余 1936μs 用于正常的存储器操作，如图 3-11(a)所示。

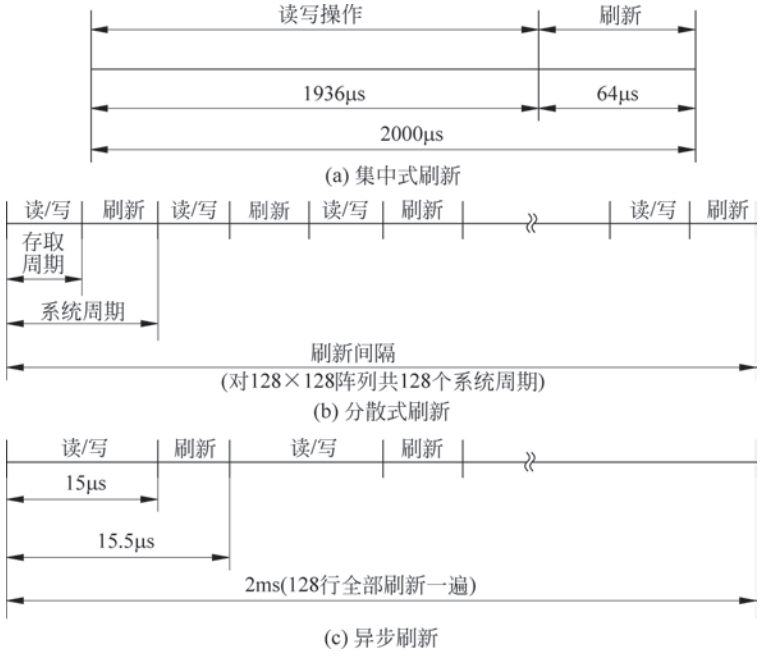

图 3-11　动态存储器的 3 种刷新方式

2）分散式刷新方式

在这种刷新方式中，定义系统对存储器的存取周期是存储器本身的存取周期的两倍。把系统的存取周期平均分为两个操作阶段，前一个阶段用于对存储器的正常访问，后一个阶段用于刷新操作，每次刷新一行，如图 3-11(b)所示。显然这种刷新方式没有“死区”，但由于没有充分利用所允许的最大的刷新时间间隔，导致刷新过于频繁，人为降低了存储器的速度。就上面的例子而言，仅每隔 128μs 就对所有的存储单元电路实施了一遍刷新操作。

3）异步式刷新方式

异步式刷新方式是上述两种方式的折中。按上述例子，每隔 2ms/128＝15.625μs 时间间隔刷新一次(128 个存储单元电路)即可。取存取周期的整数倍，则每隔 15.5μs 时间间隔刷新一次，在 15.5μs 中，前 15μs(30 个存取周期)用于正常的存储器访问，后 0.5μs 用于刷新，时间

分配情况如图3-11(c)所示。异步式刷新方式既充分利用了所允许的最大的刷新时间间隔,保持了存储器的应有速度,又大大缩短了"死区"时间,所以是一种常用的刷新方式。

4) 透明刷新(隐含式刷新)

前3种刷新方式均延长了存储器系统周期,占用CPU的时间。实际上,CPU在取指周期后的译码时间内,存储器为空闲阶段,可利用这段时间插入刷新操作,这不占用CPU时间,对CPU而言是透明的。这时设有单独的刷新控制器,刷新由单独的时钟、行计数与译码独立完成,目前高档微机中大部分采用这种方式。

3.2.4 半导体存储器的组成

CPU对存储器进行读写操作,首先要由地址总线给出地址信号,然后要发出相应的读/写控制信号,最后才能在数据总线上进行信息交流。所以,存储芯片与CPU的连接主要包括地址信号线的连接、数据信号线的连接和控制信号线的连接。

但由于一块存储器芯片的容量总是有限的,因此内存总是由一定数量的存储器芯片构成。要组成一个主存储器,首先要考虑如何选芯片以及如何把许多芯片连接起来,之后按照上述3部分将整个存储器与CPU连接起来。

存储芯片的选择通常要考虑存取速度、存储容量、电源电压、功耗及成本等多方面的因素。就主存所需芯片的数量而言,可由下面的公式求得:

$$\text{芯片总数}=\frac{\text{主存储器总的单元数}\times\text{位数}/\text{单元}}{\text{每片存储芯片的单元数}\times\text{位数}/\text{单元}} \tag{3-1}$$

例如用64K×1位的芯片组成256K×8位的存储器,则所需的芯片数为:

$$\frac{256\text{K}\times 8\text{ 位}}{64\text{K}\times 1\text{ 位}}=32(\text{片})$$

通常存储器芯片在单元数和位数方面都与要搭建的存储器有很大差距,所以需要在字方向和位方向两个方面进行扩展,按扩展方向分为下列3种情况。

1. 位扩展

如果芯片的单元数(字数)与存储器要求的单元数是一致的,但是存储芯片中单元的位数不能满足存储器的要求,就需要进行位扩展,即位扩展只是进行位数扩展(加大字长),不涉及增加单元数。

位扩展的连接方式是将所有存储器芯片的地址线、片选信号线和读/写控制线一一并联起来,连接到CPU地址和控制总线的对应位上。而各芯片的数据线单独列出,分别接到CPU数据总线的对应位。

例3-1 用1K×4位的芯片构成lK×8位的存储器。

解: 存储器要求容量为1K×8位,单元数满足,位数不满足。由式(3-1)可知,需要(1K×8)/(1K×4)=2片芯片来构成存储器。

具体的连接方式如图3-12所示。1K×8位的存储器共8根数据线$D_7\sim D_0$,两片芯片各自的4位数据引脚分别连接数据总线的$D_7\sim D_4$和$D_3\sim D_0$。芯片本身有10位地址线,称为片内地址线,与存储器要求的10根地址线一致,所以只要将它们并接起来即可。电路中CPU的读/写控制线R/$\overline{\text{W}}$与芯片的读/写控制线$\overline{\text{WE}}$信号并接。$\overline{\text{MREQ}}$为CPU的访存请求信号,作为芯片的片选信号连接到$\overline{\text{CS}}$上。

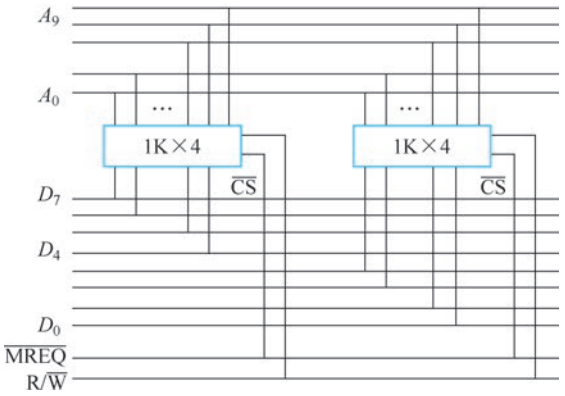

图 3-12　存储器位扩展连接图

2. 字扩展

字扩展仅是单元数扩展，也就是在字方向扩展，而位数不变。在进行字扩展时，将所有芯片的地址线、数据线和读/写控制线一一对应地并联在一起，连接到 CPU 的地址、数据、控制总线的对应位上。利用选片信号来区分被选中的芯片，选片信号由高位地址（除去用于芯片内部寻址的地址之后的存储器高位地址部分）经译码进行控制。

例 3-2　用 16K×8 位的存储器芯片构成 64K×8 位的存储器。

解： 16K×8 位的芯片可以满足 64K×8 位的存储器数据位的要求，但不满足单元数的要求。由式(3-1)可算出共需要 4 片 16K×8 位的芯片采用字扩展方式来构成存储器。具体的连接方式如图 3-13 所示。

64K×8 位的存储器共 8 根数据线 $D_7 \sim D_0$，分别接 4 块芯片的对应数据引脚。CPU 的读/写控制线（R/$\overline{W}$）与 4 块芯片的 $\overline{WE}$ 信号并接。64K×8 位的存储器需要 16 位地址线 $A_{15} \sim A_0$，而 16K×8 位的芯片的片内地址线为 14 根，所以用 16 位地址线中的低 14 位 $A_{13} \sim A_0$ 进行片内寻址，高两位地址 A_{15}、A_{14} 用于选片寻址，作为片选译码器的输入，译码器的 4 位输出分别接 4 块芯片的 $\overline{CS}$ 引脚。访存请求信号 $\overline{MREQ}$ 接译码器的使能端。若存储器从 0 开始连续编址，则 4 块芯片的地址分配如下。

第一片地址范围为 0000H～3FFFH（高两位地址 A_{15}、A_{14} 为 00 时，选中第一片芯片）。

第二片地址范围为 4000H～7FFFH（高两位地址 A_{15}、A_{14} 为 01 时，选中第二片芯片）。

第三片地址范围为 8000H～BFFFH（高两位地址 A_{15}、A_{14} 为 10 时，选中第三片芯片）。

第四片地址范围为：C000H～FFFFH（高两位地址 A_{15}、A_{14} 为 11 时，选中第四片芯片）。

3. 字和位同时扩展

在构建主存储器空间时，往往需要字和位同时扩展，是位扩展与字扩展的组合，可按下面的规则实现。

（1）确定组成主存储器需要的芯片总数。

（2）所有芯片对应的地址线接在一起，接到 CPU 引脚的对应位，所有芯片的读写控制线接在一起，接入 CPU 的读写控制信号上。

（3）所有处于同一地址区域芯片的选片信号接在一起，接到选片译码器对应的输出端。

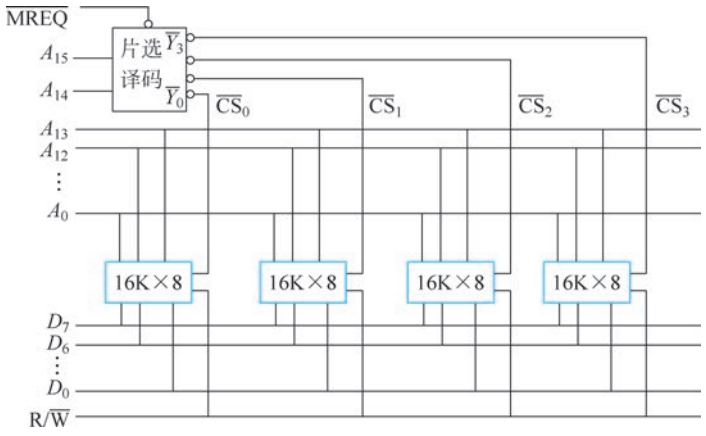

图 3-13 存储器字扩展连接图

(4) 所有处于不同地址区域的同一位芯片的数据输入/输出线对应地接在一起,接到CPU数据总线的对应位。

例 3-3 用1K×4位的芯片组成4K×8位的存储器。

解: 用1K×4位的芯片构成4K×8位的存储器,由式(3-1)可知,所需芯片数为$\frac{4K\times8\text{位}}{1K\times4\text{位}}=8$(块)。8块芯片分成4组,每组组内按位扩展方法连接,两组组间按字扩展方法连接。

图3-14为该例中芯片的连接图。

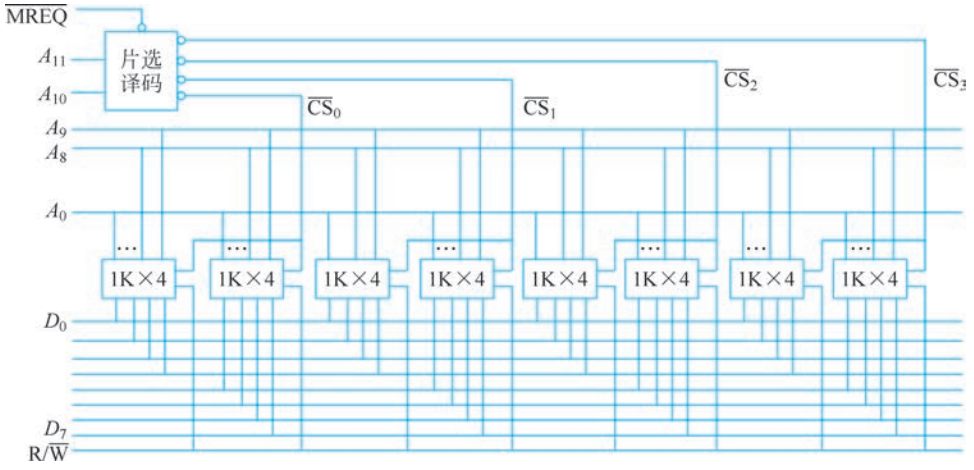

图 3-14 存储器字位扩展连接图

4K×8位的存储器共8位数据线$D_7\sim D_0$,每组两块芯片各自的4位数据引脚$I/O_3\sim I/O_0$分别连接到数据总线的$D_7\sim D_4$和$D_3\sim D_0$。电路中CPU的读/写控制线R/$\overline{W}$与8块芯片的读写控制线$\overline{WE}$信号并接。4K×8位的存储器共12根地址线$A_{11}\sim A_0$,而1K×4位的芯片的片内地址线为10根,所以用12位地址线中的低10位$A_9\sim A_0$进行片内寻址,高两位地址A_{11}、A_{10}用于选片寻址。译码器的每位输出接同一地址区域的两块芯片的$\overline{CS}$引脚。若存储器从0开始连续编址,则4组芯片的地址分配如下。

第一片地址范围为0000H~03FFH(地址A_{11}、A_{10}为00时,选中第一组两块芯片)。

第二片地址范围为0400H~07FFH(地址A_{11}、A_{10}为01时,选中第二组两块芯片)。

第三片地址范围为 0800H～0BFFH(地址 A_{11}、A_{10} 为 10 时,选中第三组两块芯片)。

第四片地址范围为 0C00H～0FFFH(地址 A_{11}、A_{10} 为 11 时,选中第四组两块芯片)。

4. 多种数据的传输

多种数据的传输是指存储器按照 CPU 的指令要求,在 CPU 间传输 8 位、16 位、32 位或 64 位数据的情况。此时,CPU 要增加控制信号,控制存储器传输不同位数的数据。

整数边界存储是指当计算机具有多种信息长度(8 位、16 位、32 位等)时,应当按存储周期的最大信息传输量为界存储信息,保证数据都能在一个存储周期内存取完毕。例如,假设计算机字长为 64 位,一个存储周期内可传输 8 位、16 位、32 位、64 位等不同长度的信息。那么一个 8 位、两个 16 位、两个 32 位、一个 64 位等信息如何在主存储器中存放呢?

若信息存储不合理,即不考虑整数边界问题,虽然一个存储周期的最大数据传输量为 64 位,但也会出现两个周期才能将数据传送完毕的情况。如图 3-15(a)所示,图中每个方格代表一个字节的存储空间,则第一个 16 位、第二个 32 位和 64 位都需要两个存储周期才能完成访问。无边界规定有可能造成系统访存速度下降。

图 3-15(b)是按整数边界要求的信息存储情况图,图中画“○”的字节单元代表未存放任何有效信息。此时,各种数据的整数边界地址(地址码用二进制表示)安排如下。

8 位(1 字节)	地址码最低位为任意值	XXXXXB
16 位(2 字节)	地址码最低 1 位为 0	XXXX0B
32 位(4 字节)	地址码最低 2 位为 00	XXX00B
64 位(8 字节)	地址码最低 3 位为 000	XX000B

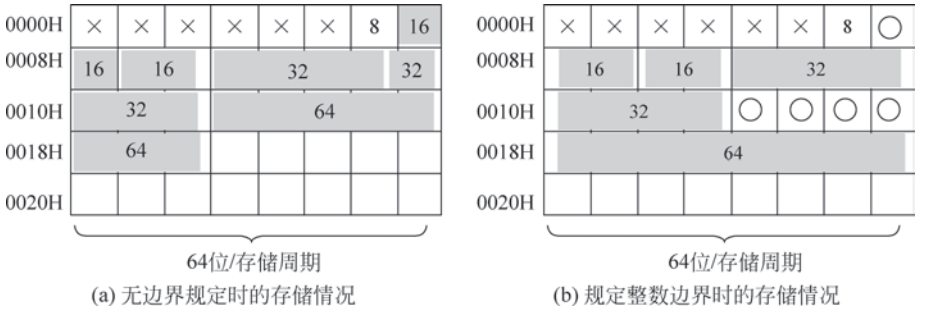

图 3-15　多数据长度的存储信息图

整数边界存储虽然浪费空间,但随着半导体存储器的扩容,以空间换取时间势在必行。

例 3-4　请用 2K×8 位的 SRAM 芯片构成一个 8K×16 位的存储器,要求当 CPU 给出的控制信号 $B=0$ 时访问 16 位数据,$B=1$ 时访问 8 位数据。存储器以字节为单位编址。

解:由式(3-1)可知,该存储器所需要的芯片总数为$\dfrac{8\text{K}\times 16\text{ 位}}{2\text{K}\times 8\text{ 位}}=8$(块)。根据前面的讨论可知,8 块芯片分成 4 组,每组两块芯片,用于实现位扩展,4 组芯片实现了字扩展。

由于存储器以字节为单位编址,总容量为 8K×16 位,所以 8K×16b=8K×2×8b=$2^{14}\times$8b,所以 CPU 控制线中有 14 位地址线 A_{13}～A_0,16 位数据线 D_{15}～D_0。

因为存储器需要访问 16 位数据,但每块芯片数据线只有 8 位,考虑整数边界的要求,将 16 位数据分为高 8 位和低 8 位,分别用奇存储体和偶存储体存放,即 4 组芯片中每组的两块芯片,一片位于奇存储体,一片位于偶存储体。选中位于偶存储体的芯片工作时,CPU 送来的

最低位地址 A_0 为0,选中位于奇存储体的芯片时,CPU送来的地址 A_0 为1,这也称为交叉编址。访问8位数据时可以访问任意主存地址,访问16位数据时必须同时访问同组芯片的地址,分别提供高8位(存放在奇存储体)和低8位(存放在偶存储体)数据,以此保证在一个存储周期中完成16位数据的存/取操作。所以用最低位地址 A_0 区分奇存储器、偶存储体。

规定14根地址线中,A_0 与 B 组合用于控制8位、16位数据的存取。由于每块SRAM芯片容量是2K,所以CPU地址总线中的 $A_{11}\sim A_1$ 用于片内地址,高位地址 A_{13}、A_{12} 用于片选译码,得到4位译码输出信号 $\overline{Y}_0$、$\overline{Y}_1$、$\overline{Y}_2$ 和 $\overline{Y}_3$,与 A_0、B 组成每块芯片的片选信号。

设选中偶存储体时,$C=1$,选中奇存储体时,$D=1$,则得出表3-1的真值表。

由此真值表得下面的逻辑表达式:

$$C=\overline{A_0},\quad D=\overline{B\oplus A_0},$$

则8块芯片的片选信号的逻辑表达式为:

$$\overline{CS_0}=\overline{C\cdot Y_0}\quad \overline{CS_2}=\overline{C\cdot Y_1}\quad \overline{CS_4}=\overline{C\cdot Y_2}\quad \overline{CS_6}=\overline{C\cdot Y_3}$$

$$\overline{CS_1}=\overline{D\cdot Y_0}\quad \overline{CS_3}=\overline{D\cdot Y_1}\quad \overline{CS_5}=\overline{D\cdot Y_2}\quad \overline{CS_7}=\overline{D\cdot Y_3}$$

存储器结构及与CPU连接的示意图如图3-16所示。

表3-1 C、D 取值真值表

B A_0	C D	说 明
0 0	1 1	访问16位数据
0 1	0 0	不访问,地址不满足整数边界要求
1 0	1 0	访问偶存储体,存取8位数据
1 1	0 1	访问奇存储体,存取8位数据

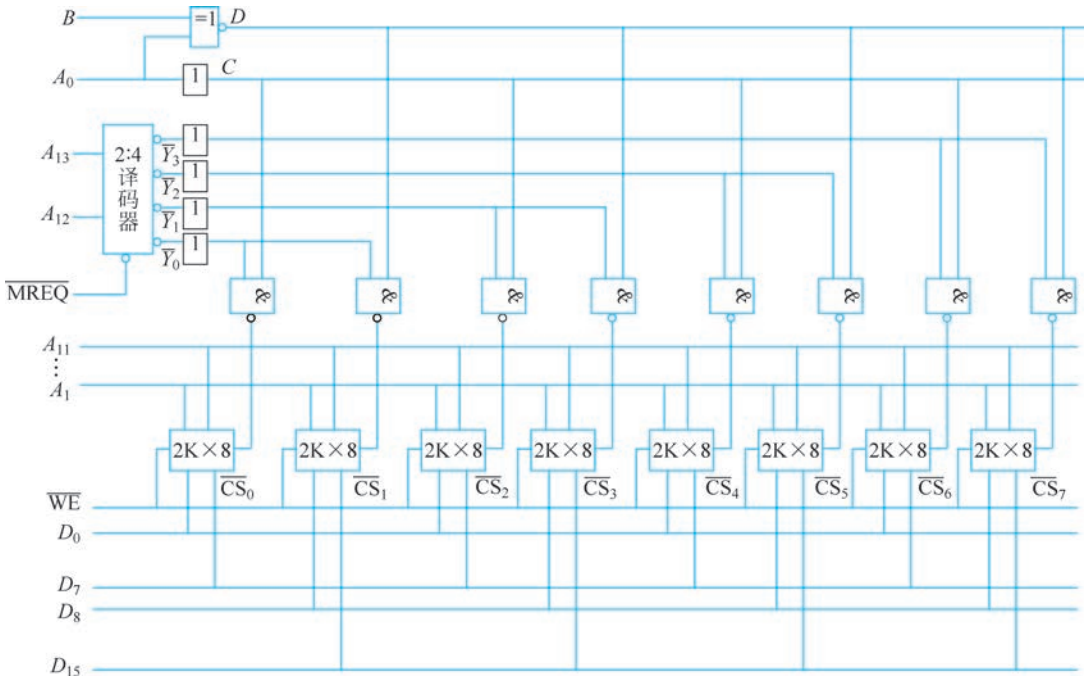

图3-16 存储器多数据传输举例

3.3　非易失性存储器

非易失性存储器(Non-Volatile Memory,NVM)是指当关机后,存储器中的内容不会随之消失的计算机存储器。在非易失性存储器中,根据存储器中的内容是否能在计算机工作时随时改写为标准,可分为 3 类:只读存储器(Read-Only Memory,ROM)、闪速存储器(Flash Memory)和新型的非易失性存储器。

3.3.1　只读存储器

只读存储器的特点是,在系统断电以后,只读存储器中所存储的内容不会丢失。因此,只读存储器是非易失性存储器。半导体只读存储器常作为主存的一部分,用于存放一些固定的程序,如监控程序、启动程序、磁盘引导程序等。只要一接通电源,这些程序就能自动运行。此外,只读存储器还可以用作控制存储器、函数发生器、代码转换器等。在输入、输出设备中,常用 ROM 存放字符、汉字等的点阵图形信息。

只读存储器的类型多种多样,如掩膜 ROM、可编程 ROM、紫外线擦除可编程 ROM、电擦除可编程 ROM、闪速可擦除可编程 ROM。下面对它们分别做出简要说明。

1. 掩膜 ROM

掩膜 ROM 中的内容是由半导体存储芯片制造厂家在制造该芯片时直接写入 ROM 中的,即掩膜 ROM 不是用户可编程 ROM。掩膜 ROM 的主要优点是比其他类型的 ROM 便宜,但是一旦掩膜 ROM 中的某个代码或数据有错误,整批的掩膜 ROM 都得扔掉。

2. 可编程 ROM

可编程 ROM(Programmable ROM,PROM)是提供给用户,将要写入的信息“烧”入 ROM。PROM 为一次可编程 ROM(One Time Programmable ROM,OTPROM)。对 PROM 写入信息需要用一个叫 ROM 编程器的特殊设备来实现这个过程。

3. 紫外线擦除可编程 ROM

人们发明用紫外线实现擦除的可编程 ROM(Erasable Programmable ROM,EPROM)的目的是使已写入 PROM 中的信息能被修改(与 PROM 有本质的不同),且可被编程、擦除几千次。EPROM 的问题是:需要紫外设备,EPROM 芯片有一个窗口用于接收紫外线,通过紫外线照射擦除其内容。擦除芯片的内容耗时为分钟级。

4. 电擦除可编程 ROM

与 EPROM 比,电擦除可编程 ROM(Electrically Erasable Programmable ROM,EEPROM)有许多优势。其一是用电来擦除原有信息,实现瞬间擦除,而 EPROM 需要 20min 左右的擦除时间。此外,使用者可以有选择地擦除具体字节单元的内容,而 EPROM 擦除的是整个芯片的内容,并且 EEPROM 的使用者可直接在电路板上对其进行擦除和编程,不需要额外的擦除和编程设备。这要求系统设计者在电路板上设置对 EEPROM 进行擦除和编程的电路。EEPROM 的擦除一般使用 12.5V 的电压,即在 V_{PP} 引脚上要加有 12.5V 的电压。

5. 闪速可擦除可编程 ROM

闪速可擦除可编程 ROM,又称闪速存储器,简称闪存,起源于 20 世纪 90 年代初,是深受欢迎的用户可编程存储芯片。闪存正在逐渐替代原来个人计算机中的 BIOS ROM。有的设计人员认为闪存将来可能替代硬盘,如此将大大改善计算机的性能,因为闪存的存取时间在 100ns 之内,而磁盘的存取时间为毫秒级。

闪存替代硬盘有两个问题必须解决:一是成本因素,即同等容量的 U 盘价格应与同等容量的硬盘价格相差不大;二是闪存可擦写的次数,必须像硬盘一样在理论上是无限的(这是硬盘的工作原理所决定的),而闪存的可擦写次数是有限的。

3.3.2 闪速存储器

闪速存储是只读存储器的一种,由于其是一种较为常见的只读存储器,下面对其做出详细介绍。

东芝公司的发明人 Fujio Masuoka 于 1984 年首先提出了快速闪存的概念。与传统计算机内存不同,闪存的特点是拥有非易失性。

Intel 是世界上第一个生产闪存并将其投放市场的公司。1988 年,公司推出了一款 256Kb 的闪存芯片。它如同鞋盒一样大小,内嵌于一个录音机里。后来,Intel 发明的这类闪存被统称为 NOR 闪存。它结合了 EPROM 和 EEPROM 两项技术,并拥有一个 SRAM 接口。

第二种闪存是日立公司于 1989 年研制的 NAND 闪存,它被认为是 NOR 闪存的理想替代者。NAND 闪存的写周期是 NOR 闪存的 1/10 倍,保存与删除处理的速度快,存储单元只有 NOR 的一半,但读取速度要慢于 NOR 型闪存。

1. 闪速存储器的基本原理

闪存以单晶体管作为二进制信号的存储单元,它的结构与普通的半导体晶体管(场效应管)非常类似,如图 3-17 所示,区别在于闪存的晶体管加入了浮动栅(Floating Gate)和控制栅(Control Gate)。前者用于存储电子,表面被一层硅氧化物绝缘体所包覆,并通过电容与控制栅相耦合。当负电子在控制栅的作用下被注入浮动栅中时,该 NAND 单晶体管的存储状态就由 1 变成 0。反之,当负电子从浮动栅中移走后,存储状态就由 0 变成 1;而包覆在浮动栅表面的绝缘体的作用就是将内部的电子困住,以达到保存数据的目的。如果要写入数据,就必须将浮动栅中的负电子全部移走,令目标存储区域都处于 1 状态,这样只有遇到数据 0 时才发生写入动作。

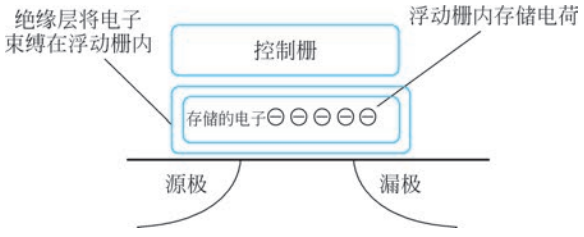

图 3-17 闪存的基本存储单元结构图

闪存有几种不同的电荷生成与存储方案,应用最广泛的是通道热电子编程(Channel Hot Electron,CHE),该方法通过对控制栅施加高电压,使传导电子在电场的作用下突破绝缘体的

屏障进入浮动栅内部，反之亦然，以此来完成写入或者擦除动作；另一种方法被称为隧道效应法(Fowler-Nordheim，FN)，该方法直接在绝缘层两侧施加高电压形成高强度电场，帮助电子穿越氧化层通道进出浮动栅。

2. 闪速存储器的特点

固有的非易失性：SRAM 和 DRAM 断电后保存的信息随即丢失，为此 SRAM 需要备用电池来保存数据，而 DRAM 一般需要磁盘作为后援存储器。由于闪存具有可靠的非易失性，因此它是一种理想的存储器。

廉价和高密度：和 SRAM 及 DRAM 相比，相同存储容量的闪存具有更低的成本。

可直接执行：NOR 型闪速存储器中存储的应用程序可以直接在闪存内运行，不必再把代码读到系统 RAM 中，而磁盘中存储的应用程序要先加载到 RAM 中，才能执行。

固态性能：闪速存储器是一种低功耗、高密度且没有机电移动装置的半导体技术，访问速度也快于传统磁盘，因而特别适合便携式微型计算机系统，如固态硬盘(Solid State Disk，SSD)。但固态硬盘要想完全替代磁盘至少要解决两个问题：一是固态硬盘每位存储成本高于磁盘；二是固态硬盘的访问次数是有限的，而磁盘的访问次数在理论上是无限的。

3. 闪速存储器的分类

根据技术架构的不同，闪速存储器可分为如下两类。

1) NOR 型闪存

NOR 型闪存工作时同时使用通道热电子编程和隧道效应法两种方法。通道热电子编程用于数据写入，支持单字节或单字编程；隧道效应法则用于擦除，但 NOR 不能单字节擦除，必须以块为单位或对整片区域执行擦除。由于擦除和编程速度慢，块尺寸也较大，使得 NOR 闪存的擦除和编程花费时间长，无法胜任纯数据存储和文件存储之类的应用。

NOR 型闪存带有 SRAM 接口，有足够的地址引脚来寻址，可以很容易地存取其内部的每一个字节，因此它支持代码本地直接运行，即应用程序可以直接在闪存内运行，不必再把代码读到系统 RAM 中，这是嵌入式应用经常需要的一个功能。但其单位存储价格比较高，容量比较小，比较适合频繁随机读写的场合。

2) NAND 型闪存

NAND 型闪存工作时采用隧道效应法写入和擦除，单晶体管的结构相对简单，使其存储单元只有 NOR 的一半，因而存储密度较高。与 NOR 相比，NAND 型闪存的写和擦除操作的速度快，但其随机存取的速率慢。NAND 型闪存的基本存取单元是页(Page)。每一页的有效容量是 512 字节的倍数，类似于硬盘的扇区，所谓的有效容量是指用于数据存储的部分。例如，每页的有效容量是 2048 字节，外加 64 字节的空闲区，空闲区通常被用于纠错码、损耗均衡(Wear Leveling)等。

与磁盘和 DRAM 不同，类似于 EEPROM 技术，对闪存的写操作会损耗存储位，所以闪存的操作次数是受限的。为了尽量延长其寿命，大多数闪存产品都有一个控制器，用来将写操作从已经写入很多次的块中映射到写入次数较少的块中，从而使写操作尽量分散。这种技术称为损耗均衡。该技术也可以将芯片在制造过程中出错的存储单元屏蔽掉。

根据闪存颗粒中单元存储密度的差异，NAND 型闪存又分为 SLC(Single-Level Cell，单层单元)、MLC(Multi-Level Cell，多层单元)、TLC(Triple-Level Cell，三层单元)、QLC(Quad-

Level Cell,四层单元)等多种类型。SLC每个单元存储一位二进制数据,这种设计提高了耐久性、准确性和性能。它的价格最高。MLC每个单元存储两位二进制数据。尽管在存储单元中存储多位能够在相同空间内获得更大容量,但它的代价是使用寿命降低,可靠性降低。TLC每个单元存储三位二进制数据,通常用于性能和耐久性要求相对较低的消费级电子产品,最适合包含大量读取操作的应用程序。QLC每个单元存储四位二进制数据,这种产品在大数据等应用中发挥了很大的作用。

NAND型闪存主要用来存储资料,它常常被应用于诸如数码照相机、数码摄像机、闪存卡、固态硬盘等产品。

3.3.3 新型的非易失性存储器

新型的非易失性存储器通常包括铁电存储器(Ferroelectric Random Access Memory,FRAM)、相变存储器(Phase Change Random Access Memory,PCRAM)、磁性存储器(Magnetic Random Access Memory,MRAM)和阻变存储器(Resistive Random Access Memory,RRAM)等。其中,FRAM是一种在断电时不会丢失内容的非易失性存储器,具有高速、高密度、低功耗和抗辐射等优点。PCRAM利用特殊材料在晶态和非晶态之间相互转化时所表现出来的导电性差异来存储数据。MRAM利用磁性隧道结的隧穿磁电阻效应对数据进行存储。RRAM是以非导性材料的电阻在外加电场的作用下,在高阻态和低阻态之间可逆转换为基础的非易失性存储器。相比其他非易失性存储技术,RRAM是高速存储器。下面重点介绍RRAM芯片的重要电子元件——忆阻器(Memristor)。

忆阻器是表示磁通与电荷关系的电路器件。其中,忆阻的阻值是由流经它的电荷来确定的。因此,通过测定忆阻的阻值便可知道流经它的电荷量,从而具有记忆电荷的作用。简单来说,忆阻器是一种有记忆功能的非线性电阻,通过控制电流的变化可改变其阻值。例如,把高阻值定义为1,低阻值定义为0,这种电阻就可以实现存储数据的功能。

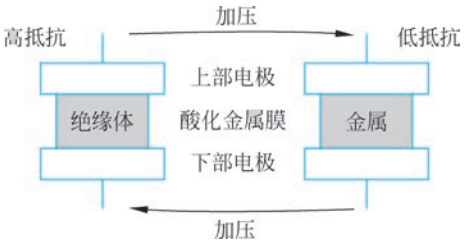

图3-18 忆阻器的基本原理

在每个忆阻器中,底部和顶部的导线分别与器件的两边接触。忆阻器是由两个金属电极夹着的二氧化钛层构成的双端与双层交叉开关结构的半导体。其中一层二氧化钛掺杂了氧空位,成为一个半导体,而相邻一层不掺任何东西,保持绝缘体的自然属性,通过检测交叉开关两端电极的阻性就能判断RRAM的"开"或者"关"状态,如图3-18所示。

忆阻器除了其独特的记忆功能外,还有两大特性:一是有更短的访问时间和更快的读写速度,它整合了闪速存储器和DRAM的部分特性;二是存储单元小,尺寸可以做到几纳米。由于忆阻器尺寸小、能耗低,因此能很好地存储和处理信息。例如,一个忆阻器的工作量相当于一枚CPU芯片中十几个晶体管共同产生的效用,而且其可以与CMOS技术相兼容,是下一代非易失性存储技术的发展趋势。

3.4 并行存储器

随着计算机的不断发展,虽然存储器系统速度也在不断提高,但始终跟不上CPU速度的提高,因而使之成为限制系统速度的一个瓶颈。为了解决两者的速度匹配问题,通常采用的方

法有：一是采用更高速的主存储器，或加长存储器的字长；二是采用并行操作的双端口存储器；三是在每个存储器周期中存取几个字，即采用并行存储器；四是在 CPU 和主存储器之间插入一个高速缓冲存储器(Cache)。

本节先介绍双端口存储器，然后介绍并行主存系统，最后介绍相联存储器，下一节介绍高速缓冲存储器。

3.4.1　双端口存储器

常规的存储器是单端口存储器，每次只接收一个地址，访问一个编址单元，从中读取或存入一个字节或一个字。在执行双操作数指令时，就需要分两次读取操作数，工作速度较低。在高速系统中，主存储器是信息交换的中心。一方面 CPU 频繁地访问主存，从中读取指令、存取数据，另一方面外围设备也需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，要么读要么写，这也影响了工作速度。为此，在某些系统或部件中使用双端口存储器，已有集成芯片可用。

如图 3-19 所示，双端口存储器具有两个彼此独立的读/写口，每个读/写口都有一套独立的地址寄存器和译码电路、数据总线和控制总线，可以并行地独立工作。

当送达两个端口的访存地址不同时，在两个端口上进行读写操作，一定不会发生冲突。每个端口都可以独立对存储器进行读写，就像是两个存储器在同时工作，实现了并行存储操作。

当两个端口地址总线上送来的是存储器同一单元的地址时，便发生读写冲突。为解决此问题，双端口存储器芯片特设置 $\overline{\text{BUSY}}$ 标志。在这种情况下，芯片上的判断逻辑可以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 $\overline{\text{BUSY}}$ 标志(使其变为低电平)，即暂时关闭此端口。换句话说，读写操作对 $\overline{\text{BUSY}}$ 变为低电平的端口是不起作用的。优先端口完成了读写操作，才将被延迟端口的 $\overline{\text{BUSY}}$ 复位(使其变为高电平)，开放此端口允许延迟端口进行存取。

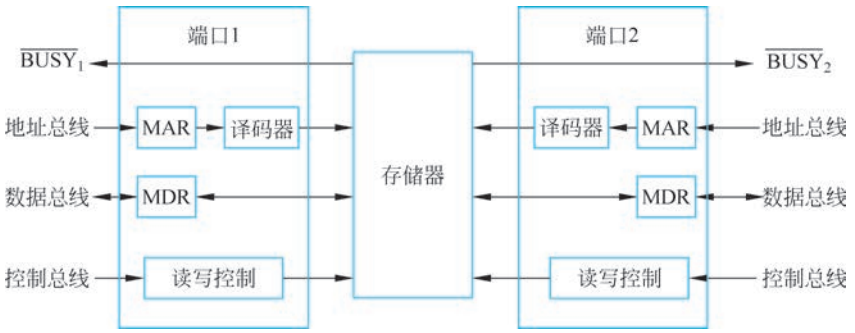

图 3-19　双端口存储器

双端口存储器的常见应用场合有：一种应用是在运算器中采用双端口存储芯片，作为通用寄存器组，能快速提供双操作数，或快速实现寄存器间的传送；另一种应用是让双端口存储器的一个读/写口面向 CPU，通过专门的存储总线(或称局部总线)连接 CPU 与主存，使 CPU 能快速访问主存，另一个读/写口则面向外围设备或输入输出处理机 IOP，通过共享的系统总线连接，这种连接方式具有较大的信息吞吐量。此外，在多机系统中，常采用双端口存储器甚至多端口存储器，作为各 CPU 的共享存储器，以实现多 CPU 之间的通信。

3.4.2 并行主存系统

为解决主存与CPU之间的速度差异,在高速的大型计算机中普遍采用并行主存系统,在一个存储周期内可并行存取多个字,从而提高整个存储器系统的吞吐率(数据传送率),以解决CPU与主存间的速度匹配问题。通常有两种方式。

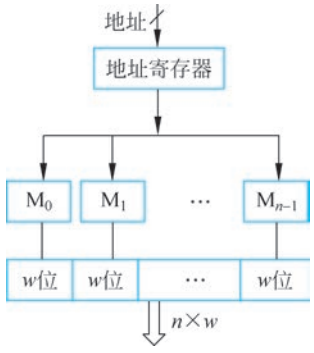

图 3-20 单体多字并行主存系统

1. 单体多字并行主存系统

如图3-20所示,多个并行存储器共用一套地址寄存器,按同一地址码并行地访问各自的对应单元。例如读出沿这 n 个存储器顺序排列的 n 个字,每个字有 w 位。假定送入的地址码为 A,则 n 个存储器同时访问各自的 A 号单元。我们也可以将这 n 个存储器视作一个大存储器,每个编址对应于 n 字×w 位,因而称为单体多字方式。

单体多字并行主存系统适用于向量运算一类的特定环境。在执行向量运算指令时,一个向量型操作数包含 n 个标量操作数,可按同一地址分别存放于 n 个并行主存之中。例如矩阵运算中的 $a_ib_j=a_0b_0$、a_0b_1…,就适合采用单体多字并行存取方式。

2. 多体交叉并行主存系统

在大型计算机中使用更多的是多体交叉存储器,如图3-21所示,一般使用 n 个容量相同的存储器,或称为 n 个存储体,它们具有自己的地址寄存器、数据线、时序,可以独立编址地同时工作,因而称为多体方式。

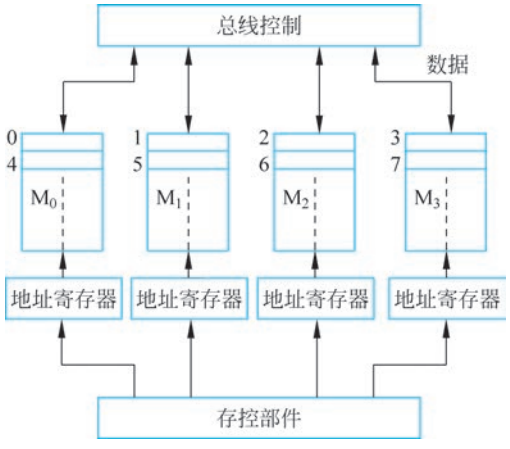

图 3-21 多体交叉并行主存系统

各存储体的编址大多采用交叉编址方式,即将一套统一的编址按序号交叉地分配给各个存储体。以4个存储体组成的多体交叉存储器为例:M_0 体的地址编址序列是0,4,8,12,…,M_1 是1,5,9,13,…,M_2 是2,6,10,14,…,M_3 是3,7,11,15,…。换句话说,一段连续的程序或数据,将交叉地存放在几个存储体中,因此整个并行主存是以 n 为模交叉存取的。

相应地,对这些存储体采取分时访问的时序,如图3-22所示。仍以4个存储体为例,模等于4,各体分时启动读/写,时间错过四分之一存取周期。启动 M_0 后,经 $T_M/4$ 启动 M_1,在

$T_M/2$ 时启动 M_2，在 $3T_M/4$ 时启动 M_3。各体读出的内容也将分时地送入 CPU 中的指令栈或数据栈，每个存取周期可访存 4 次。

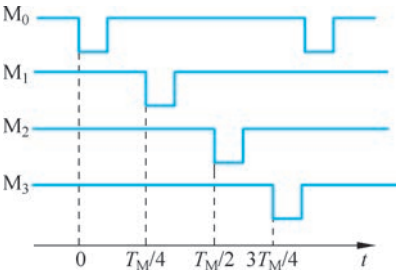

图 3-22　多存储体分时工作示意图

采取多体交叉存取方式，需要一套存储器控制逻辑，简称为存控部件。它由操作系统设置或控制台开关设置，确定主存的模式组合，如所取的模是多大；接收系统中各部件或设备的访存请求，按预定的优先顺序进行排队，响应其访存请求；分时接收各请求源发来的访存地址，转送至相应的存储体；分时收发读写数据；产生各存储体所需的读/写时序；进行校验处理等。显然，多体交叉存取方式的存控逻辑比较复杂。

当 CPU 或其他设备发出访存请求时，存控部件按优先排队决定是否响应请求，并按交叉编址关系决定该地址访问哪个存储体，然后查询该存储体的“忙”触发器是否为 1。若为 1，则表示该存储体正在进行读/写操作，需等待；若该存储体已完成一次读/写，则将“忙”触发器置 0，然后可响应新的访存请求。当存储体完成读/写操作时，将发出一个回答信号。

这种多体交叉存取方式很适合支持流水线的处理方式，而流水线处理方式已是 CPU 中的一种典型技术。因此，多体交叉存储结构是高速大型计算机的典型主存结构。

3.4.3　相联存储器

常规存储器是按地址访问的，即送入一个地址编码，选中相应的一个编址单元，然后进行读写操作。在信息检索一类工作中，需要的却是按信息内容选中相应单元，进行读写。例如，从一份学生档案中查找某生的学习成绩，送出的检索依据是该生的姓名（字符串），找到相应的存储区，读出他的成绩。当使用常规存储器进行检索时，就需要采用某种搜索算法，依次按地址选择某个存储单元，读出姓名（字符串），与检索依据进行符合比较。若不符合，则按算法修改地址，再读出另一姓名信息，进行比较。直到二者相符，表示已找到所需寻找的学生姓名，然后从此找到对应的存储区域，读出成绩数据。可见，用常规存储器进行信息检索，需将检索依据的内容设法转化为地址，因此效率往往很低。能否将有关的这些姓名与检索依据同时进行符合比较，一次就找到相符内容所存的单元呢？于是出现了相联存储器（Associative Memory）。

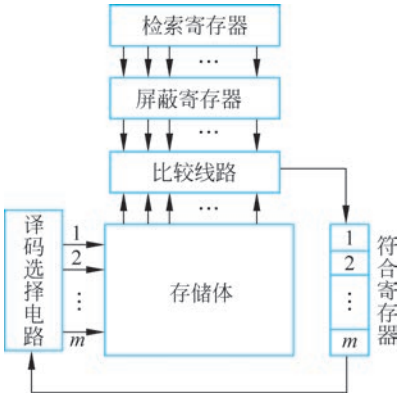

图 3-23　相联存储器的逻辑结构图

相联存储器又称为联想存储器，它是根据所存信息的全部特征或部分特征进行存取的，即一种按内容寻址的存储器。图 3-23 描述了其逻辑结构，它由存储体、检索寄存器、屏蔽寄存器、符合寄存器、比较线路、数据寄存器以及控制线路组成。检索寄存器和屏蔽寄存器的位数与存储体中存储单元的位数（n）相等，符合寄存器的位数则跟存储单元数（m）相等，即符合寄存器的每一位对应存储体中的一个存储单元。

当需要查找某一数据时，先把数据本身或数据特征标志部分（检索项）送入检索寄存器。由于每次检索时一般只用到其中的一部分，例如只输入学号，或只输入姓名，因此屏蔽寄存器中存放着屏蔽字代码。例如本次检索只用到高 8 位，即输入的检索字为高 8 位，

则屏蔽字的高8位为1,其余低位均为0,将本次不用的无效位屏蔽掉。高8位的检索信息将与相联存储器中的 m 个字的高8位同时进行符合比较,其余被屏蔽掉的无效位将不参与比较。如果输入的检索字是另外的检索项,则修改屏蔽字。屏蔽字中为1的诸位像一个窗口,只允许窗口对应的检索项进行比较操作。

比较线路的作用是把检索项同时和相联存储阵列中的每一个存储单元的相应部分进行逻辑比较,若完全相同,则把与该字对应的符合寄存器相应位置1,表示该字就是所要查找的字,并利用这个符合信号去控制该字单元的读/写操作,实现数据的读出或写入。数据寄存器则存放要读出或写入的数据。

综上所述,存储体中的每个单元都应有一套比较线路,最终产生使符合寄存器相应位置1的信号。这样相联存储器的成本高,且容量有限,所以相联存储器一般用来存放检索中可能要查询的关键信息。所以若其他信息存放在另外的常规数据存储器中,则符合寄存器的各位经编码产生地址,据此到数据寄存器中读/写。

在计算机系统中,相联存储器主要用于在虚拟存储器中存放分段表、页表和快表;在高速缓冲存储器中,相联存储器用于存放Cache的行地址。在这两种应用中,都需要快速查找。

3.5 高速缓冲存储器

3.5.1 Cache在存储体系中的地位和作用

由于集成电路技术不断进步,导致生产成本不断降低,CPU的功能不断增强,运算速度也越来越快;同时,微型计算机的应用领域也不断拓展,使得系统软件和应用软件都变得越来越大,客观上需要大容量的内存支持软件的运行,因此需要计算机配备较大容量的内存。从成本和容量两个因素考虑,现代计算机广为采用DRAM构成的内存,因为DRAM的功耗和成本低,容量大。但DRAM的速度相对较慢,很难满足高性能CPU在速度上的要求。那么如何解决这个矛盾呢?

1. 程序的局部性

根据冯·诺伊曼计算机的特点,我们注意到:在较短的时间内,程序的执行仅局限于某个部分,相应地,CPU所访问的存储器空间也局限于某个区域(至少在一段时间内是这样的),这就是程序的局部性(Locality of Reference)原理。程序的局部性表现为时间局部性和空间局部性。由于程序中存在着大量的循环结构,程序中的某条指令一旦执行,不久以后该指令可能再次执行,如果某数据被访问过,则不久以后该数据可能再次被访问,这就是时间局部性。程序的另一个典型情况就是顺序执行,一旦程序访问了某个存储单元,在不久以后,其附近的存储单元也将被访问,表现为空间局部性。

基于程序的局部性原理,高速缓冲存储器(Cache)的设计理念就是:只将CPU最近需要使用的少量指令或数据以及存放它们的内存单元的地址,复制到Cache中提供给CPU使用,即用少量速度较快的SRAM构成Cache,置于CPU和主存之间,以提高CPU的工作效率。这种设计思想利用了SRAM的速度优势和DRAM的高集成度、低功耗及低成本的特点,是多级存储层次的一个重要层级。

例如在33MHz 80386构成的系统中,如果CPU需要从内存中读取指令或数据,在不需要

插入任何等待状态的理想情况(零等待)下,最大的延迟时间为 60ns,即 CPU 发出内存地址和读命令后,最多等待 60ns 就需要在数据总线上取得它需要的指令或数据。换一个角度讲,当内存得到了 CPU 送来的地址和读命令后,只有 60ns 的时间完成地址译码、读出指令或数据并将读出的指令或数据稳定地放到数据总线上。当时在市场上满足成本要求的 DRAM 芯片,在速度上都不能满足要求,只有访问时间为 45ns 的 SRAM 在速度上才能满足 CPU 的要求。因此,在这样的系统中使用 Cache 是完全必要的。

不难想象,随着大规模集成电路技术不断进步,CPU 的工作频率进一步提高,虽然 DRAM 技术和生产工艺也在不断进步,DRAM 的读写周期在不断缩短,即速度也在不断提高,但是仍然达不到同阶段的 CPU 对内存速度的要求。问题依然存在,且变得更加严重,所以在目前的系统中均采用 Cache 和 DRAM 内存的组合结构。

2. 多级 Cache 概念

基于目前的大规模集成电路技术和生产工艺,人们已经可以在 CPU 芯片内部放置一定容量的 Cache。CPU 芯片内部的 Cache 称为一级(L1)Cache,CPU 外部由 SRAM 构成的 Cache 称为二级(L2)Cache。目前最新的 CPU 内部已经可以放置二级甚至三级 Cache。

同时也应该看到,若 CPU 随机地访问存储器,不遵循局部性原理,则 Cache 的设计理念根本无法发挥作用。目前看来,频繁且无规则地在程序中使用 CALL 或 JMP 指令将会严重地影响基于 Cache 的系统性能,但这种情况在实际应用中并不多见,也可以考虑通过加大 Cache 的容量来提高系统性能。

在带有 Cache 的计算机系统中,Cache 对于程序员是透明的。从逻辑上讲,程序员并不会感觉到 Cache 的存在,只会感觉到主存的速度加快了。

3.5.2　Cache 的结构及工作原理

Cache 的总体结构如图 3-24 所示。Cache 存储阵列由高速存储器构成,用于存放主存信息的副本。其容量虽小于主存,但编址方式、物理单元长度均与主存相同。Cache 中用于存放数据的部分称为数据 Cache,存放指令的部分称为指令 Cache,有时二者也统称为内容 Cache。在带有 Cache 的计算机系统中,Cache 和主存均被分割成大小相同的块(也称为行),信息以块为单位调入内容 Cache。Cache 中数据块(行)的大小一般为几到几百字节。

由于 Cache 容量有限,只能复制主存中小部分内容,因此 Cache 中有专门用于记录主存内容存入 Cache 时两者的对应关系的部件,称为标识 Cache,一般由相联存储器组成。在标识 Cache 中,内容 Cache 中的每个块都有一个对应的标识(标记),表明存入 Cache 当前块中主存内容的特征。另外,标识 Cache 中还有一位有效位,用于判定当前 Cache 块中是否包含有效信息。这是因为处理器刚启动时,Cache 中没有有用的数据,所有有效位都为无效的。即使在执行了一些指令后,Cache 中某些块依然是空的。

当 CPU 存取数据或指令时,按数据或指令的内存地址去访问 Cache,与标识 Cache 中的标记相比较,若相等且有效位为有效,说明 Cache 中找到了数据或指令(称为 Cache 命中),则 CPU 无须等待,Cache 就可以将信息传送给它;若标记不相等,说明数据或指令不在 Cache 中(称为未命中),此时存储器控制电路从内存中取出数据或指令传送给 CPU,同时复制一份该信息所在的数据块到 Cache 中。若此时 Cache 已满,则 Cache 中的替换策略实现机构按照某种替换算法调出某一 Cache 块,然后从内存中装入所需的块。之所以这样做,是为了防止

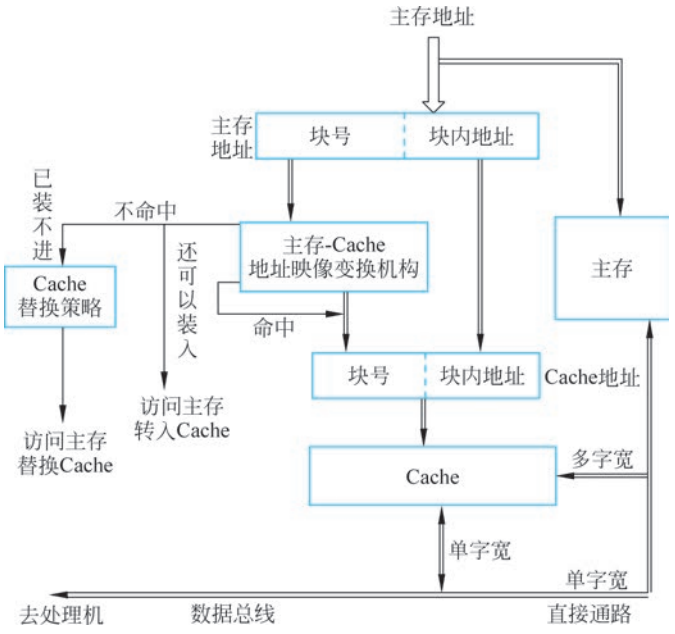

图 3-24 Cache 的总体结构

CPU 以后再访问同一信息时又会出现不命中的情况,以便尽量降低 CPU 访问速度相对较慢的内存的概率。换言之,CPU 访问 Cache 的命中率越高,系统性能就越好。目前,在绝大多数有 Cache 的系统中,Cache 的命中率一般能做到高于 85%。

Cache 的命中率取决于 Cache 的大小、Cache 的组织结构和程序的特性 3 个因素。容量相对较大的 Cache,命中率会相应地提高,但容量太大成本就会变得不合理。遵循局部性原理的程序在运行时,Cache 命中率也会很高。另外,Cache 的组织结构的好坏对命中率也会产生较大的影响。就 Cache 的组织结构而言,有 3 种类型的 Cache:直接映像方式、全相联映像方式和组相联映像方式。

由于 CPU 仍以主存地址访问 Cache,因此先用主存地址中的一部分与标识 Cache 的标记比对,判定是否命中。用访存地址中的哪部分与标记比对,以及与标识 Cache 中所有标记比对,还是只比对某个标记,或是比较某几个标记,这些都取决于 Cache 的组织结构,即主存信息按什么规则装入 Cache。通常将主存与 Cache 的存储空间划分为若干大小相同的块。例如,某机 Cache 容量为 16KB,块大小为 64B,则 Cache 可划分为 256 块;主存容量为 1MB,可划分为 16 384 块。下面以此为例介绍 3 种 Cache 的组织结构,也称为 Cache 映像方式。

1. 直接映像方式

所谓直接映像,是指任何一个主存块只能复制到某一固定的 Cache 块中。它实际是将主存以 Cache 的大小划分为若干区,每一区的第 0 块只能复制到 Cache 的第 0 块,每一区的第 1 块只能复制到 Cache 的第 1 块,……。如图 3-25 所示,在前述实例中,把主存按照 Cache 的大小分为 64 个区,每个区 256 块。当 CPU 访存时,给出 20 位主存地址,其中高 14 位给出主存块号,低 6 位给出块内的字节地址。

为了实现与 Cache 间的地址映像与变换,主存高 14 位地址又分为两部分:高 6 位给出主存区号,选择 64 区中的某一区;低 8 位为区内块号,实际就是 Cache 块号,选择区内 256 块中

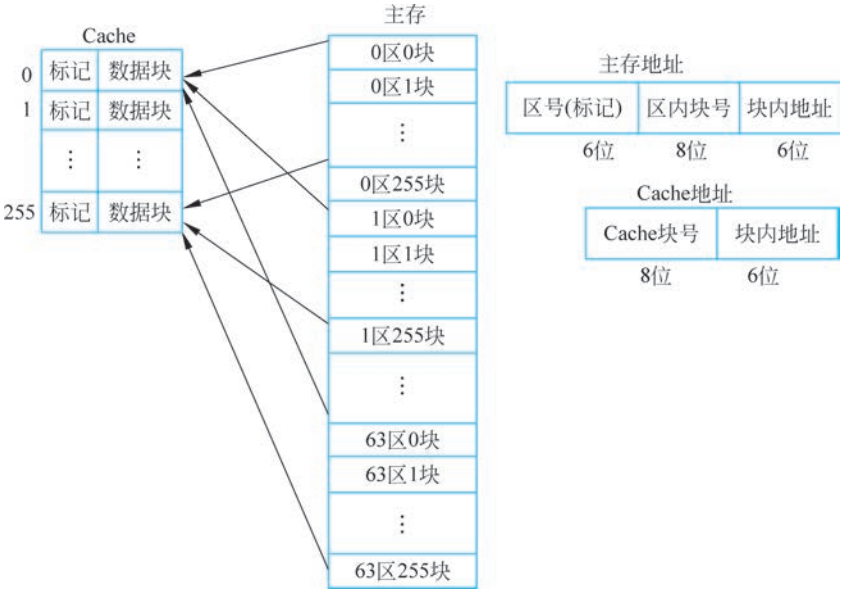

图 3-25　直接映像的 Cache 组织

的某一块。由于主存块在 Cache 中的位置固定，一个主存块只能对应一个 Cache 块，故标识 Cache 中只需存储每一块所对应的主存区号。如图 3-26 所示，访存时，以主存地址中的区内块号为索引定位到标识 Cache 的相应位置，再将主存地址中的区号与标识 Cache 中的相应块的标记比较。如果相等且有效位为有效，表示 Cache 命中，则所需的数据由 Cache 相应单元读出或写入。若不等，表示所需块未装入 Cache，此时需访问主存获取信息，同时将所需内容对应的块从主存复制到 Cache 中并修改对应的标识。

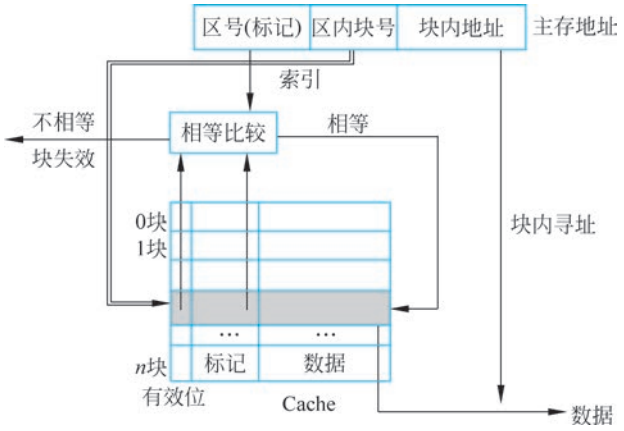

图 3-26　直接映像中 Cache 的工作方式

在直接映像方式下，主存中存储单元的数据只可调入 Cache 中的一个固定位置，如果主存中另一个存储单元的数据也要调入该位置，则将发生冲突。

直接映像方式的硬件实现简单，地址变换速度快。由于主存块在 Cache 中的位置固定，一个主存块只能对应一个 Cache 块，因此没有替换策略问题，但块的冲突率高，Cache 利用率也降低了。若程序连续访问两个相互冲突的块，将会使命中率急剧下降。

2. 全相联映像方式

在全相联映像方式的Cache中,任意主存单元的内容可以存放到Cache的任意单元中,两者之间的对应关系不存在任何限制,如图3-27所示。

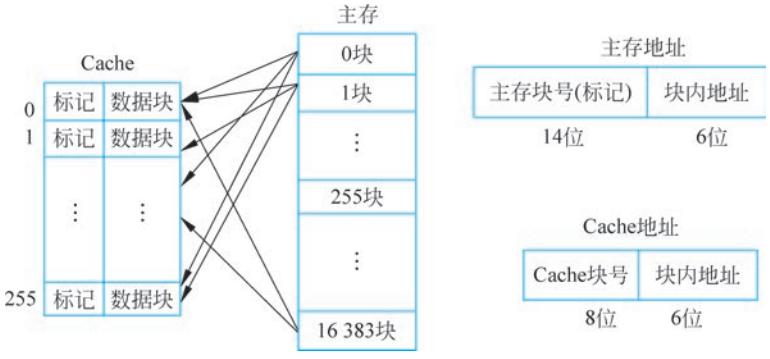

图3-27 全相联映像的Cache组织

在全相联映像方式中,CPU送到Cache的访存地址被分为两部分,如上例中高14位为主存块号,低6位为块内地址。每块Cache的标记也为14位,用于指示装入Cache对应位置的主存块号。当CPU访存时,将主存块号与Cache标记全相联比较,若有相符者,则表示被访主存块已装入Cache,相应块的内容被读出或写入。若没有相符者,则表示被访主存块未复制到Cache中,此时若Cache中有空块,则从主存调入所需块并建立标记;若Cache中无空块,则需淘汰某一Cache块,再调入新块,并修改Cache标记。

全相联方式Cache空间利用率高,只有在Cache中的块全部装满后才会出现块冲突,所以块冲突概率小。缺点是需相联比较,因而硬件逻辑复杂,成本高。

3. 组相联映像方式

全相联映像方式灵活性和命中率高,但地址映像电路中的比较器复杂,而直接映像方式正好与之相反。组相联映像是这两种方式的一种折中,它将Cache进行分组,每组中的块数固定,同时将主存按照Cache的块尺寸分割成若干块。主存中的任何一块只能存放到Cache中的某一固定组中,但存放在该组的哪一块是灵活的。

如果Cache组的大小为1,则变成了直接映像。如果组的大小为整个Cache的尺寸,则又变成了全相联映像。若组相联映像中每组的块数为k,则又称为k路组相联。

仍使用前面的例子。如图3-28所示,假设Cache中每组大小为4块,则Cache共有64组。12位Cache地址分为3部分:6位组号、2位组内块号和6位块内地址。1MB主存共有16 384块,20位主存地址的高14位为块号,低6位为块内地址。设主存中某一块的块号为s,则它所在的Cache组号$k=s\text{MOD}64$,即主存地址中14位块号的低6位就是s块所在的Cache组号,而14位块号中的高8位作为标记存储在标识Cache中,用于访问Cache时的相联比较。这样主存的s块只能存放在Cache的k组中,但可放于k组4块中的任意位置。由此可以看出,组的映像是直接映像,而组内是全相联映像。

如图3-29所示,按照图3-28所示的结构,Cache按照四路组相联组织,每行是一组,每组由4个块构成。当CPU以20位主存地址访问Cache时,其地址被分为3部分:8位标记、6位Cache组号和6位块内地址。以主存组号为索引查找标识Cache,在标识Cache中将对应

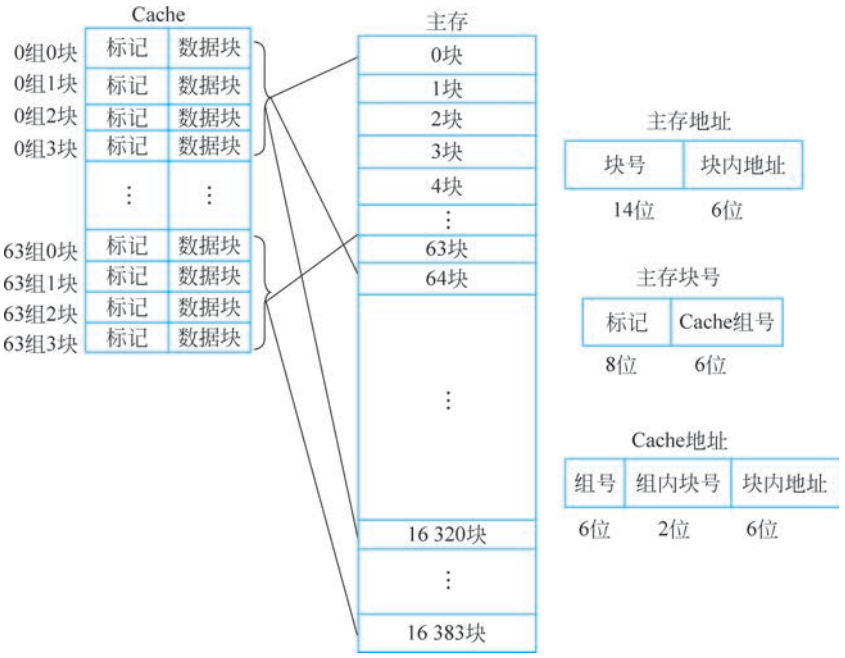

图 3-28　组相联映像的 Cache 组织

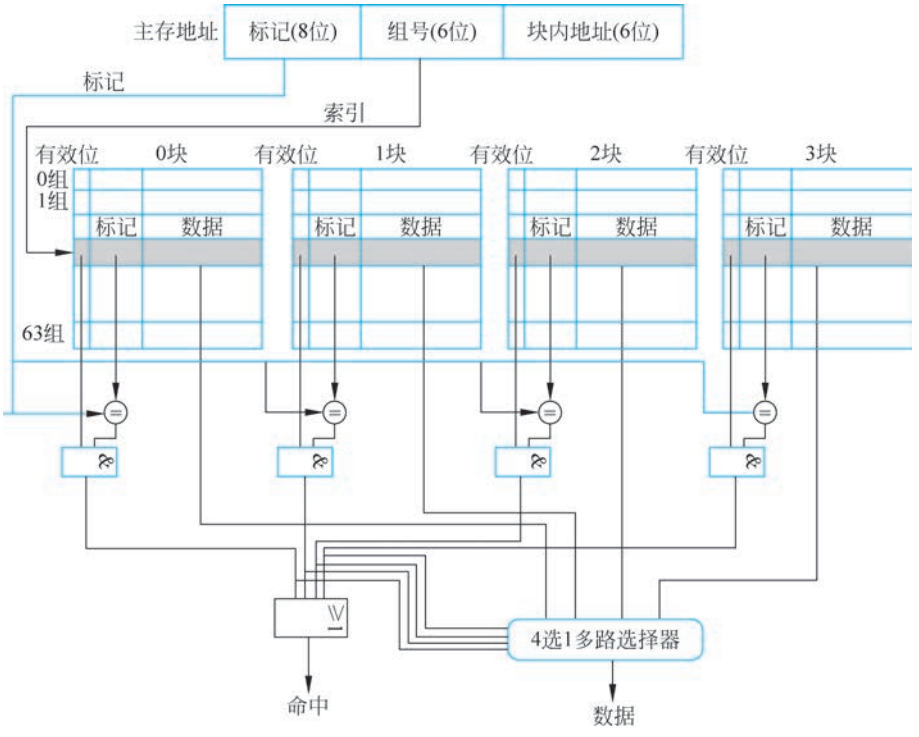

图 3-29　四路组相连 Cache 的工作方式

组的 4 个标记与主存地址中的标记进行相等比较，如果有相等的，且有效位为有效，表示 Cache 命中，由四选一多路选择器选择出匹配的数据块，再根据主存地址中的块内地址找到块内要访问的数据读出送给 CPU。如果没有相等的标记，表示不命中，对主存进行访问并将主

存中的块调入Cache中,同时将主存地址中的标记写入标识Cache中,以改变映像关系。在新的数据块调入时,可能还需确定将组内的哪一个数据块替换出去。

例3-5 假设某Cache有4096块,块大小为4字,字长为32位,主存地址为32位,分别计算在直接映像、两路组相连映像、四路组相联映像和全相联映像中,Cache的组数以及总的标记位数。

解: 由于字长为32位,即4字节,块大小为4字,因此块内地址为4位,Cache有$4096=2^{12}$块,所以Cache地址共16位。

① 直接映像时,主存地址结构如下:

区号(标记16位)	区内块号(12位)	块内地址(4位)

直接映像时,一组只有一个块,所以Cache共4096组。标记的总位数$=4096\times16=64$Kb。

② 两路组相连映像时,主存地址结构如下:

标记(17位)	组号(11位)	块内地址(4位)

Cache地址结构如下:

组号(11位)	组内块号(1位)	块内地址(4位)

所以Cache组数为$2^{11}=2048$组。标记的总位数$=4096\times17=68$Kb。

③ 四路组相联映像时,主存地址结构如下:

标记(18位)	组号(10位)	块内地址(4位)

Cache地址结构如下:

组号(10位)	组内块号(2位)	块内地址(4位)

所以Cache组数为$2^{10}=1024$组。标记的总位数$=4096\times18=72$Kb。

④ 全相联映像时,主存地址结构如下:

块号(标记 28位)	块内地址(4位)

Cache地址结构如下:

块号(12位)	块内地址(4位)

全相联映像时Cache只有一组。标记的总位数$=4096\times28=112$Kb。

3.5.3 Cache的替换算法与写策略

1. Cache的替换算法

CPU访问Cache在未命中的情况下,需要访问主存找到所需的信息,同时需要把该信息所在的块装入Cache。若此时在全相联映像结构中,所有Cache块都装满了,或在组相联映像结构中,主存地址所对应的组也满了,这时就必须按照某种策略替换Cache中的一个块中的数据或指令,以便腾出位置存放新的数据或指令,这个过程称为Cache刷新。

根据计算机的设计目的和使用意向,可以采用随机的、顺序的、先进先出(First In First

Out,FIFO)和最近最久未使用(Least Recently Used,LRU)算法,来决定被替换的数据。LRU 算法是将最近一段时间内 CPU 最久未使用的数据替换掉。考虑到实现的方便性,一般采用最近最少使用(Least Frequently Used,LFU)算法来决定被替换的数据,这种方法是让 Cache 控制器记录 Cache 中每块数据最近使用的次数,当要为新数据腾出空间时,最近使用次数最少的数据块被替换。

在替换数据时,若内存中已有被替换数据块的副本,则无须将其再写回内存,直接丢弃即可,否则将被替换的 Cache 块写回主存,以保证数据的一致性。

Cache 未命中时,CPU 必须阻塞,等待要访问信息所在的数据块由主存复制到 Cache 中,为了减少阻塞延迟,可采用提前重启(Early Restart)或关键字优先(Critical Word First)技术。提前重启就是当访问主存返回所需字时,处理器马上继续执行,不需要等待整个块都装入 Cache 后再执行。这种技术应用于指令 Cache 可以保证存储器系统每个时钟周期都能传送一个字。这是因为大多数指令访问都是连续的。但应用于数据 Cache 效率要低一些,因为 CPU 对数据的访问可能是不连续的。在当前访问的块从主存传送到 Cache 前,CPU 很可能访问的是另一块的数据,如果此时数据传输正在进行,处理器必然阻塞。关键字优先技术是重新组织存储器,使得被请求的字先从主存传送到 Cache,再传送该块剩余的部分,从所请求字的下一个地址开始传送,再回到块的开始。这种技术也被称为请求字优先(Requested Word First),它比提前重启要快一些,但如果 CPU 连续访问的字不在同一块中,是离散分布的,同样可能会引起处理器阻塞。

2. Cache 的写策略

在具有 Cache 的系统中,由于对应同一地址的数据有两份副本,一份在主存中,另一份在 Cache 中,因此必须确保在操作过程中不丢失任何数据,使 CPU 使用的任何数据都是最新的。这就必须采用一个完美的 Cache 写策略,以确保写入 Cache 中最新的数据也写入了主存。目前有两种 Cache 写策略:写直达法(Write-through)和写回法(Write-back)。

1) 写直达法

采用写直达法,数据被同时写入主存和 Cache。因此,任何时刻内存中都有 Cache 中有效数据的副本。这种策略保证了内存中的数据总是最新的。如果 Cache 中的内容被覆盖,可以从内存中访问到最新数据。但这种做法增加了 CPU 占用系统总线的时间。

2) 写回法

采用写回法,CPU 将最新的数据只写入 Cache 中,但不写入内存。仅当 Cache 要替换数据时,才由 Cache 控制器将 Cache 中被替换的那个数据写入内存。采用此策略的 Cache 中增加了一位状态位,称为修改位(Dirty Bit)。当 Cache 要替换其中的某个块(行)中的数据时,首先查看与该块(行)对应的修改位:若修改位为 0,则表明 Cache 中的数据未被修改过,其内容与内存中对应块的内容是一致的,可以直接丢弃;若修改位为 1,则表明 Cache 中的数据是新数据,只有 Cache 中有,而内存中没有,在替换之前需要将其写入内存。当把 Cache 中的数据复制到内存中后,修改位将被清 0。采用写回法实现了在必要时才更新内存的内容,减少了 CPU 占用系统总线的时间。写回法不像写直达法那样,当 CPU 每次向 Cache 中写入数据时,都要同时向内存中写入数据,而无端占用系统总线。

在多处理器或有 DMA 控制器的系统中,不止一个处理器可以访问内存,此时必须确保 Cache 中总是有最新的数据。当一个处理器或 DMA 控制器改变了内存中的某些单元的数据

时,必须通知其他处理器内存的数据已经被修改。如果在其他处理器所使用的 Cache 中存放的是被修改的内存单元修改前的内容,要将 Cache 中的旧数据标记为“旧的”。这样,若处理器要使用旧数据,Cache 会告知它数据已被更改,需要到内存中重取。在多处理器共享内存中的同一组数据时,必须采取策略确保所有处理器用到的都是最新的数据。

3.6 虚拟存储器

根据程序的局部性原理,应用程序在运行之前没有必要全部装入内存,仅将那些当前需要运行的部分代码先装入内存运行,其余部分暂留在磁盘上即可。如果程序所要访问的代码或数据尚未调入内存,此时系统产生中断,由操作系统自动将所缺部分从磁盘调入内存,以使程序能继续执行下去。如果此时内存已满,无法再装入新的代码或数据,则操作系统利用置换功能将内存中暂时不用的内容调至磁盘上,以腾出足够的内存空间装入要访问的内容,使程序继续执行下去。这种存储器管理技术称为虚拟存储器。

所谓虚拟存储器,是指具有请求调入功能和置换功能,能从逻辑上对内存容量加以扩充的一种存储器系统。其逻辑容量由内存容量和外存容量之和决定,其运行速度接近内存速度,而每位的成本又接近外存。利用虚存技术,程序不再受有限的物理内存空间的限制,用户可以在一个巨大的虚拟内存空间上写程序。此时,CPU 执行指令所生成的地址称为逻辑地址或虚地址,由程序所生成的所有逻辑地址的集合称为逻辑地址空间或虚地址空间。而内存单元所看到的地址,即加载到内存地址寄存器中的地址称为物理地址或实地址。程序执行时,从虚地址到物理地址的映射是由内存管理部件 MMU 完成的。

虚拟存储器的管理方式有 3 种:页式、段式和段页式。

3.6.1 页式虚拟存储器

1. 基本原理

在页式虚拟存储系统中,把程序的逻辑地址空间分为若干大小相等的块,称为逻辑页,编号为 0,1,2,…。相应地,把物理地址空间也划分为与逻辑页相同大小的若干存储块,称为物理块或页框,编号为 0,1,2,…。设逻辑地址空间大小为 2^n,页面大小为 2^m,则页式虚拟存储系统中的逻辑地址结构如下:

逻辑页号 p	页内地址 d
$n-m$ 位	m 位

操作系统将程序的部分逻辑页离散地存储在内存中不同的物理页框中,并为每个程序建立一张页表。页表中的每个表项(行)分别记录了相应页在内存中对应的物理块号,该页的存在状态(是否在内存中),以及对应的外存地址等控制信息。程序执行时,通过查找页表即可找到每个逻辑页在内存中的物理块号,实现由逻辑地址到物理地址的映射。

如图 3-30 所示,当程序执行时产生访存的逻辑地址,页式虚存地址变换机构将逻辑地址分为逻辑页号和页内地址两部分,并以逻辑页号为索引去检索页表(检索操作由硬件自动执行)。地址变换机构根据页表基地址与逻辑页号,找到该逻辑页在页表中的对应表项,得到该页对应的物理块号,装入物理地址寄存器,同时将逻辑地址寄存器中的页内地址送入物理地址寄存器,就得到了该逻辑地址对应的物理地址,完成了地址映射。

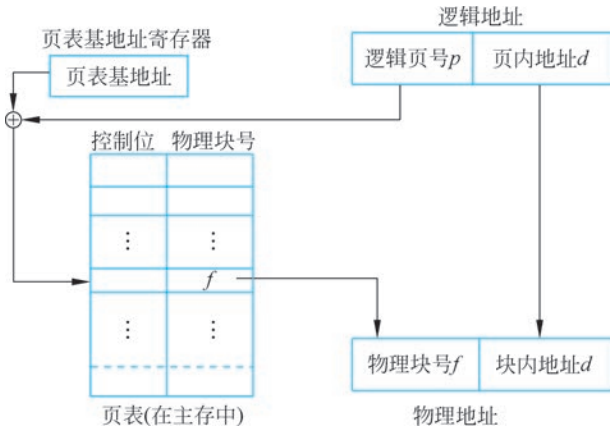

图 3-30　页式虚拟存储器的地址映射

在页式虚拟存储系统中，当地址变换机构根据逻辑页号查找页表时，若该逻辑页不在物理内存中（页表对应表项的存在位为“0”），此时产生缺页中断，请求操作系统将所缺的页调入内存。缺页中断处理程序根据该逻辑页对应页表项指明的外存地址在硬盘上找到所缺页面，若物理内存中有空闲物理块，则直接装入所缺页。否则，缺页中断处理程序转去执行页面置换功能，根据页面置换算法选择一页换出内存，再将所缺页换入内存。常用的页面置换算法有 FIFO、LRU、Clock、LFU 算法等。

2. 快表

一般页表存放在内存，使得 CPU 执行指令时每次访存操作至少要访问两次主存：第一次是访问内存中的页表，从中找到指定页的物理块号，第二次访存才是获得所需的数据或指令。这使计算机的处理速度降低近 1/2。

通常的解决办法是：在地址变换机构增设一组由关联存储器构成的小容量特殊高速缓冲寄存器（通常只存放 16～512 个页表项），又称为联想寄存器（Associative Memory），或称为快表，用以存放当前访问的那些页表项。而内存中的页表则称为慢表。

在具有快表的页式虚拟存储系统中，逻辑地址映射为物理地址的过程如图 3-31 所示。在 CPU 给出访存的逻辑地址后，由地址变换机构自动地将逻辑页号 p 送入快表，并与快表中的所有页号同时并行比较，若其中有与此相匹配的页号，便表示所要访问的页表项在快表中。于是，可直接从快表中读出该逻辑页所对应的物理块号，并送到物理地址寄存器中。若在快表中未找到对应的页表项，则再访问内存中的页表（慢表），找到后，把从页表项中读出的物理块号送入物理地址寄存器，同时还要将此页表项存入快表的一个单元中。若快表此时已满，则操作系统必须按照一定的置换算法从快表中换出一个页表项。

3.6.2　段式虚拟存储器

段式虚拟存储系统把程序按照其逻辑结构划分为若干逻辑段，如主程序段、子程序段、数据段等，逻辑段号为 0，1，2，…。每个段的大小不固定，由各段的逻辑信息长度决定。逻辑段内的地址从 0 开始编址，并采用一段连续的逻辑地址空间。段式虚拟存储系统中的逻辑地址结构如下：

逻辑段号 s	段内地址 d

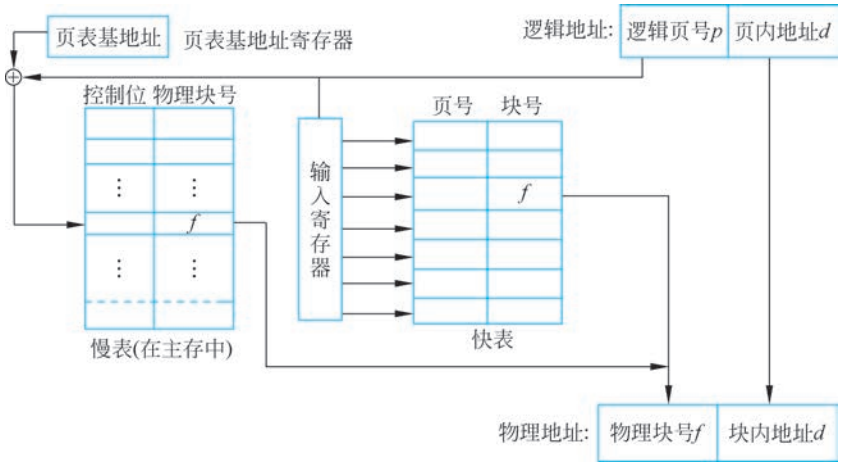

图 3-31 具有快表的页式虚拟存储器的地址映射

操作系统在装入程序时,将程序的若干逻辑段离散地存储在内存不同的区块中,每个逻辑段在物理内存占有一个连续的区块。为了能在内存中找到每个逻辑段,并实现二维逻辑地址到一维物理地址的映射,系统为每个程序建立了一张段表。程序的每个逻辑段都有一个对应的表项,记录该段的长度、在物理内存的起始地址、该段的存在状态(是否在内存中)、对应的外存地址等控制信息。

如图 3-32 所示,若程序执行时产生了访存的二维逻辑地址,段式虚存地址变换机构以逻辑段号为索引检索段表,得到该逻辑段在内存的起始物理地址,将起始物理地址与逻辑地址中的段内地址相加,即可得到一维物理地址,完成地址映射。

与页式虚拟存储系统相似,当地址变换机构查找段表时,若该逻辑段不在物理内存中(段表中对应表项的存在位为 0),此时产生缺段中断,请求操作系统将所缺的段调入内存。缺段中断处理程序根据该逻辑段对应段表项中指明的外存地址在硬盘上找到所缺段,若物理内存中有足够大的空闲区块,则直接装入所缺段。否则,缺段中断处理程序按照一定的置换算法换出内存中的一个或几个段(空出足够大的内存区块),再装入所缺段。

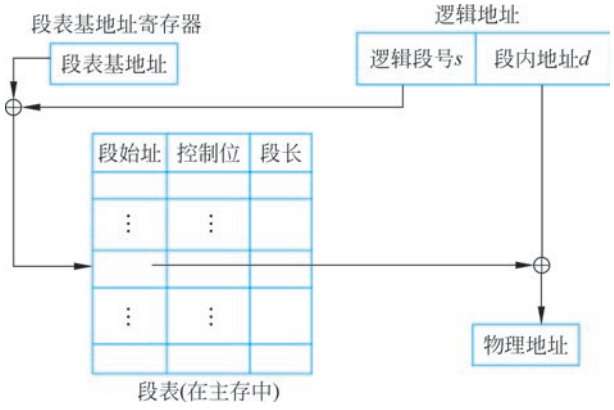

图 3-32 段式虚拟存储器的地址映射

3.6.3 段页式虚拟存储器

段页式虚拟存储器将程序按照其逻辑结构分为若干段,每段再划分为若干大小相等的逻

辑页；物理内存被划分为若干同样大小的页框。操作系统以页为单位为每个逻辑段分配内存，这样不仅段与段之间不连续，一个逻辑段内的各逻辑页也离散地分布在物理内存中。

图 3-33 给出了段页式虚拟存储器的地址映射关系。为了实现地址映射，系统为每个程序建立一张段表，为每个逻辑段建立一张页表。段表记录程序各个逻辑段的页表在内存的起始地址、段长、存在状态等控制信息。每个逻辑段对应的页表记录着本段各页对应的物理块号。

CPU 执行指令时产生的访存逻辑地址分为 3 部分：段号、段内页号和页内地址。进行地址映射时，首先利用段号 s 和段表起始地址的和求出该段所对应的段表项在段表中的位置，从中得到该段的页表起始地址，并利用逻辑地址中的段内页号 p 来获得对应页的页表项位置，从中读出该页所在的物理块号 b，再利用块号 b 和页内地址来构成物理地址。

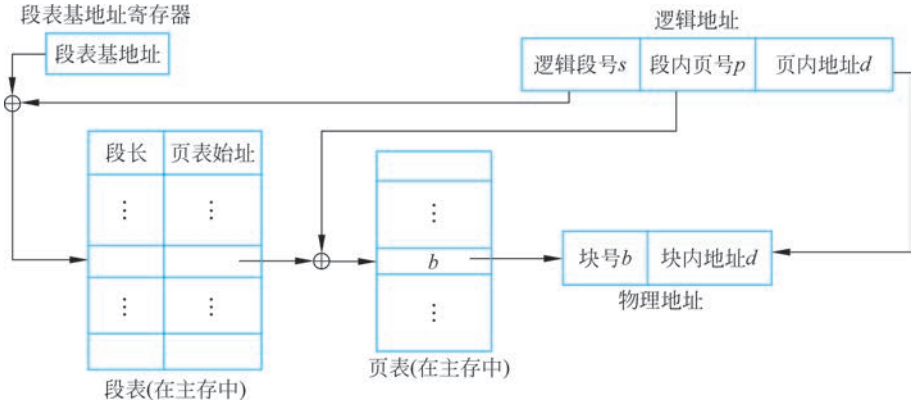

图 3-33　段页式虚拟存储器的地址映射

在段页式虚拟存储系统中，为了从主存中取出一条指令或数据，至少要访存三次。第一次访问的是内存中的段表，从中取得该逻辑段对应的页表起始地址；第二次访问的是内存中的页表，从中取出要访问的页所在的物理块号，并将该块号与页内地址一起形成指令或数据的物理地址；第三次访存才是真正取出指令或数据。为了提高执行速度，可在地址变换机构中增加类似页式虚拟存储器的高速缓冲寄存器，即段页式快表。段页式快表将段表和页表合成一张表，表项如下：

段号	逻辑页号	物理块号	其他控制位

地址变换时，先查找快表，仅当快表中没有找到时，才去查找慢表，提高了访问效率。

习题

3.1　简答题。

（1）静态 MOS 存储器与动态 MOS 存储器存储信息的原理有何不同？为什么动态 MOS 存储器需要刷新？一般有哪几种刷新方式？

（2）什么是程序的局部性原理？

（3）并行存储和交叉存储的特点是什么？

（4）Cache 的写直达法和写回法指什么？二者各有何优缺点？

（5）虚拟存储解决什么问题？遇到缺页情况如何处理？

3.2　某一 64K×1 位的动态 RAM 芯片，采用地址复用技术，则除了电源和地引脚外，该

芯片还应有哪些引脚？各为多少位？

3.3 假设某存储器地址长为22位，存储器字长为16位，试问：

(1) 该存储器能存储多少字节信息？

(2) 若用64K×4位的DRAM芯片组织该存储器，则需多少片芯片？

(3) 在该存储器的22位地址中，多少位用于选片寻址？多少位用于片内寻址？

3.4 某8位计算机采用单总线结构，地址总线17根($A_{16\sim0}$，A_{16}为高位)，数据总线8根双向($D_{7\sim0}$)，控制信号R/$\overline{W}$(高电平为读，低电平为写)。已知该机的I/O设备与主存统一编址，若地址空间从0连续编址，其地址空间分配如下：最低16K为系统程序区，由ROM芯片组成；紧接着48K为备用区，暂不连接芯片；接着60K为用户程序和数据空间，用静态RAM芯片组成；最后4K为I/O设备区。现有芯片如图3-34所示。

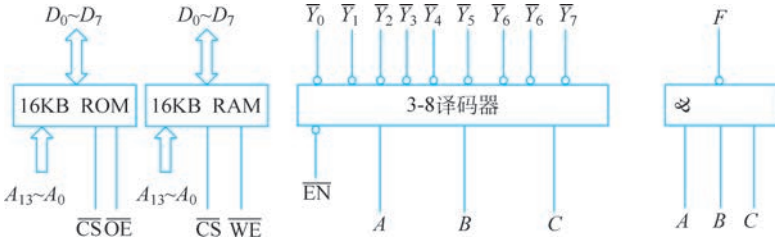

图3-34 现有芯片

ROM：16K×8位。其中$\overline{CS}$为片选信号，低电平有效；$\overline{OE}$为读出控制，低电平读出有效。

静态RAM：16K×8位。其中$\overline{CS}$为片选信号，低电平有效；$\overline{WE}$为写控制信号，低电平写，高电平读。

译码器：3-8译码器。输出低电平有效。$\overline{EN}$为使能信号，低电平时译码器功能有效。

与非门：扇入系数不限。

试画出主存芯片连接的逻辑图并写出各芯片地址分配表(设存储器从0开始连续编址)。

3.5 某8位计算机采用单总线结构，地址总线17根($A_{16\sim0}$，A_{16}为高位)，数据总线8根双向($D_7\sim D_0$)，控制信号R/$\overline{W}$(高电平为读，低电平为写)。已知该机存储器地址空间从0连续编址，其地址空间分配如下：最低8K为系统程序区，由ROM芯片组成；紧接着40K为备用区，暂不连接芯片；而后78K为用户程序和数据空间，用静态RAM芯片组成；最后2K用于I/O设备(与主存统一编址)。现有芯片如图3-35所示。

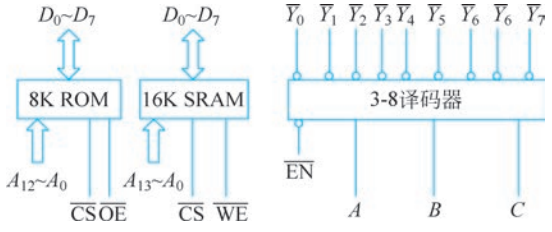

图3-35 现有芯片

SRAM：16K×8位。其中$\overline{CS}$为片选信号，低电平有效；$\overline{WE}$为写控制信号，低电平写，高电平读。

ROM：8K×8位。其中$\overline{CS}$为片选信号，低电平有效；$\overline{OE}$为读出控制，低电平读出有效。

译码器：3-8译码器，输出低电平有效。$\overline{EN}$为使能信号，低电平时译码器功能有效。

其他与、或等逻辑门电路自选。

(1) 请问该主存需多少 SRAM 芯片?

(2) 试画出主存芯片与 CPU 的连接逻辑图。

(3) 写出各芯片地址分配表。

3.6　已知某 8 位机的主存采用 4K×4 位的 SRAM 芯片构成该机所允许的最大主存空间,并选用模块板结构形式,该机地址总线为 18 位,问:

(1) 若每个模块板为 32K×8 位,共需几个模块板?

(2) 每个模块板内共有多少块 4K×4 位的 RAM 芯片? 请画出一个模块板内各芯片连接的逻辑框图。

(3) 该主存共需要多少 4K×4 位的 RAM 芯片? CPU 如何选择各个模块板?

3.7　64K×1 位 DRAM 芯片通常排成 256×256 阵列。若存储器的读/写周期为 0.5μs,则对集中式刷新而言,其"死区"时间是多少? 如果是一个 256K×1 位的 DRAM 芯片,希望能与上述 64K×1 位 DRAM 芯片有相同的刷新延时,则它的存储阵列应如何安排?

3.8　请用 2K×8b 的 SRAM 设计一个 8K×32b 的存储器,写出各芯片片选信号的控制逻辑表达式,并画出存储器与 *CPU* 的连接原理图。要求:

(1) 存储器可分别被控制访问 8、16、32 位数据。控制位数的信号 B_1B_0 由 CPU 提供:

当 $B_1B_0=00$ 时访问 32 位数据。

当 $B_1B_0=01$ 时访问 16 位数据。

当 $B_1B_0=10$ 时访问 8 位数据。

(2) 存储芯片地址按交叉方式编址,即一列为奇地址,一列为偶地址。

(3) 满足整数边界地址的安排。

3.9　某计算机主存容量为 16MB,采用 8 位数据总线; 数据 Cache 容量为 32KB,主存与数据 Cache 均按 64B 的大小分块。请回答下列问题:

(1) 设标识 Cache 中每个单元只包含标识位和 1 位有效位。若 Cache 采用直接映像方式,则该计算机中标识 Cache 的容量是多少?

(2) 若 Cache 采用直接映像方式,则主存地址为 123456H 的存储单元有可能装入 Cache 中哪个地址对应的单元中?

(3) 若 Cache 采用组相联映像方式,每组块数为 4 块。写出主存与 Cache 地址的结构格式并标出各个字段的位数。

(4) 若 Cache 采用组相联映像方式,每组块数为 4 块,则主存地址为 123456H 的存储单元有可能装入 Cache 中哪几个地址对应的单元中?

3.10　选择题。

(1) 需要定期刷新的存储芯片是_____。

A. EPROM　　B. DRAM　　C. SRAM　　D. EEPROM

(2) _____存储芯片是易失性的。

A. SRAM　　B. UV-EPROM　　C. NV-RAM　　D. EEPROM

(3) 下面叙述不正确的是_____。

A. 半导体随机存储器可随时存取信息,掉电后信息丢失

B. 在访问随机存储器时,访问时间与单元的物理位置无关

C. 内存中存储的信息均是不可改变的

D. 随机存储器和只读存储器可以统一编址

(4) 动态 RAM 与静态 RAM 相比,其优点是_____。

A. 动态 RAM 的存储速度快

B. 动态 RAM 不易丢失数据

C. 在工艺上,比静态 RAM 的存储密度高

D. 控制比静态 RAM 简单

(5) 某 512×8 位 RAM 芯片采用一位读/写线控制读写,该芯片的引脚至少有_____。

A. 17 条　　B. 19 条　　C. 21 条　　D. 522 条

(6) 在下列存储器中,允许随机访问的存储器是_____。

A. 半导体存储器　　B. 磁带　　C. 磁盘　　D. 光盘

(7) 在下列存储器中,不能脱机保存信息的是_____。

A. 磁盘　　B. 磁带　　C. RAM　　D. 光盘

(8) 以下关于主存的叙述中正确的是_____。

A. CPU 可直接访问主存,但不能直接访问辅存

B. CPU 可直接访问主存,也能直接访问辅存

C. CPU 不能直接访问主存,也不能直接访问辅存

D. CPU 不能直接访问主存,但能直接访问辅存

3.11 判断题。

(1) 数据引脚和地址引脚越多,芯片的容量越大。(　　)

(2) 存储芯片的价格取决于芯片的容量和速度。(　　)

(3) SRAM 每个单元的规模大于 DRAM 的规模。(　　)

(4) 当 CPU 要访问数据时,它先访问虚存,之后再访问主存。(　　)

(5) 能够在 CPU 和主存之间增加一级 Cache,是基于程序局部性原理。(　　)

(6) 主存与磁盘均用于存放程序和数据,一般情况下,CPU 从主存取得指令和数据,如果在主存中访问不到,CPU 才到磁盘中取得指令和数据。(　　)

(7) 半导体存储器是一种易失性存储器,电源掉电后所存信息均将丢失。(　　)

(8) Cache 存储器保存 RAM 存储器的信息副本,所以占部分 RAM 地址空间。(　　)

(9) EPROM 只能改写一次,故不能作为随机存储器。(　　)

3.12 填空题。

(1) Cache 使用的是_____存储芯片。

(2) 主存由_____(DRAM、硬盘)构成,虚存由_____(DRAM、硬盘)构成。

(3) SRAM 依据_____存储信息,DRAM 依据_____存储信息。SRAM 与 DRAM 相比,速度高的是_____,集成度高的是_____。

(4) Cache 存储器的主要作用是解决_____。

(5) 存储器的取数时间是衡量主存_____的重要指标,它是从_____到_____的时间。

(6) 某存储器数据总线宽度为 32 位,存取周期为 250ns,则其带宽是_____。

(7) 存储器带宽是指_____,如果存储周期为 T_M,存储字长为 n 位,则存储器带宽为_____,常用的单位是_____和_____。为了加大存储器的带宽,可采用_____和_____。