图书在版编目(CIP)数据

文秘手册／金 炜 主编
—企业管理出版社，1996.3 ISBN 7-80001-669-2
Ⅰ.文… Ⅱ.金… Ⅲ.行政—文秘手册

文 秘 手 册

金 炜 主编

出版发行：企业管理出版社出版
开本：787mm × 1092mm 1 ／ 16 印张： 132
字数：2108千字
印刷：1996年3月第1版
版次：1996年3月第1次印刷
书号：ISBN 7-80001-669-2

定价：76.00元（图书共19册）

目　录

文秘微机操作指南之一

文秘微机操作指南之一

第一章　微机系统工作原理

当“蓝色巨人”IBM在20世纪80年代初插足当时几乎被Apple公司的Apple II机垄断的个人计算机市场时，它也没有想到能够那么迅速，那么彻底地超越竞争对手。当年IBM公司推出的PC机凭借的是什么优势？

它凭借的是两点：①优异的性价比。当年Intel公司的准十六位80x86芯片8088与8位的8085和8086及16位的8086相比，性能自然是先进多了，并且8088与外部通信采用8位数据线，因而能够使用以前大量生产的硬件及软件资源，并且价格也比较相宜。②开放的结构设计。IBM公司在推出它的PC机时就公开了几乎全部的技术资料，包括全部的硬件电路图和基本的底层支撑软件BIOS，使得广大的计算机工作人员能够为它进行深入的开发工作。

就这两点，使其以狂风卷残云之势扫荡了整个计算机市场。各种PC机兼容机也纷沓而出。到IBM后来推出PS/2机时，IBM跟随者也是潮拥一般。直到目前为止，IBM PC机（主要是PC/AT机兼容机）仍然以其开放的结构设计，良好的性价比和丰富的软件资源继续占领着个人机市场的绝大部分份额——大约80%以上的个人计算机市场由IBM PC机兼容机占领。

随着技术的进步，目前的主流PC机型已经发展到了采用Pentium，Pentium II，甚至Pentium III等高档芯片的PC机。不过由于它们向下兼容的特性（在原始PC机上能够运行的程序在如今的奔腾机上一样能够运行），它们的总体组成结构还没有太大的变化，要想学习把握采用Pentium，Pentium II，Pentium III等先进芯片的PC机，一个较好的途径还是从采用80486芯片的PC机入手。事实上，高档的80x86芯片虽然引入了更新的工作模式，但是处于实地址工作环境下基本等同一个快速的8086，并且，对计算机各种接口从设计到编程均具有一致的思路和方法。从此人手，逐步深入，无疑是一个良好的循序渐进过程。

所以说，熟悉PC机的结构及工作原理，能够对于掌握更加高档的机器打下一个好的基础。当然在本章的许多地方，讲述的内容已经不只是局限在PC机上了。

第一节　PC机系统构成总述

从外部看，PC机由主机箱、显示器、键盘构成（如果是用作一个计算机系统的话，通常还包含一个输出的硬拷贝设备——打印机），事实上目前几乎所有的通用微机都是这样一种构成，区别也就在于机箱的大小，安放形式，显示器的大小等。

如果有机会打开PC机的机箱，就能看到密密麻麻的电路板、集成块、连线。不过不要被这表面现象搞迷惑了。我们可以把这些复杂器件按照功能分成几类，这样就不会有杂乱

的感觉了，用俗语说，就是“外行看热闹，内行看门道”这句话了。

机箱中的零部件可以分为以下几部分：电源盒、系统板（主板）、主板扩充插卡、硬盘、软驱、一个小喇叭和几个与外部其他部件进行互连的插口。

系统板是计算机的核心部件。该板集中了[illegible]机的核心控制及运算部件以及其他一些辅助控制电路连线。它是一块多层印刷电路板，水平地安装在机箱底部。系统板通过两个六芯接插件连入由电源盒输出的一组电源线，获取电力来源；它的后边有一个圆形的插口用来连接外部标准键盘，接受外部的键盘输入；主板上有两根线接到机箱内的小喇叭上；还有[illegible]个长插槽，用来插入各种具有不同功能的电路板——扩充插卡，对系统进行各种功能扩充，称为扩充插槽。系统板上还有双列直插式标准拨动开关（[illegible]），其开关状态可以人为调整，用以表明机器的硬件配置，它的开关状态可以由程序检测到，因此检测程序可以得知机器硬件配置情况，如系统板上的存储器的容量，显示卡的类型及缺省的工作方式，有多少个软盘驱动器等。有关主板上的器件，更细些讲有[illegible]，[illegible]的[illegible]，[illegible]的[illegible]及其他一些集成电路芯片：时钟发生器芯片，总线控制芯片，定时器芯片，[illegible]控制器芯片，中断控制器芯片，并行输入输出芯片等等，在下面还要分开详细讲述。现今生产出的主板功能更强，具有更高的时钟频率，总线速度更快，并且能够支持不同类型的[illegible]芯片（同一系列），但是容易发现芯片却少了很多，主要是由于集成度提高后，许多控制芯片都集成在了统一的大规模集成电路中。但是不要担心我们以后讲述的器件不存在了，它们只是被浓缩进了更少的部件内部，我们可以使用同样的方法进行控制，程序也一样正常工作。

主板上备有许多扩充插槽，可以插入各类接口卡（或称适配器，就是前面讲述的主机与外设的接口电路），用来扩充主机的输入输出功能。如插入显示适配器就可以得到显示输出，插入软硬盘适配器可以给主机增加外部辅助存储器，插入打印适配器可以连接上打印机，插入异步通讯适配器可以进行远程通讯……。当然仅仅插入扩展卡还不能完全实现卡本身的功能，还需要主机执行相应的控制程序（称为驱动程序），这些驱动程序控制扩充插卡，实现其功能。在原始的[illegible]中，这些适配器大都为单独的插卡形式。不过在目前，由于集成度的提高，人们能够把多个外设接口做在一块插卡上，广泛应用于[illegible]兼容机中的多功能卡，就集成了软硬盘接口、打印并行接口、串行通讯接口和游戏棒接口等多个外设接口，可以同时连接多个外设进行工作。

现在我们可以简单浏览一些基本扩充插卡的连接及功用：显示接口卡（或称为显示适配器）是主机与外边的显示器之间的接口；软硬盘接口卡（软硬盘适配器）用以连接主机与硬盘和软驱；并行打印输出口用来连接并行打印机；串行输出口可连接鼠标和其他串行设备或进行计算机之间的通讯。

第二节　控制核心

[illegible]机的控制核心集成在主机板上，主要由[illegible]，[illegible]时钟发生器，[illegible]总线控

制器构成。

除了 8259A 进行系统控制，还有进行外部中断管理功能的 8259A 中断控制器和进行 DMA 传送控制用的 8237A DMA 控制器也能够归入系统的功能控制部分，它们本身就是对最基础的计算机所做的功能扩充。不过有关中断控制器的讲述将安排在 32 位机的中断系统中，有关 DMA 控制器的讲述将安排在 32 位机的输入输出系统中。这里，仅仅讲述 32 位机中最为核心的控制主机部分 80386 及其周边支持器件。

一、80386 的基本构成

80386 是微处理器家族中的两种第三代微处理器。80386 具有 32 位数据通道可以与外界交换信息，或者说其外部数据总线是 32 位宽的。而 80386 与外界联络的数据通道是 16 位宽的。其他方面，两个微处理器都是相同的。为其中一个 CPU 写的软件可以不加修改地在另外一个 CPU 上执行。原始的 IBM PC 系列机及兼容机上广泛地采用了 80x86 微处理器，所以以下我们仅提到 80386，其实在对 CPU 功能方面所作的描述，对 80x86 同样适用。80386 就功能而言分成两大部分：总线接口单元 BIU（Bus Interface Unit）和执行单元 EU（Execution Unit）。

在这里，我们没有采用以前讲述模型化计算机时，把 CPU 内部划分为控制器和运算器两部分，而是分为总线接口和执行单元，这是出于以下考虑：

（1）在实际的计算机中，内部部件（包括运算器与控制器）与外部总线之间的数据通路是一个很复杂的接口，担负着地址运算，读取指令，存取数据的重任。并且是我们在编程时可以使用汇编指令直接控制的部件——能够在程序中设置一些总线接口部件，如地址寄存器的内容，因此被作为实际 CPU 的一个重要组成部分。

（2）控制器的主体部分——指令译码控制部分对于程序员来说是不可视见的。我们仅仅只要了解它的功能即可，并且它与运算器可以被共同看作是指令的执行器件，它们联合完成指令译码执行功能。

BIU 负责 80386 与存储器和外部设备之间的信息传送。具体地说，一是 BIU 根据程序指针的值负责从内存中指定部位取出指令送到指令流排队机构中排队；二是在执行指令时，所需要的操作数，也由 BIU 根据指令要求的寻址方式负责从内存的指定区域取出，传送给 EU 部分去执行。BIU 的组成分为段寄存器（CS，DS，ES，SS），指令指针 IP，地址加法器和指令队列寄存器等。

EU 部分负责对总线接口部分送来的指令译码执行，包括数学逻辑运算，移位操作，串操作等等；另外还有一些其他的功能管理，如中断管理、总线管理，也由这里的“控制中心”完成。计算机工作的核心部分其实就在这里，我们以前也曾用了很大章节介绍它的工作原理。它主要由下列部件构成：指令译码器、时序控制逻辑、算术逻辑部件 ALU，EU 通用寄存器、标志寄存器等。

这两部分是既分工又合作的两个独立器件，它们的操作是并行的，分别完成取指令和执行指令的任务，所以在 EU 译码执行一条指令的过程中，BIU 可以同时进行下一条或多条指

令的读取工作，把它们放在 悦孕哉内部的指令队列缓冲区中，当 耘孕哉执行完一条指令后就可以不加等待地执行下一条指令，否则执行完一条指令之后再到存储器取指，就会使 悦孕哉处于等待状态（消耗的时间甚至比 悦孕哉进行一般的运算操作还要多），从而提高了 悦孕哉的利用率，加快了程序的执行速度。

当然，需要指出的是，当程序不是顺序往下执行时（如上一条是跳转指令），指令队列缓冲区中预先读取的指令就会失效，需要重新根据上条指令的结果到存储器中取指执行。按照功能划分，愿愿愿悦孕哉的外部引脚可以分为如下几组：

（员）数据线引脚

愿愿愿悦孕哉数据信号线是 酝阅园原酝阅苑一个字节宽，即一次能够与外部进行一个字节数据的交换。

（圆）地址线引脚

愿愿愿悦孕哉的地址信号线是 酝阅猿原酝阅园共 圆园根。我们看到 酝阅园原酝阅苑既要输入输出数据，还要输出地址信号，为了保证不互相干扰，它们是分时使用的。即在某个时钟周期内输入输出数据信号，在另外某些时钟周期内输出地址信号，时间关系是由 悦孕哉的控制逻辑部分安排好的，不会造成冲突。因为在 悦孕哉传送数据时，地址信号与数据信号可能同时要起作用，这就有必要在 悦孕哉的外部附加有地址锁存器，用来暂时将先行从 酝阅苑原酝阅园输出的地址信号锁存起来，保证与在后面的时钟周期中开始发生效用的数据线能够同步工作。酝阅猿原酝阅园也是多功能引脚，因而也需要地址锁存器，而 酝阅猿原酝阅愿没有分时使用，故而在 悦孕哉内部就使用了锁存，在整个总线操作期间均有效，所以仅仅使用一个增加驱动能力的数据缓冲器即可，而用不着锁存器。

（猿）控制线引脚

愿愿愿悦孕哉有两种工作方式：最小模式和最大模式。由它的 酝阅悦猿号引脚所接的电压决定，接高电平 灾则悦孕哉处于最小模式工作；接低电平 灾则悦孕哉处于最大模式工作。鉴于 孕悦悦悦的使用情况，我们只是介绍最大模式的工作情况。

由于以前集成电路制造工艺的限制，使得 愿愿愿芯片的引出管脚数目受到限制而不能太多。这样，愿愿愿工作在最大模式时的许多信号不能全部直接通过芯片外部的引脚输出，而只能通过编码之后使用少量的管脚输出，在外部再经过“译码”后得到原控制信号。愿愿愿就是配合 愿愿愿在最大模式下工作的译码芯片，或者说，愿愿愿最大模式工作必须使用总线控制器 愿愿愿协助。愿愿愿是一个 圆园脚的芯片，与 悦孕哉的连接及输入输出信号。它接收处理器的杂园、杂员、杂圆及时钟信号，产生最终的存储器读写和 陨辕阅读写命令信号、中断响应信号，并对 悦孕哉与总线连接的数据收发器、地址锁存器的工作进行控制。

为了保证系统操作的同步协同性，时钟脉冲信号在任何计算机系统中都是必要的，愿愿愿芯片内部没有设计时钟产生电路，而是通过 员愿脚引入外部时钟发生器产生的时钟信号。

愿愿愿悦孕哉能够处理各种中断请求，有关这方面的控制信号线有 员愿脚不可屏蔽中断请求输入线 晕酝陨；员怨脚可屏蔽中断请求输入线 陨晕吁；表征 悦孕哉是否响应目前发生中断的中断响应输出线 陨晕粤是通过 愿愿愿对 杂园、杂员、杂圆译码后产生的。

愿愿愿裙裁为与其他的总线控制器件协同工作，还提供了下面四类信号：圆原圆脚的 匝匝员匝匝是 悦裁内部指令队列状态输出信号，使得辅助处理器能够跟踪 悦裁的指令队列。猿员和猿圆脚的 砸匝辕蕴园砸匝辕蕴员未标上划线）用于裁决总线的使用权，输入低电平表示外部器件请求使用总线，输出低电平表示 悦裁同意让出总线。若二线同时有请求，则园优先。圆缘脚为测试信号输入端，悦裁在执行指令时，每隔 缘个时钟周期就测试此端，若为低就继续执行下面指令，否则等待。通常该信号与协处理器的 月栽猿端相连，用于两者的协同工作。圆愿脚是锁定信号输出端，若执行的指令带有一个字节的前缀 缄酝月运，那么在执行此指令过程中，此输出端输出信号一直保持为低，此信号一般送到总线仲裁电路，使该指令的执行期间不得发生总线控制权的转让而使该指令能够被连续执行完。

员怨脚是 愿愿愿裙裁的复位端，用于 悦裁的复位操作。

圆园脚为准备就绪信号输入端（砸藻葬凿赠），当 悦裁检测到该端信号为低时，就处于等待状态一个时钟时间。一般用来与速度慢的存储器等协同工作。

此外，悦裁还有电源引脚，用来获取电力供应。

二、愿愿愿的存储器分段结构与地址信号的形成

愿愿愿内部的寄存器都是 员远位的，而处理器输出的是 圆园位地址线，那么这 圆园位地址线上的信号是怎样形成的呢？愿愿愿裙裁采用的是地址分段的方法。

愿愿愿内部有四个 员远位的段寄存器，悦裁在形成向外输出的 圆园位地址线时，把段寄存器的 员远位内容补上四个零（二进制位），这就形成了一个 圆园位的二进制数据，然后再加上一个员远位的二进制数（孕悦机中这个 员远位数可以是一个 员远位寄存器的内容，还可能直接是一个员远位的二进制数），输出到外部就形成 圆园位的最终地址信号。

我们把参与相加的段寄存器的内容称为段值，参与相加的 员远位二进制数称为段内偏移量。由此可以看出，当保持段值不变，仅仅改变段偏移量，就可以改变输出的 圆园位地址。我们把仅仅通过改变段偏移量所能够寻址的空间称为一个段。因为 员远位二进制数可表示的数的范围为 园原（远源园原员，所以一个段的长度为 远源园远 并且在段偏移量值为 园时称为该段的段首，其值为（远源园原员时称为段尾。因为段寄存器的内容在形成真实地址时要右移 源位，所以它每改变 员形成的输出到外部的真实物理地址信号变化 员远故每个段的段首并不是可以从任何地方开始，而是从 员远的整数倍处开始。

显然，每个存储器单元仅对应一个确定的 圆园位二进制数地址，但是由于 愿愿愿裙裁形成地址信号的方式，使得一个固定的单元地址可以由不同的段值与段偏移量搭配形成。比如段值为 园园园园日，段偏移量为 园园园园日，与段值为 园园园员日，段偏移量为 园园园园日所指示的地址其实是一个。

三、愿愿愿的内部寄存器

愿愿愿内部有 员源个寄存器，用于提供辅助运算，控制指令执行，处理内存寻址等功能。它们都是 员远位寄存器。

这些寄存器按照功能可以分为六类：

（1）通用数据寄存器。这些寄存器用作一般用途，比如字符串操作、算术运算、循环移位、输入输出数据。我们使用 AX、BX、CX、DX 代表它们。实际使用时，每个寄存器还可以分成高 8 位和低 8 位分别使用，即 AH、AL、BH、BL、CH、CL、DH、DL。这些寄存器在实际工作中，是使用得最为频繁的。操作数一般要先送到这些寄存器中才能进行别的操作。

（2）段寄存器。我们已经讲到，8086 CPU 中 20 位地址信号是有两部分——段值与偏移量共同形成。此处的段寄存器就是用来存放其中的段地址的 16 位寄存器。段寄存器包括代码段寄存器 CS、数据段寄存器 DS、堆栈段寄存器 SS 和附加段寄存器 ES 四个，分别用于程序中不同存储区域的寻址。当 8086 输出地址信号时，将能够自动或按照指令要求利用四个段寄存器的内容作为段地址形成地址信号。

（3）堆栈指针寄存器 SP。一般情况下，我们使用堆栈时，仅仅使用压入和弹出指令，在 8086 机使用的汇编语言中，汇编指令为 PUSH 和 POP，没有直接指明地址，但是 8086 机的堆栈位于主存中，8086 执行这些堆栈操作指令时，因为要对内存进行读写操作，所以一定要使用地址。这个地址（栈顶地址）就是 8086 按照当前堆栈段寄存器 SS 与堆栈指针寄存器 SP 中的内容共同形成的，并且使用一次 PUSH、POP，SP 内容自动减 2。需要指出的是，由于使用是自动进行的，不需要亲自处理地址问题，所以在使用堆栈操作时，往往忽视了堆栈的实际位置和大小，导致与别的程序或数据产生重叠（称为堆栈溢出）。比如堆栈底端与一块别的数据或程序仅仅相距 2 个字节，那么连续使用 3 次 PUSH 指令后，存入堆栈的数据就与其他数据发生了重叠。实际上由于中断程序也要占用堆栈，恐怕用不了 3 次操作堆栈就会发生溢出现象了。这一点在高级语言编程中（主要是 C 语言）尤其要注意，因为高级语言中子程序的变量一般是安排在堆栈中，不注意的话，很容易就导致由于堆栈溢出而死机的情况的发生。

（4）指令指针寄存器 IP。用来指示当前执行程序指令的地址，或者说，8086 依据它的内容决定将要取出执行存储器中哪一单元存放的指令。由于 8086 机上真实地址的形成方式是段寄存器与段偏移量按一定方式相加产生的，故实际工作中不仅仅是由一个指令指针 IP 决定，而是由代码段段寄存器 CS 与指令指针寄存器 IP 的内容共同决定程序执行方向。

（5）基数指针寄存器 BP 与变址寄存器 SI、DI。BP、SI、DI 都可以存放操作数的偏移地址。这几个寄存器一般用作存储器中操作数的寻址，基数指针主要用作堆栈中的数据寻址，而变址寄存器用作常规内存数据块传送。另外，变址寄存器可以像通用数据寄存器一样在运算过程中存放操作数（不过不能分成两部分使用，只能以 16 位的一个字使用）。

（6）标志寄存器。程序有时需要依据上一个指令执行的结果来判断往下执行的指令位置，为此 8086 中均设置一个标志 8086 运行状态的寄存器，通常称为标志寄存器 FLAGS。它也是 16 位的寄存器，但是只使用了一些位。

四、有关 愿愿愿地址空间使用的进一步说明

如上所述，愿愿愿系统中，每个存储单元均有两种地址，一种就是一个 圆圆位的二进制数，它可以代表信息在存储器中存放的实际位置，我们称之为物理地址；另外一种地址使用段地址、偏移量的表示方式，它也代表了 愿愿愿的分段地址空间中的一个存储单元，我们称之为逻辑地址。我们与 悦悦悦打交道，使用低级语言，如汇编编程时，要处理地址问题均是使用逻辑地址，悦悦悦在进行操作时总线接口部件 月月月将逻辑地址自动转换为与外部存储器相连的 圆圆位物理地址。

愿愿愿悦悦悦内部有四个段寄存器，它们均可以被用来作为段地址形成 圆圆位的物理地址，但是 悦悦悦把它们用于不同用途上。在到存储器中取指执行时使用的 圆圆位物理地址总是由悦悦悦代码段寄存器与 阅阅指令指针形成的。在使用堆栈操作指令时，悦悦悦总是使用 杂杂堆栈段寄存器与 杂杂堆栈指针形成 圆圆位物理存储器地址。只是在寻址存储器中各种数据时，比较灵活，可以使用四个段寄存器的的任何一个作为段地址，但是 悦悦悦在进行数据寻址时有一个约定，如果汇编指令中不明确表明使用哪一个段寄存器，则缺省使用 阅阅数据段寄存器。实际工作中，由于 悦悦悦杂杂杂均各有用途，故极少使用它们寻址数据。耘悦悦也可以用来在内存中进行数据的寻址工作。

实际上，一个程序一般分为 猿个部分：代码部分、数据部分、堆栈部分。由于 悦悦悦在寻址这些部分时，缺省使用不同的段寄存器，所以程序员在编制低级语言程序中，可以通过控制各个段寄存器内容，决定程序这些部分在内存中的位置和大小。一般情况下，各段在存储器的分配是由操作系统负责的。每个段可以独立地占用 远源远字节存储器，也允许重叠，这时各段就可能没有 远源远长。

五、愿愿愿指令系统介绍

任何一个 悦悦悦都有一个指令集合，每个指令是它能够识别并执行的一个操作。指令在计算机内部表现为一个个的二进制数，依指令长度的不同，可以分为单字节指令和多字节指令。更为广泛的分类方式是依据指令的功能进行。

愿愿愿的指令系统可以分为以下几个功能组：

（员）数据传送；

（圆）算术运算；

（猿）逻辑运算；

（源）串操作；

（缘）程序控制；

（远）处理器控制。

由于不是专门介绍汇编编程的书，所以这里仅就其中一部分基本且重要的指令加以介绍，我们可能会在后面讲述接口控制时使用到它们。

员、数据传送指令

数据传送指令包括 酷宏指令，交换指令 载波郎地址传送指令，堆栈操作指令，输入输出指令，标志位传送指令几类。这里只挑出几个简要介绍。

（员）酷宏指令

使用格式 酷宏参数 员，参数 圆

酷宏是操作码，参数 员和参数 圆分别是进行数据传递的目的操作数和源操作数。本指令把一个字节或一个字的操作数从源传送至目的。

需要指出的是存储器之间、段寄存器之间不能互相传送。要想进行这种传送，需要使用中间媒介，如使用通用寄存器作为桥梁。下面两个语句实现把 恍杂段寄存器的内容送到 耘杂段寄存器：

酷宏 粤载，恍杂

酷宏 耘杂，粤载

上面凡是涉及到存储器传递的地方均没有明确写出地址表示，这是因为涉及到存储器操作时，有一个存储器寻址问题。由于 愿遵愿循载的地址分段方式，所以存储器寻址比寄存器寻址和立即数寻址复杂得多，计算机提供的寻址方式的多少一般也就体现在存储器的各种寻址方法上。下面我们以存储器与通用寄存器之间进行数据传送为例，介绍各种存储器地址寻址方式：

①操作数地址的 员远位偏移量直接包含在指令中，同时用方括号括起来：

酷宏 粤载，[圆圆圆圆]

省略段寄存器的说明则 恍载自动使用 阅杂段寄存器，即上语句实际效果等效于：

酷宏 粤载，阅杂：[圆圆圆圆]

也可以明确说明使用到哪个段寄存器：

酷宏 粤载，恍杂：[圆圆圆圆]

这种方式称为直接寻址。

②使用寄存器说明操作数的内存地址，或说操作数的内存地址的偏移量用寄存器的内容表示。寄存器是 杂阅酝阅月孕，月载之一，并用方括号括起来：

酷宏 粤载，[杂阅]

这种方式寻址操作数时，使用不同的寄存器将对应与不同的段寄存器，当使用 杂阅酝阅月载时，恍载使用 阅杂段寄存器的内容作为段值，使用 月孕时，恍载使用 耘杂段寄存器的内容作为段值。当然，如果明确指出段寄存器，恍载就按指出的段寄存器作为段值：

酷宏 粤载，耘杂：[月孕]

这种方式称为寄存器间接寻址。

③把①，②两种寻址方式两项合并，得到这种方式：

酷宏 粤载，圆圆圆圆[杂阅]

圆圆圆圆为立即数，杂阅为②项中所述的 源个寄存器之一。两者相加作为偏移量，段寄存器的使用与②一致，可以使用缺省的，也可以直接指定。

这种方式称为变址寻址。

④在 [illegible]中，通常把 [illegible]和 [illegible]称为基址寄存器，把 [illegible]称为变址寄存器，两者结合起来可以用作存储器寻址：

[illegible]

现在是三者相加形成偏移量，段寄存器缺省由使用的基址寄存器决定。如果其中使用基址寄存器使用 [illegible]则缺省的段寄存器为 [illegible]；如果使用的基址寄存器为 [illegible]则缺省的段寄存器为 [illegible]。另外也可以明确指出所使用的段寄存器。

这种寻址称为基址加变址寻址。

这些有关存储器的寻址方式，同样可以用于立即数到寄存器的传送，存储器与段寄存器的传送。如：

[illegible]

[illegible]

等等。不过要注意的是代码段寄存器 [illegible]的赋值需要通过通用寄存器而不能直接传送。

(圆堆栈操作指令

[illegible]的堆栈操作指令就是两条：[illegible]和 [illegible]，分别是数据进栈和出栈指令。使用格式为：[illegible]和 [illegible]。其中 [illegible]可以按照上述的操作数寻址方式使用。常见的情况 [illegible]就是寄存器名字，如 [illegible]

还有两条有关标志寄存器的进出栈指令：[illegible]，一般用来在中断发生保存标志寄存器之值。

(猿输入输出指令

输入输出指令用于 [illegible]端口与 [illegible]之间的通信，由 [illegible]和 [illegible]组成；其中 [illegible]指令完成端口到寄存器 [illegible](员个字)或 [illegible]一个字节)的数据传送。[illegible]输出 [illegible]或 [illegible]的内容到端口。这里需要注意的是，[illegible]方进行通信用的寄存器只能是累加器([illegible]或它的低 [illegible]位 [illegible]。

[illegible]端口使用端口地址指定，[illegible]的端口编址只使用了 [illegible]位地址(即相当于不使用段寄存器，或其值为 [illegible]，端口地址范围为 [illegible]，其中前 [illegible]个地址可以直接在指令中指定。如果端口地址超过 [illegible]时，就需要使用 [illegible]寄存器来指定端口地址：先把端口地址送入 [illegible]寄存器，然后进行输入输出工作。

指令格式如下([illegible]和 [illegible]仅数据传送方向不一样，故这里仅列出 [illegible]的格式)：

①[illegible] [illegible]，[illegible]

这里 [illegible]是 [illegible]的一个数，直接指出端口地址，前面使用 [illegible]时，仅传送 [illegible]地址处一个字节内容到 [illegible]；如果使用 [illegible]则还把 [illegible]+员地址处端口内容传送到 [illegible]中，共传送一个字(员个[illegible])的内容到 [illegible]寄存器。

②[illegible] [illegible]，[illegible]

与上一条指令区别在于端口地址使用 [illegible]指定，[illegible]是 [illegible]位寄存器，所以可以指定 [illegible]端口中的任何一个，传送内容如上所示。

[illegible]算术运算指令

[illegible]提供加减乘除四种基本算术运算。这些操作都可以用于字节运算或字(两个字

节,或一个寄存器长度数)运算。其中参加运算的数据可以是带符号数,也可以是无符号数。下面仅把加减发指令作一个简单说明。

(员加法指令 :粤源院参数 员,参数 圆

此指令完成两个操作数相加,然后把结果送到参数 员保存,即 :参数 员←参数 员+参数 圆

(圆减法指令 :杂哉月参数 员,参数 圆

此指令完成两个操作数的减法运算,结果存放在第一个参数处。

猿.逻辑运算及移位指令

(员逻辑非指令 :晕韵栽参数,操作数 的址寻址取反→的址寻址,结果不影响标志位。

(圆逻辑与指令 :粤晕阅 的址寻址员,的址寻址圆或者 的址寻址员,的址寻址圆→的址寻址员,结果影响标志位。

(猿逻辑或指令 :的耘 的址寻址员,的址寻址圆或者 的址寻址员,的址寻址圆→的址寻址员,结果影响标志位。

(源异或指令 :辕韵耘 的址寻址员,的址寻址圆或者 的址寻址员,的址寻址圆→的址寻址员,结果影响标志位。

(缘移位指令 :算术/逻辑左移 杂匀粤/蚤蕴 杂匀蕴 的址寻址,皂,将操作数 的址寻址左移 皂位。每移动一次 的址寻址的最低位补 园,最高位进入标志寄存器的 悄云位。操作也将影响其他标志位。此外还有算术右移指令 杂匀砸,逻辑右移指令和几条循环移位指令 砸韵造等。

源.串操作指令

这里的串指字符串,串操作是指计算字符串的长度,在字符串中查找某一个字符,改换字符,判断两个串是否一致等等。愿惫愿中有一些串操作指令,完成基本的字符串操作:

(员串传送指令 :贼韵灾 杂的址寻址员,的址寻址圆;

(圆串比较指令 :悄酝责 杂的址寻址员,的址寻址圆;

(猿串搜索指令 :杂悦粤杂 杂的址寻址;

(源取串指令 :造耘essage 杂的址寻址;

(缘存串指令 :杂贼韵则 杂的址寻址;

缘.程序控制指令

程序控制指令也称为程序计数器控制指令,这类指令能够改变程序计数器的值,使程序不再按顺序执行,而是发生一次跳跃。程序控制指令包括程序转移指令、子程序调用指令和中断指令三类。

(员转移指令

转移指令又分为无条件转移指令和条件转移指令两类。

①无条件转移指令

指令格式 :悦粤责 的址寻址

其中 的址寻址是汇编程序的一个语句标号或存储器地址或寄存器。此指令使用按照 的址寻址寻址方式确定的地址内容改变 陨孕(段内转移)或 陨孕和 悄杂(段间转移)的内容,从而使程序控制转移到新的入口。

②条件转移指令

愿惫愿有一系列的条件转移指令,一般是以 愿惫愿状态寄存器内部的标志寄存器的某些位的状态作为逻辑判断依据,若满足指令规定的条件则程序转移,否则程序继续按照顺序执行。这些

条件转移指令的转移目标位置不能太远，要求在本转移指令后面一条指令的 [illegible] 字节之内，即使用一个字节整数作为转移的偏移量。下面就是几个单标志位的条件转移指令：

[illegible] 标志位 [illegible] 为 [illegible] 时转移到 [illegible] 为一个语句标号。

[illegible] 为 [illegible] 时转移到 [illegible]

[illegible] 为 [illegible] 时转移到 [illegible]

例子：[illegible] [illegible]；[illegible] 与 [illegible] 的内容相加，结果送 [illegible]

[illegible] 为 [illegible] 时转移到 [illegible]，把 [illegible] 赋值为 [illegible]

[illegible] 为 [illegible]，结果不为 [illegible] 时，把 [illegible] 赋值 [illegible]

[illegible]

[illegible]

[illegible]

(3) 调用和返回指令

调用指令用于在一个程序中插入执行另外一个程序。指令格式为：

[illegible]

[illegible] 指示目标程序的起始地址。

当程序执行到调用指令时，保存当前断点地址 [illegible] 压入堆栈，段内调用仅保存 [illegible] 内容)，然后把调用子程序的起始地址(按照 [illegible] 的寻址方式确定)送入 [illegible] 和 [illegible]；段间调用，地址是两个字——段值：偏移量，分别送入 [illegible] 和 [illegible]；如果是段内调用，地址仅是 16 位的偏移量，只送入 [illegible] 即可)。

子程序执行到最后使用调用返回指令 [illegible]，执行此指令后，保存在堆栈中的 [illegible] 和 [illegible] 重新送回，程序继续执行 [illegible] 指令下面的内容。

(4) 中断指令

[illegible] 有三条中断操作指令：软件中断 [illegible]，溢出中断 [illegible]，中断返回指令 [illegible]。在 [illegible] 中断系统中将详细讲述。

(5) 处理器控制指令

[illegible] 提供的这些指令允许程序控制各种 [illegible] 功能，如修改 [illegible] 的标志位，与外部事件同步工作以及空操作指令等。

(6) 标志操作指令

有些指令的执行会影响到 [illegible] 内部的标志寄存器的内容，而这里提供的指令可以直接按位控制标志寄存器内容：

[illegible] 进位位置 1；[illegible] 进位位求反；[illegible] 进位位置 0

[illegible] 方向标志置 1；[illegible] 方向标志置 0

[illegible] 中断标志置 1；[illegible] 中断标志置 0

(7) 其他控制指令

[illegible]，空操作指令，占用一个字节，不执行任何操作，用于调试程序。

与 [illegible] 停机指令，使机器暂停指令的执行，等待 [illegible] 信号或一次外部中断信号的到来，中断结束后继续执行下面程序。

[illegible] 等待指令，执行后处于 [illegible] 状态，与 [illegible] 指令类似，不同的是在中断返回后重新处于 [illegible] 状态继续等待。

[illegible] 封锁指令，用于总线控制。

六、[illegible] 汇编语言编程简介

[illegible] 汇编语言介绍

汇编语言的思路就是用一组字母、数字、符号来代替一条条用二进制码表示的指令（称为指令助记符），通过把一系列的指令助记符排列起来，这就形成了一个能够完成一定功能的汇编程序。

汇编程序中使用指令助记符来代替计算机指令代码，使我们不必强记那一个个毫无规律可言的二进制数学表示的指令代码，给我们实际编程以极大的方便。比如我们使用一条汇编语句 [illegible] 表示对 [illegible] 这个通用数据寄存器的内容加 [illegible]，如果写成机器内部的指令代码（机器语言），就是没有规律的两个字节 [illegible]——这样记下来实在太费劲了！

指令语句的一般格式为：

标号： 指令助记符 参数，参数 ；注释语句

在我们看来，指令助记符代表着 [illegible] 进行的操作，而后面的操作数在不同的指令中可能有两个，也可能只有一个，还可能一个也没有，主要是看该步的操作内容是否需要明确指出一个或几个操作数。如 [illegible] 指令使 [illegible] 内部标志寄存器中表征 [illegible] 对外部中断是否响应的 [illegible] 位置为 [illegible]，这样就允许 [illegible] 对 [illegible] 端的中断请求作出反应，这条指令就没有列出明显的操作数。像 [illegible] 这样的指令后面就有 [illegible] 这个寄存器的名字表示对它加 [illegible]，而 [illegible] 则使用两个参数 [illegible]，[illegible]，表示把 [illegible] 寄存器的内容减去 [illegible] 寄存器的内容，并将结果存放在 [illegible] 寄存器中。

我们还看到，在汇编程序中可以使用了一个标号放在汇编语句前面，并用冒号把它们隔开，其实在各种高级语言中，也有这种使用方法。这个标号是与这条指令所处的存储器地址联系着的，或者说，这个标号代表了该条指令的地址。我们可以使用标号作为各种转移指令如 [illegible] 跳转指令和 [illegible] 程序调用指令的操作数："[illegible] 标号" 和 "[illegible] 标号"。"[illegible] 标号" 表示程序不再顺序往下执行，而是从 [illegible] 处标号指令直接转移到标号所处的指令行开始执行。显然，不使用程序转移指令的指令行其标号可以不要，另外一个程序中不能有两个指令行的标号一样，这样做使得编译程序在编译时不能分辨这个标号到底代表哪一个地址。

在语句后面可以写上一句注释语句，注释语句使用一个分号与前面指令语句分开。实际上，任何地方分号后面的语句将作为注释而对程序的执行不起作用。

在实际的汇编语言程序中，除了使用以上介绍的指令语句还有伪指令、宏指令等语句格式。伪指令一般用来定义使用的数据，并为之分配存储器空间。在 [illegible] 的汇编程序设计中还用来进行段定义。程序指令的定位，一个过程（子程序）也使用伪指令定义。宏指令主要

是把一组常用的指令用一个符号(宏指令)代替,这样可以缩短程序长度,编程方便。编译时编译程序还是要把该宏指令换成它代替的指令集合后再编译。

圆汇编语言程序设计

随便编制一个程序是比较简单的,而设计一个好的程序就不是简单的工作。好程序不仅要完成预定功能,还要满足一些别的要求 如结构化,简明易懂,执行速度高,长度短等。

目前我们的微机上有各种高级语言可以使用,是否采用汇编语言是一个值得考虑的问题。一般情况下我们是不用汇编语言进行程序设计的。但是在某些时候,如编写实时控制系统,智能化仪器仪表,以及高性能软件方面,汇编语言是一种不可缺少的工具。当然,对于了解计算机的详细工作原理,汇编语言也是不可缺少的,因为计算机各个构成器件的工作可以直观简单地使用汇编语言进行控制。

在实际设计汇编语言程序时,首先要对完成的任务或实现的目的有一个透彻了解,如果比较复杂,还要确定算法,然后根据上述思想绘制一个程序的流程图(即使不画在纸上,也起码要在大脑中有一个流程)。这个图还可以帮助他人理解程序,尤其对以后自己回头查看程序时有的大的帮助。

在开始编制程序时,决定存储空间的分配和使用,因为 愿要对存储结构的分段使用,因此需要定义各个段,分配变量存储空间。然后开始具体工作的处理。还有许多具体的问题需要注意,如参数的传递,悦载内部寄存器的充分应用等。

猿汇编语言使用过程

我们编制好汇编语言程序后,要在机器上运行,需要以下几个步骤来做:

(员运行一个文本编辑程序(比如 阅编软提供的 耘则载或 宰贼则载等),建立并修改源程序。

(圆使用汇编程序(如 酝糟辑藏公司的 配编辑或 月则载公司的 栽则载编辑等)把编辑好的源程序进行汇编,得到目标程序(扩展名为 韵月允。

(猿使用 在则载运程序处理上述目标程序,得到最终的执行文件(扩展名为 耘则载。

(源键入程序的文件名(可以省略扩展名)即可运行该程序。

第三节　孕悦机的主存空间分配

计算机的主存就象一套大房子,能够容纳大量的程序代码和数据。悦载是它的使用和管理者(其实还是我们编制的程序控制者和存储空间的使用者),它与存储器通过总线连接在一起进行通信。

悦载通过地址总线输出地址信号到存储器选择存储单元,通过数据总线与选中的存储单元交换数据。悦载输出地址总线的宽度决定了它能够识别的存储器的容量或该 悦载主存空间的大小,地址总线上信号变化的范围称为 悦载的寻址空间,以前的 悦载芯片地址线较少,早期的 愿位机寻址空间仅为 远源(员远根地址线),每个单元装配一个字节存储器,可以