第 3 章　半导体存储器及其接口

学习目标

存储器是计算机系统中存储信息的主要部件。本章从存储器的分类、性能以及存储器基本结构着手，重点讨论随机存储器（RAM）和只读存储器（ROM）的工作原理，介绍存储器的容量扩展以及与微处理器的连接方法，最后介绍微型计算机系统中的多体存储器、高速缓冲存储器（Cache）和虚拟存储器技术。

通过本章的学习，读者应了解半导体存储器的分类、主要性能指标，掌握随机存储器和只读存储器的功能特性，掌握存储器容量扩展以及与微处理器的连接方法，领会存储系统的层次结构以及多体存储器、高速缓冲存储器和虚拟存储器的技术特点。

3.1　存储器概述

存储器是计算机系统中的存储部件，用于存储计算机工作时所用的程序和数据。计算机工作时，CPU 自动、连续地从存储器中取出指令并执行指令所规定的操作，每执行完一条或几条指令，要把处理的结果保存到存储器中。因此，存储器是计算机的记忆部件，是计算机系统的重要组成部分。

随着计算机技术的发展及广泛应用，存储系统的读写速度也在不断地提高，存储容量不断增加。特别是近些年多媒体技术的发展以及计算机网络的应用，要求计算机存储和处理的信息量越来越大，并对存储器的存取速度不断提出更高的要求，因而在存储系统中应用了存储器层次结构、多体结构、高速缓冲存储器、虚拟存储器等技术，外存储器容量也可以无限地扩充成为海量存储器。

3.1.1　存储器的分类

存储器的分类方法很多，通常从以下几方面对存储器进行分类。

1. 按系统中的作用分类

根据存储器在微型计算机系统中的不同作用，可分为内存储器和外存储器。

（1）内存储器。又称主存储器，简称内存或主存，是计算机主机的一个重要组成部分，用来存放当前正在运行或将要使用的程序和数据。CPU 可以通过指令直接访问内存。

系统对内存的存取速度要求较高，为了与 CPU 的处理速度相匹配，内存一般使用快速存储器件构成。但是由于受到地址总线宽度的限制，内存空间远远小于外存容量，例如在 8086/8088 系统中，由于地址总线为 20 位，所以最大内存空间只能达到 1MB（2^{20}）。

尽管内存容量远不及外存，但是由于它具有访问速度快的特点，使得内存被用来存放计算机工作时必须的系统软件、参数以及当前要运行的应用软件和数据；而更多的系统软件和所

有的应用软件则存放于外存中，在需要时再由外存调入内存。

（2）外存储器。又称辅助存储器，简称外存或辅存，也用于存储程序和数据，但它存储的信息却是 CPU 当前操作暂时不用的。外存位于主机外部，CPU 不能直接对其进行读写操作，当 CPU 需要使用外存中的信息时，要先通过专门的部件将其从外存调入到内存，然后再对内存中调入的数据进行直接的读写操作。

系统对外存速度的要求相对于内存可以慢一些，但对外存的容量却要求很大。由于外存具有长久保存信息的特点，所以多被用来保存和备份数据。

目前微型计算机主要采用硬盘作为外存储器，某些高档微型计算机中还配有速度更快的固态硬盘。此外，还有光盘、移动硬盘、U 盘等大容量的可移动式存储器都是微型计算机系统中常见的辅助存储器，使计算机的外存成为“海量存储器”。

尽管计算机存储系统包括内存和外存，但内存是主机的一部分，而外存则属于外部设备。现代微型计算机的内存采用半导体存储器技术，主要以动态随机存储器构成内存，它具有断电则信息丢失的特点，考虑微型计算机启动的需要，在内存的高端地址还配有小容量的只读存储器，作为操作系统的引导程序存储空间。而外存作为内存的后备存储器，存储的数据可以方便地修改和永久地保存，不受断电影响。

另外，在现代高档微型计算机中，为了加快信息传递速度和提高计算机处理速度，广泛应用高速缓冲存储器（Cache）。Cache 是微型计算机系统中一种高速小容量的存储器，位于 CPU 和内存之间，用于存储内存的部分副本，向 CPU 快速提供指令和数据。内存一般采用动态随机存储器，而 Cache 则主要由高速的静态随机存储器组成。

2. 按存储信息的可保存性分类

根据存储器中信息的可保存性，可将存储器分为易失性存储器和非易失性存储器。

（1）易失性存储器。是指断电后信息消失的存储器，如半导体随机存储器 RAM，由于它具有读写速度快的特点，所以多被用于计算机内存。易失性存储器中的程序及数据可以在开机启动后由非易失性存储器调入，在断电前，要及时保存到非易失性存储器中。

（2）非易失性存储器。是指断电后仍然保持信息的存储器，如半导体只读存储器 ROM、磁盘、光盘、U 盘等。微型计算机系统软件中有一部分软件如引导程序、监控程序或者基本输入/输出服务程序 BIOS，是计算机系统正常工作所必须的，而且不能被随意修改的，所以它们必须常驻内存，于是应用了半导体存储器 ROM。希望长久保存的信息可存于硬盘、光盘、移动硬盘、U 盘等非易失性存储器中。

在微型计算机系统中，要求系统至少有一部分存储器必须是非易失性的。

3. 按存储介质分类

存储二进制信息的物理载体称为存储介质，根据所使用存储介质的不同，存储器可分为半导体存储器、磁介质存储器、光盘存储器。半导体存储器多用于微型计算机内存，而磁介质存储器和光盘存储器则用于外存。

（1）半导体存储器。用半导体器件做成的存储器称为半导体存储器。半导体存储器包含只读存储器（Read Only Memory，ROM）和随机存储器（Random Access Memory，RAM），而随机存储器又可分为静态随机存储器（Static RAM，SRAM）和动态随机存储器（Dynamic RAM，DRAM）。

（2）磁介质存储器。目前计算机系统使用的磁介质存储器主要是硬盘。硬盘采用涂有磁性介质的盘片存储二进制数据，具有存储容量大、价格低、断电后数据不丢失的特点。但硬盘在数据存取时需要硬盘驱动器的机械驱动，所以数据存取速度远低于半导体存储器。

在微型计算机中，硬盘用于存储操作系统、其他各种系统软件、应用程序和数据，供计算机运行过程中随时调用。

（3）光盘存储器。光盘利用介质材料的光学性质（对聚集光束的反射率、偏振方向等）的变化来表示所存储的信息，光盘具有体积小、价格低、便于携带、易于长久保存等特点。但光盘在数据存取时需要光盘驱动器的机械动作，数据存取速度慢于硬盘。

目前的微型计算机中，光盘主要用作辅助存储器，多用于软件发布和数据备份。

3.1.2 存储器的主要性能指标

存储器是计算机系统中的重要部件，它的性能直接影响整机的性能。微型计算机系统存储器的性能指标很多，如存储容量、存取速度、存储器可靠性、功耗、价格、性能价格比等，但就功能和接口技术而言，最重要的性能指标是存储容量和存取速度。

1. 存储容量

存储容量是指存储器可以容纳的二进制信息总量。容量越大，意味着所能存储的二进制位（bit）越多。1 位二进制数是最小存储单位，8 位二进制数为 1 个字节（Byte，简写为 B），由于微型计算机都是按字节编址的，因此 B 是存储容量的基本单位，存储容量单位还有 KB、MB、GB、TB、PB，它们之间的关系为：

1 KB ＝1024 B ＝2^{10} B＝1 024 B

1 MB＝1024 KB ＝2^{20} B＝1 048 576 B

1 GB ＝1024 MB＝2^{30} B＝1 073 741 824 B

1 TB ＝1024 GB ＝2^{40} B＝1 099 511 627 776 B

1 PB ＝1024 TB ＝2^{50} B＝1 125 899 906 842 624 B

存储容量越大，计算机系统的功能就越强，所以人们总是希望尽量提高存储容量。但是存储容量的提高受到 CPU 的寻址范围、所选用的存储芯片的速度、成本等诸多因素的限制，故不能设计得很大。

2. 存取速度

存储器的存取速度通常由存取时间来衡量，存取时间又称读写时间，是指从 CPU 发出有效的存储器地址从而启动一次存储器读/写操作，到读出或写入数据完毕所经历的时间。存取时间越短，则存取速度越快。内存存取速度的度量单位采用 ns 表示，如 5ns。

目前市场上也常用“内存主频”表示内存的速度，它代表着内存所能达到的最高工作频率，以 MHz 为单位，如 2400MHz。内存主频越高在一定程度上代表内存所能达到的速度越快。

3.1.3 内存储器的基本结构

内存储器由存储体、地址寄存器、地址译码器、读写驱动器、数据寄存器以及时序控制电路等部件组成，如图 3-1 所示。

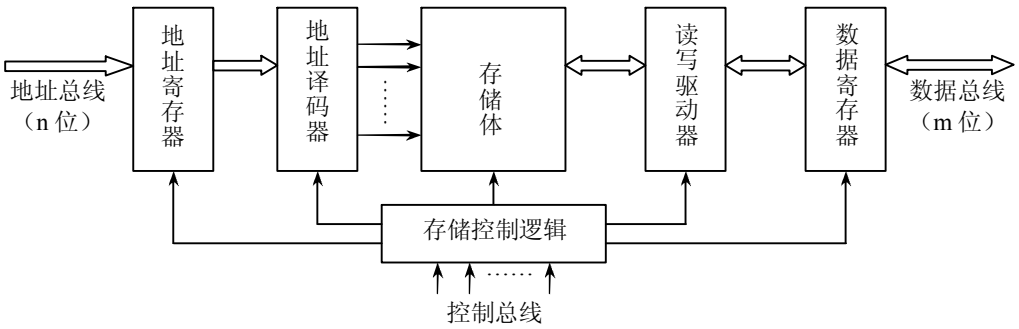


图 3-1　内存储器的基本组成

存储体是具体存储信息的场所，是存储单元的集合。假设存储器有 m 位数据总线、n 位地址总线和若干控制总线。地址总线给出所需访问的存储单元地址，最多可访问 2^n 个存储单元；数据总线用于在 CPU 与存储单元之间传送数据，一次可传送 m 位数据；控制信号则用来控制存储器的读/写等操作。

当 CPU 要访问内存时，首先通过地址总线把地址码送入地址寄存器中锁存，再传送给地址译码器，经译码后使对应于该地址的某一根选择线有效，从而选中相应的存储单元；接着 CPU 发出读/写命令，于是时序控制电路产生对存储单元的读/写操作控制信号。

在进行存储器读操作时，控制电路中的读信号线有效，于是把所选中的存储单元的信息读出并送入读/写驱动器放大，然后送至数据寄存器，再经数据总线送给 CPU。在进行存储器写操作时，CPU 不仅要把地址码送入地址寄存器，还要把待写入的数据传送给数据寄存器，并使控制电路中的写信号线有效，于是数据寄存器中的数据被存入选中的存储单元。

对内存写操作时，存储单元中原有的内容将被新写入的数据取代；读操作时，存储单元中的内容不受影响。所以说对内存的写是“破坏性”的写，对内存的读是“非破坏性”的读。

3.1.4　半导体存储器

半导体存储器具有工艺简单、集成度高、成品率高、可靠性高、存取速度快、体积小、功耗低等特点；其存储电路所占的空间小，可以和译码电路以及缓冲寄存器制作在同一芯片中。所以现代微型计算机的内存储器普遍采用半导体存储器。

半导体存储器从器件制造的工艺角度，分为双极型和 MOS 型。双极型存储器由 TTL 电路制成，其特点是存取速度快、集成度低、功耗大、价格较高。MOS 型存储器由金属氧化物半导体电路制成，与双极型存储器比较，其集成度高、功耗低、价格低，但存取速度慢。

从功能和应用的角度，半导体存储器又分为随机存储器 RAM 和只读存储器 ROM 两大类，如图 3-2 所示。

1. 半导体随机存储器 RAM

半导体随机存储器 RAM 可以对每个存储单元随机进行读出或写入，使用灵活，但它是一种易失性存储器，断电后信息会丢失。

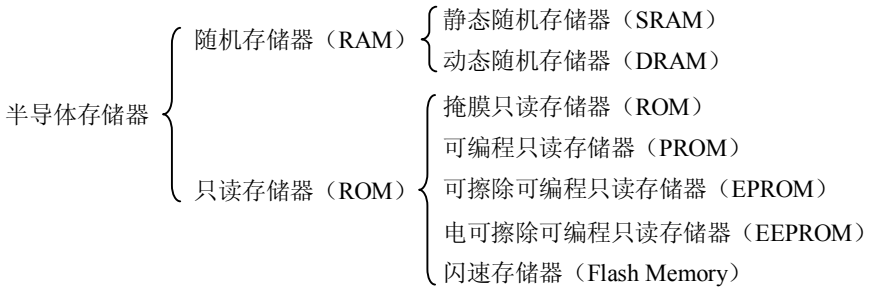


图 3-2　半导体存储器的分类

根据存储原理，随机存储器 RAM 又分为静态随机存储器 SRAM 和动态随机存储器 DRAM。SRAM 存放的信息在不断电的情况下可以长时间保存不变；而 DRAM 保存的内容即使在不掉电的情况下，隔一定时间后也会自动消失，因此要对其进行定期的刷新。

与 DRAM 相比，SRAM 不需要定时刷新，访问速度明显快于 DRAM，但是电路复杂，集成度低，且价格较高，因此多用于高速缓冲存储器 Cache，而 DRAM 具有价格低廉、集成度高等优点，内存条基本都采用 DRAM。

2. 半导体只读存储器 ROM

半导体只读存储器 ROM 是一种只能读出但不能随机写入信息的存储器，所存储的信息可以长久保存，掉电后存储的信息仍不会改变。一般 ROM 用于存放固定程序，如启动程序、BIOS 程序等。

按存储单元的结构和生产工艺的不同，ROM 又可分成掩膜只读存储器（ROM）、可编程只读存储器（Programmable ROM，PROM）、可擦除可编程只读存储器（Erasable PROM，EPROM）、电可擦除可编程只读存储器（Electrically EPROM，EEPROM）、闪速存储器（Flash Memory）。

3.2　随机存储器 RAM

3.2.1　静态 RAM（SRAM）

1. SRAM 的基本存储电路

所谓基本存储电路是指存储一位二进制数的电路，又称单元电路。对于各种基本存储电路，不论其内部结构如何，对外呈现的特性均为用一个信号线选中该电路，电路中所存储的信息通过数据线与外界交换。

典型的 SRAM 基本存储电路如图 3-3 所示，T1、T2、T3、T4 这 4 个晶体管形成两个交叉耦合的反相器，存储单元有两个稳定的状态，分别表示二进制“1”和“0”，此外还需要 T5、T6 这两个晶体管为访问存储单元提供控制信号，因此 SRAM 至少需要 6 个晶体管才能存储和访问一位二进制信息。

在进行写操作时，字线上送来高电平有效信号，写入的信号从 D 线和 $\overline{D}$ 线输入。例如写入 1，则使 D 线为 1，$\overline{D}$ 线为 0，当字线信号消失后，T5、T6 截止，于是 T1～T4 组成的交叉耦合的反相器保持数据 1，只要不掉电这个状态会一直保持，直至重新写入新的数据。

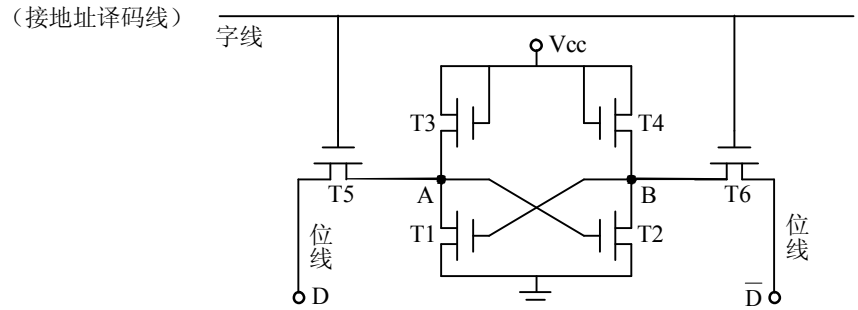


图 3-3　6 管静态 RAM 基本存储电路

在进行读操作时，字线上送来高电平有效信号，使 T5、T6 导通，A 点的状态被送到 D 线上，B 点的状态被送到 $\overline{D}$ 线上，如此读走了原来存储的信息。信息读出后，基本存储电路中原来存储的内容仍然保持不变。

基本存储电路的工作必须有电源，存入的数据才可以保留和读出，若掉电，存入的数据全部丢失。

由以上基本存储电路组成的 SRAM 芯片具有以下特点：

（1）可靠性高、速度快。不必配合内存刷新电路，可提高整体的工作效率。

（2）高稳定性。只要不掉电，SRAM 芯片中存储的信息永远不会丢失，所以无须外加刷新电路，故外围电路简单。

（3）集成度低。由于 SRAM 存储电路中所用的晶体管较多，因而集成度较低。

（4）功耗较大。由于 T1、T2 管组成的双稳态触发器总有一个是导通的，即电路中一直有电流通过，故功耗较大。

由于 SRAM 具有以上特点，它的每位价格较高，用它来构造大容量的存储器显然不划算。因此，SRAM 一般用作高速缓冲存储器 Cache，这可以充分发挥 SRAM 速度快和可靠性高的优势，采用小容量的 SRAM 作为 Cache，价格也不至于太高，这样可以保证微型计算机的高性能价格比。

2. SRAM 的结构

利用多个基本存储电路排成行列矩阵，再加上地址译码电路和读写控制电路，就可以构成读写存储器。下面以 4 行 4 列基本存储电路构成的 16×1 位 SRAM 为例说明它的结构。

如图 3-4 所示，这个可读写的静态随机存储器主要由存储体、读写控制电路、地址译码电路和 I/O 电路构成。

（1）存储体。存储体由多个基本存储电路有规则地组合而成。为了减少片内的连线，便于译码寻址，存储体内所有基本存储电路排列成行列矩阵。由图中可见，存储体排成 4 行×4 列矩阵，每个基本存储电路可存储一个二进制位，构成 16×1 位的存储体。

（2）地址译码电路。存储矩阵中，基本存储电路的地址译码一般有单译码和双译码两种方式。实际存储芯片的地址译码电路与存储体集成在一个芯片中，所以当存储容量较小时，可以使用单译码方式，即使用一个地址译码器。当存储容量较大时，为避免因地址译码器的输出选择线过多，导致集成电路内部结构复杂，所以采用双译码方式，即地址译码电路分为行地址译码和列地址译码两部分。

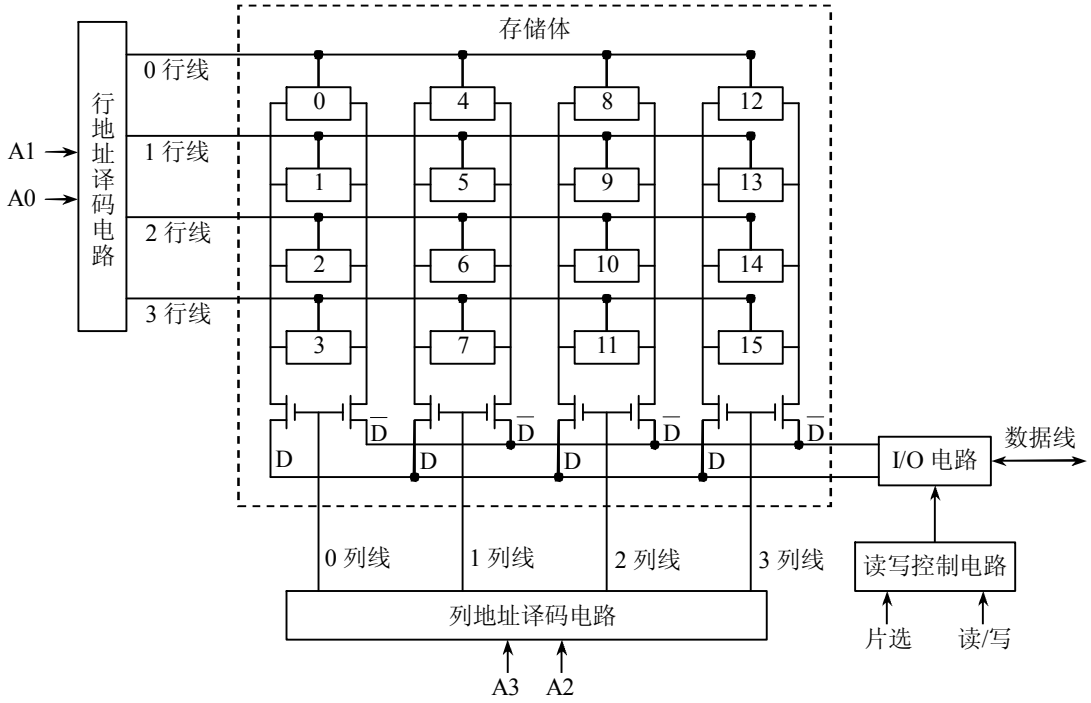


图 3-4　典型的 SRAM 结构

双译码方式中，行、列地址译码器分别接收地址总线上送来的高位、低位地址信号。若行译码器的地址输入信号为 n 位，则行译码器的输出选择线有 2^n 根，一根行选择线对应存储矩阵中的一行，与该行上各个基本存储电路的字线并联。同理，存储矩阵中同一列上各个基本存储电路的字线与同一列选择线并联。

如图 3-4 所示，当给定的地址码为 A3A2A1A0＝0000 时，A1A0 经行地址译码器译码后，使 0 行线为有效的高电平，A3A2 经列地址译码器译码后，使 0 列线为有效的高电平，于是 0 行与 0 列交叉处的 0 号基本存储电路被选中。又如，当给定的地址码为 A3A2A1A0＝0110 时，2 行与 1 列交叉处的 6 号基本存储电路被选中。如此，只要给定一个地址码，就会唯一选中一个存储单元。

（3）读写控制电路。读写控制电路接收 CPU 发来的片选及读/写控制信号，控制对本存储器的数据读写方向。

由于单片存储器的容量限制，所以一般的系统存储器都是由多个芯片组成的，由于地址信号和读写控制信号同时加到所有芯片上，若想选中一个芯片中的某一单元，那么其他芯片中相同地址的单元就不应该同时被选中，所以每个芯片还要有一个片选控制信号。当片选端送来有效信号时，该存储芯片被唯一选中，这时才能进行读写操作。

读/写控制信号用来规定对存储器进行读或写操作。所以，一个存储芯片中某单元的信息能否与系统数据总线的信息交换，是受地址、片选和读写控制三个信号同时控制的。

（4）I/O 电路。I/O 电路处于数据总线与存储体之间，具有数据传送功能，并具有放大信号的作用。I/O 电路内部的三态双向缓冲器，连接存储体中各个基本存储电路的位线及系统数据总线，在存储芯片未被选中时，三态缓冲器呈高阻态，使存储器与系统数据总线隔离；当片

选信号有效时，在读写控制信号的控制下，可以对被选中的单元进行数据的读出或写入。

3. 典型的 SRAM 芯片

常用的 SRAM 芯片有 1K×4 位（如 2114）、2K×8 位（如 2128、6116）、4K×8 位（如 6132、6232）、8K×8 位（如 6164、6264、3264、7164）、32K×8 位（如 61256、71256、5C256）、64K×8 位（如 64C512）等。

（1）Intel 2114 芯片。

Intel 2114 为 NMOS 静态 RAM，单一+5V 电源，4 位数据输入/输出端，采用三态控制。所有的输入端和输出端都与 TTL 兼容，其引脚排列如图 3-5 所示。

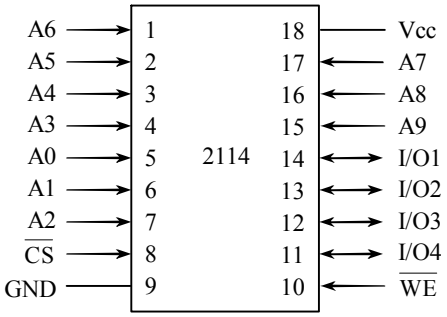


图 3-5 Intel 2114 引脚图

Intel 2114 容量为 1K×4 位，即 1024 个字，每字 4 位。芯片内部共有 4096 个基本存储电路，排列成 64×64 的矩阵。地址线共 10 根（A9～A0），其中 A8～A3 这 6 根用于行译码，产生 64 个行选择信号；A9、A2、A1、A0 这 4 根地址线用于列译码，产生 16 个列译码信号，并且每个列译码信号同时接 4 位。各信号线的意义如表 3-1 所示。

表 3-1 Intel 2114 引脚功能表

引脚	功能	说明	
A9～A0	地址	输入，三态	
I/O4～I/O1	数据线	双向，三态	
$\overline{CS}$	片选	输入，低电平有效	
$\overline{WE}$	写允许	=0	写
		=1	读
Vcc	电源		
GND	地		

在 $\overline{CS}$ 片选信号低电平有效的情况下，当写允许控制端 $\overline{WE}$ 为低电平时，可以对存储芯片 2114 进行写操作，当 $\overline{WE}$ 为高电平时，可以对 2114 进行读操作。

（2）Intel 6116 芯片。

Intel 6116 采用 CMOS 工艺制造，与 TTL 兼容，完全静态，无需时钟脉冲或定时选通脉冲，24 引脚封装，引脚信号排列如图 3-6 所示。

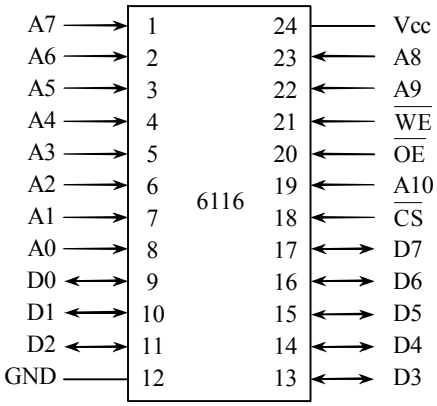


图 3-6　Intel 6116 引脚图

Intel 6116 芯片的存储容量为 2KB，即 2048 个字，每字 8 位。芯片内部有 16384 个基本存储电路，排列成 128×128 的矩阵。需要 11 条地址线，A10～A4 这 7 根地址线用于行译码，产生 128 个行选择线，A3～A0 这 4 根地址线产生 16 个列选择线，每个列选择线同时接 8 位。各信号线的意义如表 3-2 所示。

表 3-2　Intel 6116 引脚功能表

引脚	功能	说明
A10～A0	地址	输入，三态
D7～D0	数据线	双向，三态
$\overline{CS}$	片选	输入，低电平有效
$\overline{WE}$	写允许	输入，低电平有效
$\overline{OE}$	读允许	输入，低电平有效
Vcc	电源	
GND	地	

在$\overline{CS}$片选信号低电平有效的情况下，当写允许端$\overline{WE}$ =0 且$\overline{OE}$ =1 时，可以对 Intel 6116 进行写操作，当$\overline{WE}$ =1 并且$\overline{OE}$ =0 时，可以对 Intel 6116 进行读操作。

其他 SRAM 芯片的结构与 Intel 6116 相似，不过容量及地址线数量有所不同。如 8KB 容量的 Intel 6264 芯片为 28 个引脚的双列直插式，使用单一的+5V 电源。

SRAM 的外部引脚设置与 EPROM 引脚兼容，从而使接口电路的连线更为方便。

3.2.2　动态 RAM（DRAM）

动态 RAM 的基本存储电路是利用 MOS 管的栅极分布电容充电、放电来保存信息的。

1. 单管 DRAM 基本存储电路

单管 DRAM 基本存储电路如图 3-7 所示，由一只 MOS 管和一个与源极相连的电容 C 组成。DRAM 中信息的存放依靠电容 C，电容 C 有电荷时表示存储的信息为 1，无电荷时表示存储的是 0。但是由于任何电容都存在漏电现象，所以即使电容 C 有电荷，过一段时间后随着

电荷的泄漏，信息也就丢失了。解决的办法就是定时刷新，每隔一定时间刷新一次，使电容中的电荷得到补充。

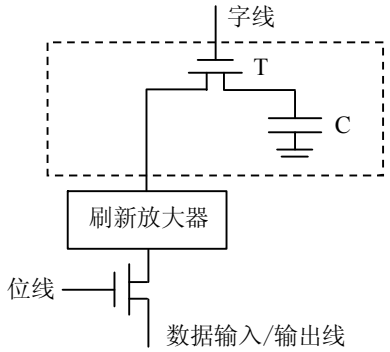


图 3-7　单管动态 RAM 基本存储电路

在未进行读写操作时，字线处于低电平，MOS 管 T 截止，电容 C 与外电路断开，不能进行充电、放电，电路保持原状态。

读操作时，字线为高电平，基本存储电路中的 T 管导通，刷新放大器读取存储于电容 C 上的电压值。刷新放大器的灵敏度很高，放大倍数较大，它能将读来的电压值转换为对应的逻辑电平 0 或 1。位线上的选择信号的作用是驱动基本存储电路，从而可以由输入/输出线将信息输出。

在读出过程中，由于基本存储电路中的电容受到干扰，为了在读出之后，仍能保存原来信息，刷新放大电路在对这些电容上的电压值读取之后，又立即重写回存储电容 C 中，即在读操作时完成刷新。由于刷新时位线信号总为低电平，所以电容 C 上的信息不会再被送到数据总线上。

DRAM 的存取速度不及 SRAM，需要定时刷新。但由于 DRAM 所用的 MOS 管少，集成度较高，功耗降低，价格低，所以在微型计算机中被大量用作内存。

2. DRAM 的刷新方式

动态存储器刷新就是周期性地对动态存储器进行读出、放大、再写回的过程。

DRAM 是利用电容存储电荷的原理来保存信息的，由于电容会泄漏放电，所以为了保证电容中的电荷不丢失，每隔一定时间必须对 DRAM 读出、放大、再写回一次，从而使原来处于逻辑电平 1 的电容上所泄漏的电荷得到补充，而原来处于电平 0 的电容仍保持 0，这就是对 DRAM 的刷新。

一般来说，DRAM 应在 2ms 时间内将全部基本存储电路刷新一遍。但是，由于读写操作的随机性，不能保证在 2ms 内对 DRAM 的所有行都能遍访一次，所以需要依靠专门的存储器刷新周期来系统地完成对 DRAM 的刷新。

在存储系统中，刷新是按行进行的，即一行内所有的基本存储电路同时读出、放大、再写回，一个刷新周期内对所有行中所有的基本存储电路都刷新一遍。例如，对一个 64 行×32 列的存储矩阵进行刷新，一次对一行中 32 个基本存储电路同时刷新，在一个刷新周期 2ms 内必须将 64 行全部刷新完毕，所以对每行的刷新必须在 31μs（2ms/64≈31μs）内完成。

DRAM 的刷新常采用两种方法：一是利用专门的 DRAM 控制器实现刷新控制，如 Intel 8203 控制器；二是在每个 DRAM 芯片上集成刷新控制电路，使存储器件自身完成刷新，这种

器件叫综合型 DRAM，如 Intel 2186/2187。

3．典型 DRAM 芯片

常用的 DRAM 芯片有 64K×1 位、64K×4 位、256K×1 位、256K×4 位、1M×1 位、1M×4 位、4M×4 位等。

Intel 2164 是容量为 64K×1 位的典型 DRAM 芯片，片内有 65536 个存储单元，每个单元存放 1 位数据，用 8 片 2164 就可以构成 64KB 的存储器。Intel 2164 片内 64K×1 位存储体分为 4 个 128×128 存储矩阵，每个 128×128 矩阵都采用双译码方式，行、列各需要 7 位地址，如图 3-8 所示。

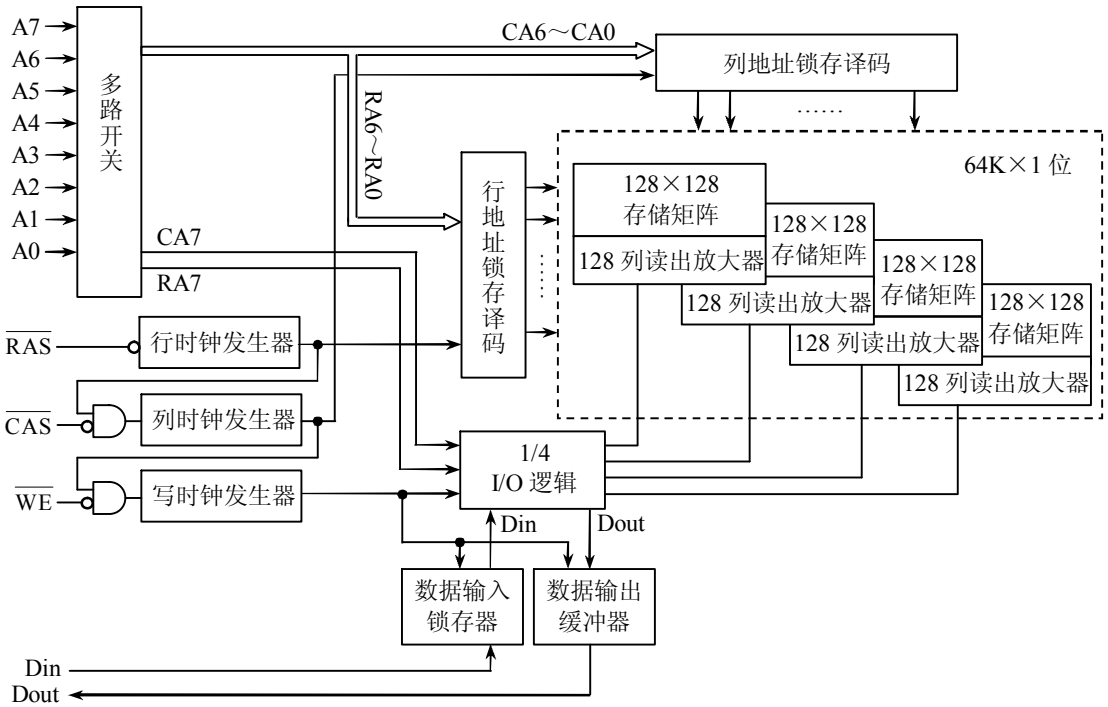


图 3-8　Intel 2164 的内部结构

若想在片内寻址 64K 个单元，通常需要 16 条地址线。为了减少地址线引脚数，Intel 2164 采用分时复用技术，将片内地址线分为行地址线和列地址线，这样芯片对外只需引出 8 条地址线，于是 16 位地址信号可以拆分成低 8 位和高 8 位，分两次送至芯片中。芯片内部利用多路开关和地址锁存器，首先由行地址选通信号 $\overline{\text{RAS}}$（低电平有效）把先送来的 8 位地址中的低 7 位（RA6～RA0）送至行地址锁存器中，随后列地址选通信号 $\overline{\text{CAS}}$（低电平有效）把后送来的 8 位地址中的低 7 位（CA6～CA0）送至列地址锁存器中。

需要说明一点，7 位行地址和 7 位列地址是同时加到存储体内 4 个 128×128 存储矩阵上的，也就是说，经行、列地址译码后，4 个存储矩阵中的同位置存储单元被同时选中。那么到底选择 4 个存储矩阵中的哪一个呢？由芯片内部行、列地址中的最高位 RA7 和 CA7 决定。RA7 和 CA7 接至 1/4 的 I/O 逻辑，在对芯片读/写时，通过这两位地址的 4 种不同编码，实现对 4 个矩阵的选 1 操作。

Intel 2164 的数据输入和输出是分开的，由 $\overline{\text{WE}}$ 信号控制对芯片的读写。当 $\overline{\text{WE}}$ 为低电平

时，数据输入缓冲器允许，于是 Din 引脚上的数据经数据输入缓冲器对选中单元进行写入；当 $\overline{WE}$ 为高电平时，数据输出缓冲器允许，于是所选中单元的内容经过数据输出缓冲器由 Dout 引脚读出。

Intel 2164 芯片的引脚如图 3-9 所示，各引脚的说明如表 3-3 所示。Intel 2164 芯片没有片选信号，实际上用 $\overline{RAS}$ 和 $\overline{CAS}$ 作为片选信号。

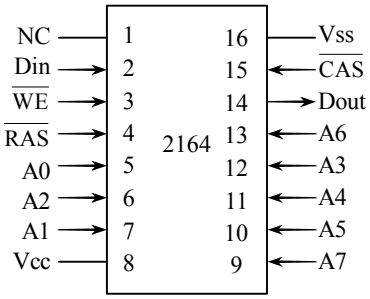


图 3-9　Intel 2164 引脚图

表 3-3　Intel 2164 引脚功能表

引脚	功能	说明	
A7～A0	地址	输入，三态	
Din	数据输入线	输入，三态	
Dout	数据输出线	输出，三态	
$\overline{RAS}$	行地址选通	输入，低电平有效	
$\overline{CAS}$	列地址选通	输入，低电平有效	
$\overline{WE}$	写允许	=0	写
		=1	读
Vcc	电源		
Vss	地		

Intel 2164 的 8 条地址线也用于刷新（刷新时地址计数，逐行刷新，2ms 内全部刷新一遍）。刷新时只用 $\overline{RAS}$ 信号和低 7 位行地址 RA6～RA0，RA7 不用。$\overline{RAS}$ 的低电平状态会使 $\overline{CAS}$ 和 $\overline{WE}$ 禁止，从而保证刷新时不使用列地址，并且暂时禁止新的数据写入。$\overline{RAS}$ 为有效低电平时，行地址 RA6～RA0 被锁存到行地址锁存器中，因为这 7 位行地址是同时加到 4 个 128×128 存储矩阵上的，所以一次行地址译码就有 4 个存储矩阵的同一行被选中。刷新是周期性的，平均每隔 15μs（2ms/128≈15μs）刷新一次，每次刷新一行，一行中 512 个存储单元的信息被选通送至 512 个读出放大器，经过鉴别后再重新写回。

3.3　只读存储器 ROM

半导体只读存储器 ROM 具有如下特点：

（1）信息一旦写入就可以长期保存，不受电源掉电的影响。

（2）信息一旦写入只能读出，不能再随机写入新的内容。

（3）结构简单、成本低、集成度高。

（4）可靠性高。

因此，在微型计算机系统中，ROM 常用来存储一些固定的程序，如监控程序、启动程序、基本输入/输出服务程序等。

3.3.1 掩膜只读存储器 ROM

掩膜式 ROM 中的信息是生产厂家在制造过程中写入的。掩膜式 ROM 制成后，存储的信息就不能再改变了，用户在使用时只能读出。如图 3-10 所示是一个简单的 4×4 位 MOS 管 ROM，采用单译码方式，两位地址 A1、A0 输入，经译码后产生 4 条选择字线，可分别选中 4 个单元，每个单元有 4 位。

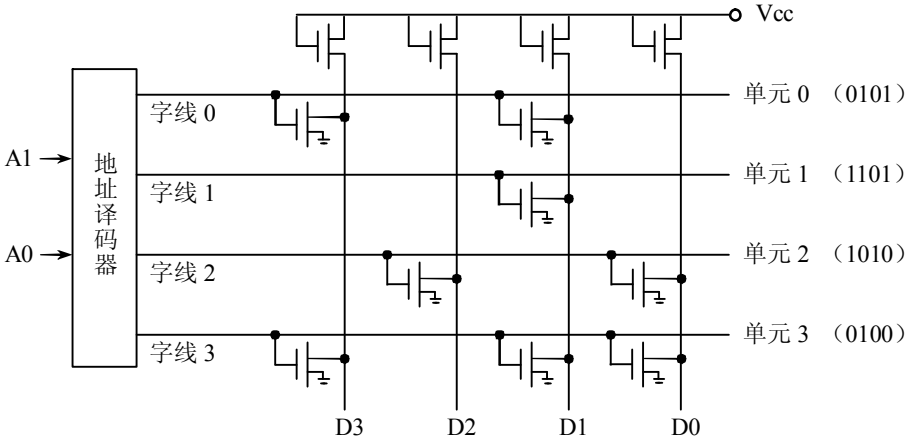


图 3-10 4×4 位掩膜 ROM 示意图

图中所示的矩阵中，在行与列的交叉点上，有的接有管子，有的没有，这是厂家根据用户提供的程序对芯片图形（掩膜）进行二次光刻所决定的，所以称掩膜 ROM。最上排的 4 个 MOS 管起到上拉电阻的作用，它永远处于导通状态，但具有一定的导通电阻，保证了无 MOS 管的位输出为 1，有 MOS 管的位输出为 0。

例如，当地址线 A1A0＝00 时，经地址译码后选中 0 号单元，即字线 0 为高电平，若有管子与其相连，如图中的 D3 和 D1，其相应的 MOS 管导通，输出为 0；而 D2 和 D0 没有管子与字线 0 相连，则输出为 1，故单元 0 的 D3D2D1D0＝0101。

因为掩膜式 ROM 需要专门制作掩膜板，成本很高，制作周期较长，所以只在生产批量较大时，才做成这种掩膜式 ROM。

3.3.2 可编程只读存储器 PROM

半导体 PROM 适用于用户根据自己的需要来写入存储信息的场合。厂家生产的 PROM 芯片不事先存入固定内容，存储矩阵的所有字线和位线的交叉处均连接有二极管或三极管，即出厂时，存储单元的内容是全 1。使用时，用户可根据自己的需要，将某些位的内容改写为 0，但只能改写一次。

如图 3-11 所示为用双极型晶体管和熔丝组成的 PROM 的基本存储结构。晶体管的集电极接 Vcc，基极连接字线，发射极通过一个熔丝与列线相连，所以也称为熔丝式 PROM。

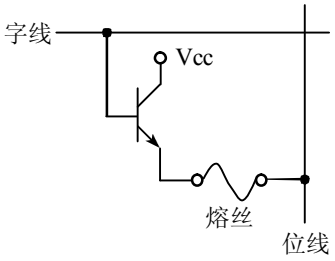


图 3-11　PROM 基本存储电路

PROM 在出厂时，晶体管阵列的熔丝均为完好状态。编程时，通过字线选中某个晶体管，当写入信息时，可在 Vcc 端加高电平。若某位写 0，则向相应位线送低电平，此时管子导通，控制电流使该位熔丝烧断，即存入 0；若某位写 1，向相应位线送高电平，此时管子截止，使熔丝保持原状，即存入 1。

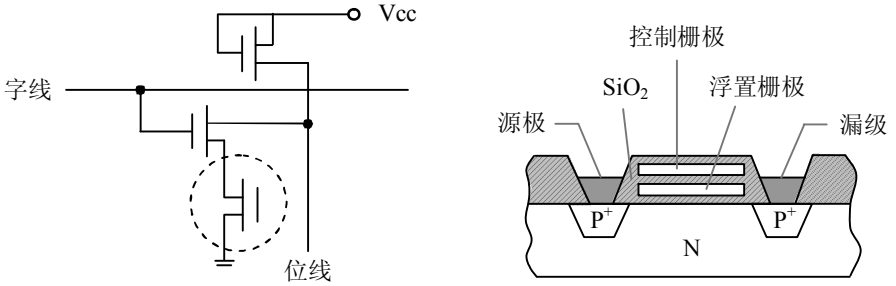
可见，熔丝一旦烧断，就不能再复原，所以这种 PROM 是一种一次性写入的只读存储器。PROM 的电路和工艺要比 ROM 复杂，所以价格较贵。

3.3.3　可擦除可编程只读存储器 EPROM

由于 PROM 的内容在一次写好后就不能再改变，因此能够重复擦写的 EPROM 被广泛应用。EPROM 芯片的顶部开有一个圆形的石英玻璃窗口，通过紫外线的照射可将片内存储的原有信息“擦除”，恢复为出厂时的原始状态。根据需要可利用 EPROM 的专用编程器对其编程写入，以不透光的贴纸或胶布封住窗口后，信息可长久保持。这种芯片可反复使用。

EPROM 基本存储电路如图 3-12（a）所示，在每个字线与位线的交叉点都设计一个浮栅 MOS 管，根据 MOS 管的导通或截止状态确定该位为 0 或 1。

浮栅 MOS 管结构如图 3-12（b）所示，该管与普通 P 沟道增强型 MOS 管相似，只是它的栅极没有引出端，而被 SiO2 绝缘层包围，处于浮置状态。出厂时浮置栅极（简称浮栅）上没有电荷，源极与漏极之间不形成导电沟道，MOS 管处于截止状态，此时存储的信息为 1。



（a）基本存储电路　　（b）浮栅 MOS 管结构

图 3-12　EPROM 基本存储电路

在对其编程时，如果在漏源极之间加上较高电压（通常为 15V 以上，称为编程电压），会产生雪崩击穿现象，获得能量的电子会穿过 SiO2 注入到浮栅中，编程结束后，在漏源极之间感应出的导电沟道将会保持下来，MOS 管处于导通状态，表示存入 0。

EPROM 芯片在出厂时，浮栅中没有积存电荷，存储的信息为全 1。EPROM 的编程过程实际是对某些单元写入 0 的过程，即借助专门的编程器对某些位施加高压，从而向浮栅注入电子的过程。需要擦除时可用一定强度的紫外线灯照射 EPROM 芯片的透明窗口，经 15～20 分钟后浮栅上的电荷泄漏掉，芯片内所有存储单元全部擦除，为全 1。读操作时，由于控制栅极

施加的电压较小，不足以改变浮栅中原有的电荷量，所以不会改变芯片的原有数据。

常用的 EPROM 有 2716（2K×8 位）、2732（4K×8 位）、2764（8K×8 位）、27128（16K×8 位）、27256（32K×8 位）、27512（64K×8 位）等典型芯片。

3.3.4 电可擦除可编程只读存储器 EEPROM

尽管 EPROM 可以实现多次编程，但在编程过程中，整个芯片即使只写错一位，也必须用紫外线擦除器全部擦除重写，因此操作起来比较麻烦。另外，EPROM 可被擦除后重写的次数也是有限的，一块芯片的使用寿命往往不太长。

EEPROM 也是一种可用电擦除和编程的只读存储器，原理与 EPROM 类似，但是擦除的方式是使用高电场来完成，因此不需要透明窗。

EEPROM 主要特点是能在应用系统中进行在线读写，并且在断电情况下保存的数据信息不会丢失，它既能像 RAM 那样随机地改写，又能像 ROM 那样在掉电的情况下非易失地保存数据，可作为系统中可靠保存数据的存储器。其擦写次数可达 1 万次以上，数据可保存 10 年以上，故 EEPROM 比 EPROM 具有更大的优越性。

由于 EEPROM 兼有 RAM 和 ROM 的双重优点，因此，在计算机系统中使用 EEPROM 以后，可使整机的系统应用变得更加灵活和方便。

3.3.5 闪速存储器 Flash Memory

闪速存储器（Flash Memory），简称闪存，它与 EEPROM 类似，也是一种电擦写型 ROM。EEPROM 是按字节擦写，速度慢，而闪存是以块为单位进行擦写，速度快。

闪存是近年来发展最快的一种新型半导体存储器。由于闪存是以 EPROM 的存储单元为基础的，因此具有非易失性，在断电时也能保留所存储的内容，这使它优于需要持续供电才能存储信息的易失性存储器；它的单元结构和具有的 EPROM 基本特性，使得它的制造特别经济，其价格低于 DRAM；它在密度增加时仍保持可测性，具有可靠性；可实现大规模电擦除，它的擦除功能可迅速清除整个存储器的所有内容；采用快速脉冲编程方法，整个芯片编程时间短，可实现高速编程；可重复使用。

总之，由于闪存具有在线电擦写、功耗低、容量大、擦写速度快、抗震性好、存储可靠性高、成本低等明显优势，兼具了 ROM 和 RAM 的性能，使得闪存在移动设备中获得广泛的应用，如 U 盘、数码相机、笔记本、随身听等大都采用闪存作为存储介质。目前闪存是非易失性存储器中应用最广泛的存储器。

综上所述，各种半导体存储器都有各自的存储特点，它们的主要应用如表 3-4 所示。

表 3-4 各种半导体存储器的应用

存储器	应用	存储器	应用
SRAM	Cache	EPROM	用于产品试制阶段试编程序
DRAM	内存储器	EEPROM	IC 卡上存储信息
ROM	固化程序	闪速存储器	固态盘、U 盘、存储卡
PROM	自编程序，用于工业控制或电器中		

3.4 半导体存储器接口

通常内存储器是由多个半导体存储芯片组成的。内存储器与 CPU 的连接就是多个半导体存储芯片与 CPU 芯片的连接。就 CPU 而言，CPU 是通过数据总线 DB、地址总线 AB 和控制总线 CB 与外部交换信息的，因此，半导体存储芯片与 CPU 的连接也就是与 CPU 三种总线的连接。

3.4.1 存储器的选址

微型计算机系统中，CPU 对内存进行读/写时，首先要对存储芯片进行选择，使相应芯片的片选端 $\overline{CS}$ 有效，称为片选，然后在选中的芯片内部再选择某一存储单元，称为字选。片选信号和字选信号均由 CPU 发出的地址信号经译码电路译码后产生。

片选信号由存储芯片本身不使用的高位地址经外部译码电路按一定方式译码后产生，这部分译码电路在存储芯片外部，是需要自行设计的部分。每个存储芯片都有一定数量的地址输入端，在接收 CPU 输出的低位地址后，经内部译码电路译码后产生字选信号，选中芯片内部指定的存储单元，这部分译码电路在芯片内部，不需要用户设计。

实现片选的存储芯片外部译码电路的具体接法决定了存储芯片的寻址空间。设计微型计算机的存储系统时，要保证存储芯片中的每个存储单元与实际地址一一对应，这样才能通过准确寻址对存储单元进行读写操作。在存储系统中，通常使用存储芯片本身不用的高位地址实现片选，一般有以下 3 种：

（1）线选法。所谓线选法，是指用某一条高位地址线直接作为存储芯片的片选信号的方法。在简单的微型计算机系统中，由于存储容量不大，存储芯片数也不多，可以使用存储芯片本身不使用的单根地址线作片选信号，每个存储芯片只用一根地址线选通。

这种方法的优点是连接简单，无须专门的译码电路。但是局限性较大，首先表现在存储器寻址空间可能不连续，会造成大量的地址空间浪费；其次，若有多片存储器均使用线选法，可能会出现地址不唯一现象，造成数据冲突，这是不允许的。所以线选法不适合于系统中存在多片存储器的情况。

（2）部分译码法。所谓部分译码法，是指将部分高位地址通过译码电路或译码器产生存储器片选信号的方法。该方法只使用部分高位线进行译码，从而产生片选信号，剩余高位线可以空闲或直接用作其他存储芯片的片选控制信号，使用这种方法可能会出现地址不唯一的现象。

（3）全译码法。全译码法是指存储芯片本身不使用的高位地址全部参与译码的方法。在 CPU 输出的所有地址总线中，除了将低位地址线直接连至各存储芯片的地址线外，余下的高位地址线全部送至译码电路或译码器，译码输出作为各芯片的片选信号。

这种方法可以提供对全部存储空间的寻址能力；还可以使每片存储器的寻址空间唯一确定，而且是连续的，若安排合理，可避免空间浪费。

3.4.2 存储器的容量扩展

在实际应用中，由于单个存储芯片的容量总是有限的，很难满足实际存储系统的要求，因此需要将多个存储芯片连接在一起进行容量扩展，从而满足 CPU 数据总线宽度的需要或提

供给 CPU 更大的存储空间。

在进行存储器容量扩展时，首先要确定内存储器的大小，并根据所选择的存储芯片容量确定需要的存储芯片数目，最后再将选择好的存储芯片与 CPU 有机地连接起来。下面具体介绍几种连接方法。

1. 位扩展

位扩展又称横向扩展，指扩展存储单元的位数，增加字长，而不需要增加单元数。

RAM 芯片具有 1 位、4 位和 8 位等不同结构，如 64K×1 位、256K×4 位、128K×8 位等。当内存储器的单元数与存储芯片的单元数相等，而存储芯片的位数小于内存储器的字长时，就要进行位扩展。

位扩展的连接方法是地址线、片选信号线、读/写信号线分别并联，而数据线串联。例如，要将 64K×1 位的芯片扩展为 64K×8 位的内存储器，其扩展连接方法如图 3-13 所示。

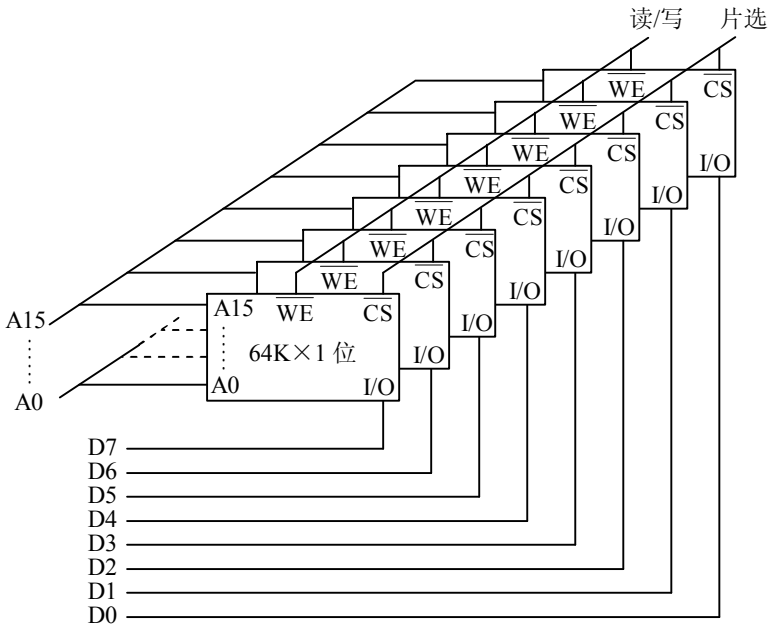


图 3-13　位扩展的连接示例

用 64K×1 位的芯片扩展为 64K×8 位的内存储器，需要 8 片存储芯片，连接时，将 8 个芯片的地址线 A15～A0、读/写信号线 $\overline{WE}$ 、片选信号线 $\overline{CS}$ 分别并接在一起，而 8 个芯片的数据线（每片 1 位）分别与 CPU 的数据线 D7～D0 相连。

2. 字扩展

字扩展又称纵向扩展，是指扩展存储单元的个数，存储单元的位数并不改变。

其方法是将地址线、数据线、读/写信号线分别并接在一起，而将片选信号线单独引出，用以决定每一芯片的地址范围，使存储器的地址空间为各个芯片地址空间之和。例如用容量为 16K×8 位的存储芯片组成一个 64K×8 位的内存储器，则需要 4 片这样的存储芯片，其扩展连接方法如图 3-14 所示。

连接时，将 4 个芯片的地址线 A13～A0、读/写信号线 $\overline{WE}$ 、数据线 D7～D0 分别并接在一起，而 4 个芯片的片选信号线 $\overline{CS}$ 单独引出，连至地址译码器的四个输出选择端。

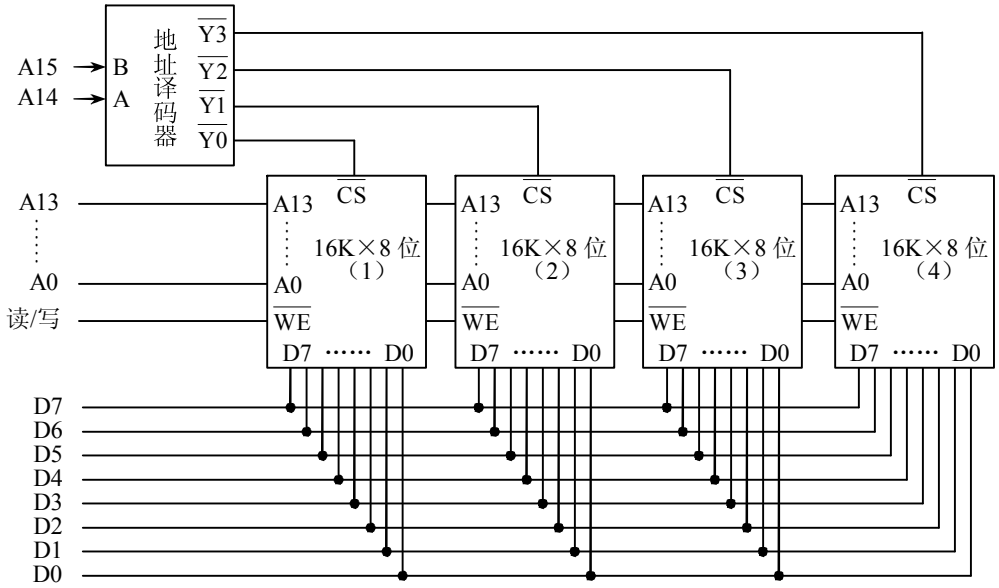


图 3-14　字扩展的连接示例

参与片选译码的应为高位地址。由于每个存储芯片的容量为 16K×8 位，芯片内的存储单元所需的地址线为 A13～A0（16K＝2^{14}），故由 A15 和 A14 经地址译码器译码后，得到 4 个芯片的片选信号线 $\overline{Y3}$～$\overline{Y0}$，分别对应 A15 和 A14 的 11～00 编码；每个芯片内部地址 A13～A0 的地址范围为 0000H～3FFFH。由此将片选地址 A15、A14 与片内地址 A13～A0 统一在一起，最后可以得出 4 个存储芯片的存储单元地址范围，具体如表 3-5 所示。

表 3-5　64KB（4 片 16KB 芯片字扩展）存储器单元地址范围

芯片 \ 地址范围	A15 A14	A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0	寻址空间
芯片（1）	0　0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 …… 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0000H～3FFFH
芯片（2）	0　1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 …… 1 1 1 1 1 1 1 1 1 1 1 1 1 1	4000H～7FFFH
芯片（3）	1　0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 …… 1 1 1 1 1 1 1 1 1 1 1 1 1 1	8000H～BFFFH
芯片（4）	1　1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 …… 1 1 1 1 1 1 1 1 1 1 1 1 1 1	C000H～FFFFH

3. 字位扩展

实际的存储器常常需要在字方向和位方向同时扩展。例如，若采用 16K×4 位的存储芯片

组成 64K×8 位的内存储器，需要多少个这样的芯片呢？各芯片又如何连接在一起呢？

因为每个芯片的存储单元位数（4 位）以及存储单元数目（16K）都不满足内存储器的容量要求（64K×8 位），所以需要进行位、字两个方向的扩展。

首先进行位扩展，满足每个存储单元为 8 位的要求，根据每芯片 4 位，分析出需要两片构成一组。然后以组为单位进行字扩展，满足存储单元的数目要求，因为在每组内存储单元数仍为 16K，所以需要 4 组才能达到存储器单元数目为 64K 的要求。因此，需要 4 组，共计 8 片，其扩展连接方法如图 3-15 所示。

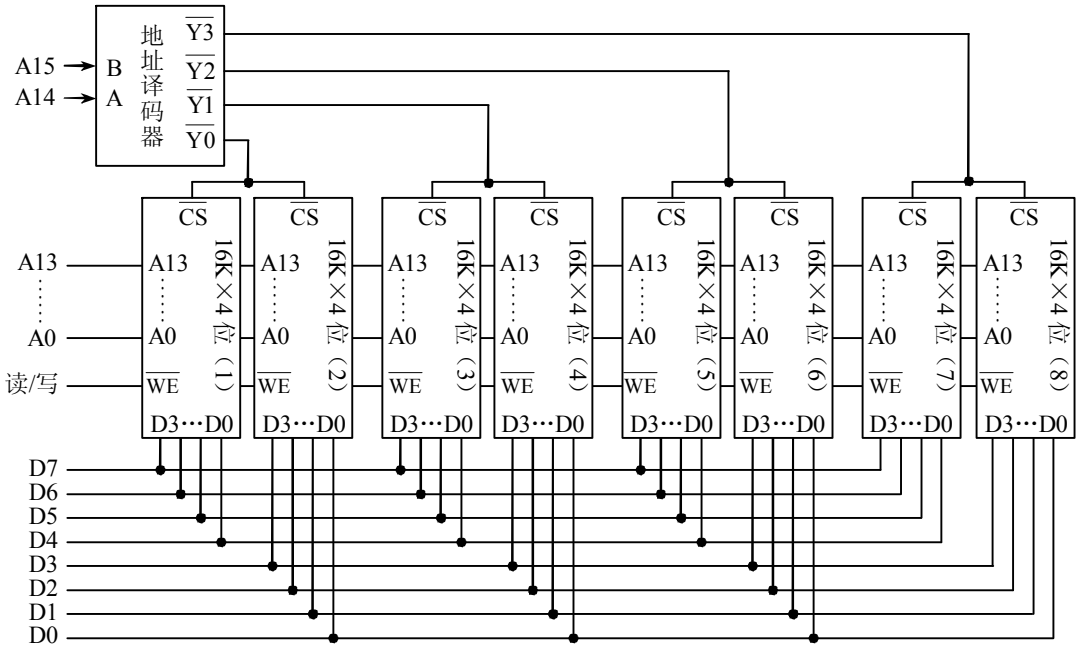


图 3-15　字位扩展的连接示例

由图中可见，寻址 64KB 容量的存储器需要 16 根地址线。这 16 根地址分为高位地址和低位地址两部分。首先各个芯片内的 14 根地址线 A13～A0 直接与系统地址总线的 A13～A0 并联在一起，用于片（组）内寻址；还需要两根高位地址线 A15 和 A14 经地址译码器译码后产生 4 根选择信号线，用作组间寻址。同组芯片的$\overline{CS}$端并接后，分别与 4 根选择信号线$\overline{Y3}$～$\overline{Y0}$相接。同组内两个芯片的各自 4 位数据线 D3～D0 分别与系统数据总线 D7～D0 相连。读/写信号线与 8 个芯片的$\overline{WE}$端并接在一起。这样，经过组合后就形成了容量为 64KB 的存储器。依此类推，就可以得到容量更大的存储器结构。

【例 3.1】使用 2114（1K×4 位）存储芯片组成 2K×8 位的内存储器，需要多少片这样的芯片？如何连接？

分析过程如下：

（1）根据存储器总容量及单片 2114 的容量计算出所需的芯片数：$\frac{2K\times 8位}{1K\times 4位}=4$（片）。

（2）根据存储器及单片 2114 的单元数计算出字扩展所需的芯片组数：$\frac{2K}{1K}=2$（组）。

（3）根据存储器及单片 2114 的单元位数计算出位扩展所需的芯片数：$\frac{8\text{位}}{4\text{位}}=2$（片/组）。

（4）每个 2114 芯片内部的 A9～A0 分别与系统地址总线 A9～A0 并接，A10 用来作片选端，接至两组芯片（4 片）的$\overline{CS}$端，A10 与第一组芯片的$\overline{CS}$直接相连、经非门取反后与第二组芯片的$\overline{CS}$相连。

（5）系统读/写信号直接与各个芯片的$\overline{WE}$端相连。

（6）同组内两个芯片的各自 4 位数据线 I/O4～I/O1 分别与系统数据总线 D7～D0 相连。

（7）存储器容量扩展连接方法如图 3-16 所示。

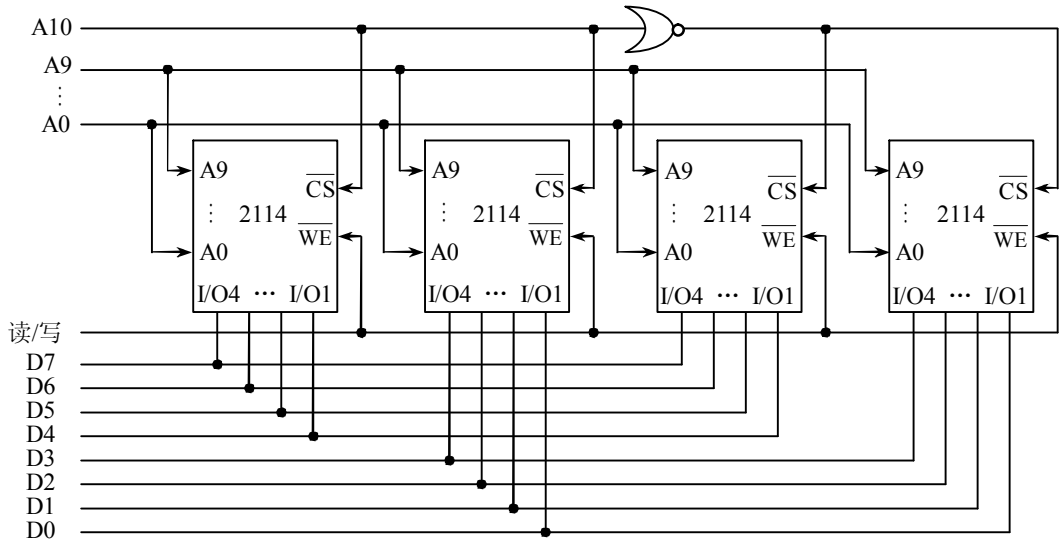


图 3-16　4 片 2114 扩展组成 2K×8 位存储器

【例 3.2】利用 74LS138 作为地址译码器（其功能表如表 3-6 所示），使用 4 片 6116（2K×8 位）存储芯片组成 8KB 的存储器，存储器容量扩展连接方法如图 3-17 所示，试分析各存储芯片的寻址范围。

表 3-6　74LS138 功能表

G_1	$\overline{G}_{2A}$	$\overline{G}_{2B}$	C　B　A	输出
1	0	0	0　0　0	$\overline{Y0}$=0，其他输出均为 1
			0　0　1	$\overline{Y1}$=0，其他输出均为 1
			0　1　0	$\overline{Y2}$=0，其他输出均为 1
			0　1　1	$\overline{Y3}$=0，其他输出均为 1
			1　0　0	$\overline{Y4}$=0，其他输出均为 1
			1　0　1	$\overline{Y5}$=0，其他输出均为 1
			1　1　0	$\overline{Y6}$=0，其他输出均为 1
			1　1　1	$\overline{Y7}$=0，其他输出均为 1

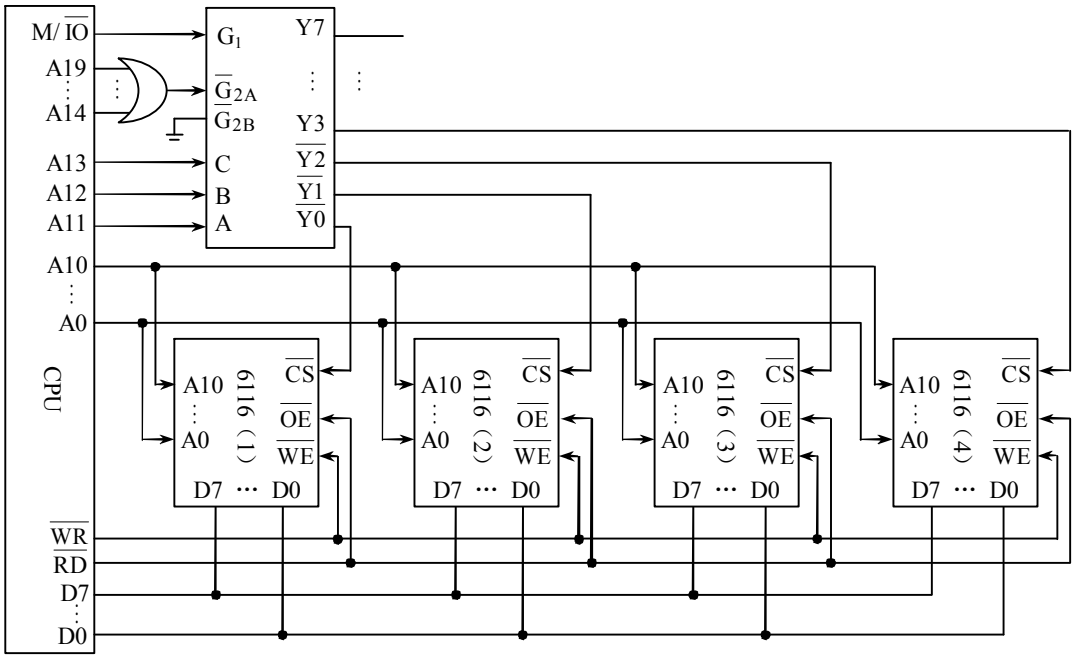


图 3-17　4 片 6116 扩展组成 8KB 存储器

分析过程如下：

（1）根据 74LS138 功能表可知，在 M/$\overline{IO}$ ＝1，A19～A14＝000000 的情况下，当 A13～A11 分别为 000、001、010、011 时，分别选中 1、2、3、4 号芯片。

（2）各芯片的片内地址为 A10～A0，片内地址范围为 0000H～07FFH。

（3）将 A19～A0 统一起来，得出 4 个存储芯片的存储单元地址范围，如表 3-7 所示。

表 3-7　8KB（4 片 6116）存储器单元地址范围

芯片 \ 地址范围	A19～A13	A12 A11	A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0	寻址空间
6116（1）	0000000	0　0	0 0 0 0 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1 1 1 1	00000H～007FFH
6116（2）		0　1	0 0 0 0 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1 1 1 1	00800H～00FFFH
6116（3）		1　0	0 0 0 0 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1 1 1 1	01000H～017FFH
6116（4）		1　1	0 0 0 0 0 0 0 0 0 0 0 …… 1 1 1 1 1 1 1 1 1 1 1	01800H～01FFFH

3.4.3 典型 CPU 与内存储器的连接

一台微型计算机无论内存储器容量多大，不管字长是 8 位、16 位、32 位还是 64 位，都是以 8 位为 1 个字节，以字节为单位进行编址，每个字节单元拥有一个唯一的物理地址。

8086 CPU 和 8088 CPU 地址总线均为 20 位，可管理 1MB 的内存空间；8086 的内部总线和外部总线均为 16 位，而 8088 的内部结构与 8086 基本相同，但外部数据总线却为 8 位，由此导致两种 CPU 与内存储器在连接上的不同。

1. 8088 CPU 与内存储器的连接

8088 存储系统与 CPU 的连接比较简单，其存储器结构如图 3-18 所示。

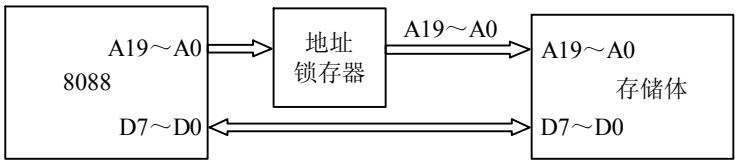


图 3-18 8088 存储器结构

在一个总线周期内，8088 CPU 对内存只能按字节操作，而 8086 CPU 对内存既可以按字节操作，也可以按 16 位的字操作。显然，由于 8088 CPU 对 1 个 16 位数据的存取必须经过两次，所以 8086 CPU 对内存的访问速度必然高于 8088 CPU。

2. 8086 CPU 与内存储器的连接

在 8086 系统中，内存采用分体结构，即将 1MB 的内存空间分成两个 512KB 的存储体，一个存储体中包含偶数地址单元，称为偶地址存储体，另一个存储体包含奇数地址单元，称为奇地址存储体。偶地址存储体与 8086 CPU 的低 8 位数据总线（D7～D0）相连，奇地址存储体与 8086 CPU 的高 8 位数据总线（D15～D8）相连。A19～A1 是体内地址线，它们并行连接到两个存储体上，如图 3-19 所示。

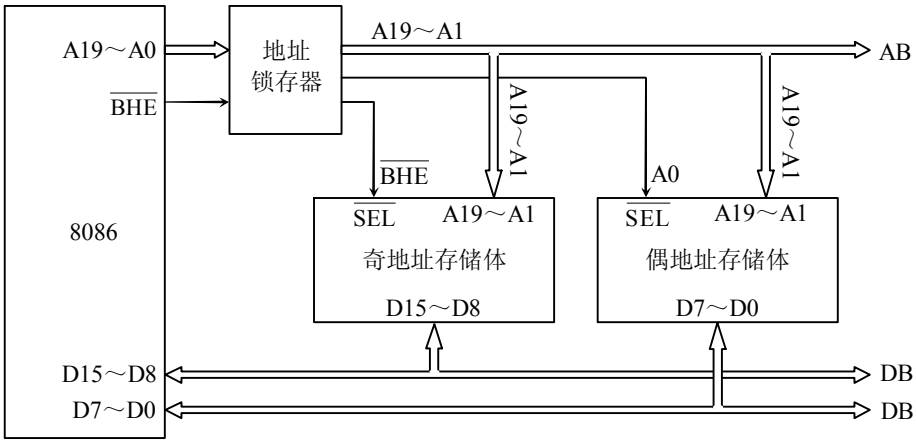


图 3-19 8086 存储器结构

为使 CPU 不仅能访问内存中的一个字节（只访问一个存储体），也能访问内存中的一个字

（同时访问两个存储体），8086 给出了一个有用的控制信号 $\overline{BHE}$ 。 $\overline{BHE}$ 称为高字节数据总线允许信号，当 $\overline{BHE}$ =1 时，奇地址存储体被禁止读/写；当 $\overline{BHE}$ =0 时，可以对奇地址存储体进行读/写操作。当 A0＝0 时，可以对偶地址存储体进行读/写。 $\overline{BHE}$ 与地址线 A0 配合使用实现了对字和字节寻址的控制，如表 3-8 所示。

表 3-8　8086 存储体的操作

$\overline{BHE}$	A0	传送的信息
0	0	同时访问两个存储体，传送一个字（D15～D0）
0	1	只访问奇地址存储体，传送高位字节（D15～D8）
1	0	只访问偶地址存储体，传送低位字节（D7～D0）
1	1	无效

3．80286 CPU 与内存储器的连接

80286 CPU 的数据总线也为 16 位，地址总线为 24 位，最大寻址空间可达 16MB，其存储器结构与 8086 存储器结构相似，也将内存分为奇地址存储体和偶地址存储体，由 $\overline{BHE}$ 和 A0 作为奇、偶存储体的选择信号，具体如图 3-20 所示。

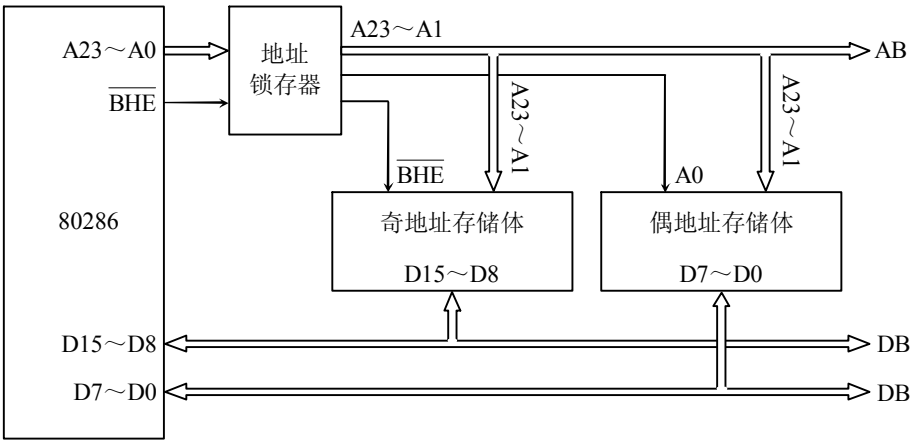


图 3-20　80286 存储器结构

4．80386/80486 CPU 与内存储器的连接

80386、80486 CPU 是 32 位的微处理器，有 32 位数据总线和 32 位地址总线。为了实现对一个 32 位数据中的 8 位、16 位或 32 位的灵活访问，系统将 32 位数据以 8 位一组，分成 4 个字节，同时 CPU 设有 4 个引脚 $\overline{BE3}$ ～ $\overline{BE0}$，以控制对不同字节数据的访问。 $\overline{BE3}$ ～ $\overline{BE0}$ 由 CPU 根据指令类型产生， $\overline{BE3}$ ～ $\overline{BE0}$ 的功能如表 3-9 所示。

80386、80486 系统在进行存储系统设计时，常把内存分为 4 个存储体，依次存放 32 位数据的 4 个字节，每个存储体的 8 位数据线依次并行连接到系统数据总线 D31～D24、D23～D16、D15～D8、D7～D0 上，如图 3-21 所示。CPU 的 32 位地址总线中，高位地址 A31～A2 直接输出，低 2 位地址 A1、A0 经由内部编码产生 $\overline{BE3}$ ～ $\overline{BE0}$，以选择不同字节。

表 3-9 $\overline{BE3}$～$\overline{BE0}$ 功能表

字节允许				允许访问的数据位			
$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	D31～D24	D23～D16	D15～D8	D7～D0
1	1	1	0	——	——	——	D7～D0
1	1	0	1	——	——	D15～D8	——
1	0	1	1	——	D23～D16	——	——
0	1	1	1	D31～D24	——	——	——
1	1	0	0	——	——	D15～D8	D7～D0
1	0	0	1	——	D23～D16	D15～D8	——
0	0	1	1	D31～D24	D23～D16	——	——
1	0	0	0	——	D23～D16	D15～D8	D7～D0
0	0	0	1	D31～D24	D23～D16	D15～D8	——
0	0	0	0	D31～D24	D23～D16	D15～D8	D7～D0

尽管 CPU 具有 4GB（2^{32}=4G）的寻址能力，但实际应用中无需那么大的内存容量。若每个存储体的容量为 256KB，则 4 个存储体构成的内存储器容量为 1MB，所以至少需要 20 位（2^{20}=1M）地址进行内存单元寻址，如图 3-21 所示，每个存储体的 18 位（256K=2^{18}）地址 A17～A0 接至 CPU 的系统地址线 A19～A2，片选信号 $\overline{CE}$ 由高位地址的译码结果与 $\overline{BE3}$～$\overline{BE0}$ 相“或”后产生。

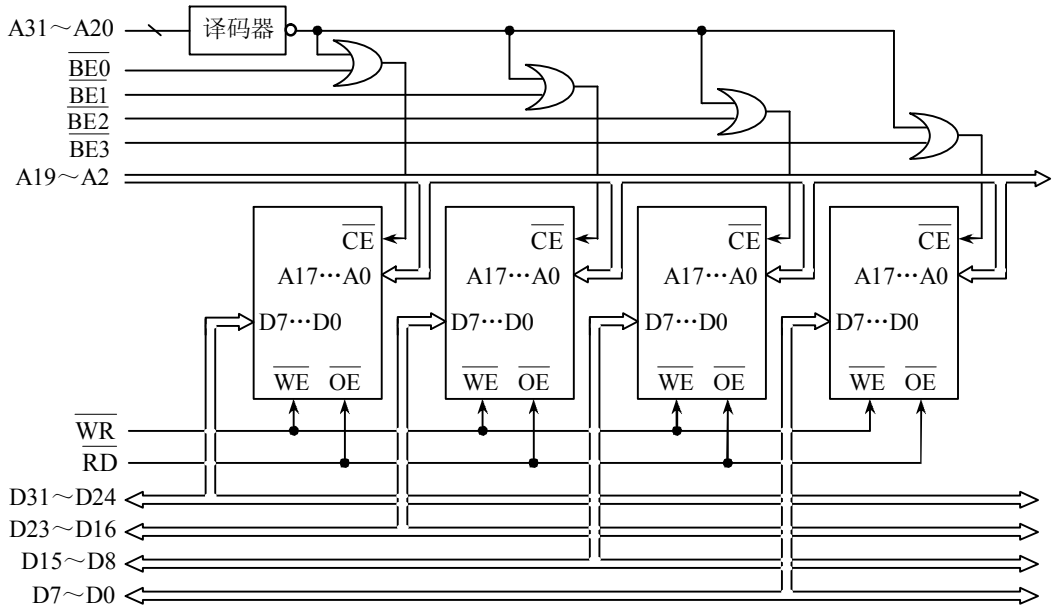


图 3-21 80386/80486 存储器结构

一旦地址码确定，$\overline{BE3}$～$\overline{BE0}$ 决定某个或某几个存储体被选中，A19～A2 确定被选中的存储体内指定的字节单元，然后即可对选中单元进行读/写操作。

3.5 存储体系结构

存储器有三个主要性能指标：容量、速度和价格（每位价格），计算机对存储器性能指标的基本要求是容量大、速度快和成本低。但是要想在一个存储器中同时兼顾这些指标是很困难的，有时甚至是相互矛盾的。速度越快，每位价格就越高；容量越大，每位价格就越低；容量越大，速度就越慢。为了解决存储器的容量、速度和价格之间的矛盾，人们在不断地研制新的存储器件和改进存储性能的同时，还从存储系统体系上不断研究合理的结构模式。

3.5.1 存储系统的层次结构

人们把各种不同存储容量、存取速度和价格的存储器按层次结构组织起来，并通过管理软件和辅助硬件有机地组成统一的整体，使所存放的程序和数据按层次分布在各级存储器中，形成存储系统的多级层次结构。一般计算机存储系统的多级层次结构如图 3-22 所示。

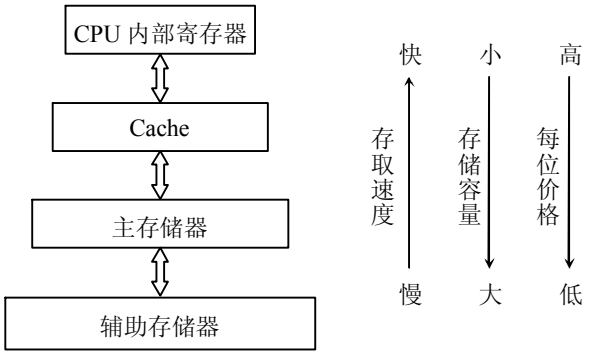


图 3-22 存储系统的多级层次结构

由图 3-22 可见，计算机存储系统的多级层次结构主要由 CPU 内部寄存器、高速缓冲存储器 Cache、主存储器和辅助存储器组成。辅存、主存、Cache 以及 CPU 内部的寄存器都具有数据存储功能，由它们构成的存储器组织能够充分发挥存储速度快、容量大、价格低的特点。

CPU 内部寄存器可读可写，并与 CPU 速度相匹配。它们的功能很强，但是每位价格较高、数量也不能太多。主要用于 CPU 执行指令过程中指令和数据的暂存。

主存的容量要比 CPU 内部寄存器的容量大得多，可作为 CPU 内部寄存器的后备支持。主存是计算机工作时必不可少的存储部件，它和 CPU 一起构成主机的骨干部件。由于主存的每位成本较高，数量也不能很大，并且在断电后 RAM 中的数据全部丢失，所以必须有更大容量的非易失性存储后援的支持。

辅存由硬盘以及光盘、U 盘等可移动式存储器构成。相对于主存，它具有存储容量大、成本低、断电后信息不丢失等优点，所以辅存作为海量存储器，主要用于计算机的大容量程序和数据的长久保存。但是它也有如读写速度慢等缺点。

随着计算机技术的发展，CPU 的速度不断提高，远远超过了以 DRAM 为基础的主存储器，为了不影响 CPU 的效率，人们在微型计算机的 CPU 与主存之间使用了高速缓冲存储器 Cache，Cache 以 SRAM 为存储介质，存取速度接近 CPU 的工作速度，作为主存储器的副本，可直接

向 CPU 提供程序和数据，从而进一步提高向 CPU 提供数据的速度。

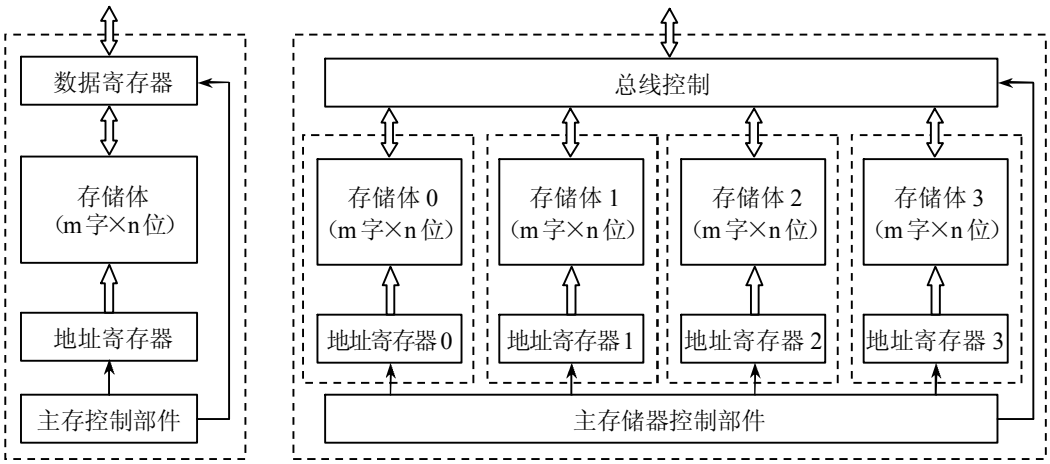
如图 3-22 所示，在存储系统多级层次结构中，由上而下，其容量逐渐增大，速度逐级降低，成本则逐次减少。这样构成的存储系统可以接近 CPU 的高速度，并获得大容量辅存的支持，价格也能为用户所承受。如此，多级层次结构的计算机存储系统有效地解决了存储器速度、容量和价格之间的矛盾。

整个结构又可以看成两个层次：主存－辅存层次、Cache－主存层次。这两个层次系统中的每种存储器都不再是孤立的存储器，而是一个有机的整体。它们在计算机操作系统和辅助硬件的管理下工作，可把主存－辅存层次作为一个存储整体，形成的可寻址空间远远大于主存空间，而且由于辅存容量大、价格低，使得存储系统的整体平均价格降低。另外，由于 Cache 的存取速度和 CPU 的工作速度相当，故 Cache－主存层次可以缩小主存与 CPU 之间的速度差距，从整体上提高存储系统的存取速度。

3.5.2 多体存储结构

CPU 需要频繁访问主存储器，但是主存的存取速度跟不上系统需求，这是影响整个计算机系统性能的重要问题。解决这个问题最直接的办法就是采用多个存储器并行访问和交叉访问，从而使 CPU 在一个存储周期内可以同时访问多个数据。

例如，一般主存储器在一个存储周期内只能访问到一个存储字，如图 3-23（a）所示，把多个这样的存储器组合后可构成多体存储结构，如图 3-23（b）所示。



（a）一般存储器　　（b）多体存储器

图 3-23　多体存储器与一般存储器比较

对多体存储器交叉访问通常有两种工作方式：地址码高位交叉、地址码低位交叉。

1. 高位交叉访问存储器

以每个存储体内有 4 个存储单元、4 个这样的存储体构成的存储器为例，地址码高位交叉访问存储器如图 3-24 所示。

地址码的低位部分直接送至各存储体，作为各存储体的体内地址；高位地址部分送至译码器，用来区分存储体的体号。高位交叉访问存储器要求每个存储模块都有各自独立的存储体和控制部件，控制部件包括地址寄存器、数据寄存器、驱动放大电路、地址译码电路和读写控

制电路等，每个独立存储模块均可独立工作。因此高位交叉访问存储器具备了并行工作的条件，这种存储器不仅提高了存取速度，更主要的目的是用来扩大存储器容量。

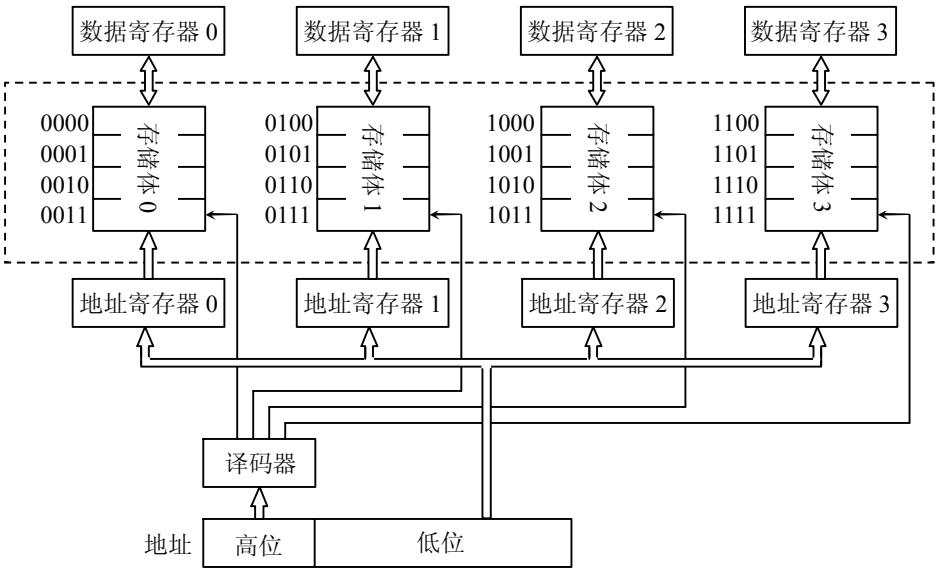


图 3-24　高位交叉访问存储器结构示例

目前，大部分计算机系统中的主存储器都采用模块结构（内存条），用户可以根据自己的需要随时改变主存储器的容量，例如用 2 个 2GB 的内存条可以构成一个 4GB 的主存储器。

2. 低位交叉访问存储器

以每个存储体内有 4 个存储单元、4 个这样的存储体构成的存储器为例，地址码低位交叉访问存储器如图 3-25 所示。

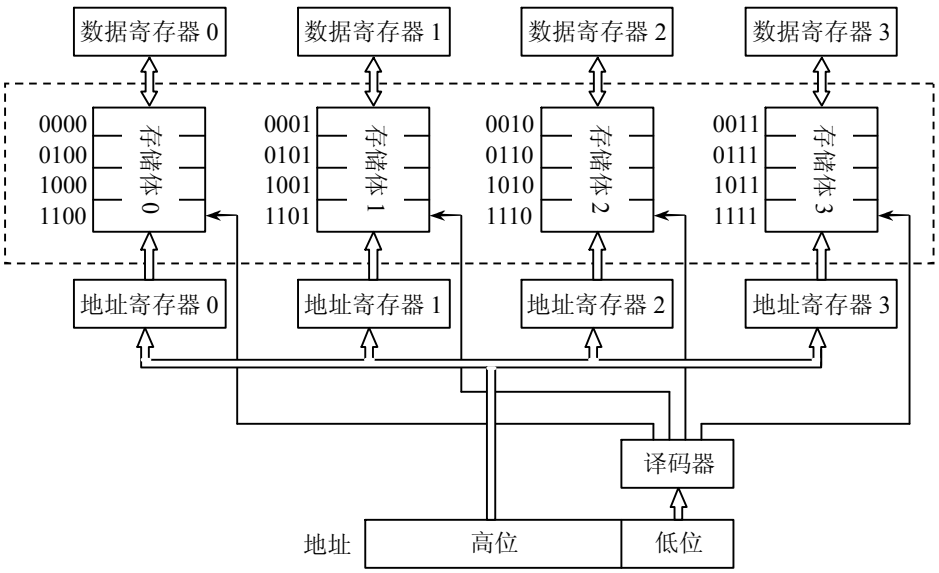


图 3-25　低位交叉访问存储器结构示例

低位交叉访问存储器的地址码的使用方法与高位交叉访问方式恰恰相反，其低位地址送至译码器，用来区分存储体的体号；高位地址直接送存储体，作为各个存储体的体内地址。

低位交叉访问存储器的主要目的是提高存储器的访问速度。当然，在提高访问速度的同时，由于增加了存储器模块的数目，也就增加了存储器的容量。

3.5.3 高速缓冲存储器

高速缓冲存储器（Cache）是位于 CPU 和主存之间的一种存储器，容量比主存小，但速度比主存快。它用来存放 CPU 频繁使用的指令和数据。由于使用 Cache 后可以减少对慢速主存的访问次数，缓解了 CPU 与主存之间的速度差异，所以提高了 CPU 的工作效率。目前，在微型计算机中广泛使用高速缓冲存储器技术。

1. Cache 工作原理

在半导体存储器中，只有 SRAM 的存取速度与 CPU 的工作速度接近，但 SRAM 的功耗大、价格较贵、集成度低，而 DRAM 的功耗小、成本低、集成度高，但是存取速度却跟不上 CPU 的工作速度。于是产生了一种折衷的分级处理办法：在以 DRAM 为基础的大容量主存与高速 CPU 之间增加一个容量较小的 SRAM 作为 Cache，用于保存那些使用频率较高的主存副本。CPU 需要数据时，首先在 Cache 中查找，Cache 中没有才从主存中读取。

由于程序执行和数据访问具有局域性，通过指令预测和数据预取技术，可以尽可能将 CPU 需要的指令和数据预先从主存中读出，存放在 Cache 中，据统计，CPU90%以上的存储器访问都发生在 Cache 中，只有不到 10%的几率需要访问主存，即命中率可达 90%以上，因此少量 Cache 可以极大地提高存储系统的访问速度，缓解由于主存存取速度比 CPU 工作速度慢而产生的性能瓶颈问题，进而提高系统性能。

Cache－主存结构存储系统一般由高速缓冲存储器 Cache、主存储器和 Cache 控制器组成，如图 3-26 所示。

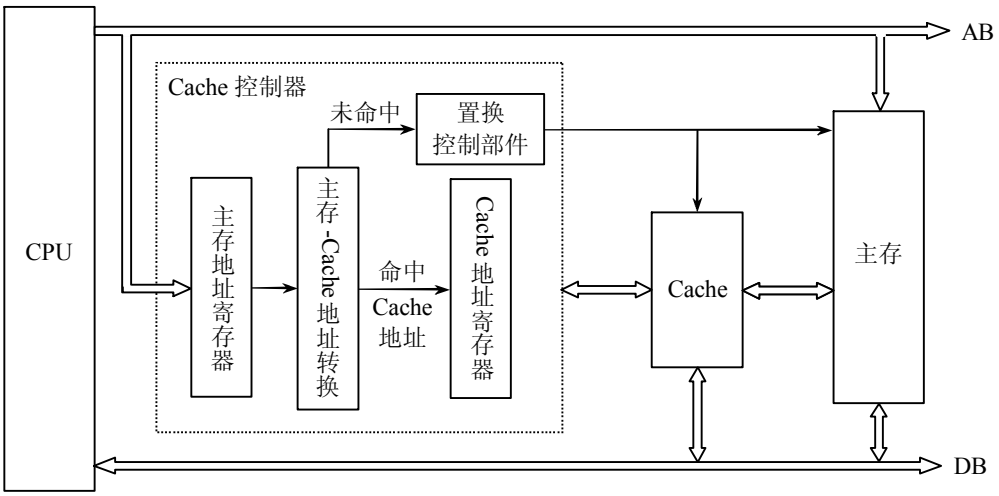


图 3-26 Cache 工作原理图

在高速缓冲存储系统中，所有数据和指令都存放在主存中，而使用频率较高的数据和指令则复制到 Cache 中。Cache 的内容是在读写过程中逐步调入的，是主存中部分内容的副本。

主存和 Cache 的存储区都划分成多个块，每块由多个单元组成，主存与 Cache 之间以块为单位交换信息。

信息块调往 Cache 时的存放地址与它在主存时的地址不一致，但二者之间有一定的对应关系，这种对应关系称为地址映射函数。将主存地址变换成 Cache 地址的过程是通过 Cache 控制器内的地址转换机构按照所采用的地址映射函数自动完成的。

当 CPU 访问存储器时，首先检查 Cache，如果要访问的信息已在 Cache 中，CPU 就能很快地访问它，这种情况称为“命中”，否则称为“未命中”。若未命中，CPU 必须从主存中提取信息，同时还要把含有这个信息的整个数据块从主存中取出送到 Cache 中。CPU 访问 Cache 时，找到所需要信息的百分比称为命中率，命中率是 Cache 的一个重要指标，命中率越高，在 Cache 中找到所需指令和数据的可能性就越大，CPU 去访问慢速主存的次数就越少。

Cache 的全部功能由硬件实现，并且对程序员来说是“透明”的，就像不存在一样，程序员不需要明确知道 Cache 及 Cache 控制器的存在。

2. 替换算法

当新的主存块需要调入 Cache，而它的可用位置已被占用时，就需要替换算法来解决。一个好的替换算法既要考虑提高访问 Cache 的命中率，又要考虑容易实现替换，从而减少 CPU 访问主存的机率。替换算法有多种，通常采用以下两种算法：

（1）先进先出法（First Input First Output，FIFO）。这种算法是把最早调入 Cache 的信息块替换掉。为了实现这种算法，需要在地址变换表中设置一个历史位，每当有一个新块调入 Cache 时，就将已进入 Cache 的所有信息块的历史位加 1。需要进行替换时，只要将历史位值最大的信息块替换掉即可。这种算法比较简单，容易实现，但不一定合理，因为有些信息块虽然调入较早，但可能仍需使用。

（2）最近最少使用法（Least Recently Used，LRU）。LRU 算法是把近期使用最少的信息块替换掉。这种算法利用了程序访问的局部性原理：在时间上，程序即将用到的信息很可能就是正在使用的信息；在空间上，程序即将用到的信息很可能与当前正在使用的信息临近。所以最近使用的信息块应尽量保留在 Cache 中。

LRU 算法要求随时记录 Cache 中各信息块的使用情况。这样，要为每个信息块设置一个计数器，以便确定哪个信息块是近期最少使用的。LRU 算法还可用堆栈来实现，也称为堆栈型算法。当堆栈已满，却又有一个信息块需要调入 Cache 时，首先检查堆栈中是否已经有该信息块。若有，则将该信息块从堆栈中取出并压入堆栈的栈顶；如果没有，则将该信息块直接压入栈顶，于是原来在栈底的信息块就成为被替换的信息块而被压出堆栈，这样就能够保证任何时候栈顶的信息块都是刚被访问过的信息块，而栈底的信息块总是最久没有被访问过的块。这种算法与 FIFO 相比可以获得较高的命中率。

3. 多层次 Cache

（1）多层次 Cache 结构。

现在微型计算机中的高速缓冲存储器采用三级，分别称为一级缓存（L1 Cache）、二级缓存（L2 Cache）、三级缓存（L3 Cache），都集成在 CPU 内部，两级 Cache 之间有专用总线相连。

超高速 L1 Cache 的存取速度与 CPU 的工作速度相匹配，L1 Cache 又分为数据 Cache 和指令 Cache，但容量较小，一般为几 KB～几十 KB；过去 L2 Cache 放在 CPU 外，随着制造

工艺的提高，Pentium Ⅱ以后 L2 Cache 被集成在 CPU 内部，但不区分指令和数据，容量一般为几百 KB～几 MB；现代高端微型计算机的 CPU 中还集成有 L3 Cache，容量为几 MB～几十 MB。

由 L1 Cache 到 L3 Cache，存储容量逐级增大，存取速度逐级降低。L1 Cache 与 L2 Cache 之间、L2 Cache 与 L3 Cache 之间、以及 L3 Cache 与主存之间的映射、替换算法和读写操作全部由辅助硬件来完成，从而实现了 Cache 的高速处理功能。

（2）指令 Cache 和数据 Cache。

80486 之后，Pentium 开始在 CPU 内部将 Cache 分为指令 Cache 和数据 Cache 两部分，CPU 可以同时访问指令 Cache 和数据 Cache，从而减少了指令与数据存取的冲突。

例如，Pentium 内有 8KB 数据 Cache 和 8KB 指令 Cache，原生四核的 Intel Core i7 9××微处理器内，每核有 32KB 数据 L1 Cache 和 32KB 指令 L1 Cache。

4. Cache 一致性问题

当要访问的信息在 Cache 时，若是读操作，则 CPU 可以直接从 Cache 中读取数据，不涉及主存；若是写操作，则 Cache 和主存中相应两个单元的内容都需要改变。这时有两种处理办法：一种是 Cache 单元和主存中相应单元同时被修改，称为“直通存储法”；另一种是只修改 Cache 单元的内容，同时用一个标志位作为标志，当有标志的信息块从 Cache 中移去时，再修改相应的主存单元，把修改信息一次性写回主存。显然“直通存储法”比较简单，但对于需要多次修改的单元来说，可能导致不必要的主存重复写工作。

3.5.4 虚拟存储器

虚拟存储器（Virtual Memory）是在“主存－辅存”层次结构上进一步发展和完善的存储管理技术，是现代操作系统的重要特征之一。

虚拟存储器把主存和辅存视为一个统一的虚拟主存，提供比实际主存容量大得多的、可使编程空间不受限制的虚存空间；在程序中使用虚地址，使程序不必作任何修改，即可用接近主存的速度在这个虚拟存储器上运行。使得计算机系统好像只有一个大容量、高速度、使用方便的存储器，而没有主存、辅存之分。

1. 虚拟存储器工作原理

虚拟存储系统对主存和辅存构成的虚存空间重新进行统一编址，称为虚地址。虚存空间并不是主存容量加上辅存容量，而是一个比主存大得多的虚拟空间，它与主存和辅存空间的容量无关；虚地址并非主存的实际物理地址，例如 80386/80486 的虚地址码长为 46 位，虚存空间可达 64TB（2^{46}），这是任何计算机系统中主存储器所不可能达到的容量，而主存实际物理地址只有 32 位。

计算机程序员可以按虚地址空间来编制程序。在程序运行时，只把虚地址空间的一小部分映射到主存，其余大部分虚地址空间则映射到辅存（如大容量硬盘）上。当按虚地址访问虚拟存储器时，存储器管理部件首先查看该虚地址所对应的内容是否已在主存中，若已在主存中（命中），就将该虚地址自动转换为主存实际物理地址，对主存进行访问；若不在主存中（未命中），就将虚拟存储器中待访问的程序或数据由辅存调入主存，再进行访问。故每次访问虚拟存储器时，都必须进行虚地址向实地址的转换。

虚拟存储器是通过存储器管理部件和操作系统自动管理和调度的，主存和辅存的地位和

性质并未改变，但虚拟存储器相对于每个用户来说是透明的，这样大大方便了用户。

2. 虚拟存储器与 Cache 的对比

“主存－辅存”层次与“Cache－主存”层次从原理上看是相同的，因而它们有很多相似之处，如它们都采用地址变换及映射方法和替换策略，对于程序员来说，Cache 和虚拟存储器都是透明的。但虚拟存储器和 Cache 仍有很明显的区别：

（1）Cache 用于弥补主存与 CPU 的速度差距，而虚拟存储器则用来弥补主存容量较小的缺憾。

（2）CPU 可以直接访问 Cache，但 CPU 却不能直接访问辅存。

（3）Cache 每次传送的信息块是定长的，且信息块短小，只有几个或几十个字节；而虚拟存储器每次调动的信息块较大，可达几百或几千个字节，可以分页（信息块定长）或分段（信息块可变长）调动。

（4）Cache 操作全部由辅助硬件实现，而虚拟存储器则由辅助硬件（存储器管理部件）和辅助软件（操作系统的存储管理软件）综合管理。

3. 虚拟存储器的分类

在实际应用中，根据如何对主存空间与磁盘空间进行分区管理、虚实地址怎样转换、采取何种替换算法等，可以有页式、段式和段页式三类虚拟存储器。

（1）页式虚拟存储器。页式虚拟存储器是以“页”为信息传送单位的虚拟存储器。在页式虚拟存储系统中，将虚拟空间等分成固定大小的页，页面大小随机器而异，一般计算机系统中一页的大小为 1KB～16KB。虚存空间中所划分的页称为“虚页”，主存空间也等分成同样大小的页，称为“实页”。页面都是由 0 开始顺序编号的，分别称为虚页号和实页号。

虚地址分成虚页号和页内地址两部分：虚页号占用高位部分，低位部分则为页内地址。主存的实际物理地址也由两部分组成：实页号和页内地址，实页号占用高位部分，低位部分为页内地址。页内地址的长度取决于页面大小，实页号的长度取决于主存的容量。

信息是以页为单位由虚页向主存中的实页调入的。因为虚页和实页大小相等，所以调入时只要页边界对齐，页内地址无需修改就可以直接使用。这样的话，虚－实地址的转换主要是虚页号向实页号的转换，这个转换是由“页表”实现的。

每道程序都有一个独立的虚存空间，在程序运行时，存储管理软件根据主存的运行情况自动为每道程序建立一张独立的“页表”。在页表中，对应每个虚页号有一行表目，每个表目记录着虚页号对应到实页号的一些信息。所有页表都存放于主存的特定区域——页表区。当一个虚页号向实页号转换时，首先找到对应该程序的页表在页表区的首地址，然后在页表中按虚页号顺序找到该虚页所对应的表目，找到表目就可以实现虚地址向实地址的转换。

页式虚拟存储器的页表简单，地址变换速度较快，页面长度较小，主存空间的利用率高，对辅存的管理比较容易，因而得到了广泛应用。但是页式虚拟存储器的页面大小固定，无法反映程序内部的逻辑结构，不便于程序的执行、保护和共享。

（2）段式虚拟存储器。一个复杂的大程序总可以分解成多个在逻辑上相对独立的模块（程序段），每个模块的大小可以各不相同。段式虚拟存储器就是以程序的逻辑结构所自然形成的模块作为主存分配单位进行存储器管理的。其中每个模块（段）的长度可以不同，也可以独立编址，有的段还可以在执行时动态决定大小。

程序运行时，以段为单位整段从辅存调入主存，一段占用一个连续的存储空间。CPU 访

问时仍需要进行虚－实地址的转换，段式虚拟存储器与页式虚拟存储器技术十分相似，每道程序都有一个“段表”，存放该程序中各段装入主存的状态信息，每个段的信息在段表中占一行，主要包括段号、段起点、段长度等，根据段表可进行虚地址到实地址的转换，从而确定其在主存中的位置。

段式虚拟存储器有效配合了模块化程序设计，各段之间相互独立，程序按逻辑结构分段，便于程序和数据的共享，段的大小、位置可变，模块可独立编址，可提高按段调度的命中率。但是段式虚拟存储器的地址变换所花费的时间较长，主存空间的利用率往往比较低，对辅存的管理比较困难。

（3）段页式虚拟存储器。为了能够同时获得页式虚拟存储器在管理主存和辅存空间方面的优点和段式虚拟存储器在程序模块化方面的优点，把二者结合起来就形成了段页式虚拟存储器，即存储空间仍按程序的逻辑模块分段，每段又等分为若干个页，页面大小与实存页面相同。

虚地址包括段号、页号和页内地址三部分，实地址则只有页号和页内地址。虚存与实存之间的信息调度以页为基本单位。每个程序有一张段表，每段对应有一张页表，CPU 访存时，由段表指出每段对应页表的起始地址，而每一段的页表可指出该段的虚页号对应在实存空间的实页号，最后与页内地址拼接即可确定 CPU 要访问信息的实存地址。

段页式虚拟存储器的特点表现为分两级查表实现虚实地址转换，以页为单位调进或调出主存，按段来共享与保护程序和数据。

习题与思考

3.1 在某微型计算机中，地址总线为20位，可寻址的内存空间应该是（　　）。

A．640KB　　B．16MB

C．1MB　　D．1GB

3.2 在微型计算机中，主存储器以（　　）方式进行编址。

A．以字节为单位　　B．以字为单位

C．以数据块为单位　　D．以磁道、扇区为单位

3.3 下列关于微型计算机存储系统的说法错误的是（　　）。

A．内存的存取速度高于外存

B．外存的存储容量远大于内存

C．内存和外存都具有非易失性

D．DRAM一般用作微型计算机内存，SRAM用作高速缓冲存储器

3.4 内存某一单元中存储着数据60H，CPU对其读操作后，该单元的内容是（　　）。

A．00H　　B．FFH

C．60H　　D．不确定

3.5 某微型计算机的内存容量为1MB，按字节编址，它的寻址范围是（　　）。

A．00H～FFH　　B．000H～FFFH

C．0000H～FFFFH　　D．00000H～FFFFFH

3.6　下列SRAM芯片各需要多少位地址输入线，多少位数据信号线？

A．512K×4位　　B．1K×8位　　C．2K×1位

D．16K×8位　　E．64K×1位

3.7　对下列存储器组成方案，各需要（　　）个芯片。

A．用64K×1位芯片组成64K×8位的存储器

B．用1K×4位的芯片组成4K×8位的存储器

C．用2K×1位的芯片组成32K×8位的存储器

3.8　已知一个具有 16 位地址和 8 位数据的存储器，回答下列问题：

（1）该存储器能存储多少字节的信息？

（2）如果存储器由16K×4位RAM芯片组成，需要多少片？

（3）至少需要多少位地址作芯片选择？

3.9　存储器有哪些主要性能指标？

3.10　简述半导体存储器的分类。

3.11　半导体存储器 SRAM、DRAM、ROM、PROM、EPROM、EEPROM 和闪速存储器各有何读写特点？一般应用于何种场合？

3.12　DRAM 为什么必须定期刷新？Intel 2164 芯片中的 $\overline{RAS}$ 和 $\overline{CAS}$ 信号有何作用？

3.13　在某一微型计算机系统中，CPU有20位地址线和8位数据线，主存储器扩展了两片Intel 6116（2K×8位），它们的硬件连接如图3-27所示，试分析这两片6116的寻址范围。

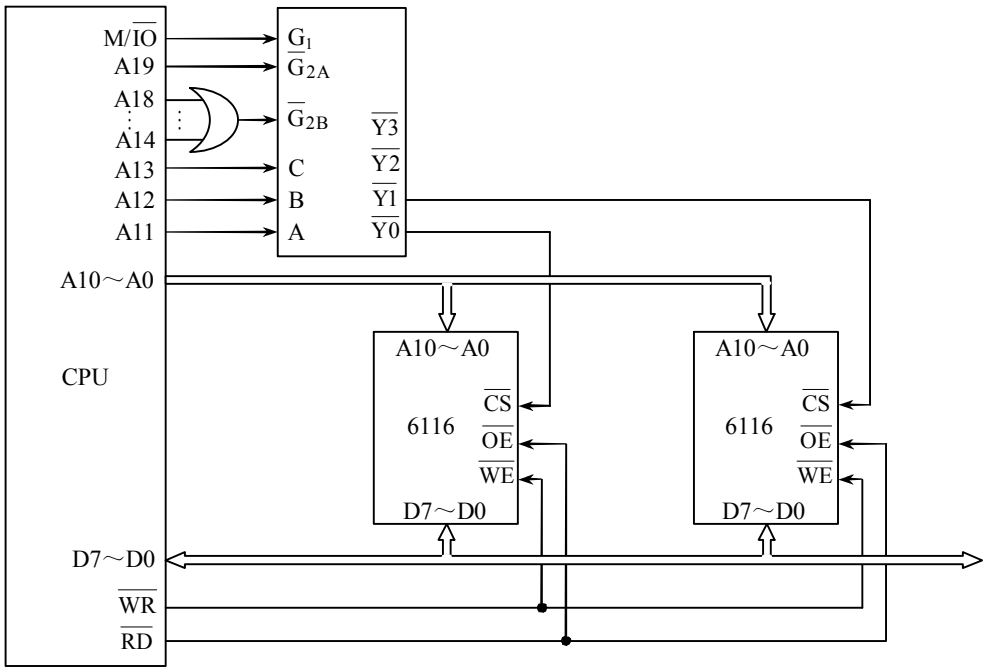


图 3-27　存储器容量扩展实例

3.14　微型计算机中某存储器芯片的首单元地址为2000H，末单元地址为9FFFH，试分析该存储器芯片的容量是多少？

3.15　在微型计算机的存储器中，某 RAM 芯片容量为 4KB，它的首单元地址为 4000H，那么最后一个单元的地址是多少？

3.16　某微型计算机有20位地址线、8位数据线，现用两片Intel 2114（1K×4位）组成2K×8位的存储器，并将它们的起始地址设置为04800H，试画出硬件连接图。

3.17　为什么微型计算机存储系统要采用层次结构？

3.18　在微型计算机系统中，为什么要使用 Cache？为什么要使用虚拟存储器？