**1微型计算机的基本结构和运算基础**

1.1 重点内容提要

（一）微型计算机的基本结构

微型计算机系统是由硬件系统、软件系统以及通信网络系统组成的一个整体系统。

微型计算机硬件系统是指构成微机的所有实体部件的集合。这些部件包括集成电路芯片、机械等物理部件，通常称为“硬件”。微型计算机的硬件主要由输入设备、输出设备、运算器、存储器和控制器 5 个部分组成。

微型计算机的软件系统可分为系统软件和应用软件。

（二）数制转换

在计算机内部，一切信息的存取、处理和传送均采用二进制形式。但为了方便，常采用八进制、十六进制和十进制。这样它们之间就存在一种对应的转换关系。

任意进制数转换为十进制数就是按权展开求多项式之和。十进制转换为二进制数时，对于整数部分，采用除基数取余数法；对于小数部分，则采用乘基数取整数法。八进制、十六进制和二进制之间的转换非常简单，分别按 3 位二进制数对应 1 位八进制数、4 位二进制数对应 1 位十六进制数的关系转换即可。

（三）计算机中带符号数的表示方法

在计算机中表示的数叫机器数。数有带符号数和不带符号数之分，在计算机中，对于带符号数，其正和负必须符号化。带符号数的机器数最常用的有原码、反码和补码 3种形式。

正数的原码、反码和补码形式一样，其符号位都为“0”，数值位同真值；对于负数，其符号位都为“1”，而数值位有区别，原码的数值位同真值，反码的数值位是其真值按位取反，补码的数值位为其反码末位加 1。

补码表示的机器数其符号位能和有效数值位一起参加数值计算，并能使减法运算变为加法运算，从而简化运算器的线路设计。

补码加法规则：[X + Y]补 = [X]补 + [Y]补

[X – Y]补 = [X]补 + [–Y]补

溢出判别：计算机在进行补码运算时，由于位数的限制可能产生溢出。对于带符号数而言，溢出是由于数值位侵犯符号位造成的，可采用双高位法判别溢出。显然，在两个同号数相加或两个异号数相减时才可能溢出。溢出时，符号位的“1”和“0”已不能准确表示数的符号了。对于不带符号数，因所有位均是有效数值，可根据最高位是否产生进位或借位来判别溢出。（四）计算机中数的小数点表示方法

计算机中数的小数点表示方法有定点表示法和浮点表示法。

在定点表示法中，小数点在数中的位置是固定不变的。而在浮点表示法中，小数点的位置是不固定的，用阶码和尾数来表示。通常尾数为纯小数，阶码为整数，尾数和阶码均为带符号数。尾数的符号代表数的正、负；阶码的符号表明小数点的位置。

（五）常用的二进制编码

计算机只能识别二进制数“0”和“1”，因此在计算机中任何信息都是通过一定的编码实现的。常用的二进制编码有 BCD 码、ASCII 码、汉字国标码等。

**2 Intel 32 位微处理器**

2.1 重点内容提要

微型计算机中的运算器和控制器合起来称为微处理器，因微处理器通常集成在一块大规模集成电路芯片上，所以人们又把微处理器称作 CPU。Intel 公司生产的 80386、80486、Pentium 都是 32 位的微处理器（Pentium 4 已是 64 位）。32 位微处理器是指在微处理器内部以 32 位二进制数为单位进行数据处理。

（一）微处理器的基本结构

80486 微处理器内部由 8 大部件组成：总线接口部件、高速缓存部件、代码预取部件、指令译码部件、浮点数部件、执行部件、段部件、分页部件。图 2.1 为 80486 微处理器的内部基本结构图，它反映了微处理器内部的基本逻辑部件、部件之间的基本联系、各部件的主要功能、数据在微处理器中的主要流动方向。



图 2.1 80486 内部基本结构图

微处理器外部引脚分为数据总线引脚、地址总线引脚及控制总线引脚。32 位微处理器都采用栅格阵列插针或封装，在微机主板上有相应的插座安装。

80486 CPU 有 3 种工作方式：实地址方式、保护方式和虚拟 8086 方式。Pentium 微处理器除了上述 3 种工作方式外，增加了一种系统管理方式（SMM）。

（二）寄存器

寄存器是微处理器内部用来放置数据或地址的存储单元。在微处理器的各个部件中，都有一些寄存器。有些寄存器是编程不可见的，有些是编程可见的。对于编程可见的寄存器，根据功能可分为 9 组，分别是通用寄存器、指令指针寄存器、标志寄存器、段寄存器、系统地址寄存器、调试寄存器、测试寄存器、控制寄存器及浮点寄存器。

在通用寄存器中，32 位寄存器有 EAX、EBX、ECX、EDX、EBP、ESP、ESI 和 EDI，其低 16 位分别是 AX、BX、CX、DX、BP、SP、SI 和 DI。其中 AX、BX、CX、DX 的每个字节均另有一个名字，高字节分别称为 AH、BH、CH、DH；低字节分别称为 AL、BL、CL、DL。指令指针寄存器 EIP（低 16 位称为 IP），其内容为下一条要取入 CPU 的指令在内存中的偏移地址。标志寄存器 EFLAGS（低 16 位称为 FLAGS），共有 3 类标志：状态标志、控制标志和系统标志。80486 中的标志包括进位标志 CF、奇偶标志 PF、辅助进位标志 AF、零标志 ZF、符号标志 SF、溢出标志 OF、方向标志 DF、中断允许标志 IF、陷阱标志 TF、嵌套任务标志 NT、输入输出特权级标志 IOPL、恢复标志 RF、虚拟 8086 方式标志 VM 和对准检查标志 AC。

段寄存器有 CS、DS、ES、SS、FS 和 GS。它们分为两部分，一部分是编程可见的选择寄存器，为 6 个 16 位寄存器，对应在另一部分有 6 个 64 位的编程不可见的描述符寄存器。在实地址方式或虚拟 8086 方式下，描述符寄存器不起作用，选择寄存器退化成16 位 CPU 的段寄存器功能，存放内存段的段基址（段首地址的高 16 位）。其中 CS 对应于代码段，DS 对应于数据段，ES 对应于附加数据段，SS 对应于堆栈段，FS 和 GS 没有定义。在保护方式下，选择寄存器的低二位是特权标志。其中 D2位是描述符表类型标志，高 13 位是选择码，指出本段的段描述符在由 D2位指出的描述符表中的逻辑排序。

其他寄存器都是与系统有关的寄存器。控制寄存器和系统地址寄存器是实现保护模式功能的基础，控制着新的存储器管理方式、多任务切换等功能的实现；调试和测试寄存器用于调试微码执行、数据访问的断点自陷和测试芯片逻辑、高速缓冲存储器等；浮点寄存器主要用于浮点运算。

（三）逻辑地址、线性地址和物理地址

一般来说，用户使用的是逻辑地址，由内存管理软件和相关硬件将逻辑地址变为物理地址。

在 16 位模式下，逻辑地址由 16 位的段基址和 16 位的偏移地址组成，物理地址 20位。其关系为：

物理地址=段基址´10H+偏移地址

在 32 位模式下，地址转换比较复杂，逻辑地址是一个 16 位的段选择符和一个 32位的偏移地址。段部件根据段选择符获得相应的段描述符，取出 32 位的段基址再加上32 位偏移地址得到 32 位的线性地址。然后再由段部件传送给分页部件。由其转换为 32位的物理地址。若分页部件被禁止，那么计算出的线性地址就是物理地址。

（四）指令流水线操作32 位微处理器的一个重要的特点就是采用了指令流水线技术。这一技术大大加快了指令执行速度，加大了信息流量。这是一种同时进行若干操作的并行处理方式。在流水线结构中，每条指令分成若干步骤来执行，每个子过程各在一个专门的硬件站点上执行，这样一条指令的全部操作需顺序地经过流水线中多个站点的处理才能完成。但前后连续的几个操作可以依次进入流水线中，在各个站点间重叠执行，以此来实现并行操作。

**3 80x86 寻址方式和指令系统**

3.1 重点内容提要

（一）80x86 数据寻址方式

通常，一条指令由操作码和操作数组成。数据寻址方式即指寻找指令中操作数所在地址的方法。80x86 操作数可以放在寄存器、存储器中，也可以以立即数的形式放在指令代码中。因此，从大体上来分，80x86 的寻址方式分为 3 大类，即立即寻址、寄存器寻址和存储器寻址。

1．立即寻址

操作数包含在指令中，作为指令的一部分，存放在代码段中。当取指令时，操作数也随之取出。如：

MOV AL，80H

MOV EAX，0

注意：立即寻址主要用来给通用寄存器或存储器赋值，它只能用于源操作数，不能用于目的操作数。

2．寄存器寻址

在这种方式中，操作数放在 32 位、16 位或 8 位的通用寄存器中。执行时不需使用总线周期，因而运算速度较快。如：

MOV AX，BX

3．存储器寻址

操作数在存储器中，其地址可以在指令中直接给出，也可以放在 16 位或 32 位规定的寄存器中间接给出。寻址所需要的偏移地址即为有效地址 EA，它代表被寻址的操 作数在存储器中距离该段起点的字节数。有效地址 EA 的计算随所采用的寻址方式不同而异。

（1）16 位指令模式下的存储器寻址方式：80x86 实方式为 16 位指令模式。寻址的寄存器为 BX、BP、SI、DI。如果有效地址在 BX、SI 或 DI 中，则以 DS 寄存器内容为段基址；如果有效地址在 BP 中，则以 SS 段寄存器之内容为段基址。如果使用段超越前缀（CS：ES：DS：SS：），操作数则可以放在冒号前指定的段。16 位指令模式寻址结构由 4 部分组成。

段基址 ´ 10H + 基址 + 变址 + 偏移量

基址寄存器 BX 和 BP 存放基址，变址寄存器 SI 和 DI 存放变址，偏移量为 8 位或16 位用补码形式表示的带符号数。操作数的有效地址为 16 位，段内寻址范围为 64KB，物理地址为 20 位，使用存储器的第一个 1MB 空间。① 直接寻址 操作数的有效地址 EA 就在指令中，可以是数值地址，也可以是符号 地址。符号地址必须是赋过值的。如：

MOV AX，[2000H] MOV BX，BLOCK

注意：EA 是在指令中，而操作数本身在数据段的存储单元中。

① 直接寻址 操作数的有效地址 EA 就在指令中，可以是数值地址，也可以是符号

地址。符号地址必须是赋过值的。如：

MOV AX，[2000H]

MOV BX，BLOCK

注意：EA 是在指令中，而操作数本身在数据段的存储单元中。

② 寄存器间接寻址 操作数的有效地址在 BX、BP 或 SI、DI 中，可表示为：



例如：MOV AX，[BX]

③ 寄存器相对寻址 操作数在存储单元中，其有效地址是一个 8 位或 16 位的偏移量与一个基址或变址寄存器的内容之和，偏移量和寄存器均由指令给出，可表示为：

例如：MOV AL，[BX + 5]

④ 基址变址寻址 操作数在存储单元中，其有效地址是一个基址寄存器与一个变 址寄存器的内容之和，可表示为：



例如：MOV AX，[BP + DI]

⑤ 相对基址变址寻址 操作数在存储单元中，其有效地址是一个 8 位或 16 位偏移

量、一个基址寄存器与一个变址寄存器 3 部分内容之和，可表示为：

例如：MOV AX，[BX+SI+2000H]

（2）32 位指令模式下的存储器寻址方式：在保护方式下，32 位指令模式寻址结构

由 5 部分组成。段址 + 基址 + 变址X 比例因子 + 偏移量

其中，基址寄存器或变址寄存器可以是除 ESP 以外的任何 32 位通用寄存器。当基址寄存器为 EBP 时，默认段寄存器 SS 存放段选择符，否则，默认 DS 存放段选择符。也可使用段超越前缀来指定。比例因子为 1、2、4、8。偏移量为 8 位或 32 位。① 直接寻址 指令中直接包含 32 位的有效地址。EA 为 32 位有效地址。

② 间接寻址 EA 为基址寄存器的内容。

③ 相对基址寻址 EA 为基址寄存器和一个 8 位或 32 位偏移量之和。④ 相对比例变址寻址 EA 为变址寄存器的内容乘以比例因子，再加偏移量。

⑤ 相对比例基址变址寻址 EA 为变址寄存器的内容乘以比例因子，加上基址寄存

器的内容，再加上偏移量。（二）80x86 指令系统

80x86 指令系统，按指令的功能可分为数据传送指令、算术运算指令、逻辑运算指

令、控制转移类指令、串操作指令、处理器控制指令和保护方式指令。

1．传送指令这类指令完成数据、地址等的传送操作，除了标志寄存器传送指令外，其他传送指令都不影响标志位。在使用传送指令 MOV、堆栈指令 PUSH 和 POP 以及数据交换指令 XCHG 时应注意：①不允许将立即数直接送段寄存器；②不允许目的操作数为立即数和 CS 段寄存器； ③不允许两个存储单元之间直接传送数据；④不允许两个段寄存器之间直接传送数据；⑤操作数数据类型要匹配；⑥对堆栈的操作不允许字节操作；⑦不能改变 CS 和指令指针寄存器（E）IP 的值。

地址传送指令 LEA、LDS 和 LES 的功能是将存储单元的地址送入特定的寄存器中。

标志传送指令 LAHF、SAHF、PUSHF、POPF、PUSHFD、POPFD 主要用来获取标志寄存器的值或改变标志寄存器的值，通过 AL 寄存器或堆栈来实现。查表指令 XLAT 实现直接查表功能。规定 BX 寄存器存放表的首地址的偏移地址，AL 寄存器存放表内偏移量，查得结果存放在 AL 中。

符号扩展指令是将累加器中带符号数进行符号扩展，正数高半部分扩展为全“0”，负数高半部分扩展为全“1”。2．算术运算指令

算术运算指令用来完成加、减、乘、除等算术运算，包括 4 种基本算术运算操作及十进制算术调整指令。指令执行结果均影响标志位。ADD 和 ADC 为加法指令，SUB 和 SBB 为减法指令，ADC 和 SBB 要考虑 CF 值。INC 和 DEC 分别为加 1 和减 1 指令，不影响 CF 值。比较指令 CMP 完成两个数的相减操作，但不回送相减的结果，只是使结果影响标志位，常用于分支程序设计中。

乘法指令分为无符号数乘 MUL 和带符号数乘 IMUL，只影响 CF 和 OF 标志。除法指令也分无符号数除 DIV 和带符号数除 IDIV，被除数必须是除数的双倍字长，一般要进行高位扩展。当进行无符号数除法时，被除数高位按 0 扩展为双倍除数字长。当进行有符号数除法时，被除数以补码表示，可使用扩展指令 CBW、CWD、CWDE 和 CDQ进行高位扩展。

BCD 算术运算指令的功能是将二进制运算结果调整为 BCD 码表示形式。BCD 码分为组合型（压缩格式）的 BCD 码和非组合型的 BCD 码（ASCⅡ码）。组合型 BCD 码的加、减调整指令分别为 DAA 和 DAS；非组合型 BCD 码的加、减、乘、除调整指令分别为 AAA、AAS、AAM 和 AAD。DAA 和 AAA 一般跟在 ADD 或 ADC 指令之后，DAS和 AAS 一般跟在 SUB 或 SBB 之后，AAM 跟在 MUL 之后，但 AAD 则用于 DIV 操作之前。注意，BCD 码总是被作为无符号数看待的。

使用算术运算指令时，如果没有特别规定，参与运算的两个操作数类型要匹配，且不能同时为存储器操作数。操作数不允许为段寄存器。若操作数类型在指令中不明确，要用 PTR 伪指令予以说明。

3．逻辑运算指令

逻辑运算指令是按位操作，除 NOT 指令不影响标志位外，其余指令将使 CF=OF=0，影响 SF、ZF 和 PF。TEST 和 AND 指令执行同样的操作，但 TEST 不送回操作结果，而仅仅影响标志位。TEST 指令用来测试某位的状态。AND 指令用来使某些位屏蔽，即清0。OR 指令常用来将一些位置 1。XOR 指令常用在使某个寄存器清 0 或使某些位变反。

移位指令包括有算术移位、逻辑移位和循环移位指令。算术移 N 位，可相当于把二进制数乘以或除以 2N。逻辑移位操作用于截取字节或字或双字的若干位。循环移位指令在移位时移出的目的操作数并不丢失，而循环送回目的操作数的另一端。

4．控制转移类指令

这类指令用来控制程序的执行流程，即改变程序的走向。分为转移指令和循环控制指令。转移指令可分为无条件转移指令和条件转移指令两类。无条件转移指令 JMP 是指程序无条件地转移至其指定的地址去执行从该地址开始的指令。转移可分为段内转移和段间转移，段内转移只改变 IP，而段间转移改变 IP 和CS。段内转移可以是段内直接转移（转向的有效地址是当前的 IP 寄存器的内容和一个偏移量之和），也可以是段内间接寻址（转向的有效地址在一个寄存器或存储器单元中，即用寄存器或存储器的内容取代 IP 的内容）。段间转移可以是段间直接转移（直接提供CS 和偏移地址 IP），也可以是段间间接转移（用存储器相邻两个字的内容取代 IP 和 CS的内容）。

条件转移指令比无条件转移指令多了一个条件判断的功能，若满足指令规定的条件，则程序转移，否则程序顺序执行。条件转移指令的条件主要是根据标志位来判别的，对于 CF、ZF、SF、OF 和 PF 都有相应的判断指令。对于无符号数和带符号数大小的判别也有两组指令，是根据标志位的组合状态来判别的。还有测试 CX 条件转移指令。条件转移指令只允许段内转移，不允许段间转移。循环控制指令 LOOP 用来管理程序循环的次数，它与条件转移不同的是，循环指令要对（E）CX 寄存器的内容进行测试，判断其是否为 0 作为转移条件。每循环执行一次。 （E）CX 的内容自动减 1，直到（E）CX 为 0 时循环结束。

5．串操作指令

串操作指令都用寄存器（E）SI 对源操作数进行间接寻址，并且默认是在 DS 段中；用（E）DI 对目的操作数进行间接寻址，并且默认是在 ES 段中。（E）SI 和（E）DI 这两个地址指针在每次串操作以后都自动进行修改，这与方向标志 DF 有关。当 DF=1 时，（E）SI 和（E）DI 做自动减量修改；当 DF=0 时，（E）SI 和（E）DI 做自动增量修改。增、减量根据串元素的数据类型不同而不同，字节型对应是 1，字型对应是 2，双字型对应是 4。可以通过加重复前缀来实现串操作。6．输入/输出指令

使用 IN 或 OUT 指令时，要注意：一个操作数为 AL 或 AX 或 EAX，另一个操作数以 8 位端口地址方式直接给出，或者以 DX 寄存器间接给出。直接寻址范围为 256B，间接寻址范围为 64KB。每个 I/O 地址对应的端口的数据长度为 8 位，传送 8 位数据占用一个端口地址，传送 16 位数据占用两个端口地址，传送 32 位数据占用 4 个端口地址。

**4 汇编语言、程序设计 及其与高级语言调用**

4.1 重点内容提要

（一）汇编语言

汇编语言是符号化了的机器语言。用汇编语言编写的程序被称为汇编语言源程序。源程序只有经过汇编过程才能得到目标程序。汇编时一般采用 MASM 汇编程序。

1．汇编语言格式

汇编语言的语句一般有两类，伪指令语句和指令语句。指令语句有其对应的机器码，汇编程序在对源程序进行汇编时，把指令语句翻译成机器指令。而伪指令没有其对应的机器码，只是指示汇编程序如何进行汇编。

指令语句格式：[标号：]助记符 操作数 ；注释

伪指令语句格式：[名字]定义符 参数 ；注释程序中标号定义了指令的逻辑地址。标号具有段地址、偏移地址和类型属性。标号的段地址和偏移地址是指标号对应的指令首字节所在的段地址和段内的偏移地址。标号的类型属性有两种：NEAR 和 FAR 类型。NEAR 指示近程（段内）标号，该标号只能为本段内指令使用；FAR 指示远程（段间）标号，该标号可以为其他段使用。在转移和调用指令中常将标号作为转移目标地址使用。

变量定义了数据的逻辑地址。变量具有段地址、偏移地址和类型属性。类型属性有BYTE、WORD、DWORD、PWORD、QWORD 等。2．常用的伪指令（1）数据定义伪指令：该指令格式为：变量 定义符 表达式 1，表达式 2，…

定义符一般是以下几种。

DB——定义字节变量。

DW——定义字变量。

DD——定义双字变量。

（2）简化的段定义伪指令：高版本的 MASM 6.X 提供了简化的段定义伪指令。简化的段结构中常用伪指令说明如下。X86——用于选择 80x86 指令系统。

.X86P——用于选择 80x86 保护模式指令系统。

.STARTUP——指示程序开始，初始化 DS 和 SS 寄存器。

.EXIT——使程序返回 DOS 操作系统。

.CODE——定义程序段。

.DATA——定义数据段。

.STACK——定义堆栈段，其后可根据参数，定义堆栈大小。

.MODEL——内存模式说明。

（3）完整段定义伪指令：该指令格式为：段名 SEGMENT [属性]

…

段名 ENDS

属性：[段合并属性][段类属性][段对齐属性][段长度属性]

当定义 DS、ES 和 SS 时，在 SEGMENT／ENDS 伪指令中间的语句，只能包括伪指令语句，不能包括指令语句。只有当定义 CS 时，中间的语句才能为指令语句以及与指令有关的伪指令语句。

段合并属性用来告诉连接程序本段与其他段的组合关系。可供选择的组合类型有 5 种。

PRIVATE——段不与其他同名段合并（默认）。

PUBLIC——段与其他同名段合并为单个连续段。

COMMON——定位段起始处放置其他同名段，段长为最长段长。

MEMORY——将它定位于其他段之上的地址。

AT 表达式——定位段起始于表达式指定的 16 位段值处。

段类属性表明该段的类别，有如下 4 种。

‘CODE’——代码段。

‘DATA’——数据段。

‘STACK’——堆栈段。

‘EXTRA’——附加段。

类别必须是用单引号括起来的字符串。具有相同类别的段，不管其段名是否相同，在连接时，将所有相同类别的逻辑段连接在一起，形成一个统一的物理段。

段对齐属性对段的起始边界进行定位。它有如下 5 种定位方式。BYTE——段从任意字节开始。

WORD——段从下一字地址处开始。

DWORD——段从下一个双字地址处开始。

PARA——段从下一节地址处开始（默认 16B 为一节）。

PAGA——段从下一页地址处开始（256B 为一页）。

段长度属性有：

USE16——段为 16 位，段长 64KB。

USE32——段为 32 位，段长 4GB。（4）段寄存器定义伪指令：其格式如下所示。ASSUME 段寄存器：段名[，段寄存器：段名]

该指令用来规定给定段的段地址寄存器，段寄存器包括 CS、DS、ES、SS、FS 和 GS。（5）段结束伪指令：如果采用简化的段定义结构，则其格式为 END。如果采用完整

的段定义结构，则其格式为 END 标号。

这里，标号为第一条可执行语句的标号。3．控制汇编语言程序语句

MASM 6.X 版本提供了控制程序流程的 3 种汇编语句。

（1）IF 语句。

格式： .IF 表达式

语句 1

.ELSE

语句 2

.ENDIF

（2）DO–WHILE 语句。

格式： .WHILE 表达式

语句

.ENDW

（3）REPEAT–UNTIL 语句。

格式： .REPEAT

语句

.UNTIL 表达式（二）汇编语言程序设计

1．顺序程序设计

顺序程序是一种最简单的程序，每条指令按其在程序中的排列顺序执行。

2．分支程序设计

程序的分支主要靠条件转移指令来实现。这里需要注意的是，条件转移指令都是近程跳转。若程序所要转移的地址超出其范围时，需利用一条无条件转移指令作为中转。

3．循环程序设计

循环程序设计主要用于某些需要重复进行的操作，主要使用循环指令 LOOP 或条件转移指令来实现循环。循环程序的设计可分为设置循环初始状态、循环体和循环控制条

件 3 部分。

（1）设置循环初始状态：该状态主要是指设置循环次数的计数初值，以及其他为能使循环体正常工作而设置的初始状态等。

（2）循环体：它是循环操作（重复执行）的部分，包括循环的工作部分及修改部分。循环的工作部分是实现程序功能的主要程序段；循环的修改部分是指当程序循环执行时，对一些参数如地址、变量的有规律的修正。

（3）循环控制部分：它是循环程序设计的关键。每个循环程序必须选择一个控制循环程序运行和结束的条件。

4．子程序设计

子程序是一段独立的指令组，一般具有通用性，可共享、可被一个程序反复使用或被多个程序调用。采用子程序设计可以缩短程序长度、节省内存空间。在 80x86 汇编程序中，子程序常常以过程的形式出现。

（1）过程定义

格式：过程名 PROC[TYPE][USES REG]

…

RET

过程名 ENDP

TYPE：过程类型，分为近程 NEAR 和远程 FAR，当类型为 NEAR 时，该过程可以被本段调用，当类型为 FAR 时，该过程可以被其他程序段调用。

USES REG：规定需要保护的寄存器，在过程开始将指定通用寄存器内容压入堆栈，在返回之前再恢复这些寄存器的内容。（2）过程调用指令

段内直接调用：CALL 过程名

段内间接调用：CALL NEAR PTR[REG]或 CALL REG

段间直接调用：CALL FAR PTR 过程名

段间间接调用：CALL FAR PTR [REG]

（3）过程返回指令

格式：RET 或 RET N

过程返回指令一般放在过程体的末尾，用来返回主程序。RET 后面可以指定一个 N值（为偶数），其功能为完成 RET 操作后根据 N 值修改堆栈指针（E）SP，目的是用于清除过程调用时入栈的参数。（4）寄存器内容的保护和恢复：即保护现场和恢复现场。除了在定义过程时用 USES

REG 伪指令实现外，还可以在过程一开始，先把过程中要改变的寄存器的内容用 PUSH

指令压入堆栈，然后在返回前，用 POP 指令恢复这些寄存器的内容。

（5）主程序和过程间的参数传递：主程序调用过程时，往往要将入口参数传递给子

程序；子程序在执行完时，要将出口参数传递给主程序。参数传递可以有 3 种形式：①通

过寄存器传递参数；②通过地址表传递参数；③通过堆栈传送参数或参数地址.

**5 内存储器及其管理**

5.1 重点内容提要

（一）存储器概述1．存储器的类型

微型计算机系统中的存储器可分为两大类，即内存和外存。内存也称为主存储器，可通过系统总线直接与微处理器联系，用来存放正在执行的程序和正在处理的数据；外存也称辅助存储器，需通过专门的接口电路和微处理器联系，用来存放暂时不执行或不被处理的数据。

内存主要由半导体存储器组成，其类型如图 5.1 所示。

图 5.1 半导体存储器的分类

外存主要有磁盘、磁带和光盘。2．存储器多级结构

在计算机的发展过程中，CPU 与主存储器速度匹配的矛盾越来越突出。为此，在CPU 与主存之间再增加一级或多级高速缓冲存储器（Cache）来提高存储器的性能价格比。目前，微型计算机的存储器系统已经发展成多级结构，如图 5.2 所示。3．32 位内存储器的组成方式

80486 微处理器分为 4 个存储体，依次存放 32 位数据的不同字节。每个存储体的 8位数据线依次并行连接到系统数据线 D31～D0 上。每个存储体的 15 位地址 A14～A0 接CPU 的地址线 A16～A2，片选信号由高位地址 A31～A17的译码结果和 BE3 ～ BE0 相“与”后产生。一旦地址确定，A31～A1 将确定 4 个存储体中的相同地址单元，BE3 ～ BE0 决定某一个或某几个字节单元选中，然后可对选中单元同时进行读/写操作。

**6 微型计算机的输入/输出**

6.1 重点内容提要

（一）接口及端口

CPU 与外设之间不能直接进行通信，必须要借助于接口。这是因为 CPU 与外设之间速度不匹配或信息格式不同等原因。

对于 I/O 接口，在其内部往往有多个可寻址读写的寄存器，我们称之为端口。主机和外设之间交换信息都是通过操作接口电路的 I/O 端口来实现的。

CPU、I/O 接口和外设之间的接口信息如图 6.1 所示。

图 6.1 CPU、I/O 接口和外设之间的接口信息

（二）I/O 端口的编址

CPU 对 I/O 端口的编址分为存储器映象的 I/O 寻址和 I/O 映象的 I/O 寻址两种方式。存储器映象的 I/O 寻址是把 I/O 端口的地址和存储单元的地址统一编址；I/O 映象的I/O 寻址是将两者分开独立编址。所谓的“独立”和“统一”是相对于存储器地址空间而言的。独立编址时，I/O 地址空间完全独立于存储器空间，于是一个 I/O 端口的地址可以与一个存储单元的地址重叠，CPU 通过控制总线来确定要访问哪一个空间。

存储器映象的 I/O 寻址的优点是指令丰富、编程方便，但程序易读性差、存储器地址范围相对减小。I/O 映象的 I/O 寻址需要用专门的 I/O 指令，但程序方便阅读。80x86采用的是 I/O 映象的 I/O 寻址.

（三）I/O 端口地址的形成

当系统中使用存储器映象的 I/O 寻址方式时，系统中仅需一个译码器芯片，从译码器的输出端接至 I/O 接口芯片的控制端或片选端形成 I/O 端口地址，接至存储器芯片的片选端形成存储单元地址。注意，CPU 的 M/IO 线接在译码器的控制输入端时，要接在高电平有效的那一端，因为在执行存储器访问指令时，CPU 的 M/IO 管脚输出高电平有效信号。

当系统中使用 I/O 映象的 I/O 寻址方式时，系统中的 I/O 端口地址需要单独的一个译码器芯片，译码器的输出端只允许接至 I/O 接口芯片的控制端或片选端形成 I/O 端口地址。注意，CPU 的 M/IO 线接在译码器的控制输入端时，要接在低电平有效的那一端，这是因为在执行 IN 或 OUT 指令时，CPU 的 M/IO 管脚输出低电平有效信号。

（四）输入/输出方式

CPU 与外设通信时，数据传送的控制方式有程序控制传送方式、中断控制传送方式、直接存储器存取方式（DMA 方式）和 I/O 处理机方式。

程序控制传送方式分为同步传送方式和异步查询方式。同步传送方式只有在外设的各种操作时间已知且固定的场合下才能使用。异步查询方式在传送前，必须先查询一下外设的状态。当外设准备好了，CPU 就立即与外设进行数据传送，否则，CPU 就处于循环查询状态，直到 CPU 查询到外设准备好了才进行传送。这种方法能较好地解决外设与CPU 速度上的差异，但 CPU 必须不断地循环测试外设的状态，这样，大大降低了 CPU的运行效率。

在中断传送方式下，CPU 启动外设后，不用等待查询状态，CPU 继续执行自己的程序。当外设准备好了时，由外设向 CPU 发出一个中断请求。若 CPU 响应它，则去执行中断服务程序，待中断服务程序执行完后，再返回主程序。中断方式实现了 CPU 和外设的重叠工作，从而使 CPU 的效率得到提高。

若有大批量的数据要传送，采用 DMA 方式比较合适。DMA 方式是用一个硬件DMAC 芯片来完成软件的工作。CPU 把总线使用权交给 DMAC，然后由 DMAC 快速地将大批量数据与内存进行信息交换。在这段时间里，CPU 可以处理一些内部工作。

要使 CPU 完全摆脱控制 I/O 设备的负担，则提出了 I/O 处理机方式。由 I/O 处理机完成一切 I/O 处理的工作，它有自己的指令系统，可以执行程序来实现对数据的处理。（五）DMA 控制器 8237A

DMA 控制器可以像 CPU 那样得到总线控制权，用 DMA 方式实现外部设备和存储器之间的数据高速传输。8237A 是高性能的可编程 DMA 控制器，它一方面可以控制系统总线，这时通常称它为总线主模块；另一方面又可以和其他接口一样，接受 CPU 对它的读/写操作，这时它就成了总线从模块。

（1）理解 8237A 是如何向 CPU 申请总线控制权的，在控制总线期间完成什么工作。掌握 8237A 作为总线主模块（控制系统总线）和总线从模块（系统总线由 CPU 控制）时，8237A 的某些引脚的不同含义。其中IOR 、IOW 、A3～A0、DB7～DB0等在 8237A作为主模块和从模块时的定义不同，甚至传送方向也不同。例如，8237A 作为从模块时，IOW 是送入 DMA 控制器的，此信号有效，由 CPU 往 DMA 控制器的内部寄存器写入信息，即进行编程；在 8237A 作为主模块时，IOW 的方向是由 DMA 控制器送出的，此信号有效时，存储器中读出的数据被写入 I/O 接口中。8237A 作为从模块时，A3～A0是输入端，接受 CPU 发出的地址信息，对 8237A 片内的 16 个端口进行寻址；在 8237A 作为主模块时，A3～A0是输出端，是 8237A 输出的寻址存储器单元的地址信息的最低 4 位。（2）在 8237A 与 CPU 之间的连线中，注意 ADSTB 和 AEN 信号的连接。简单地说，ADSTB 用于外部地址锁存器的锁存端相连，而 AEN 则与锁存器的允许输出相连，同时AEN 信号还应控制 CPU 的地址锁存器的允许输出。当 8237A 作为主模块时，AEN 有效应允许 8237A 的地址锁存器有效输出，而禁止 CPU 的地址锁存器输出。

（3）8237A 仅提供 16 位地址线，因此在 8086/8088 计算机系统中，为能寻址 1MB存储器空间，最高 4 位地址 A19～A16需在 DMA 传输之前，要用指令将高 4 位地址送到一个 4 位的 I/O 端口中，或采用页面寄存器 74LS670 来实现。

（4）8237A 具有两种优先级管理方式：固定优先级和循环优先级。但无论采用哪种优先级管理方式，一旦某通道的请求获得服务之后，即占有总线控制权后，其他的通道请求均被禁止，直到该通道的服务结束。

注意：8237A 的优先管理不同于中断优先级的管理。

（5）8237A 的 4 种工作模式，除级联传输模式外，其他 3 种模式的不同点都体现在DMA 控制器占有总线后释放总线条件的不同上，基于这一点认识，便不难为不同的应用选择合适的工作模式。

8237A 具有 16 个端口地址，学习时注意各端口的类型及信息格式。

**7 中 断**

7.1 重点内容提要

（一）中断和异常

中断是指由于某个事件的发生（硬件的或软件的），计算机暂停执行当前的程序，转而执行另一程序，以处理发生的事件，处理完毕后又返回原程序继续原作业的过程。通常，把因外部事件而改变程序执行的流程，去处理外部事件的过程叫做中断，也称硬件中断或外部中断；把因内部意外而改变程序执行的流程，以报告出错情况和异常状态的过程叫做异常，也称软件中断或内部中断。中断和异常其实都是中断过程，本章将它们统称为“中断”。

（二）中断源

能引起中断的原因，或是能发出中断请求信号的来源，称为中断源。CPU 外部中断源有 I/O 设备、数据通道、实时时钟和故障源等；内部中断源有 CPU 指令产生了异常（如除 0、溢出、单步调试等），或执行了软件中断指令 INT n。（三）中断类型

根据中断源的不同，中断源可按图 7.1 所示 分类。

可屏蔽中断和非屏蔽中断的区别是：CPU 对可屏蔽中断是否响应要根据标志寄存器中 IF 位来决定，若 IF=1，则响应，否则不响应；非屏蔽中断不受IF的影响，常用于CPU感知突发致命事件。



图 7.1 中断类型

（四）中断优先顺序

CPU 的中断优先权从高到低排列为：除法出错中断、溢出中断，INT n→NMI→ INTR→单步中断。（五）中断类型码的分配

中断类型码是连接中断源和中断处理程序的唯一桥梁。80x86 可处理 256 级中断，中断类型码可以是 0～255，一部分由系统占用；一部分由用户支配。（六）中断处理的一般过程

对各种中断的响应和处理过程是不完全相同的，主要区别在于如何获取相应的中断类型码。对于 INT n、NMI、单步中断等中断类型码是系统规定好的自动形成的。而 INTR，则必须判断 IF 是否为 1，若是，CPU 才响应，并读取该中断源的中断类型码。在获取了中断类型码以后，中断处理过程是一样的。

（1）把 CPU 的标志寄存器压栈保护。

（2）陷阱标志 TF 的状态送入暂存器，并对 TF 和 IF 清 0。

（3）保护断点。将 CS 和 IP 压栈保护。

（4）根据中断类型码取中断服务程序的入口地址。

（5）执行中断服务程序。

（6）恢复断点地址。

（7）恢复标志寄存器的内容。

（8）返回原程序，顺序执行下一条指令，再判断有无中断。（七）中断向量表和中断描述符表

每个中断都分配有中断类型码，CPU 通过中断类型码经过处理要找到对应的中断服务程序的入口地址。在实模式下，由中断向量表将中断类型码和中断服务程序的入口地址联系起来；而在保护模式下，是由中断描述符表来对两者进行联系。

中断向量表设置在系统 RAM 的最低端 00000H～003FFH 的 1KB 内，表中共有 256个中断类型码对应的向量值，每个向量占用 4 个字节。前两个字节为中断服务程序入口地址的偏移值 IP，后两个字节为服务程序的段基址值 CS。这 4 个单元的地址中的最小地址称为向量地址。CPU 可以通过 CS 和 IP 的值得到该中断类型的中断服务程序的入口地址。

中断描述符表（IDT）在内存中的首地址保存在 CPU 内部的 IDTR 系统寄存器中。在 CPU 响应中断过程中，CPU 把中断类型码乘以 8，与 IDTR 中的基地址相加，指示中断描述符表中的某一中断门或陷阱门。CPU 将门描述符中的选择符送 CS 寄存器，并根据选择符中 TI 位从 LDT 或 GDT 中选择一个段描述符，送入 CS 的段描述符寄存器中。这时，由段描述符中的基地址确定中断服务程序的入口地址。

（八）中断向量表的建立

为了让 CPU 响应中断后正确转入中断服务，建立中断向量表是非常重要的。在建立中断向量表时，可以采用以下方法。

（1）绝对地址置入法。

（2）串指令装入法。

（3）DOS 调用法。

（4）直接接入法。