第3章 存储器技术

古典的冯·诺依曼结构计算机是以运算器为中心的，而现代计算机系统都是以存储器为中心的，可以看出存储器的重要性。

存储器是计算机中用来存储信息的记忆部件，在运行程序时，CPU 自动连续地从存储器中取出指令并执行指令规定的操作，计算机每完成一条指令，至少要执行一次访问存储器的操作，并把处理结果存储在存储器中。因此，存储器是微机系统不可缺少的组成部分，是计算机中各种信息的存储和交换中心。在微机系统中，存储器由 3 个层次组成，即辅助存储器（外存）、主存储器（内存）和高速缓冲存储器（高缓）。高缓相对速度最快、一般容量最小。高缓的引入较好地解决了存储器与 CPU 在速度上的协调性。存储器中除采用磁、光原理的辅助存储器外，其他存储器主要都是采用半导体存储器。

本章重点介绍半导体存储器及其使用的方法。

3.1 存储器概述

存储器是计算机系统中的记忆设备，用来存放程序和数据。存储器中最小的存储单位就是存储元，它可存储一位二进制代码。由若干个存储元组成一个存储单元，再由许多存储单元组成一个存储器。

3.1.1 存储器的分类

存储器按存储介质分为半导体存储器、磁表面存储器、光表面存储器等；按信息的可保存性分为非永久性记忆存储器（断电后信息消失）、永久性记忆存储器（断电后信息仍保存）；按存储方式分为随机存储器（任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关）、顺序存储器（只能按某种顺序来存取，存取时间和存储单元的物理位置有关）；按在计算机系统中的作用分为主存储器（内存）、辅助存储器（外存）、高速缓冲存储器。

半导体存储器按读写方式分为只读存储器 ROM 和可读写存储器 RAM。

3.1.1.1 可读写存储器 RAM

读写存储器 RAM 按其制造工艺又可以分为双极型 RAM 和金属氧化物 RAM。

1. 双极型 RAM

双极型 RAM 的主要特点是存取时间短，通常为几到几十纳秒(ns)。与下面提到的 MOS 型 RAM 相比，其集成度低、功耗大，而且价格也较高。因此，双极型 RAM 主要用于要求存取时间短的微型计算机中。

2. 金属氧化物(MOS)RAM

用 MOS 器件构成的 RAM 又分为静态读写存储器(SRAM)和动态读写存储器(DRAM)。

(1) 静态 RAM(SRAM)。

静态 RAM 的基本存储单元是 MOS 双稳态触发器。一个触发器可以存储一个二进制信息。静态 RAM 的主要特点是，存取时间为几十到几百纳秒(ns)，集成度比较高。目前经常使用的静态存储器每片的容量为几到几十 KB。SRAM 的功耗比双极型 RAM 低，价格也比较便宜。

(2) 动态 RAM(DRAM)。

动态 RAM 的存取速度与 SRAM 的存取速度差不多。其最大的特点是集成度特别高。DRAM 的功耗比 SRAM 低，价格也比 SRAM 便宜。DRAM 在使用中需特别注意的是，它是靠芯片内部的电容来存储信息的。由于存储在电容上的信息总是要泄露的，所以，每隔 2～4ms，DRAM 要求对其存储的信息刷新一次。

(3) 集成 RAM(i RAM)。

集成 RAM(Integrated RAM，i RAM)是一种带刷新逻辑电路的 DRAM。由于它自带刷新逻辑，因而简化了与微处理器的连接电路，使用它和使用 SRAM 一样方便。

(4) 非易失性 RAM(NVRAM) 。

非易失性 RAM(Non-Volatile RAM，NVRAM)的存储体由 SRAM 和 EEPROM 两部分组合而成。正常读写时，SRAM 工作；当要保存信息时(如电源掉电)，控制电路将 SRAM 的内容复制到 EEPROM 中保存。存入 EEPROM 中的信息又能够恢复到 SRAM 中。NVRAM 既能随机存取，又具有非易失性，适合用于需要掉电保护的场合。

3.1.1.2 只读存储器 ROM

ROM 的特点是把信息写入存储器以后，能长期保存，不会因电源断电而丢失信息。计算机在运行过程中，只能读出只读存储器中的信息，不能再写入信息。一般地，只读存储器用来存放固定的程序和数据，如微机的监控程序、汇编程序、用户程序、数据表格等。根据编程方式的不同，ROM 共分为以下 5 种。

1. 掩模工艺 ROM

这种 ROM 是芯片制造厂根据 ROM 要存储的信息，设计固定的半导体掩模板进行生产的。一旦制出成品之后，其存储的信息即可读出使用，但不能改变。这种 ROM 常用于批量生产，生产成本比较低。微型机中一些固定不变的程序或数据常采用这种 ROM 来存储。

2. 可一次性编程 ROM(PROM)

为了使用户能够根据自己的需要来写 ROM，厂家生产了一种 PROM，允许用户对其进行一次编程——写入数据或程序。一旦编程之后，信息就永久性地固定下来。用户可以读出和使用，但再也无法改变其内容。

3. 紫外线擦除可改写 ROM(EPROM)

可改写 ROM 芯片的内容也由用户写入,但允许反复擦除重新写入。EPROM 是用电信号编程而用紫外线擦除的只读存储器芯片。在芯片外壳上方的中央有一个圆形窗口,通过这个窗口照射紫外线就可以擦除原有的信息。由于阳光中有紫外线的成分,所以程序写好后要用不透明的标签封住窗口,以避免因阳光照射而破坏程序。EPROM 的典型芯片是 Intel 公司的 27 系列产品,按存储容量不同有多种型号,例如 2716(2KB×8)、2732(4KB×8)、2764(8KB×8)、27128(16KB×8)、27256(32KB×8)等,型号名称后的数字表示其存储容量。

4. 电擦除可改写 ROM(EEPROM 或 E^2PROM)

这是一种用电信号编程也用电信号擦除的 ROM 芯片,它可以通过读写操作进行逐个存储单元读出和写入,且读写操作与 RAM 存储器几乎没有什么差别,所不同的只是写入速度慢一些,但断电后却能保存信息。典型的 EEPROM 芯片有 28C16、28C17、2817A 等。

5. 快擦写 ROM(flash ROM)

EEPROM 虽然具有既可读又可写的特点,但写入的速度较慢,使用起来不太方便。而 flash ROM 是在 EPROM 和 EEPROM 的基础上发展起来的一种只读存储器,读写速度都很快,存取时间可达 70ns,存储容量可达 16～128MB。这种芯片可改写次数可从 1 万次到 100 万次。典型的 flash ROM 芯片有 28F256、28F516 等。

3.1.2 存储器的主要性能参数

(1) 存储容量:是指存储器可以存储的二进制信息量。

表示方法为:存储容量=存储单元数×每单元二进制位数。

不同的存储器芯片,其容量不一样。通常用某一芯片有多少个存储单元,每个存储单元存储若干位来表示。例如,静态 RAM 6264 的容量为 8K×8 位,即它有 8K 个单元(1K=1024),每个单元存储 8 位(一个字节)数据。

(2) 存取时间和存取周期:说明存储器的工作速度。

存取时间:从存储器接收到寻址地址开始,到完成取出或存入数据为止所需的时间。

存取周期:连续两次独立的存储器存取操作所需的最小时间间隔;略大于存取时间。

存取时间即存取芯片中某一个单元的数据所需要的时间。在计算机工作时,CPU 在读写 RAM 时,它所提供的读写时间必须比 RAM 芯片所需要的存取时间长。如果不能满足这一点,微型机则无法正常工作。

(3) 可靠性:指存储器对电磁场及温度等的变化的抗干扰能力。

微型计算机要正确地运行,必然要求存储器系统具有很高的可靠性。内存的任何错误都足以使计算机无法工作,而存储器的可靠性直接与构成它的芯片有关。目前所用的半导体存储器芯片的平均故障间隔时间(MTBF)是(5×10^6～1×10^8)小时。

(4) 功耗。

使用功耗低的存储器芯片构成存储器系统,不仅可以减少对电源容量的要求,而且还可以提高存储系统的可靠性。

其他指标:体积、工作温度范围、成本等。

3.2 存储器的连接

在CPU对存储器进行读写操作时，首先在地址总线上给出地址信号，然后发出相应的读写控制信号，最后才能在数据总线上进行数据交换，所以CPU与存储器的连接包括地址线、数据线和控制线的连接3部分。在连接时要考虑以下几个问题。

1. CPU总线的负载能力

一般来说，CPU总线的直流负载能力可带一个TTL负载，目前存储器基本上是MOS电路，直流负载很小，主要负载是电容负载。因此在小型系统中，CPU可以直接和存储器芯片相连，在较大的系统中，考虑CPU驱动能力，必要时要加上数据缓冲器（例如74LS145）或总线驱动器来驱动存储器负载。

2. CPU的时序和存储器存取速度之间的配合

CPU在取指令和读写操作数时，有它固有的时序，应考虑选择何种存储器来与CPU时序配合。若存储器芯片已经确定，应考虑如何实现 T_W 周期的插入。

3. 存储器的地址分配和片选

内存分为ROM区和RAM区，RAM区又分为系统区和用户区，每个芯片的片内地址由CPU的低位地址来选择。若一个存储器系统由多片芯片组成，则片选信号由CPU的高位地址译码后取得。应考虑采用何种译码方式，以实现存储器的芯片选择。

4. 控制信号的选择

8086 CPU与存储器交换信息时，提供了以下几个控制信号：$M/\overline{IO}$、$\overline{RD}$、$\overline{WR}$、ALE、READY、WAIT、$DT/\overline{R}$和$\overline{DEN}$，这些信号与存储器要求的控制信号要正确连接才能实现所需要的控制功能。

3.2.1 存储器的地址连接

一个存储器系统通常由许多存储器芯片组成，对存储器的寻址必须有两个部分，通常是将低位地址线连到所有的存储器芯片，实现片内寻址，将高位地址线通过译码器或线性组合后输出作为芯片的片选信号，实现片间寻址。由地址线的连接决定存储器的地址分配，下面分别叙述3种存储器的地址选择方法。

1. 线性选择方式

无论是ROM或是RAM芯片，芯片引脚都包括地址线和数据线、读写控制线和片选线，只有片选线有效时，才可能对该芯片进行操作。

【例3-1】 RAM芯片Intel 6164容量为8K×8位，用两片静态RAM芯片6164组成16K×8位的存储器系统。地址选择的方式是将地址总线的低13位（A_{12}～A_0）并行地与存储器芯片的地址线相连，而片选端与高位地址线相连。

为了区分6164两组不同的芯片，可用 A_{13}～A_{19} 中的任意一根地址线来控制，如图3-1所示，用 A_{13} 来控制。A_{13} 为“0”选中＃1芯片，A_{13} 为“1”选中＃2芯片，此时＃1芯片的段内地址为0000～1FFFH，＃2芯片的地址为2000～3FFFH。但仔细分析，只要 $A_{13}=0$，A_{14}～

A_{19}为任意值，都选中＃1 芯片，而只要 $A_{13}=1$，$A_{14}\sim A_{19}$为任意值，都选中＃2 芯片，所以它们的地址是重叠的。在一个段(64KB)中，地址重叠区有 4 个，即有 4 组地址可用于＃1 芯片寻址：0000～1FFFFH，4000～5FFFFH，8000～9FFFH，0C000～0DFFFH。

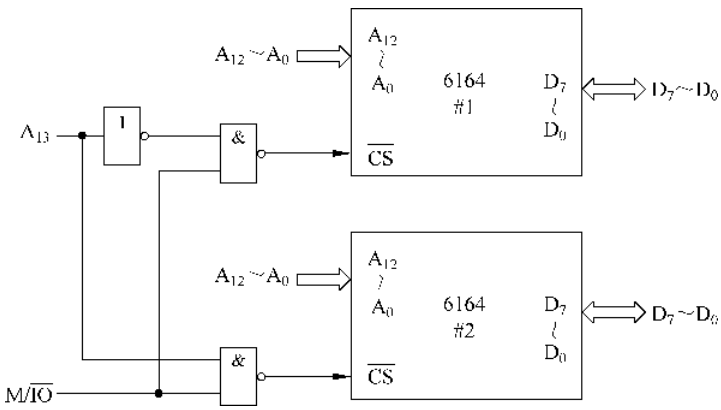


图 3-1　线性选择方式

4 组地址可用于＃2 芯片寻址：2000～3FFFFH，6000～7FFFFH，0A000～0BFFFH，0E000～0FFFFH。

至于不同的段内重叠区就更多了，这种寻址方式称为线选方式。采用线选方式时，不仅地址重叠，而且用不同的地址线作选片控制，它们的地址分配也是不同的，若【例 3-1】用 A_{14}作控制线，则它们的基本地址是：

＃1 芯片：0000～1FFFH，＃2 芯片：4000～5FFFH。

此时内存地址不连续。若用 A_{15}作控制线，把 16KB 内存地址分成上下两个区，每区 32KB，前 32KB 地址都选中第一组，后 32KB 地址都选中第二组。

总之线性选择方法简单，节省译码电路，但地址分配重叠，且地址空间不连续，在存储容量较小且不要求扩充的系统中，线性选择方法是一种简单经济的方法。

2. 全译码选择方式

全译码选择地址的方式是对全部地址总线进行译码，当有 16 根地址线时，可直接寻址 64KB 单元。

【例 3-2】 假如一个微机系统的 RAM 容量为 4KB，采用 1K×8 位的 RAM 芯片，安排在 64KB 空间的最低 4KB 位置，$A_9\sim A_0$ 作为片内寻址，$A_{15}\sim A_{10}$译码后作为芯片寻址，则 4KB 芯片占用的地址空间分别如下。

第一组：地址范围为 0000～3FFH。

第二组：地址范围为 0400～7FFH。

第三组：地址范围为 0800～0BFFH。

第四组：地址范围为 0C00～0FFFH。

其中，$A_9\sim A_0$ 并行接入内存芯片地址线，用作片内地址选择，$A_{15}\sim A_{10}$共 6 根地址线通过 6：64 译码器译码后得到各芯片的唯一片选信号，连到内存芯片的片选端，用作片间选择信号。图 3-2 给出了全地址译码选择的例子。采用全地址译码方法选择地址，译码电路比

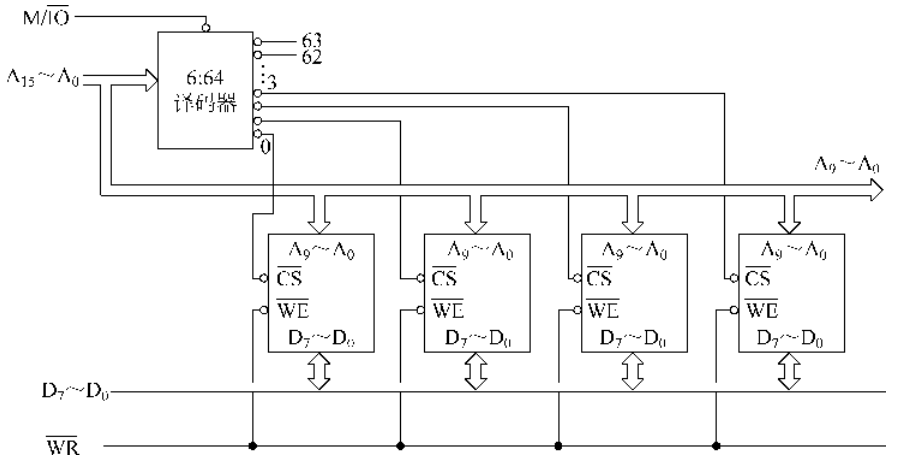


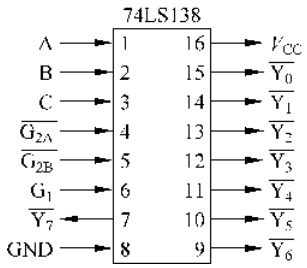
图 3-2　全译码地址选择方式

较复杂，但所得到的地址是唯一的、连续的，并且便于内存扩充。

3. 部分译码选择方式

部分译码选择方式是将高位地址线中的几位经过译码后作为片选控制，它是线性选择法与全地址译码选择法的混合方式，通常译码器采用 3-8 译码器 74LS138。

74LS138 译码器的管脚及译码输出真值表如图 3-3 所示。



G_1	$\overline{G_{2A}}$	$\overline{G_{2B}}$	C	B	A	输出
1	0	0	0	0	0	Y_0=0,其余为1
1	0	0	0	0	1	Y_1=0,其余为1
1	0	0	0	1	0	Y_2=0,其余为1
1	0	0	0	1	1	Y_3=0,其余为1
1	0	0	1	0	0	Y_4=0,其余为1
1	0	0	1	0	1	Y_5=0,其余为1
1	0	0	1	1	0	Y_6=0,其余为1
1	0	0	1	1	1	Y_7=0,其余为1

图 3-3　74LS138 译码器管脚及译码输出真值表

74LS138 的 G_1、$\overline{G_{2A}}$、$\overline{G_{2B}}$为控制端，组合为 100 时输出才有效(输出低电平有效)，当输入端 C B A 这 3 位相应组合为 000～111 时，每次译码一个输出端为 0，其余输出端为 1，具体组合如图 3-3 真值表所示。

【例 3-3】 如果要设计一个 8K×8 位的存储器系统，采用 2K×8 位的 RAM 芯片 4 片，选用 A_{10}～A_0 作为片内寻址，用 A_{13}～A_{11}作为 74LS138 的译码输入，利用输出端 Y_0～Y_3 作为片选信号，则其地址分配如下。

第一片：0000～07FFH。

第二片：0800～0FFFH。

第三片：1000～17FFH。

第四片：1800～1FFFH。

当然在存储器的一段(64KB)内，A_{14}和 A_{15}可以任意选择，所以地址仍有重叠区。若采

用 Y_4～Y_7 作为片选信号，4 片 RAM 芯片的地址分配又不同，分别如下。

第一片：2000～27FFH。

第二片：2800～3FFFH。

第三片：3000～37FFH。

第四片：3800～3FFFH。

部分译码方式的可寻址空间比线性选择范围大，比全译码选择方式的地址空间要小。部分译码方式的译码器比较简单，但地址扩展受到一定限制，并且出现地址重叠的区域。使用不同信号作为片选控制信号时，它们的地址分配也将不同，此方式经常应用在设计较小的微型计算机系统中。

总之，CPU 与存储器相连时，将低位地址线连到存储器所有芯片的地址线上，实现片内选址。将高位地址线单独选用（线选法）或经过译码器（部分译码或全译码）译码输出控制芯片的选片端，以实现芯片间寻址。连接时要注意地址分布及重叠区。

3.2.2 存储器的数据线及控制线连接

8086 CPU 与存储器芯片连接的控制信号主要有地址锁存信号 ALE，读选通信号/RD，写选通信号/WR，存储器或 I/O 选择信号 M/$\overline{IO}$，数据允许输出信号$\overline{DEN}$，数据收发扩展信号 DT/$\overline{R}$，准备好信号 READY。在最小模式系统配置中，数据线和地址线经过地址锁存器 8282 及数据收发器 8286 输出。下面是一个 CPU 与 RAM、ROM 连接的例子。

【例 3-4】 要求用 4K×8 位的 EPROM 芯片 2732，8K×8 位的 RAM 芯片 6264，译码器 74LS138 构成 8K 字 ROM 和 8K 字 RAM 的存储器系统，系统配置为最小模式。图 3-4 给出了系统连接图。

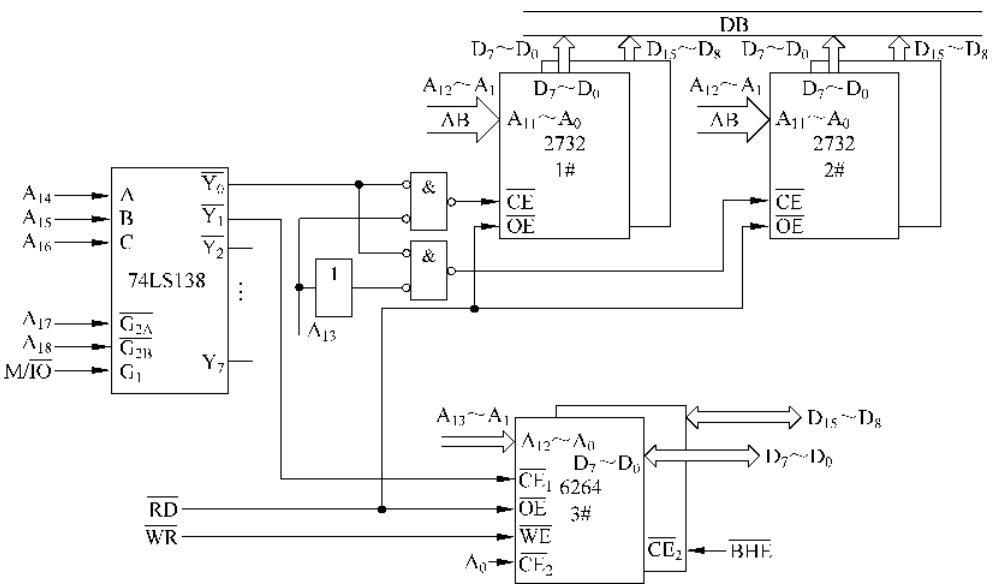


图 3-4 一个存储子系统例子

ROM 芯片，8K 字用 4 片 2732 芯片组成，片内用 12 根地址线 $A_1 \sim A_{12}$ 寻址。

RAM 芯片，8K 字用 2 片 6264 芯片组成，片内用 13 根地址线 $A_1 \sim A_{13}$ 寻址。

芯片选择由 74LS138 译码器输出 Y_0、Y_1 完成。

ROM 芯片由/RD 信号来完成数据读出。RAM 芯片由$\overline{RD}$和$\overline{WR}$来完成数据读写，A_0、$\overline{BHE}$用来区分数据线的低 8 位及高 8 位。

由于 ROM 芯片容量为 4K×8 位，RAM 芯片容量为 8K×8 位，用 A_{13} 和 Y_0 输出进行二次译码，来选择两组 ROM 芯片。

74LS138 译码器的输入端 C、B、A 分别连地址线 $A_{16} \sim A_{14}$，控制端 G_1、$\overline{G_{2A}}$和$\overline{G_{2B}}$分别连接 M/$\overline{IO}$和 A_{17}、A_{18}，计算得到存储器的地址范围如下。

#1 芯片：00000～01FFFH。

#2 芯片：02000～03FFFH。

#3 芯片：04000～07FFFH。

注意：图 3-4 中未画出对 ROM 芯片数据线的低 8 位与高 8 位的选择，可将 A_0 和$\overline{BHE}$分别与$\overline{Y_0}$信号相“与”来实现此选择。

3.3 存储器管理

以前的微型计算机在设计的时候，由于存储器芯片价格昂贵，以及软件对内存要求不高，因此设计主存储器为 640KB。随着大型软件系统的出现，多道程序要求允许不同程序同时存取，需要大量存储空间，640KB 的限制成了计算机的致命伤。但随着计算机技术的发展，CPU 的寻址空间不断扩大，80386 CPU 可寻址 4GB 存储空间，如何充分利用 CPU 庞大的寻址空间发挥其性能呢？本节简单介绍一些这方面的技术。

3.3.1 IBM PC/XT 中存储空间的分配

IBM PC/XT 采用 8088 CPU，可以寻址 1MB 存储空间。它的内存地址分配情况如图 3-5 所示。它是将 ROM 安排在高端，而把 RAM 安排在低端。在这样的多芯片组成的微机内存中，往往通过译码器实现地址分配。

其中，0000H～9FFFFH 的 640KB 为主存储器，由 DOS 进行管理。0A0000H～0BFFFFH 的 128KB 为 VRAM 区，分配给视频适配器上的存储器使用。0C0000H～0FFFFFH 为 ROM 区，供 BIOS 和其他程序使用。VRAM 区和 ROM 区合起来称为上位存储器，总共 384KB。这些 ROM 空间用户程序无法存取，此部分存储器或者在系统主机板上，或者在接口板上。

在 PC/XT 的 ROM 区，ROM BIOS 占 8KB，0FE000H～0FFFFFH，ROM BASIC 使用了从 0F6000H 开始的 32KB 区域，硬盘使用了从 0C8000H 开始的 8KB 区域。在 VRAM 区，MDA 使用了从 0B0000H 开始的 4KB 区域，CGA 使用了从 0B8000H 开始的 16KB 区域。

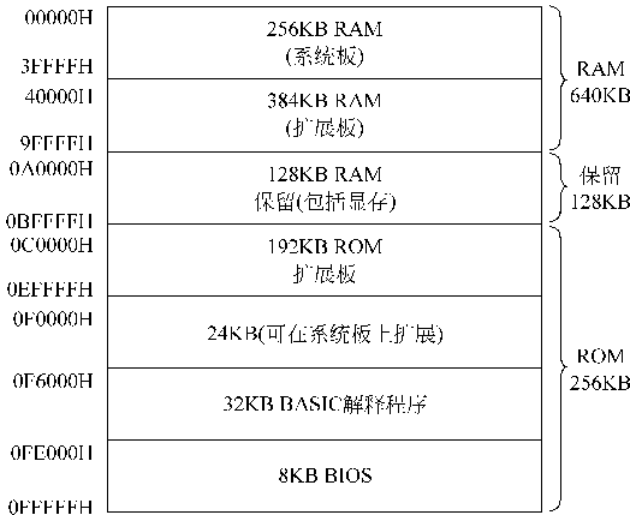


图 3-5 PC/XT 存储器地址分配

3.3.2 扩展存储器及其管理

由 80286、80386、80486 等 CPU 组成的微型计算机大多配置了 4～16MB 内存，现在 Pentium 以上的 PC 则配置了 32～512MB 甚至更多的内存，本节将介绍这些内存的管理和使用方法。

1. 寻址方式

不同的 CPU 因地址线数目的不同，其寻址范围也不同，如表 3-1 所示。

表 3-1 各型 CPU 的寻址能力

CPU	数据总线	地址总线	寻址范围	支持操作系统
8088	8 位	20 位	1MB	实方式
8086	8 位	20 位	1MB	实方式
80286	16 位	24 位	16MB	实、保护方式
80386	32 位	32 位	4GB	实、保护、V86 方式
80486	32 位	32 位	4GB	实、保护、V86 方式
Pentium	64 位	36 位	64GB	实、保护、V86 方式

Intel 8086 的地址总线为 20 根，寻址范围为 1MB。实际上 DOS 操作系统还只能管理这 1MB 中的 0～640KB 的存储空间。由于早期的 PC 应用程序规模较小，用户界面为字符形式，也不需要处理多媒体信息，在当时的计算机发展水平下，大多数的业界人士都认为，640KB 的存储容量已经足够了，而现在计算机已经配置了几百 MB 的内存。

2. 存储器管理

(1) 存储器的管理机制

计算机中的物理存储器是一个字节类型的线性数组，每一个字节占用一个唯一的地址，

称为该字节的物理地址，而在像 80386 CPU 的指令系统中，用段和偏移地址两部分寻址。这个分为两部分的地址，并不像 8088/8086 那样直接用于寻找物理存储器地址。而是由地址转换机制，把程序地址转换或映射为物理存储器地址。这种方法支持虚拟地址的概念。虚拟地址就是由这两部分组成的，之所以使用虚拟地址，是因为这样的地址并不对应于物理存储器的某一地址，而是通过物理地址映射函数间接地对应。

广泛使用的存储器管理机制是分段机制和分页机制管理，它们都是使用驻留在存储器中的各种表格，规定各自的转换函数。这些表格只允许操作系统进行访问，而应用程序不能对其修改。这样，操作系统为每个任务维护一套各自不同的转换表格，其结果是每个任务有不同的虚拟地址空间，并使各个任务彼此隔离开来，以便完成多任务分时操作。

以 80386 为例，80386 先使用分段机制，把包含两个部分的虚拟地址空间转换为一个中间地址空间的地址，这一中间地址空间称为线性地址空间，其地址称为线性地址。然后再用分页机制把线性地址转换为物理地址，如图 3-6 所示。

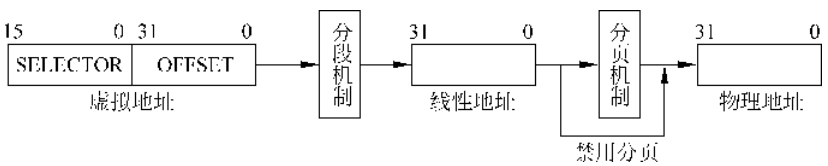


图 3-6 80386 的虚拟-物理地址转换

虚拟地址空间是二维的，它所包含的段数最大可到 16K 个，每个段最大可到 4GB，从而构成 64TB 容量的庞大虚拟地址空间。线性地址空间和物理地址空间都是一维的，其容量是 $2^{32}=4$GB。事实上，分页机制被禁止的时候，线性地址就是物理地址。

8086 和以后的微处理器均支持 3 种工作方式，即实地址模式、虚地址保护模式和 V86 模式。80286 只有实地址模式和虚地址保护模式两种工作方式。8088/8086 只工作在实地址模式。

(2) 虚拟存储器概念

虚拟存储技术是建立在主存和大容量辅存物理结构的基础上的，由附加硬件装置及操作系统内的存储管理软件组成的一种存储体系，它将主存和辅存的地址空间统一编址，提供了比实际物理内存大得多的存储空间。在程序运行时，存储器管理软件只是把虚拟地址空间的一小部分映射到主存储器，其余部分则仍然存储在磁盘上。当用户访问存储器的范围发生变化时，处于后台的存储器管理软件再把用户所需要的内容从磁盘调入内存，用户则好像是在访问非常大的线性地址空间。这种机制使程序员在编程时，不用再考虑计算机的实际内存容量，可以写出使用的存储器比实际配置的物理存储器大得多的程序。

虚拟存储器很好地解决了计算机存储系统对存储容量、单位成本和存储速度的苛刻要求，取得了三者之间的最佳平衡。

在虚拟存储系统中，基本信息传送单位可采用段、页或段页等几种不同的方式。

在 80386 及以后的 CPU 上，分页机制是支持虚拟存储器的最佳选择，因为它是使用固定大小的块。

(3) 保护模式

这里的保护有两种含义：一是为每一个任务分配不同的虚地址空间，使任务之间完全

隔离，实现任务间的保护；二是任务内的保护机制，保护操作系统存储段及其专用处理寄存器不被用户应用程序所破坏。

通常操作系统存储在一个单独的任务段中，并被所有其他任务所共享，每个任务有自己的段表和页表。

在同一任务内，定义 0～3 这 4 种特权级别，0 级最高。定义为最高级中的数据只能由任务中最受信任的部分进行访问。特权级可以看成 4 个同心圆，内层最高，外层最低，特权级的典型用法是把操作系统的核心放在 0 级，而应用程序放在 3 级，留下的部分由中间层软件使用，如图 3-7 所示。不同的软件只能在自己的特权级中运行，这样就可以避免应用程序对操作系统内核的破坏，使系统运行更加可靠。

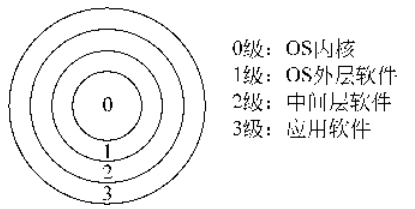


图 3-7　80386 的特权级(0 级～3 级)

3. 分段机制

80386 的分段模式使用具有两部分的虚拟地址，即段部分和偏移量。段部分是指 CS、DS、SS、ES、GS、FS 共 6 个段。80386 为地址偏移部分提供了灵活的机制，使用存储器操作的每条指令规定了计算偏移量的方法，这种规定叫做指令的寻址方法，8 个通用寄存器的任一个都可以用作基址寄存器，除堆栈指针外的其他 7 个寄存器又可以用来作变址寄存器，再把这个变址寄存器的值乘以 1、2、4、8 中的任一因子，然后再加上一个 32 位的偏移量作为地址的偏移部分。这种寻址方式提供了强有力而又灵活的寻址机制，非常适用于高级语言。

段是形成虚拟地址到线性地址转换机制的基础。每个段由 3 个参数定义。

- 段的基地址：即线性空间中段的开始地址。基地址是线性地址空间对应于段内偏移量为 0 的虚拟地址。
- 段的界限：指段内可以使用的最大偏移量，它指明该段的范围大小。
- 段的属性：如可读出段、可写入段、特权级等。

这几个参数都存储在段的描述符中，而描述符又存储在段描述表中，即描述符表是描述符的一个数组，而虚拟地址到线性地址的转换要访问描述符。

4. 分页机制

分页机制是存储器管理机制的第二部分。分页机制的特点是所管理的存储器块具有固定的大小，它把线性空间的任何一页映射到物理空间的一页。分页转换函数由称为页表的存储器常驻表来描述。可把页表看成 2^{20} 个物理地址的数组，线性到物理地址的转换就是一个简单的数组查询，线性地址的低 12 位给出页内偏移，然后将此页的偏移加到页基地址上，由于页基地址是 4KB 边境对齐的，即基地址的低 12 位是全 0，故偏移量就是物理地址的低 12 位，页基地址提供物理地址的高 20 位，该基地址是把线性的值的高 20 位作为索引，从而在页表中查询到的。在 80386 中共有 2^{20} 个表项，每项占 4 个字节，需 4MB 物理空间。为节约内存，分为两级页表机构，第一级用 1K 个表项，每项 4 个字节，占 4KB 内存。第二级再用 10 位(1KB)，这样两级表组合起来即可达到 2^{20} 个表项。

有了这些对大容量内存的管理手段，就可以支持多用户多任务的操作系统，例如 UNIX、Linux、Windows 操作系统等。