高等院校规划教材

PUTONG GAODENG YUANXIAO JISUANJI JICHU JIAOYU XILIE JIAOCAI

普通高等院校计算机基础教育系列教材

微型计算机原理及接口技术实验教程

主　编　黄　勤
副主编　唐　丹

WEIXING JISUANJI YUANLI JI JIEKOU JISHU SHIYAN JIAOCHENG

重庆大学出版社
http://www.cqup.com.cn

高等院校规划教材

PUTONG GAODENG YUANXIAO JISUANJI JICHU JIAOYU XILIE JIAOCAI

普通高等院校计算机基础教育系列教材

微型计算机原理及接口技术实验教程

WEIXING JISUANJI YUANLI JI JIEKOU JISHU SHIYAN JIAOCHENG

主　编　黄　勤

副主编　唐　丹

编　者　李　楠　吴传生　胡　青　王延川

重庆大学出版社

内容提要

本书既是《微型计算机原理及接口技术》一书的配套教材，也是一本独立的实验教程。

本书由3篇4章构成，第1篇PC机篇，包含汇编语言程序设计、PC系列微机中断及接口实验，主要涉及汇编语言的编程与调试、汇编语言常用程序设计，以及中断、键盘/显示器、声音接口的硬件结构和编程等；第2篇扩展I/O接口基础及应用篇，其内容主要涉及多种外设接口芯片的硬件结构及编程应用等，其实验内容与教学内容紧密结合；第3篇综合应用设计篇，通过多个实例的介绍，培养学生的应用系统构建能力。

本书可作为高等院校非计算机专业本专科学生微型计算机原理及接口技术课程、计算机硬件技术基础课程教学的实验教程，亦可作为微型计算机原理及接口技术课程设计、计算机硬件技术基础课程设计、计算机控制技术课程设计等的参考教材，还可用作高等教育自学教材和工程技术人员参考书。

图书在版编目(CIP)数据

微型计算机原理及接口技术实验教程/黄勤主编.
—重庆：重庆大学出版社，2015.9
普通高等院校计算机基础教育系列教材
ISBN 978-7-5624-9454-6

Ⅰ.①微… Ⅱ.①黄… Ⅲ.①微型计算机—理论—高等学校—教材②微型计算机—接口技术—高等学校—教材
Ⅳ.①TP36

中国版本图书馆CIP数据核字(2015)第219032号

普通高等院校计算机基础教育系列教材
微型计算机原理及接口技术实验教程
主　编　黄　勤
副主编　唐　丹
责任编辑：章　可　　版式设计：章　可
责任校对：贾　梅　　责任印制：赵　晟
*
重庆大学出版社出版发行
出版人：邓晓益
社址：重庆市沙坪坝区大学城西路21号
邮编：401331
电话：(023) 88617190　88617185(中小学)
传真：(023) 88617186　88617166
网址：http://www.cqup.com.cn
邮箱：fxk@cqup.com.cn (营销中心)
全国新华书店经销
重庆升光电力印务有限公司印刷
*
开本：787×1092　1/16　印张：15.75　字数：364千
2015年9月第1版　2015年9月第1次印刷
印数：1—2 000
ISBN 978-7-5624-9454-6　定价：29.50元

前 言

本书是机械工业出版社出版的《微型计算机原理及接口技术》一书的配套教材，同时也可作为一本独立的实验教程使用。

本书是作者通过10多年的教学实践及教学改革，为达到理论联系实际、培养学生解决实际问题的能力，在原教程基础上进行大量修改精心编写而成。全书共分3篇(4章)，第1篇包含两章内容，第1章简述了80486微处理器的编程结构、指令系统、汇编语言编程实验的一般步骤；汇编语言程序设计的编辑、调试方法及常用汇编语言程序设计实验。第2章介绍了中断接口芯片、定时器接口芯片、并行输入/输出接口芯片等在PC系列微机中的应用，共给出了显示器接口、定时器接口、声音接口等5个接口实验。第2篇(第3章)介绍了扩展I/O接口实验系统即PD-32实验系统的组成、各功能模块作用、使用方法及其8个扩展I/O接口实验。第3篇(第4章)为提高学生应用计算机进行实际系统设计的能力，以5个典型实例介绍了微机应用系统的设计方法。

本实验教程为帮助学生更好地理解和掌握所学理论知识与实际应用的联系，在各章中均给出了相应的例题，同时为巩固所学知识并培养学生独立解决问题的能力，每个实验后都给出相应的思考题，使学生通过本实验教程的学习，进一步提高利用微型计算机解决实际问题的能力。

本教程第1篇和第2篇参考学时数为36学时，教师可根据教学要求对本实验教程实验内容进行取舍；第3篇可在后续课程设计中完成。

本书由黄勤主编，并完成了第1.1、2.1、2.3、3.3、3.6、3.7、3.8节的编写及修改工作；唐丹任副主编，并完成了第1.2、2.2、3.4、3.5、4.2、4.5节、附录的编写及修改工作；李楠完成了第1.3、3.2节的编写及修改工作；吴传生完成了第2.4、3.1、4.1节的编写及修改工作；胡青完成了第4.3节的编写工作；王延川完成了第4.4节的编写工作。

由于编者水平有限，书中难免有疏漏之处，敬请读者批评指正。

编 者

2015年7月

目 录

第 1 篇 PC 机篇

第2篇 扩展I/O技术基础及应用篇

第3篇 综合应用设计篇

第 1 篇

PC 机篇

第1章

汇编语言程序设计

1.1 80486 微处理器的内部结构及指令系统

1.1.1 80486 微处理器的内部结构

1)内部结构

80486 微处理器的内部结构如图 1.1 所示。它主要由 8 个逻辑单元组成:总线接口单元、指令预取单元、指令译码单元、指令执行单元、段管理单元、页管理单元、高速缓冲存储器单元和浮点运算单元。

• 总线接口单元(Bus Interface Unit, BIU) 主要负责与存储器和 I/O 接口传送数据(如预取指令、读/写数据),其功能是产生访问存储器和 I/O 接口所需的地址、数据和控制信号。

• 指令预取单元(Instruction Prefetch Unit, IPU) 负责从存储器取出指令,放到一个 32 B 的指令预取队列中。当指令预取队列不满且总线空闲时,IPU 通过 BIU 从存储器读取指令并放入指令预取队列中。

• 指令译码单元(Instruction Decode Unit, IDU) 负责从指令预取队列中读取指令,进行预译码,译码后的可执行指令放入已译码指令队列中等待执行。如果预译码时发现是转移或调用指令,可提前通知总线接口部件去新的目标地址取指令,以刷新指令预取队列。

• 指令执行单元(Execution Unit, EU) 包括算术逻辑单元(ALU)、8 个 32 位的通用寄存器、桶形移位寄存器和控制单元等,它的作用是完成各种算术逻辑运算和变址地址生成。在控制单元中,大多数指令采用微程序控制执行(控制 ROM),常用基本指令采用硬件逻辑控制执行。

• 段管理单元 用于进行存储器分段管理,将逻辑地址变换为 32 位线性地址。

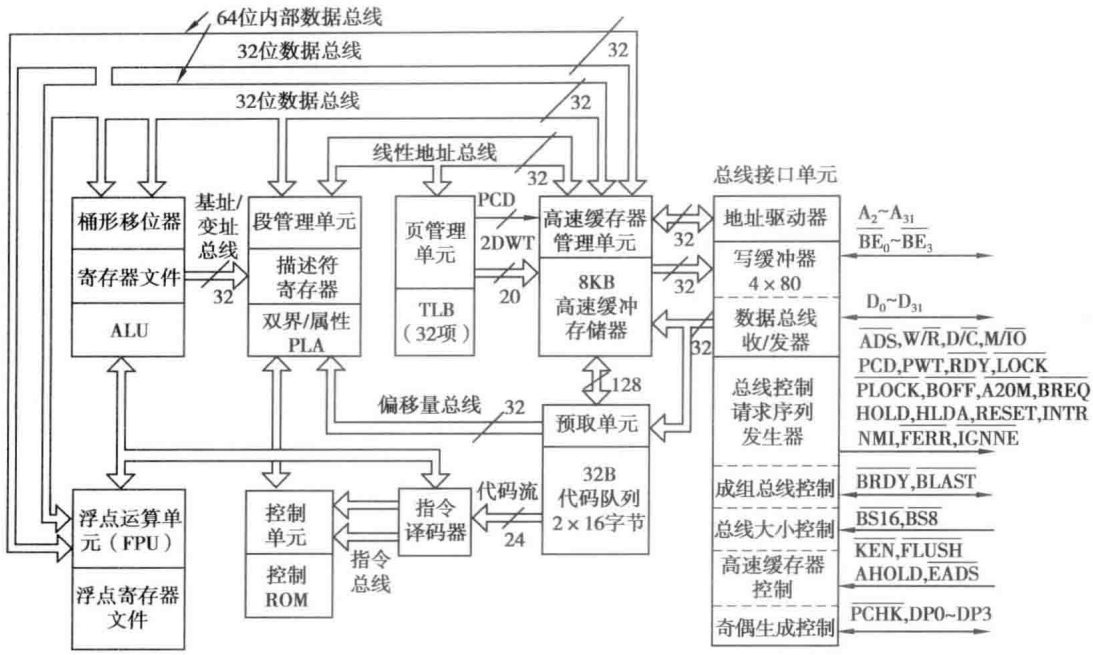

图 1.1　80486 微处理器的内部结构示意图

• 页管理单元　用于进行存储器分页管理，将线性地址变换为 32 位物理地址。该单元属可选单元，若不需用分页管理，线性地址就是物理地址。

注意：段管理单元和页管理单元也可统称为存储器管理单元 MMU。

• 高速缓冲存储器单元　用于加速指令或数据的访问过程。片内 Cache 比片外 Cache 存取速度更快。

• 浮点运算单元　主要负责专门完成浮点数运算、双精度运算等执行单元不擅长的数学运算任务，可与 ALU 的整数运算并行进行。

从图 1.1 可看出，在 Cache-ALU-FPU 之间采用了 64 bit 数据总线直接相连，双精度数据（64 位）可一次传送完；Cache 和指令预取单元之间采用了 128 bit 数据总线，一次可预取 16 B 的指令。

2）内部寄存器组

80486 的寄存器组可分为基本寄存器、系统级寄存器、浮点寄存器、调试和测试寄存器。其中基本寄存器和浮点寄存器是可被应用程序访问的，其他两类只有系统程序才能访问，在此我们不对它们加以讨论。

（1）基本寄存器

基本寄存器包括通用寄存器、指令指针寄存器、标志寄存器和段寄存器，如图 1.2 所示。

• 通用寄存器　80486 有 8 个 32 位通用寄存器 EAX、EBX、ECX、EDX、ESI、EDI、EBP、ESP，它们是由 8086/8088 相应的 8 个 16 位通用寄存器扩展而来。为了与 8086/8088 兼容，它们的低 16 位 AX、BX、CX、DX、SI、DI、BP、SP 可以单独使用。其中 AX、BX、CX、DX 还可进一步分成 8 位寄存器 AH、AL、BH、BL、CH、CL、DH、DL 使用。

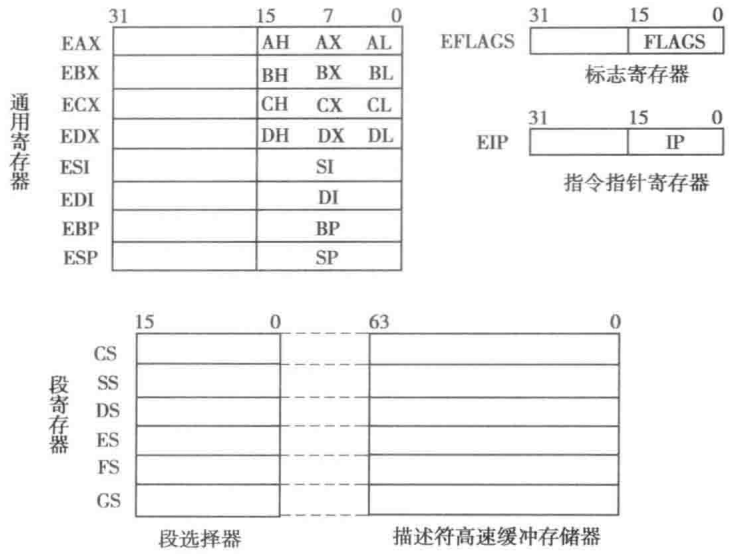

图 1.2　基本寄存器

该类寄存器均可用于存放参加运算的操作数或操作结果(ESP 和 SP 一般不用作此功能),有些还可用于存放操作数所在存储单元的偏移地址(8 位寄存器除外)。除此之外,在一些指令中有部分寄存器还有自己的习惯用法,例如:AX 作累加器;BX 作基址寄存器,存放数据块的基地址;CX 作计数寄存器,在循环和串操作中用作计数器;DX 作数据寄存器,在 I/O 指令中用于保存端口地址;指针寄存器 SP 和 BP 分别为堆栈指针寄存器和基址指针寄存器,它们常常用来存放内存单元的偏移地址;变址寄存器 DI、SI 则主要用于变址寻址方式的目的变址和源变址。

• 指令指针寄存器(EIP)　指令指针寄存器(EIP)又称程序计数器,主要用于保存下一条待取指令在当前代码段中的偏移地址。它的低 16 位 IP 也可以单独使用。当 80486 工作在保护模式下时,使用 32 位的 EIP;工作在实模式下时,使用 16 位的 IP。

• 标志寄存器(EFLAGS)　80486 的标志寄存器是一个 32 位寄存器,它共定义了 15 位 14 个标志,如图 1.3 所示。这些标志分别归类为状态标志、控制标志和系统标志。

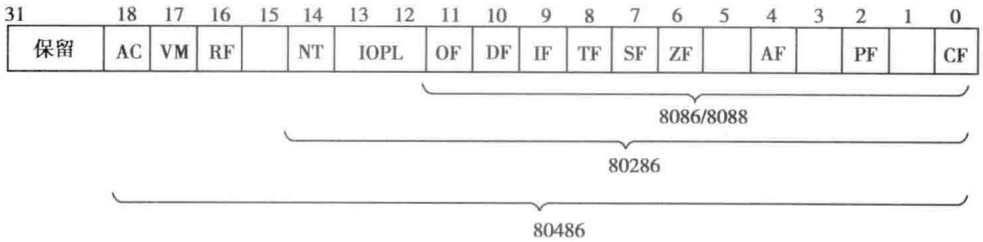

图 1.3　80486 的标志寄存器

①状态标志:状态标志反映指令执行过程和结果的一些特征,共有 6 个,它们是 CF、PF、AF、ZF、SF、OF。

CF(Carry Flag):进/借位标志,反映加/减类运算过程中的进/借位情况。有进/借位时,CF=1;反之,CF=0。

PF(Parity Flag):奇偶标志,反映运算结果中含有"1"的个数的奇偶性。偶数个"1"时,PF=1;反之,PF=0。该标志主要用于检查数据传输过程中是否出错。

AF(Accessary Carry Flag):辅助进/借位标志,在8位二进制数的加法或减法运算过程中,反映运算过程的半进/借位情况。数据D3位向D4位有进/借位时,AF=1;反之,AF=0。

ZF(Zero Flag):零标志,反映运算结果是否为零。当运算结果为零时,ZF=1;反之,ZF=0。

SF(Sign Flag):符号标志,反映运算结果的极性。当运算结果的最高位为1时,SF=1,否则SF=0。对于用补码表示的有符号数,SF=1表示结果为负,SF=0表示结果为正。

OF(Overflow Flag):溢出标志,反映运算过程中是否产生了溢出。有溢出时,OF=1;反之,OF=0。

②控制标志:控制标志仅含一个标志DF,专门用于控制串操作指令在执行过程中,其地址指针的变化方向。

DF(Direction Flag):方向标志,控制字符串操作过程中的地址方向的修改。DF=1,串操作过程中地址按减址方向修改;反之,则为增址。该标志通过指令STD和CLD分别使其为1或0。

③系统标志:系统标志用于控制I/O、屏蔽中断、调试、任务转换和控制保护方式与虚拟8086方式间的转换,共有7个,它们是:TF、IF、IOPL、NT、RF、VM和AC。

TF(Trap Flag):陷阱标志,也称跟踪标志位。TF=1表示CPU进入单步执行方式,即每执行一条指令,自动产生一个内部中断。利用它可逐条地检查指令,完成程序的调试。

IF(Interrupt Enable Flag):中断允许标志,可屏蔽中断INTR的允许/禁止标志。IF=1时,CPU可以响应外部可屏蔽中断请求;IF=0时,CPU禁止响应外部可屏蔽中断请求。

IOPL、NT、R、VM和AC标志均与系统寄存器、调试寄存器或CPU工作在保护模式下有关,在此不再一一介绍,请参阅《微型计算机原理及接口技术》教材。

• 段寄存器　80486有CS、SS、DS、ES、FS、GS共6个16位的段寄存器。它们可同时用于6个分段,CS指出当前代码段,SS指出当前堆栈段,DS、ES、FS、GS指出当前数据段。实模式下分别用来标志代码段、堆栈段、数据段的段地址,用以扩大可访问的存储空间,实现CPU对内存不同段的读写。此时每个段的大小不能超过64 kB。

保护模式下用于存放段选择符,由选择符决定段地址,标志当前可编址的存储器段。此时每个段的大小不能超过4 GB。

(2)浮点寄存器

浮点寄存器包括8个80位数据寄存器及标记字、控制、状态、指令指针、数据指针寄存器,用以完成浮点处理。

1.1.2　80486的运行模式

80486能以实模式、保护模式和虚拟8086三种模式运行程序。

1)实模式

实模式全称为实地址存储管理操作模式,实际上是8086的程序运行模式,程序与数据

运行在实际存储空间,无存储保护,内存 1 MB,采用段式存储管理。

在实模式下,80486 除了多用几种指令,就如同高速的 8086。开机时处于此模式,直到程序中明显地将它转换为保护模式。实模式是学习汇编语言的主体模式,本教材以实模式下的汇编语言程序设计为背景进行介绍。

2)保护模式

保护模式全称为保护虚地址存储管理操作模式,以实模式提供它所做的一切,程序与数据运行在虚拟存储空间,有存储保护。保护模式实际内存达 4 GB,内存管理采用段式及段页式管理。

3)虚拟 8086 模式

虚拟 8086 模式是一种既有保护功能又能执行 8086 代码的工作方式。

1.1.3 80486 指令系统

80486 的指令系统向上兼容,都是在 8086 指令的基础上发展形成的。80486 增加了 32 位操作和访问存储器的 32 位寻址方式。

80486 可以工作在实模式、保护模式和虚拟 8086 模式,为了支持系统工作模式,指令系统中设计了系统管理指令、保护模式控制指令以及高级语言支持指令等。作为汇编语言程序设计的基础,我们仅给出 80486 的基本指令集。

80486 指令系统可以分为整数指令、浮点数指令和操作系统型指令三大类,其中整数指令按功能可进一步分为数据传送指令、算术运算指令、逻辑运算与移位指令、串操作指令、控制转移指令、按条件设置字节指令、处理器控制指令、高级语言指令等;浮点数指令按功能也可分为数据传送、算术运算、比较、超越函数、常量、控制等指令。附录 2 给出了 80486 有关指令的助记符、功能及对标志的影响。

1.2 汇编语言程序的编辑与调试

计算机程序设计语言可分为高级语言与汇编语言两大类。高级语言是面向过程的语言,利用这种语言编程,程序员可以完全不考虑机器的结构特点,不必了解和熟悉机器的指令系统。

汇编语言是面向机器的语言,它是机器语言的符号表示,由汇编语言所编写的源程序,必须翻译成目标程序才能执行,该翻译过程称为汇编,完成汇编任务的程序称为汇编程序。

本节主要介绍汇编语言源程序的编辑、汇编、链接、调试过程的相关基本知识。

1.2.1 汇编语言源程序的基本结构

一个汇编语言源程序可由多个逻辑段(即若干个堆栈段、数据段、代码段)组成,但在一般情况下,对 80486CPU 而言,当前段只有 6 个,即代码段(CS)、堆栈段(SS)、数据段(DS,ES,FS,GS)。在程序中经常使用的是代码段(CS)、堆栈段(SS)和数据段(DS、ES)。

1)汇编语言源程序的一般结构

汇编语言源程序的一般结构如下:

```
SSEG    SEGMENT   [USE16/USE32]        STACK
        …
(数据定义伪指令序列)
        …
SSEG    ENDS
DSEG    SEGMENT          [USE16/USE32]
        …
(数据定义伪指令序列)
        …
DSEG    ENDS
ESEG    SEGMENT          [USE16/USE32]
        …
(数据定义伪指令序列)
        …
ESEG    ENDS
CSEG    SEGMENT   [USE16/USE32]
        ASSUME  CS:CSEG,DS:DSEG,ES:ESEG,SS:SSEG
START:  MOV     AX,DSEG
        MOV     DS,AX                ;置 DS 初值
        MOV     AX,ESEG
        MOV     ES,AX                ;置 ES 初值
        MOV     AX,SSEG
        MOV     SS,AX                ;置 SS 初值
                …
        MOV     AH,4CH               ;返回 DOS 操作系统
        INT     21H
CSEG    ENDS
        END     START
```

2)程序的正常结束方式

(1)将主程序定义为远过程

当输入文件名执行程序时,可执行文件被读入到内存中并立即执行。实际上,每次执行文件时,随着执行文件装入内存的还有一个程序的段前缀区(即 PSP)。在内存中,它被安装在执行文件的前面。PSP 中存有许多关于程序启动、执行、结束以及进程调度、进程环境地址和进程标志等重要信息。

在 PSP 的首地址字单元放着一条 INT 20H 指令,该指令的功能是返回 DOS 操作系统。

当程序执行前，PSP 所在段的段地址在 DS、ES 段寄存器中，而 INT 20H 指令所在位置的偏移量为 0。用户程序结束时，若想利用 PSP 的 INT 20H 指令返回 DOS 操作系统，可使用以下程序结构：

```
DATA    SEGMENT
        ……                          ;数据段定义内容
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA
MAIN    PROC    FAR
START:  PUSH    DS
        SUB     AX,AX
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        ……                          ;核心程序段
        RET
MAIN    ENDP
CODE    ENDS
        END     START
```

(2)采用 DOS 4CH 号功能返回

当采用 DOS 4CH 号功能返回时，程序结构如下：

```
DATA    SEGMENT
        ……                          ;数据段定义内容
DATA    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG, DS:DATA
START:  MOV     AX,DATA
        MOV     DS,AX
        ……                          ;核心程序段
        ……
        MOV     AH,4CH
        INT     21H
CSEG    ENDS
        END     START
```

3)完整源程序举例

一个完整的汇编源程序如下：

```
STACKA  SEGMENT  STACK              ;堆栈段
```

```
        DB   100   DUP(?)              ;设有 100 个字节的堆栈空间
STACKA  ENDS
DATA    SEGMENT                       ;数据段
BUF1    DB     9                      ;定义两个加数
BUF2    DB     7
RESULT  DB     ?                      ;定义结果单元
DATA    ENDS
CODE    SEGMENT
        ASSUME     CS:CODE,DS:DATA,SS:STACKA
                                      ;建立段与寄存器的关系
START:  MOV    AX,DATA                ;置 DS 初值为 DATA 段的起始地址
        MOV    DS,AX
        MOV    AX,STACKA
        MOV    SS,AX
        MOV    AL,BUF1
        ADD    AL,BUF2
        MOV    RESULT,AL
        MOV    AH,4 CH
        INT    21H
CODE    ENDS
        END    START
```

1.2.2 汇编语言源程序的开发过程

汇编语言源程序的开发过程需要经过编辑、汇编、链接和调试4个步骤，如图1.4所示。Borland公司推出的汇编编译器TASM可完成汇编语言程序的开发过程，但汇编编译器TASM无法在64位操作系统(如Windows 7 64位)上运行，此时可通过DOSBOX软件来模拟DOS环境完成汇编语言程序的开发。下面分别介绍在64位和32位操作系统上进行汇编语言程序的开发过程。

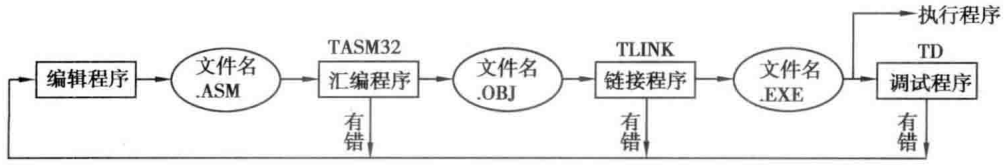

图1.4 汇编语言源程序的开发过程

1)在64位操作系统上汇编语言源程序的开发过程

在64位操作系统上通过DOSBOX与TASM软件完成汇编语言源程序的开发，其步骤如下：

(1)建立文件夹

将 TASM 软件拷贝在计算机任意硬盘分区(如 D 盘)中,在“D:\TASM\”路径下新建一个自己的文件夹。

(2)源程序的编辑

在文本文档方式下编写汇编语言源程序并以.ASM 扩展名存入指定文件夹中,如“D:\TASM\MYWORK\1.ASM”,形成汇编语言源程序文件。

(3)设置运行环境

已编辑完成的源程序,必须在规定环境下才能对其进行汇编、链接、调试,其环境设置方法如下:

①运行 DOSBOX,双击 DOSBOX 图标时会生成两个窗口,如图 1.5 所示,其中上面的窗口为 DOSBOX 命令输入窗口。

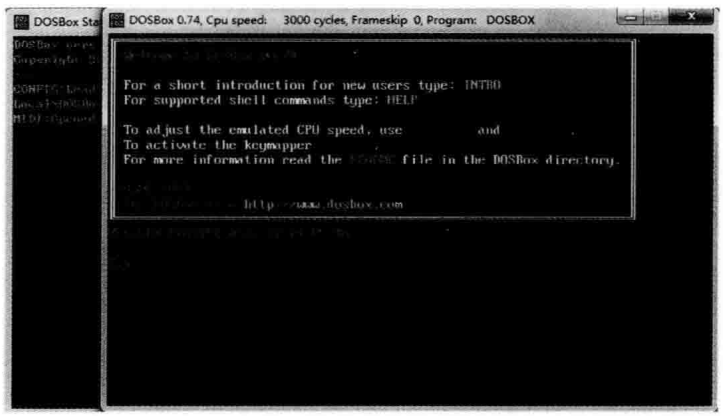

图 1.5 DOSBOX 窗口

②在 DOSBOX 命令输入窗口中依序输入下列 DOS 命令,设置运行环境。

命令 1:mount d d:\tasm\

该命令将 d:\tasm 路径映射为 DOS 窗口中的 D 盘,此后在 DOS 窗口中访问 D 盘即为访问系统中的 d:\tasm。

命令 2:d:

该命令将 DOS 窗口的当前路径切换为 D 盘根目录。

命令 3:path d:\bin\

该命令将 d:\bin\设置为 DOS 系统的搜索路径。

命令 4:cd mywork

该命令为进入存放有汇编语言源程序文件的文件夹。

其操作过程如图 1.6 所示。

运行环境设置完毕,便可在 DOSBOX 命令窗口中,通过相应命令的执行,实现对汇编语言源程序的汇编、链接和调试。

(4)源程序的汇编

汇编也称编译,即通过汇编程序 TASM32 对源程序进行翻译,生成扩展名为 OBJ 的目标文件。