6 微电子学讲座

〔日〕石田晴久 著

微型计算机程序设计

科学出版社

微电子学讲座 6

微型计算机程序设计

〔日〕石田晴久 著

吕景瑜 译

科学出版社

1988

内 容 简 介

本书以 8086 微型计算机为例，详尽地介绍了汇编语言程序设计方法，同时对几种高级语言的特点以及它们与汇编语言程序的连接进行了阐述，对 MS-DOS 操作系统和软件开发工具也作了具体说明.

全书共分八章. 第一章概述微型计算机的基本组成. 第二至五章详细介绍汇编语言程序设计方法，并给出在生产厂家的手册等资料中所没有的详细技术资料.第六章以输入/输出程序为中心，概要地介绍 FORTRAN77，Pascal，C，Ada，LISP，PROLOG，APL 等高级语言的特点. 第七章阐述几种高级语言同汇编语言程序的连接方法. 第八章以 MS-DOS 为例，具体介绍操作系统和软件开发工具.

本书内容新颖、实用且通俗易懂，书中给出的实例大都在实际中运行过.

本书可供从事计算机工程与应用的科技人员阅读，也可作为高等院校有关专业的教学参考书.

石田晴久著

岩波講座マイクロエレクトロニクス6

マイクロコンピュータのプログラミング

岩波書店，1984

微电子学讲座 6

微型计算机程序设计

〔日〕石田晴久 著

吕景瑜 译

责任编辑 孙月湘

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行　　各地新华书店经售

1988 年 2 月第 一 版　开本：850×1168　1/32

1988 年 2 月第一次印刷　印张：6 5/8

印数：0001—5,000　字数：170,000

ISBN 7-03-000181-8/TP·12

定价：1.90 元

译 者 的 话

自美国 Intel 公司于 1971 年制造出第一台 4 位微处理器 4004 以来，微处理器已有十多年的发展历史．在这十多年中，随着微处理器应用的越加广泛，其本身也在不断地更新换代．继 4 位、8 位微处理器之后，又推出 16 位、32 位微处理器，为了更好地应用这些新型微处理器，人们迫切希望掌握微型计算机的程序设计方法．

为了满足广大读者的需要，我们将日本岩波微电子学讲座第六卷《微型计算机程序设计》（マイクロコンピュータのプログラミング）一书译出．该书以 8086 微型计算机为例，详尽地介绍了汇编语言程序设计方法，扼要地介绍了 FORTRAN，Pascal，Ada，C，LISP，PROLOG，APL 等高级语言的特点以及这几种高级语言与汇编语言程序的连接方法，同时还概述了 MS-DOS 操作系统和软件开发工具．

在翻译本书的过程中，曾得到北京计算机学院王晖、毛晓宾等同志的热情帮助，在此向他们表示衷心的感谢．

由于译者水平所限，书中疏漏之处在所难免，敬请读者批评指正．

吕景瑜

1986 年 12 月

原编者的话

随着大规模集成电路（LSI）技术的不断发展，迎来了超大规模集成电路（VLSI）的时代．现在一个大规模的系统已经能够集成在一个 VLSI 芯片内，因而，面向 VLSI 的系统设计技术以及把这些 VLSI 组合起来构成巨大系统的技术，在目前均被看作是微电子学的一部分．

本讲座提到的微电子学就是这种广义的微电子学．VLSI 的发展将给今后的社会带来巨大的影响，因而，本讲座的目的在于集中介绍设计制作 VLSI、VLSI 计算机、通信、机械电子学等信息系统所必须的理论和技术．

但是，LSI 技术是一个诞生时间不长，并且发展十分迅速的领域，至于它将以何种形态向前发展尚存在着许多可能性，所以若想把它作为已确立的学科体系加以总结，还有许多困难．因此，本讲座的目的是：

（1）帮助关心微电子学的初学者和准备应用微电子学的读者入门．

（2）作为在计算机、通信、机械电子学领域工作的技术人员、研究人员的实践指南．

（3）把掌握微电子学所必须的基础知识和技术明确化、系统化．

（4）把微电子学作为一个新的学科进行系统化，并使其成为向理想教学用书迈进的里程碑．

希望本讲座成为学生、技术人员、研究人员的良师益友．

元冈 达　菅野卓雄

渡边 诚　渊 一博　石井威望

前　言

从某种意义上说，本书是微电子学讲座第五卷《微型计算机硬件》(森下岩著，マイクロコンピュータのハードウエア)的姊妹篇．在硬件篇中，重点介绍了各种微型计算机设计上的新思想．本书是软件篇，它与硬件篇的不同之处在于，集中地介绍了一个典型微型计算机——8086，重点放在其基本程序设计方法上．两书在写法上的不同，反映了作者编著方针的不同．应该说，这样做正反映了硬件和软件相对照的特点．

硬件发展迅速，而软件没有多大变化，也就是说软件进步不大，这种状况即使在今天也是如此．特别是微处理器，以前要用几片大规模集成电路（LSI）才能实现，而现在只用1片LSI即可实现．芯片的形状及向外引出的信号线的作用也在改变，同样形式的存储器芯片的存储容量一举提高了三倍，电路技术的发展促使LSI的速度成倍提高，这些都说明了硬件技术的发展极为迅速．但是，如果仔细研究一下硬件就会发现，它与软件密切相关的部分，例如用微处理器能实现的指令系统，由于功能的强化，虽然在不断地扩展，但其基本内容却没有改变．这是因为要维持软件的兼容性，所以基本内容想改变也改变不了．

最能说明这一事实的是通用大型计算机和VAX-11系列超小型计算机．通用大型计算机起源于IBM360/370，它们与IBM计算机保持兼容，发表于1964年；VAX-11系列是超小型计算机的代表，发表于1978年．但作为全部程序设计基础的指令系统，其基本内容至今没有改变．与此不同，微型计算机从8位转向16位时，可以说在微型计算机行业中人们犯了一个错误，即没有充分考虑软件的向上兼容性．因此，在16位微型计算机中，不能使用以前8位微型计算机的软件．这样一来，迫使人们必须按照新规则开发所有的软件，结果延缓了16位微型计算机的普及．

当然，为了技术的进步，抛弃兼容性也是必要的；但是如果从16位机软件储备在增加这一现状来看，今后从16位机转向32位机时，强烈要求保持其兼容性是很正确的.

本书之所以选定当前最常用的8086微处理器作为微型计算机的模型机，原因就在于此. 对于典型的微处理器来说，如能充分理解并牢固地掌握该机的程序设计基础，那么从该机获得的知识和技能也会很容易地在其他微型计算机中应用；尤其宝贵的是，这些知识和技能没有老化. 就上述意义来讲，学习程序设计类似于学习语言学.

可是，实际上要想理解程序设计技巧并通过程序设计理解微型计算机原理，则必须知道其细节. 一般说来，软件是通过一些微小要素的自由组合来共同完成一个复杂功能的. 在原理上，具有"用软件可以实现一切"的一面，但是微小的要素接近无穷，其组合方法是千变万化的.

程序设计基础常常使人感到枯燥无味，因此本书尽量用实际例子来说明. 如果读者手头有典型的个人计算机，则可立刻试一试. 在第二—五章中，列举了指令字一级的程序设计. 在这些程序设计中公开发表了厂家提供的手册等资料中所没有的详细技术资料. 这样的程序设计好象很麻烦，但就象解难题一样，如果掌握了窍门，是非常有趣的.

考虑到高级语言的重要性，本书在第六章以其他书中没有介绍过的输入输出程序为中心，对BASIC，FORTRAN，Pascal，C，APL等高级语言作了介绍. 此外，在第七章中介绍了上述高级语言程序同指令字(汇编语言)程序的连接方法，这些方法在实际中是非常重要的. 最后，在第八章中，以MS-DOS为模型，对操作系统的功能和软件开发工具进行了说明. 为了高效率地开发软件，本书的一个主张是：需要开发方便的工具.

从本书计划撰稿直至成书，岩波书店编辑部提供了各种帮助，对此，作者表示衷心感谢.

石田晴久　1984年5月

目　录

第一章　从软件看微型计算机

由于可作为开发工具使用的个人计算机（其核心部分是微型计算机，简称为微型机）的普及，因而微型计算机的软件也可以用BASIC，FORTRAN，Pascal，C 等高级语言来编制，但要想理解微型计算机的原理，或最大限度地发挥其性能，或由它来管理外部设备时，就要使用微型计算机的机器语言级，即指令级进行程序设计.

从用指令级进行程序设计的观点看，对于微型计算机的硬件，首先应知道微型计算机的内部有哪些寄存器，在执行指令的过程中怎样使用这些寄存器以及存取主存储器内的指令和数据时有哪些寻址(地址)方式等. 本章以 8086 作为典型的微处理器，重点介绍作为现代概念之一的段结构.

此外，从提高软件开发效率的观点看，最重要的是对输入输出控制方式的标准化和程序的通用化，这是因为不同机种之间的输入输出控制方式往往不一致. 本章的最后部分还涉及到由于采用了微型机的标准操作系统（如 MS-DOS），使得实现这样的通用化成为可能.

1.1　微处理器的构成

微型计算机(微型机)或个人计算机(个人机)的构成一般如图 1.1 所示，其中央处理装置（CPU）是微处理器. 在微型计算机中，一般执行任务的步骤如下：

（1）从键盘或辅助存储器将程序和数据送入主存储器.

（2）微处理器从主存储器将程序中的指令一条一条地取出并加以执行. 这时，如果需要的话，还要从主存储器取出数据加以使

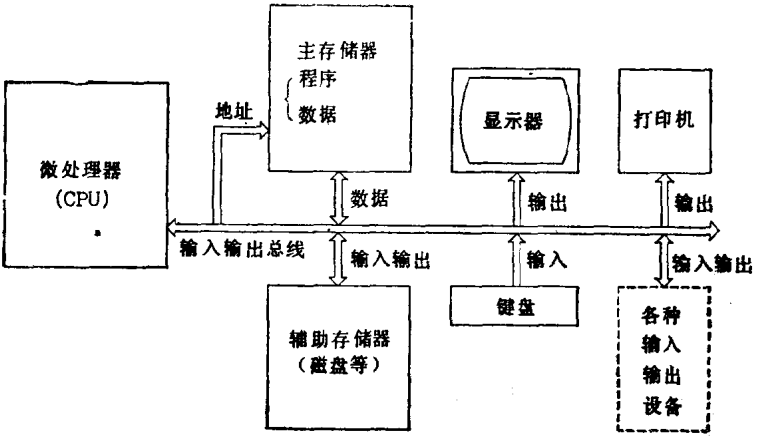


图 1.1 微型计算机的构成

用．图中的地址是表示从主存储器中的哪个单元取出指令和数据的信息．

(3) 微处理器处理的结果(输出数据)被送往主存储器或辅助存储器、显示器、打印机等．将数据存入主存储器时，用地址信息指明应将数据存入主存储器中的哪个单元．

为了具体说明上述问题，这里以 8086 微处理器为例，其内部结构如图 1.2 所示，图中的上半部分是指令的执行部分．在 8086 微处理器中，如图 1.3(a) 所示，主存储器被分成一个一个字节 (8 位) 的分区，各分区用地址 0，1，2，3 等加以区别．若向 5 号地址存放 1 个字节长的指令，则要事先用 1234 单元中的 4 个字节的数据来指定地址．

如图 1.3(b) 所示，主存储器的地址还可分成一些段．关于段的概念将在后面详细介绍，但可以把段看成相当于表示地址的第几条"街"．把几个字节(地址)区归拢在一起作为段来处理，其优点是可以把各程序部分集中在一起，把数据部分集中在一起分别加以处理．由于某一段占用主存储器的哪些地址，或每一段的容量取多大是随意确定的；因此，若把程序部分 (代码段) 分配给 ROM (只读存储器)，而把数据部分(数据段)分配给 RAM (随机

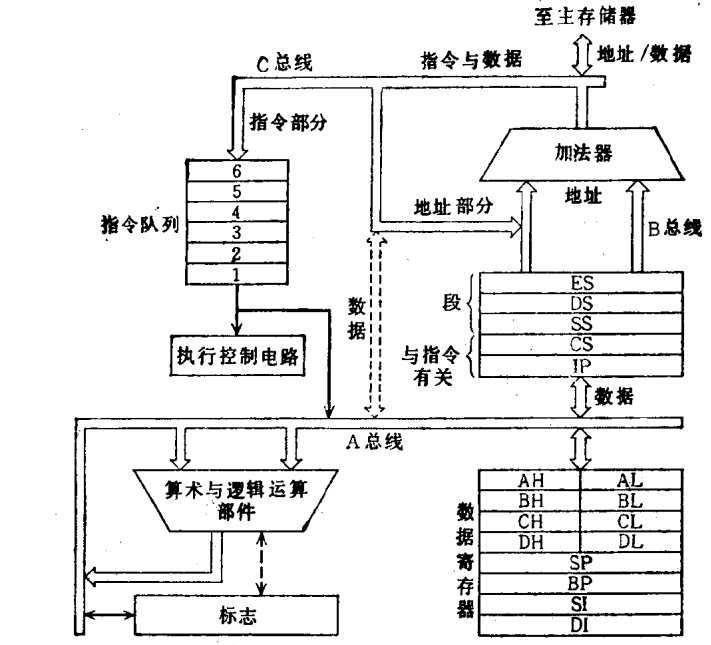
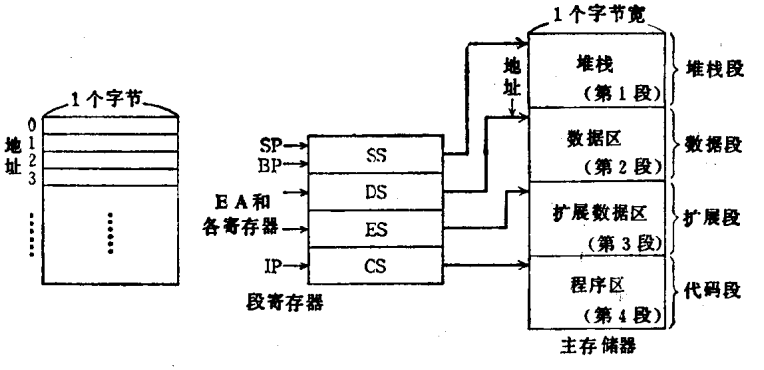


图 1.2　8086 微处理器的内部结构(虚线部分也可认为是实际存在的)



(a) 字节地址　　(b) 主存储器的段的分配

图 1.3　8086 微处理器的寻址方式

存取存储器)时,有了这种现代段的概念就很方便了.

现以图 1.3 为例加以说明,图中的堆栈部分(有关堆栈将在后面介绍)分配在第 1 段,数据部分分配在第 2 段,扩展数据部分分

配在第 3 段，程序部分分配在第 4 段. CS，SS，DS，ES 各寄存器分别表示每个段从主存储器中的哪个地址开始，其中 CS（代码段）寄存器专用于表示程序放在哪一段（第几段）. IP（指令指示器）、SP（堆栈指示器）、BP（基址指示器）是表示各段中的指令和数据所在地址的寄存器. EA 是以后取出数据的有效地址（effective address）.

下面以主存储器的分段为重点，再来看一看图 1.2，图中指令的执行过程如下：

（1）根据 CS 寄存器和 IP 寄存器（存放下一条要执行指令的地址）的值，确定下面要执行指令所在的单元（地址）属于哪一段，哪一个地址.

（2）这样得出的地址被送到主存储器.

（3）从所指定的主存储器的单元取出该指令送往微处理器. 对 8086 微处理器来说，在物理上，送往主存储器的地址同出/入主存储器的指令和数据的一部分使用同一信号线进行传输，所以把图 1.2 上侧对主存储器的信号通路（总线）归拢成一条总线，其原因就在于此.

（4）放入微处理器的 1 字节—6 字节长的指令一旦进入左上方的指令队列，就被依次执行.

（5）在指令的执行过程中需要数据时，数据在主存储器所在单元的地址是这样给出的：根据本指令的地址部分（地址）或 SP，BP，SI，DI 寄存器表示的地址信息与 DS，SS，ES 等寄存器表示的段值（第几段），在图中右上角的地址加法电路中进行计算，然后将计算出的地址送到主存储器.

（6）从主存储器指定地址取出的数据送到图 1.2 下侧的数据寄存器或算术与逻辑运算部件中，按照上述（2）的指令对数据进行处理. 其处理结果再按指令的指示存放到主存储器内某一存储单元或某一寄存器中.

重复执行上述过程称为执行程序. 应注意的是，在 8086 微处理器中寻址时需要段号（指定第几段）和普通地址（普通地址又叫

做地址或偏移). 实际上,如果不加指明, DS 值指的就是数据段. 因此,一般情况下只要指定地址就可以了; 但不要忘记,这时段值基本上是在隐含中使用.

1.2 微处理器中的寄存器

在前面给出的微处理器结构中, 编写程序时通常必须知道微处理器中的寄存器组. 现以图 1.4 的形式来表示 8086 微处理器中寄存器的构成. IP 寄存器是指令指示器,它指示下面要执行的指令放在哪个存储单元中. IP 寄存器由微处理器的硬件自动设定,不能用程序中的指令来改变,也就是程序员看不见, 因此在图 1.4 中没有将其画出.

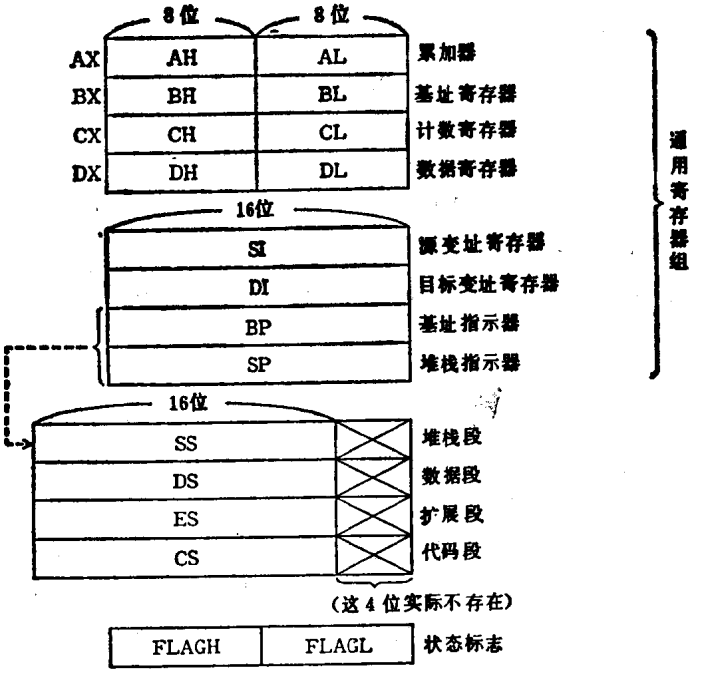


图 1.4 8086 微处理器的基本寄存器

图 1.4 上半部分的寄存器组是指令处理数据时使用的寄存器．由于这些寄存器可以随意使用，因此被称为通用寄存器．顾名思义，其中 SI, DI, BP, SP 4 个 16 位（2 个字节）长的寄存器主要用来作为源变址寄存器、目标变址寄存器、基址指示器及堆栈指示器使用，但一般为了存放 16 位数据，还可以作为多种目的使用．另外，AX, BX, CX, DX 也是 16 位长的寄存器．它们被分成左半部分（高 8 位）和右半部分（低 8 位），每 8 位可以作为 8 位独立的寄存器使用．作为 8 位寄存器使用时，寄存器名为 AH, AL, BH, BL, CH, CL, DH, DL．在这些寄存器的名字中，A, B, C, D 分别是累加器、基址寄存器、计数器、数据寄存器的英文名字的字头．各寄存器的用法完全是自由的．但是，如下所述，AX(AH, AL) 寄存器具有其他寄存器所没有的特殊功能，它是一个功能特别强的通用寄存器（累加器）．8086 微处理器所以被称为 16 位微处理器（如图 1.4 所示），主要是因为其寄存器的长度为 16 位，而且能以 16 位并行方式与主存储器交换数据．

其次，图 1.4 最下面的状态标志寄存器是由图 1.2 的算术与逻辑运算部件进行设置的寄存器，其各位表示运算结果的状态（运算结果为正、负、零等）．使用该寄存器可以写出“若运算结果为 0，则执行某种操作”这样的程序．

图 1.4 中的其余寄存器 SS, DS, ES, CS 的功能是指示已经介绍过的段区在什么地方．这 4 个寄存器的长度均为 16 位，但其内容比其他寄存器向左移了 4 位，具体含义见图 1.5．主存储器的地址用某个段（第几段）内的某个地址（又称为偏移）的形式表示；若左移 4 位，则段值作为地址被扩大了 $2^4 = 16$ 倍，也就是在低位附加了 4 位二进制数 0000．

在 8086 微处理器中进行这样复杂的设计，其目的在于基本上使用只能寻址 $2^{16} = 2^6 \times 2^{10} = 64 \times 1024$ 字节 $= 64$K 字节的 16 位长的地址来寻址 $2^{20} = 2^{10} \times 2^{10} = 1024$K 字节 $= 1$M 字节的地址；由于整个需要 20 位长的地址，因此需将段值左移 4 位后，再加上地址部分（又称为偏移），这样做具有如下意义：

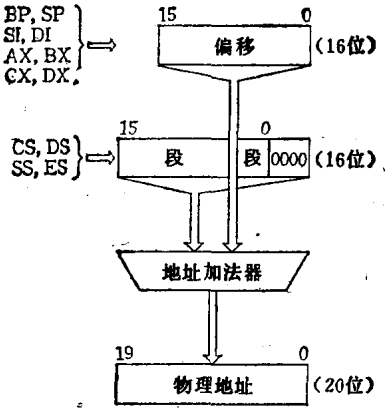


图 1.5 由偏移和段值构成的物理地址

(1) 段的大小是随便的，但段的起始地址必须是 16 的整数倍. 例如，50 号地址只说成 50 号也是可以的；但因为 $50 = 3 \times 16 + 2$，所以还可以说成 3 段 2 地址，也就是第 3 段内的第 2 号地址.

(2) 段的最小容量是 16 个字节，最大容量是 64K 字节.

(3) 各段可以向主存储器内的任何地方进行再定位（关于这一点将在后面介绍）.

对于 8086 微处理器来说，写程序时重要的是要反复强调这种段概念. 在程序和数据均为 64K 字节以内时，由于考虑一个段就可以了，所以很简单；但不管是程序还是数据，若超过 64K 字节时，就跨越两个段，于是要求用程序中的指令对段进行切换. 对前一种场合，图 1.5 的 DS，SS，ES 寄存器只在程序的开头设置一次就可以了；而对后一种场合，在程序执行过程中必须对段寄存器进行重新设置. 不管哪种场合，地址的偏移部分(各段内的地址)必须用 BP，SP，SI，DI，AX，BX，CX，DX 中某一寄存器的内容或指令的地址部分指定. 所谓可以用程序设定 DS，SS，ES；就是通过改变这些寄存器的数值，数据(的段)就可以配置(再定位)在主存储器内的任何一个单元中. 但对 CS 来说，则只是表示程

序本身所在的地址，因此不允许用程序来改变它；如第五章所述，也可以将程序本身进行再定位.

1.3 操作系统的接口

在使用以上述微处理器为中心构成的微型计算机或个人计算机时，比硬件设计更重要的是开发程序，也就是程序设计. 特别是想要知道微型计算机原理，或者利用微型机进行精密的高速控制，或最大限度地发挥微型机的性能时，一定要用汇编语言级的指令字(机器语言)来书写程序.

但是，当把编写出的程序送入计算机，检查其操作是否正确时，如有错误，需要执行改正错误的一连串操作，此时仅有图 1.1 所示的硬件是不够的，还需要有软件开发工具. 象初期的 4 位或 8 位微型计算机，其指令系统很简单，对于这样的系统，人们可以对照指令表把用汇编语言书写的程序翻译成十六进制数的形式，然后从键盘输入并加以执行. 但是，在象 8086 这样的 16 位微型计算机中，由于其指令系统相当复杂，所以类似查指令表这样的手工操作已不能满足需要，假如即便可能，采用这种手工操作，其软件开发效率也非常低.

因此，为使用户能够高效率地使用计算机，即便在个人计算机这一级上，也逐渐按照标准配置了称为操作系统 (OS) 的基本程序. 配置了操作系统的微型计算机从用户(程序员)角度来看，与其说是图 1.1，不如说是图 1.6 更合适. 在图 1.6 中，操作系统就相当于图 1.1 中的微处理器，不仅被称为软件开发工具的编辑程序、汇编程序、连接程序、调试程序这样的系统程序，就连用户编制的程序也是在操作系统的管理下执行. 以操作系统为基础，程序语言除了使用汇编语言外，一般也可以使用高级语言，如 FORTRAN，Pascal 和 C 语言等.

操作系统除控制上述程序的执行外，还具有对输入输出设备控制的标准化的功能. 接到计算机上的输入输出设备，甚至连图