微机原理与汇编语言程序设计

（上册）

主编　李乃祥

南开大学出版社

WEIJIYUANLIYUHUIBIANYUYANCHENGXUSHEJI

图书在版编目(CIP)数据

微机原理与汇编语言程序设计.上册 / 李乃祥主编.
—天津:南开大学出版社,2004.5
ISBN 7-310-02064-2

Ⅰ.微... Ⅱ.李... Ⅲ.①微型计算机—理论②汇编语言—程序设计 Ⅳ.①TP36②TP313

中国版本图书馆 CIP 数据核字(2004)第 002887 号

出版发行 南开大学出版社
地址:天津市南开区卫津路 94 号 邮编:300071
营销部电话:(022)23508339 23500755
营销部传真:(022)23508542
邮购部电话:(022)23502200
出版人 肖占鹏
承 印 南开大学印刷厂印刷
经 销 全国各地新华书店
版 次 2004 年 5 月第 1 版
印 次 2004 年 5 月第 1 次印刷
开 本 787mm×1092mm 1/16
印 张 14.75
字 数 371 千字
印 数 1—3000
定 价 21.00 元

内容简介

本书以8086微处理器和MASM宏汇编语言为背景，全面、系统地介绍了微型计算机的基本工作原理、汇编语言的基本知识及程序设计技术。全书分上、下两册，上册介绍编码与计算机逻辑器件等相关基础知识，微型计算机的编程结构、寻址方式与指令系统、伪指令以及使用汇编语言进行程序设计的基本方法和技巧。下册包括微处理器的总线与时序、存储器、输入输出接口、中断与DMA技术、高级汇编、混合编程及Windows环境编程等，并简要介绍80386以上机型所采用的各种新技术。

本书结构新颖、内容翔实、循序渐进、由浅入深，适合作为大专院校计算机及相近专业微机原理与汇编语言程序设计课程的教学用书，也可供相关人员自学使用。

本书有配套的多媒体教学课件，选用该教材的教师可直接与编者联系（022-23781331或LNXTJAC@eyou.com），免费赠送。

序

许多院校将微型计算机原理和汇编语言程序设计作为两门课程分别开设，这样，一方面造成了部分内容的重复讲授，浪费了宝贵的学时，另一方面给学生的学习带来了一定的困难。

在天津农学院领导和教务处的大力支持下，本书编者（李乃祥，天津农学院副教授）自1998年开始，进行了课程改革试验，将微型计算机原理和汇编语言程序设计合为一门课程讲授。经过对信息技术专业几届学生的教学实践，效果明显。一是学时数显著减少，二是充分利用了二者之间的有机联系，学生对知识的理解程度和运用能力有所提高。根据学生的要求和实际教学工作的需要，也考虑到社会的需求，本书编者近期对讲稿进行了整理和补充，作为一本教材正式交付出版。

本书编写遵循的原则是：既有一定的针对性，又尽量追求较宽的适用范围；既注重基本内容的介绍，又提供一定的拓展知识空间；既强调基本理论的学习，又注意应用能力的提高；既注意知识的系统性和完整性，更强调知识之间的有机结合与循序渐进。主要体现在以下几个方面：

1. 为满足非电子类专业学生及初学者的需要，书中仍简要介绍了前导课程中的编码、计算机逻辑器件等相关基础知识。

2. 仍然以8086微处理器和MASM宏汇编语言为背景，介绍微型计算机的基本工作原理、汇编语言的基本知识及程序设计技术。

3. 根据学习需要，将微型计算机原理和汇编语言程序设计的内容进行合理组织，实现了两部分内容的有机融合。

4. 介绍了汇编语言与高级语言的混合编程以及 Windows 环境下的汇编程序设计等拓展技术内容。

5. 简要介绍了80386以上机型所采用的各种新技术，包括保护模式、RISC结构、多媒体专用处理指令、高速缓存、分支预测、超标量流水线等。

学习微机原理与汇编语言程序设计的重要性主要体现在以下几个方面：

1. 微机原理和汇编语言程序设计是高等院校计算机及相近专业的两门专业基础课，前者着重介绍微型计算机的基本结构、内部信息流通和指令系统的基本原理，后者则以微型计算机的编程结构和指令系统为基础讲述汇编语言程序设计的方法，学好这两门课是进一步学习其他计算机专业课程的基础。

2. 汇编语言是计算机能提供给用户的最快而又最有效的语言，也是能够利用计算机所有硬件特性并能直接控制硬件的唯一语言，因而在那些对程序的空间和时间要求很高的场合(如实时过程控制)，常常用汇编语言来编制程序，而对于很多需要直接控制硬件的应用场合，则更是非用汇编语言不可了。

3. 学习微机原理和汇编语言程序设计，可以更清楚地了解计算机是怎样完成各种复杂工作的，从根本上认识、理解计算机的工作过程，掌握打开计算机这个“黑匣子”的钥匙。对于某些人或在某种特定条件下，把计算机作为一个“黑匣子”，不失为一种好的学习方法。但

若仅限于此则很难成为解决计算机应用领域某些深层次问题的高手。

4．现在的计算机系统中，某些功能仍然是靠汇编语言程序来实现的。例如机器自检、系统初始化、实际的输入/输出设备的操作，至今仍然是用汇编语言编制的程序来完成的。汇编语言是了解这些内容的必备工具。

本教材共有十七章，分上、下两册。包括以下四部分内容：

1．必要的概念，基础知识和工具。主要包括微型计算机基础知识，常用 DEBUG 命令及其使用等。

2．微机原理基本内容。主要包括：8086CPU 系统、总线操作和时序，I/O 接口与 I/O 实现方式，8086 的中断系统与系统功能调用，可编程接口芯片，微型计算机存储系统。

3．汇编语言基本内容。主要包括：8086 微处理器与存储器的编程结构，8086 的寻址方式与指令系统，汇编语言与基本伪指令，汇编语言程序设计的基本方法，常用处理的汇编语言程序实现，8086 的系统功能调用，I/O 程序设计，文件存取。

4．提高与拓展。主要包括：高级汇编语言技术及其使用，模块化程序设计与混合编程技术，Windows 环境下的汇编语言程序设计，PC 技术的发展。

读者在本教材的阅读和学习过程中，应注意下述方法：

1．要把微机原理和汇编语言结合起来学。这将有助于学深，学透。例如，学习了汇编指令，在学习 CPU 的译码和输出信号时，就要搞清楚执行某一条汇编指令时，译码后 CPU 主要输出信号的高低状况。为此，本教材在内容和顺序上都进行了精心组织，使两部分内容尽可能结合在一起，希望读者在学习中细心体会。

2．注意学习思想方法。微机原理中有许多地方是吸收了其他领域的思想方法，甚至社会科学的思想方法。例如，中断的思想，DMA 的思想，并行处理的思想，分层调用的思想，分级存储的思想方法等。注意思想方法的学习不仅有助于本课程的学习，也可以用来指导你学好其他课程，甚至使你受益终身。

3．注意掌握基本原理。学习中不要过多拘泥于细节，对有些部分要注意从宏观上、原理上来理解，从功能上来了解。例如，对于 8086CPU 的执行部件与总线接口部件之间如何配合工作，对于 CPU 如何进行指令译码，只能从宏观上，从功能上来理解。

4．学会灵活运用知识，做到举一反三。要逐步培养自己的能力，学会灵活运用已学过的知识去解决学习中遇到的问题。有些问题在细节上可能有区别，但在原理上却是相通的。因此，不仅要学会如何解决某个问题，而且应学会用某种方法去解决类似的一系列问题。例如，学习了译码器，在学习存储器连结设计时要做到灵活使用；介绍了一种存储器连结方法，要能根据不同型号存储器芯片的要求，设计出相应的存储器连结电路。

5．详略得当，把握住重点。学习中要做到该粗则粗，该细则细。例如，在学习中断时，不仅要弄清中断的概念，而且要作为重点，弄清中断类型码来自何处，中断向量是什么，中断向量如何存放，计算机是如何根据中断类型号找到相应的中断向量，又是如何转去执行相应的中断处理程序的。再比如，对于译码器、寄存器、加法器要能详细画出其逻辑电路图，才能真正理解其工作原理，但对于总线控制器和其他较复杂的电路则只能了解其功能。

6．多思考，多联想，多练习。只有善于把平时分散学习的内容联系到一起深入思考，才可能获得一个深刻、全面和完整的认识。只有多练习，才可能达到熟能生巧，才可能发现和纠正不正确的理解和认识。尤其是学习汇编语言，更要多练习，以便掌握每条指令的正确

使用方法以及相互之间的正确配合方式。

本书是集体合作的结果。马全意、何玉香、刘源和罗珈老师参加了第二、三章的材料整理和编写工作，马全意等老师还为本书制作了教学课件。靳润昭教授审阅了全书，并提出了宝贵的修改意见。

在本书的编写过程中得到了天津农学院领导、教务处和有关各系的大力支持，张孝义老师为本书的出版做了大量的组织和协调工作，南开大学出版社也对编者给予了热情支持与宝贵指导，在此表示衷心的感谢。此外，在本书的写作过程中，编者参阅了大量相关教材和书籍，在此也向这些书籍的作者表示谢意。

由于编者水平有限，加之成稿时间仓促，书中难免存在不足和疏漏之处，恳请有关专家和读者提出宝贵意见。

编者

2003. 12. 11

目 录

第一章　微型计算机系统导论

第一节　微型计算机发展概述

自世界上第一台电子计算机 ENIAC 问世，至今已有近 60 年的历史了。这期间，随着元器件的变化，电子计算机先后经历了四代。

第一代是电子管数字计算机，其发展年代大约从 1946 年至 1958 年。此时计算机的逻辑元件采用电子管，主存储器采用磁鼓、磁芯，外存储器已开始采用磁带，软件主要用机器语言来编制程序，后期逐步发展了汇编语言。这时的计算机主要用于科学计算。

第二代是晶体管计算机。其发展年代大致从 1958 年至 1964 年。计算机的逻辑元件为晶体管，主存储器仍用磁芯，外存储器已开始使用磁盘，软件已开始有很大的发展，出现了各种高级语言及编译程序。此时计算机的应用已发展至各种事务的数据处理，并开始用于工业控制。

第三代是集成电路计算机（1964 年至 1971 年）。计算机的逻辑元件已采用小规模和中规模的集成电路，即所谓 SSI 和 MSI。主存储器仍以磁芯为主。软件发展更快，已有分时操作系统。会话式的高级语言已出现并有相当的发展。小型计算机也随集成电路规模的增大而很快发展起来，应用的范围也日益扩大，企事业管理与工业控制都逐步引入小型计算机。

第四代是大规模集成电路计算机。这是从 1971 年开始发展起来的。所谓大规模集成电路（LSI）是指在单片硅片上可以集成 1 000 至 20 000 个晶体管的集成电路。由于 LSI 的体积小，耗能很少，可靠性很高，因而使微型计算机（Microcomputer）得以产生并不断迅速发展。

表 1-1　INTEL 80X86 系列 CPU 发展变化概况

CPU 名称	首推年份	集成度（万只）	主频(M)	位数（内/外）	执指/时钟	寻址空间	主要技术特点
4004/4040	1971	0.2	1	4			
8080/8085	1973	0.5～1	2～4	8		64K	实模式
8086/8088	1978	2～6	4～8	16		1M	内存分段、取/执并行操作①
80286	1982	13.4	6～20	16		16M	保护模式
80386	1985	27.5	13～50	32	1/5	4G	虚拟 86、SMM②模式
80486	1989	120	25～50	32	1/2	4G	RISC③、Cache 技术
P－p5	1993	310	60～150	32/64	1/1	4G	超标量双流水线、分支预测技术
P－p54c	1993	310	60～150	32/64	1/1	4G	双路 Cache④
P－pro	1995	550	66～100	32/64	2/1	64G	超标量超流水线、动态执行技术
P－MMX	1996	450	133～233	32/64		4G	增加了 MMX⑤指令
P Ⅱ	1997	750	233～400	32/64		64G	MMX 与动态执行技术
P Ⅲ	1999	950	450～1G	32/64		64G	SSE⑥
P Ⅳ	2000		1.3～1.7G	32/64		64G	SSE2
Merced	2002		≥2G	64/128			EPIC⑦
Mckinely	2003		≥2G	64/128			

①取指令与执行指令并行进行；②System Management Model；③Reduced Instruction Set Computer；④指数据和指令使用不同的高速缓存；⑤Multi-Media eXtension；⑥Streaming SIMD Extension；⑦Explicitly Parallel Instruction Computing。

如果可以用“迅速”一词来描述电子计算机的发展的话，那么微型计算机的发展则堪称为“日新月异”了。在短短 30 年的时间内，从 4 位机发展到超 32 位机，从每秒处理几千条指令发展到每秒处理几亿条指令，类型越来越多，体积越来越小，功能越来越强，用途越来越广，但价格却越来越低。表 1-1 给出了微型计算机的“心脏”——CPU 的发展变化情况，读者从中可窥见微型计算机的发展轨迹。

第二节　微型计算机系统组成与工作原理

一、微型计算机系统的基本组成

微型计算机系统包括硬件和软件两大部分，具体组成如图 1-1 所示。

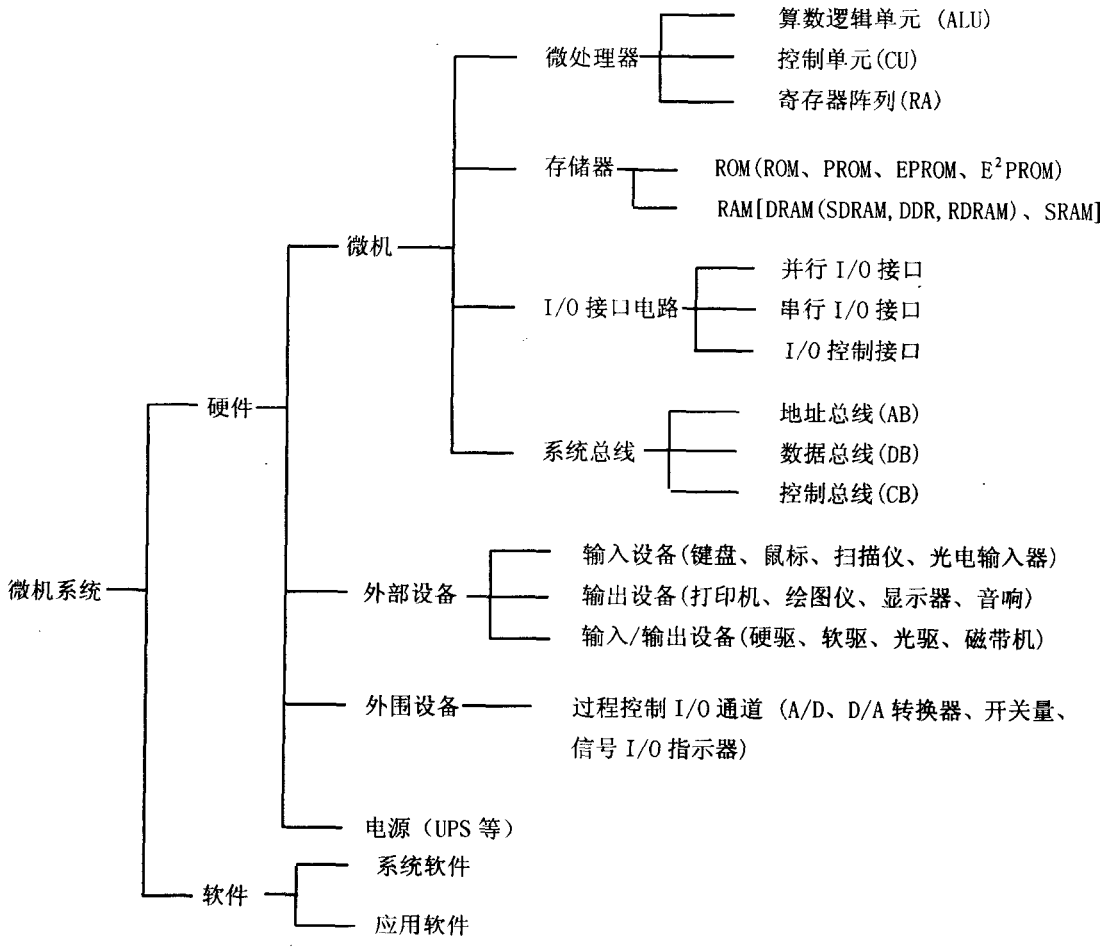

图 1-1　微型计算机系统组成

二、微型计算机硬件系统结构

所谓微机硬件系统结构系指按照总体布局的设计要求将各部件构成某个系统的连接方式。一种典型的微机硬件系统结构如图 1-2 所示。图中，用系统总线将各个部件连接起来。

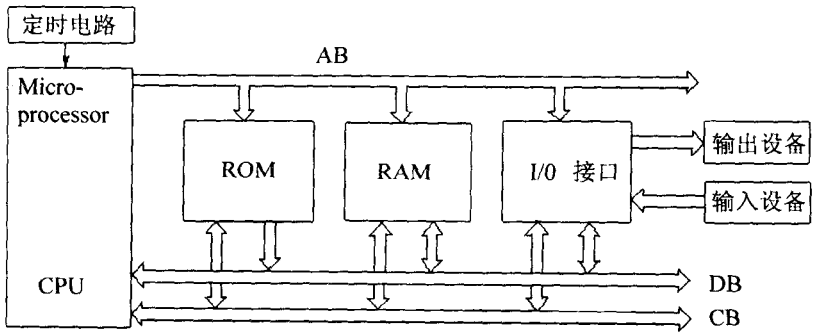

图 1-2 典型的微机硬件系统结构

系统总线是用来传送信息的公共导线，它们可以是电缆，也可以是印刷线路板上的连线。所有的信息都通过总线传送。通常，根据所传送信息的内容与作用不同，可将总线分为 3 类：数据总线 DB(Data Bus)，地址总线 AB(Address Bus)，控制总线 CB(Control Bus)。这是一种单总线系统结构，系统中各部件均挂在单总线上，所以又称为面向系统的单总线结构。

在微型计算机中有两股信息流（数据信息流和控制信息流）在流动。在单总线系统中，通过单总线实现微处理器、存储器和所有 I / O 设备之间的信息交换。由于各部件均以同一形式挂在单总线上，结构简单、易于扩充，所以目前绝大多数微机硬件系统均采用这种结构。

三、微处理器简化模型

一个简化的微处理器由运算器、控制器和内部寄存器阵列 3 部分组成，如图 1-3 所示。现将各部件的功能简述如下：

1. 运算器

运算器又称为算术逻辑单元 ALU (Arithmetic Logic Unit)，用来进行算术或逻辑运算以及位移循环等操作。通常，参加运算的两个操作数，一个来自累加器 A (Accumulator)，另一个来自内部数据总线，可以是数据寄存器 DR(Data Register)中的内容，也可以是寄存器阵列 RA 中某个寄存器的内容。运算结果往往也送回累加器 A 暂存。

2. 控制器

(1)指令寄存器 IR(Instruction Register)

指令寄存器 IR 用来存放从存储器取出的将要执行的指令(实为其操作码)。

(2)指令译码器 ID(Instruction Decoder)

指令译码器 ID 用来对指令寄存器 IR 中的指令进行译码，以确定该指令应执行什么操作。

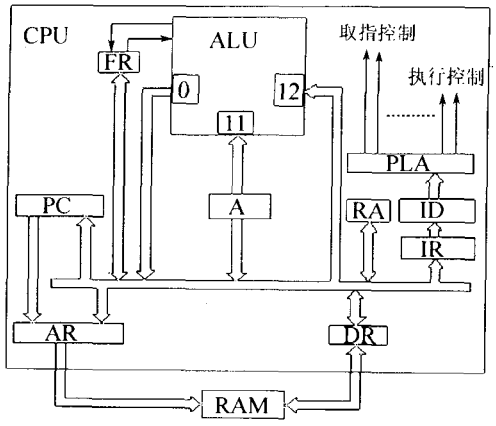

图 1-3 Microprocessor 结构图

(3)可编程逻辑阵列 PLA(Programmable Logic Array，也称为定时与控制电路)

可编程逻辑阵列用来产生取指令和执行指令所需的各种微操作控制信号。由于每条指令所执行的具体操作不同，所以，每条指令将对应控制信号的某一种组合，以确定相应的操作序列。

3. 内部寄存器阵列

通常，内部寄存器阵列包括若干组寄存器。

(1)累加器 A

累加器是用得最频繁的一个寄存器。在进行算术逻辑运算时，它具有双重功能：运算前，用来保存一个操作数；运算后，用来保存结果。

(2)数据寄存器 DR

数据寄存器 DR 用来暂存数据或指令。从存储器读出时，若读出的是指令，经 DR 暂存的指令通过内部数据总线送到指令寄存器 IR；若读出的是数据，则通过内部数据总线送到有关的寄存器或运算器。

向存储器写入数据时，数据是经数据寄存器 DR、再经数据总线 DB 写入存储器的。

(3)程序计数器 PC(Program Counter)

程序计数器 PC 中存放着正待取出的指令的地址。根据 PC 中的指令地址，准备从存储器中取出将要执行的指令。通常，程序按顺序逐条执行。任何时刻，PC 均指示要取的下一个字节或下一条指令(对单字节指令而言)所在的地址。因此，PC 具有自动加 1 的功能。

(4)地址寄存器 AR(Address Register)

地址寄存器 AR 用来存放正要取出的指令的地址或操作数的地址。在取指令时，将 PC 中存放的指令地址送到 AR，根据此地址从存储器中取出指令。在取操作数时，将操作数地址通过内部数据总线送到 AR，再根据此地址从存储器中取出操作数；在向存储器存入数据时，也要先将待写入数据的地址送到 AR，再根据此地址向存储器写入数据。

(5)标志寄存器 FR(Flag Register)

标志寄存器 FR 用来寄存执行指令时所产生的结果或状态的标志信号，关于标志位的具体设置与功能将视微处理器的型号而异；根据检测有关的标志位是 0 或 1，可以按不同条件决定程序的流向。

四、存储器组成与操作概述

1. 存储器的基本概念

存储器用来存放数据和程序。在计算机内部，数据和程序都用二进制代码的形式表示。

在微机中，一般用 8 位二进制代码作为一个字节(Byte)。用一个或几个字节组成一个字(Word)。如果用字表示一个数，称为数据字；表示一个指令称为指令字。数据字和指令字也可以用双倍字长或多倍字长表示。

一个存储器可划分为很多存储单元。存储单元中的内容为数据或指令。为了能识别不同的单元，我们分别赋予每个单元一个编号。该编号称为地址。切不可将存储器单元地址与该单元内存放的内容混淆。

2. 存储器的逻辑组成

现假定存储器由 256 个单元组成，每个单元存储 8 位二进制信息，即字长为 8 位，其结构简图如图 1-4 所示。这种规格的存储器，通常称为 256×8 位的读 / 写存储器。

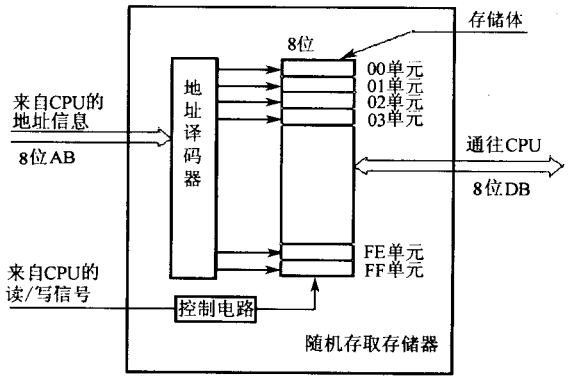

图 1-4　随机存取存储器结构简图

从图 1-4 中可见，随机存取存储器由存储体、地址译码器和控制电路组成。

存储体共有 256 个存储单元，其编号从 00H(十六进制)到 FFH，即从 00000000 到 11111111。

地址译码器接收从地址总线(AB)送来的地址码，经译码器译码选中相应的某个存储单元，以便从中读出信息或写入信息。

控制电路用来控制存储器的读 / 写操作过程。

3. 存储器的读/写操作过程

从存储器读出信息的操作过程如图 1-5(a)所示。假定 CPU 要读出存储器 04H 单元的内容 10010111 即 97H，则(1)CPU 的地址寄存器（AR）先给出地址 04H 并放到地址总线上，经地址译码器译码选中 04H 单元；(2)CPU 发出“读”控制信号给存储器，指示它准备把被寻址的 04H 单元中的内容 97H 放至数据总线上；(3) 在读控制信号的作用下存储器将 04H 单元中的内容 97H 放到数据总线上，经它送至数据寄存器（DR），然后由 CPU 取走该内容作为所需要的信息使用。

应当指出，读操作完成后，04H 单元中的内容 97H 仍保持不变，这种特点称为非破坏性读出 (Non Destructive Read Out, NDRO)。这一特点很重要，因为它允许多次读出同一单元的内容。

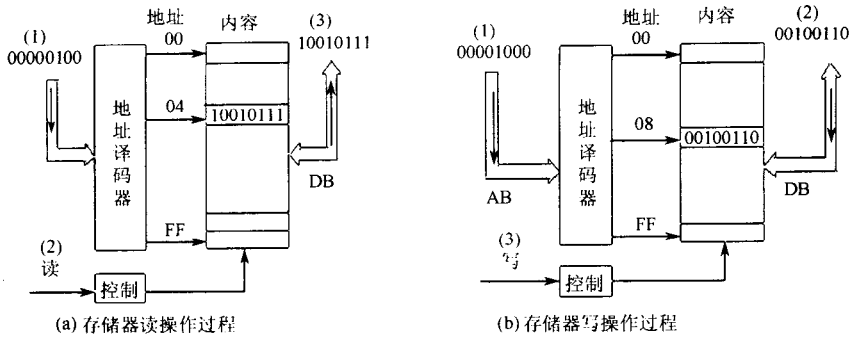

图 1-5　存储器读/写操作过程示意图

向存储器写入信息的操作过程如图 1-5(b)所示。假定 CPU 要把数据寄存器 DR 中的内容 00100110 即 26H 写入存储器 08H 单元，则（1）CPU 的地址寄存器 AR 先把地址 08H 放到地址总线上，经地址译码器选中 08H 单元；（2）CPU 把数据寄存器中的内容 26H 放到数据总线上；（3）CPU 向存储器发送“写”控制信号，在该信号的控制下，将内容 26H 写入被寻址的 08H 单元。

应当注意，写入操作将破坏该单元原存的内容，即由新内容 26H 代替了原存内容，原存内容将被清除。

上述类型的存储器称为随机存取存储器（Random Access Memory，RAM）。所谓“随机存取”即所有存储单元均可随时被访问，既可以读出也可以写入信息。

五、软件系统的层次结构

1. 计算机软件的分级层次结构

计算机软件主要分为系统软件和应用软件两大类。其中系统软件又包括操作系统和应用软件开发服务程序。它们形成一种如图 1-6 所示的具有依赖关系的分级层次结构。

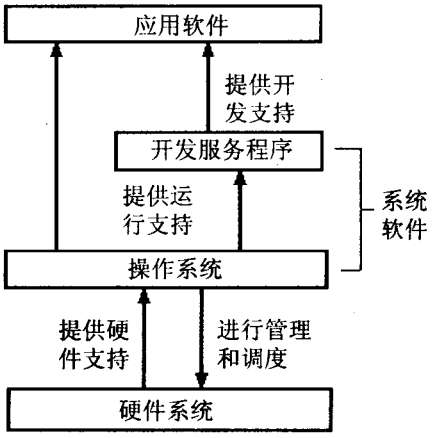

图 1-6　软件的分级层次结构

应用软件是用户为解决科学计算、办公自动化、检测与实时控制等不同领域的任务所编制的应用程序。

系统软件是指不需用户干预的能生成、准备和执行其他程序所需的一组程序。

操作系统是一套复杂的系统程序，用于提供人机接口和管理、调度计算机的所有硬件与软件资源。它所包含的系统程序的具体分类尚不完全统一。其中，最为重要的核心部分是常驻监控程序。计算机开机后，常驻监控程序始终存放在内存中，它通过接收用户命令，启动操作系统执行相应的操作。

操作系统还包括 I / O 驱动程序和文件管理程序，前者用于执行 I / O 操作，后者用于管理存放在外存中的大量数据集合。每当用户程序或其他系统程序需要使用 I / O 设备时，通常并不是由该程序执行操作，而是由操作系统利用 I / O 驱动程序来执行任务。文件管理程序与 I / O 驱动程序配合使用，用于文件的存取、复制和其他处理。

应用软件开发服务程序包括各种高级语言翻译程序、汇编程序、文本编辑程序，系统程序库，连接程序与装入程序以及辅助编写其他程序的程序。

人是通过软件系统与硬件系统发生关系的。通常，由人使用程序设计语言编制应用程序，在系统软件的干预下使用硬件系统。

2. 层次调用关系与编程

对前面的软件分层结构进一步细化，针对具体的 DOS 操作系统，可以得到图 1-7 所示的分级调用结构图。现对其中 4 个层次的功能及如何使用说明如下：

（1）应用层

DOS 应用层是系统与用户的接口，用户通过操作 DOS 命令或运行其他可执行文件而取得系统控制权。多数情况下，该层的应用程序通过高级语言编制或通过其他各种更高级的开发工具包制作。这时，开发人员不必关心 DOS 内部结构和更低层的内容；但对于过程控制等领域的一些问题，高级语言程序不能解决，应用程序要使用汇编语言编制。这时，程序中可能直接调用 DOS 内核提供的系统功能或 ROM-BIOS 提供的设备控制功能，甚至要直接和硬件控制接口打交道。开发人员必须对相应内容有深入了解。

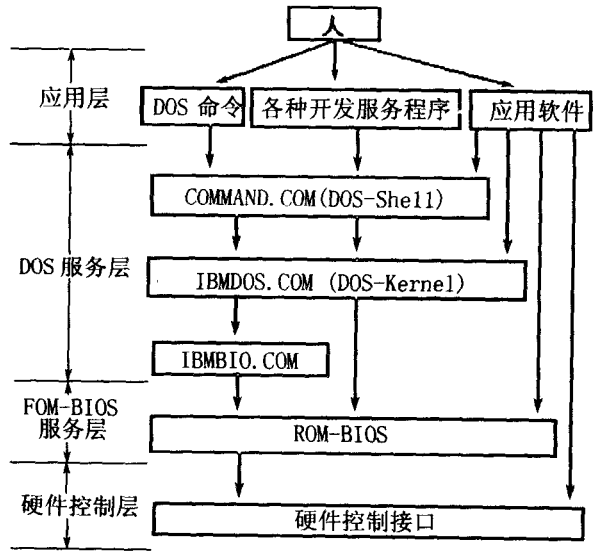

图 1-7　软件的分级调用结构

（2）DOS 服务层

DOS 服务层是为系统软件或应用软件提供系统功能服务的。该层包括 3 个模块。

DOS-Shell 模块对用户输入的 DOS 命令行或应用程序作出响应。若是内部命令，则转入暂驻部分的内部命令程序立即执行；若是外部命令，则首先要将外部命令所对应的程序加载至内存再去执行。然而，在两者具体执行的过程中，都必须调用 DOS-Kernel 模块所提供的系统功能来完成相应的功能。

DOS-Kernel 模块尽管提供了许多的系统功能，但在执行每一个系统功能过程中，完全要依赖各种设备实现指定的功能，因此，它通过一个特定的数据结构“I/O 请求头”来调用 DOS-BIOS 模块中所需的设备驱动程序。

DOS-BIOS 模块对 DOS-Kernel 传送的“I/O 请求头”内容进行解释，最终转换为对固化在 ROM-BIOS 中的设备控制程序的请求。

在 DOS 操作系统中，DOS-Kernel 模块通过 INT 21H 提供了系统功能调用。用户在使用汇编语言编程时，可以直接调用该模块。由于 DOS 的系统功能同硬件环境在逻辑上是完全隔离的，因此，凡调用系统功能所编制的应用程序均不依赖于具体的硬件设备，从而使该类程序能在运行 DOS 的各类微机上正确执行。

（3）ROM-BIOS 服务层

ROM-BIOS 服务层提供调用设备控制功能的服务。ROM-BIOS 是 8KB 长度的固化程序。它是低层的输入输出系统管理模块。除提供加电自测、自举和中断服务外，主要管理对硬件设备一级的控制。它不仅能被 DOS-BIOS 模块调用，而且也可通过 BIOS 中断（如 INT 10H、INT 13H、INT 14H、INT 16H、INT 17H）被一切应用程序所调用。用户可在使用汇编语言编程时直接调用该模块。由于这些依赖于具体硬件的设备控制程序是直接和设备控制器发生关联的（如 1NT 10H 控制显示器适配器、INT 13H 控制磁盘控制器、INT 14H 控制串行口），因此，凡调用该服务层提供的功能编制的程序，都依赖于具体硬件，这类程序仅能在 IBM PC 兼容机上正确执行。

（4）硬件控制层

硬件控制层直接控制具体硬件设备运行。凡欲与硬件设备发生关联的程序员，必须熟知该设备的基本工作原理（仅指逻辑的，并非硬件电路）和各个设备端口的调用方法，而编制的程序只能运行在与 IBM PC 机完全兼容的 PC 机上。如 DOS 或应用软件的汉化、防拷贝软件的编程以及扩展或增设系统设备功能等，都导致对硬件控制层的直接编程。

对微机应用人员而言，建立一个系统层次结构的概念是十分必要的。这将有助于在针对某一课题编程时作出采用哪一层次功能的最佳选择。当使用高级语言或更高级的开发工具时，编程简单，但执行效率低，而用汇编语言调用以上三个层次的功能，随着离硬件愈近，编程愈复杂，然而得到的执行效率愈高。

六、微型计算机工作过程简要分析

微机的工作过程就是执行程序的过程，而程序由指令序列组成，因此执行程序的过程，就是执行指令序列的过程，即逐条地执行指令；由于执行每一条指令都包括取指令（简称取指）与执行指令（简称执指）两个基本阶段，所以，微机的工作过程也就是不断地取指令和执行指令的过程。微机执行程序过程示意图如图 1-8 所示。

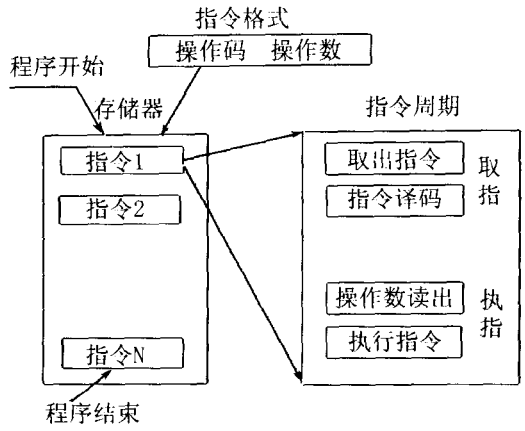

图 1-8 程序执行过程示意图

假定程序已由输入设备存放到内存中。当计算机要从停机状态进入运行状态时，首先应把第 1 条指令所在的地址赋给程序计数器（PC），然后机器就进入取指阶段。在取指阶段，CPU 从内存中读出的内容必为指令，于是，数据寄存器（DR）便把它送至指令寄存器（IR）；然后由指令译码器译码。控制器就发出相应的控制信号，CPU 便知道该条指令要执行什么操作。在取指阶段结束后，机器就进入指令执行阶段，这时，CPU 执行指令所规定的具体操作。当一条指令执行完毕以后，就转入了下一条指令的取指阶段。这样周而复始地循环，一直进行到程序中遇到暂停指令时才结束。

取指阶段都是由一系列相同的操作组成的，所以取指阶段的时间总是相同的，它称为公共操作。而执指阶段将由不同的事件顺序组成，它取决于被执行指令的类型，因此，执指阶段的时间从一条指令到下一条指令变化相当大。

为了进一步说明微机的工作过程，我们来具体讨论一个模型机怎样执行一段简单的程序。例如，计算机如何具体计算 3＋2＝？虽然这是一个相当简单的加法运算，但是，计算机却无法理解。人们必须要先编写一段程序，以计算机能够理解的语言告诉它如何一步一步地去做，直到每一个细节都详尽无误，计算机才能正确地理解与执行。

在编写程序之前，必须首先查阅所使用的微处理器的指令表（或指令系统）。它是某种微处理器所能执行的全部操作命令汇总。不同系列的微处理器各自具有不同的指令表。

假定查到模型机的指令表中可以用 3 条指令求解这个问题。表 1-2 示出了这 3 条指令及其说明。

表 1-2 模型机指令表之一

名称	助记符	机器码		说　　明
		二进制	十六进制	
立即数取入累加器	MOV A，n	10110000 n	B0 n	这是一条双字节指令，把指令第 2 字节的立即数 n 取入累加器 A 中
加立即数	ADD A，n	00000100 n	04 n	这是一条双字节指令，把指令第 2 字的立即数 n 与 A 相加，结果暂存 A 中
暂　停	HLT	11110100	F4	停止所有操作