微计算机丛书

微型计算机操作系统

[美] 马克·达姆克 著

武 强 译 王泽涵 校

电子工业出版社

微型计算机操作系统

[美] 马克·达姆克 著

武 强 译

王泽涵 校

電子工業出版社

内 容 简 介

这是第一本关于微型机操作系统方面的书，它较好地揭示了计算机操作系统将用户命令转换成所要求的系统内部操作的情况。

本书将有助于读者去评价微处理机并建立有限的开发系统，有助于理解带有若干终端和磁盘存储器的大型微型机系统的工作原理，并且有助于设计命令语言。书中涉及到的其他专门课题有多道程序设计概念、多重处理概念、数据和存储管理、软件可移植性，以及用户同系统的接口等。此外，本书还详细讨论了小型微机系统的各种类型以及它们之间的差别，书中提出了依据它们的各种预定应用来评价操作系统的观点。最后，本书还包括说明 CP/M 和 UNIX 这两种流行的操作系统的特点和用法的附录。

MICROCOMPUTER OPERATING SYSTEMS

Mark Dahmke

BYTE Publication Inc.

责任编辑　吴明卒

*

电子工业出版社出版(北京市万寿路)

新华书店北京发行所发行　各地新华书店经售

北京科技印刷厂印刷

*

开本：850×1168 1/32　印张：7.875　字数：211 千字

1986 年 6 月第 1 版　1986 年 8 月第 1 次印刷

印数：11500 册　定价：1.80 元

统一书号：15290 · 295

原　　序

五年来，我不仅读了许多有关微型计算机系统设计方面的书，而且也读了一些详细介绍操作系统设计方面的书。所有这些介绍微型机的书大都是从硬件的观点来阐述系统设计这一课题的，所有介绍大型机操作系统的书都假设读者已经对 IBM 370 有所接触，而且对计算机硬件的设计并不感兴趣。

本书从软件方面介绍微型机系统设计的概貌，而不是介绍硬件设计方面常见的细节。

本书一开始就介绍技术概貌和同现代微处理器设计有关的某些专用名词术语。全书以循序渐进的方式讨论了大型计算机中操作系统的实现方法，以及这些方法如何适应于微型机环境。书的主体部分讨论了某种操作系统的特点和内容，这种操作系统看来是大量存在的，类似于在当今大多数微型机上常用的磁盘操作系统 DOS。书中其余的部分对上述类型的操作系统作了扩充，讨论了多道程序设计、多处理机操作系统以及网络操作系统等。

本书介绍了有关操作系统设计的多方面内容，举例说明了当代流行的一些系统的动向。

目　　录

第一章 绪 论

什么是操作系统呢？一般而言，操作系统是程序，或者说，是一组相关的程序，它是计算机硬件和用户之间的桥梁。在某种小型的微处理机应用环境下［即一种开发系统或评价系统，它具有一个大约 2K 字节（2048 字节）的可编程只读存储器（PROM）和一个小容量随机存取存储器（RAM）］，其操作系统可能只不过是由一系列服务子例行程序和一个简单的命令系统组成的。这种操作系统使用户能够实现下述一些任务：从外存储器将信息装入内存储器，或将信息从内存储器转储到外存储器；对一些字节进行修改；对一些应用程序进行测试。

比较复杂一点的操作系统可能具有磁盘控制器接口和某种形式的存储管理程序。本书讨论了最高级的操作系统，它包括诸如多道程序设计（分时地运行多个独立的程序）、可切换存储体的存储器、硬磁盘控制器接口和随机磁道扇区分配等。本书还讨论了多重处理概念（多个微处理机按某种交互方式并行地工作）。

那么，为什么一个微型机系统需要有一个操作系统呢？如果所讨论的微型机有一个硬接线式的面板，那么原则上就不需要控制程序。用户可以借助各行开关或按钮来打入和更改二进制形式的指令和数据。这种方法在时间上显得有些浪费而且很不灵活。当试用了这种方法之后，用户必然会得出这样一个结论：必须使用一种更容易的方法。用户开始感到奇怪：既然微处理机总是有空闲状态，那么它为什么不能执行所有的例行任务呢？这正是为什么要将大型计算机的操作系统概念引伸到微型机中的原因。

本书试图针对几种级别的操作系统，定义那些有用的甚至是必需的功能。

第二章研究了微型机操作系统应该完成的任务，介绍了几种

不同规模的系统，以便确定每一种应用究竟需要一些什么样的功能。

第三章详细讨论了小型微机系统的监控程序。第四章研究了中、大型微机系统，以及诸如数据管理、磁盘存储空间再分配之类的课题。第五章介绍了多用户和多道程序设计环境，以及在设计用于对付多个应用程序的操作系统时应作的考虑。第六章介绍了多重处理概念全貌，此外还讨论了微型机的紧密耦合系统、松散耦合系统以及分布式网络。

第七章介绍了存储管理技术，其中包括线性地址空间、存储体切换和虚拟存储器。第八章介绍了与机器无关的环境。第九章介绍了系统实用程序。第十章讨论用户对系统的影响。第十一章是正文的结束，它讨论了下述内容：操作系统有哪些设计参数？操作系统应该包括哪些功能？其理由何在？

书末的附录 I 和 II 分别提供了微型机控制程序 CP/M 和 UNIX 操作系统的简短参考材料，包括上述两者的基本原理和内部组织。

1.1 微型机硬件和术语综述

同小型和大型计算机相比，微型机的体系结构是相当简单的。本书涉及的是 8 位微处理机，不过，它的基本原理同较新的 16 位微处理机差不多。此外，还讨论了仅仅适用于 16 位微处理机体系结构的某些特性。

1.1.1 中央处理机的组织

典型的 8 位微处理机具有如图 1.1 所示的内部体系结构。其算术逻辑运算部件（ALU）用来完成数据的算术运算和逻辑运算。一般有某些内部暂存寄存器，或者至少有一个累加器，它们用来保存来自 ALU 的中间结果，此外，还可能提供另外一些寄存器。例如：变址寄存器，它用来建立存储地址或指针；指令寄存

器，它是一种单字节缓冲器，用来暂时保存来自主存储器的当前机器指令(这种指令寄存器一般不能被程序员访问)；程序计数器(PC)，它总是指向主存储器中的当前指令字节。

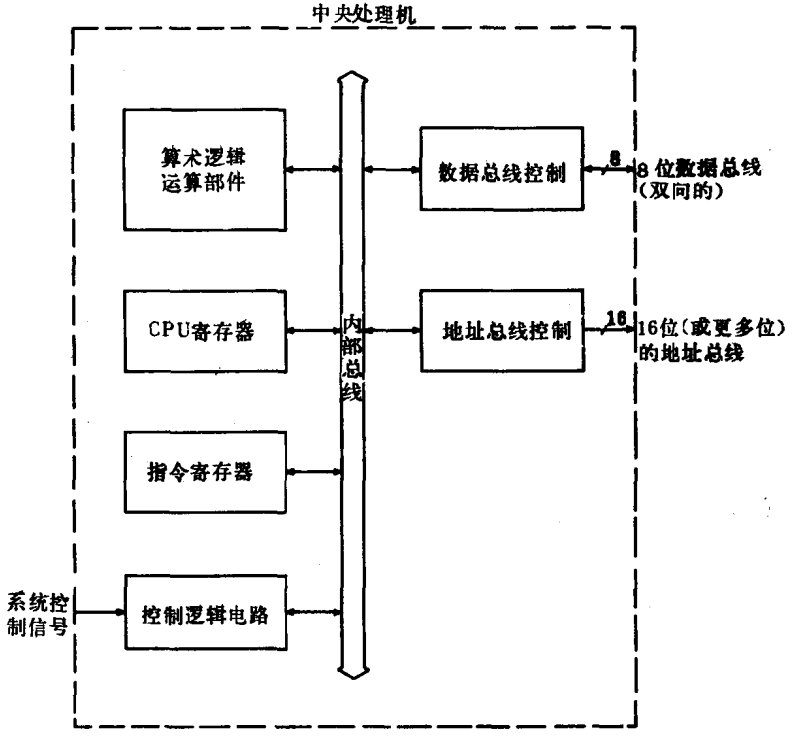

图 1.1 典型 8 位微处理机的内部组织。其算术逻辑运算部件 (ALU) 用来实现全部逻辑功能，而其控制逻辑电路则用来协调所有的机构动作。

微处理机的内部总线与系统的外部总线相连接。信息经过缓冲，沿着系统总线分布到整个微型机。外部总线包括以下三类：地址总线(其宽度通常为 16 位，有时甚至大于 16 位)；数据总线(在 8 位微型机中，其数据总线的宽度为 8 位)；控制总线[其控制信号有 READ (读)、WRITE (写)、MEMORY-REQUEST (存储器请求)、I/O REQUEST (输入输出请求)、WAIT (等待)等等]。

主存储器和输入输出端口采用这些控制信号来决定中央处理机（CPU）是否与它们"对话"。大多数微处理机都至少有一个中断输入信号。中断就是使 CPU 正在进行的那项工作暂停下来，而使控制转移到预先规定的存储单元，以便执行指定的指令序列。键盘或串行输入设备通常是建立在中断的基础上的，其目的是使异步的外部设备（如键盘）能够促使微处理机转到为某类高优先级的活动服务。这就省去了程序员周期性地查询输入数据究竟是否可用所遇到的麻烦。

当微处理机复位（借助于硬件）后，第一个活动就是从存储器取出一条指令。大多数微处理机的程序计数器都被置零，因此，其启动指令必须在存储器的头几个字节（并非全都是这种情况，而在我们的讨论中，这一点并不重要）。无论从哪里开始执行，这些指令都必须出现在使 CPU 复位之前，这可以借助于硬接线面板或者把只读存储器放在主存的开始部分来实现。CPU 将零号地址的内容装入指令寄存器，若这个内容是一条可执行的指令（操作码），则接着就要对它进行译码。

如果该指令长度大于一个字节，则 CPU 可继续取出多个字节，并且使程序计数器 PC 增量，然后执行这条完整的指令。除了转移指令（通常称为分支指令）和调用指令的情况外，程序计数器 PC 都要增量，并且这种过程是重复进行着的。转移或调用指令用来使控制转移到指定的主存地址。这就是说，要将按该指令所指出（或按该指令所指定的某寄存器中）的地址装入 PC。这个过程周期地重复下去，直到执行一条 HALT（停机）指令或者直到切断电源为止。

图 1.2 示出了微处理机的指令执行流程。

所有的存储器和输入输出操作几乎都采用相同的方案。存储单元的地址或输入输出端口的地址都置于 16 位地址总线上，数据总线则用来发送或接收 8 位数据。其控制信号用来指出要完成的应是哪种操作。

```
0000H    MVI    B, OFFH     ; 将数 "FF" 装入 B 寄存器

0003H    JMP    0020H       ; 转移到地址 0020
0006H    OUT    23H         ; 将寄存器 A 的内容输出
                            ; 到端口 23H
0008H    RET                ; 返回到调用程序
0020H    MVI    A, 10H      ; 将数 10H 装入 A 寄存器
0022H    CALL   0006H       ; 调用在 0006 的一个子例行程序
0025H    INR    A           ; A 寄存器增量
0026H    DCR    B           ; B 寄存器减量
0027H    JNZ    0022H       ; 转移到地址 0022
002AH    HLT                ; 如果完成,则使 CPU 停机
```

程序计数器	A	B	指令				
0000	?	?	06	FF		MVI	B, OFFH
0003	?	FF	C3	00	00	JMP	0020H
0020	?	FF	3E	10		MVI	A, 10H
0022	10	FF	CD	06	00	CALL	0006H
0006	10	FF	D3	23		OUT	23H
0008	10	FF	C9			RET	
0025	10	FF	3C			INR	A
0026	11	FF	05			DCR	B
0027	11	FE	C2	22	00	JNZ	0022H
0022	11	FE	CD	06	00	CALL	0006H
0006	11	FE	D3	23		OUT	23
--	--	--	--			---	
--	--	--	--			---	

图 1.2 执行流程

1.1.2 存储器组织

微计算机的存储器分为两种主要类型：读写存储器和只读存储器。现在有许多种存储器芯片(集成电路)，每片包含多达 65536 位，而且其容量还在不断地增加。如果有一种具体的存储器芯片由 16384 × 1 位组成，那么，它就要采用并联的 8 块芯片，以便得

到 16K 字节的 8 位宽的存储器。如果采用 16 位微处理机，那么就必须采用并列的 16 块存储器芯片。图 1.3 示出了一个用于 8 位微型机的典型存储器插件板。值得注意的是，在计算机术语中，16K 是指 16384，而不是 16000。因此，“K”为 2^{10}，即 1024，它比 1000 更有用，更方便。

只读存储器 ROM 可进一步细分为：掩模型只读存储器；可编程只读存储器（PROM）；可擦可编程只读存储器（EPROM）（通常缩写字母 PROM 指的是常用的 ROM、PROM 以及 EPROM）。EPROM 通常用在样机或产量较小的场合。尽管其制造成本较高，但它们可以现场编程，以后可由紫外光擦除。正因为 EPROM 在检测到程序错误时可以很容易将其擦除，故用于程序开发是很理想的。PROM 除了它不可擦之外，其余和 EPROM 没什么两样。由于 PROM 可以与 EPROM 接插兼容，因此可以用 EPROM 来开发软件和硬件，而无须从研制转到批量生产时对 EPROM 作什么变更。PROM 通常采用一种熔丝连接工艺：每一位都是一条硅“熔丝”，当其被施加一定的电压时，这种熔丝便会烧断。一旦熔丝烧断，其信息就不可能再被擦除。仅在经过全面测试从而保证了数据和程序都是正确的之后，ROM 才用于大容量存储的应用中。ROM 是通过下述工艺制作的：把数据（位组合格式）安排到芯片本身的一层“掩膜”上。这样一来，这些位就被直接制作到芯片内。

可以购买到的 ROM、PROM 和 EPROM 有各种不同的规模：128 × 8 位至 8K（即 8192）× 8 位。这些容量还在迅速增长。值得注意的是，大多数 ROM、PROM 和 EPROM 芯片内被设计成 8 位宽。由于微型机主要采用 PROM 来存储启动程序，而且其他一些批量电子产品（这里，封装数是关键因素）也需要 PROM，故采用 8 位宽度是很方便的。还可购买到 16K × 1 位的 PROM 阵列，以及其他一些规模的 PROM。

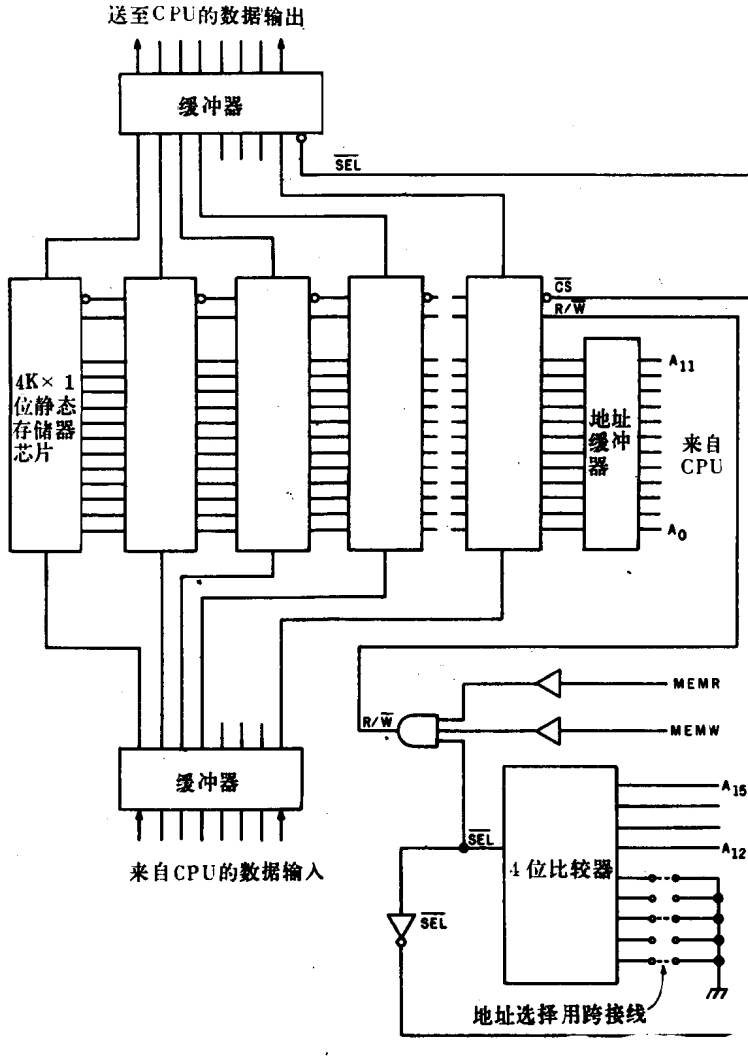

图 1.3　典型的 4K 字节存储器插件板。由 4 位的比较器实现地址译码。当选择了一个有效的存储插件板地址时，其选择线 SEL 就变成高电平(即“1”)。为了得到一个单一的信号，可用 SEL 选通存储器读 (MEMR) 和存储器写 (MEMW)。 SEL 为高电平对应于 MEMR; SEL 为低电平对应于 MEMW。片选线 CS 为低电平有效。

1.2 软件概念

最方便的微型机程序设计方法是采用汇编语言或 Pascal、BASIC 之类的高级语言。在本书所讨论的例子中，采用了具有 Z80 的扩充助记符的 8080 汇编语言。所有算法都用一种结构化的单向流程图技术(见附录 III)来表示。当开发大型程序时，采用高级语言通常较容易，但编译程序通常不能产生有效的目标代码，其结果造成了总的速度和效率的损失。

1.2.1 数据类型

在原始数据类型(特别是字节和指针)的基础上可建立起线性表、列表和其他数据结构。一个字节规定为 8 位，每位可以是 1 或 0。这就意味着，每个字节的数值范围是十进制的 0～255 [或十六进制 (以 16 为基) 的 00～FF]。计算机仅与 1 和 0 有关，而与这些位究竟代表着什么内容无关。因此，41H (十六进制的 41) 或十进制的 65 可能指十进制数 65，也可能指美国信息交换标准码 (ASCII) 的字符 A，还可能指二-十进制数 41 等等。这个字节的位组合格式是 01000001，可以将它解释为一串状态标志，或一串状态位。若该位为 1，则标志置位；若该位为 0，则标志断开或复位。采用位来表示通/断值可节省相当多的存储空间。

其他数据结构包括字符串，即存储器中一串按升序排列的数据字节。字符串通常包含字符数据和某种用以指明字符串结尾的结束字符。另一方面，字符串的长度可以在该字符串的开头一个字节中表示出来。图 1.4 所示为字符串类型。

值得注意的是，在图 1.4 中，它的头几个字符串具有不同的结束字符。这通常由程序员决定，或遵循所用系统的规则。第二种格式包括放置在该字符串的标题字节 (它包含以字节为单位的字符串长度)。若要对汇编程序中的字符串进行访问，则指针(等于