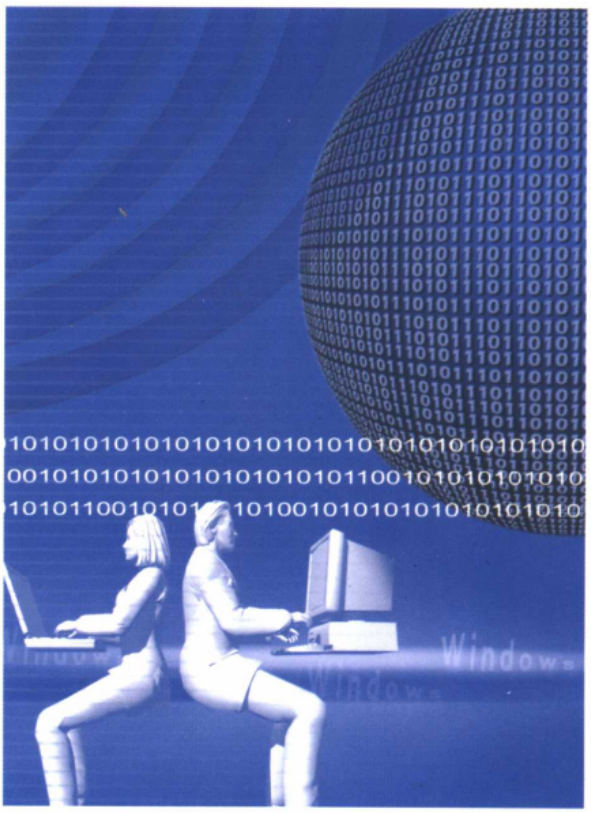
高等院校计算机应用技术系列教材

DOS/Windows 汇编语言程序设计教程

- 汇编语言16位指令
- 汇编程序设计技巧
- 硬件控制、程序加密与调试
- 32位指令与Windows汇编语言
- Windows汇编语言程序设计



赵树升 杨建军 编著

清华大学出版社

高等院校计算机应用技术系列教材

DOS/Windows 汇编语言程序设计教程

赵树升　杨建军　编著

清华大学出版社

北　京

内 容 简 介

本书以 Intel 80X86 指令系统和汇编语言开发工具 Masm611 与 Masm32 为主体，在 PC 机的 MS-DOS 和 Windows 环境中，较为系统地介绍了进行汇编语言程序设计所需要的指令、语法以及调试工具 Debug 和 W32Dasm。全书包括 MS-DOS 部分和 Windows 部分。第 1 章介绍了与软、硬件相关的基础知识。第 2 章详细介绍了主要的硬件指令、伪指令、源程序格式、程序设计与调试过程。第 3 章讲述了常用的程序设计技巧，重点介绍了分支程序、循环程序、中断程序和宏。第 4 章以实例介绍了汇编语言在 MS-DOS 下的应用。第 5 章介绍了 32 位汇编语言的特点以及 Windows 下汇编语言的特点。第 6 章介绍了 Windows 下如何实现窗口、鼠标、视频、磁盘、文件与进程相关的程序设计。本书实用性非常强，结构清晰，着重培养动手能力。

本书可以作为高等院校汇编语言课程的教材，适用于计算机科学与技术以及相关专业的学生。

图书在版编目(CIP)数据

DOS/Windows 汇编语言程序设计教程/赵树升　杨建军　编著. —北京：清华大学出版社，2005.6
(高等院校计算机应用技术系列教材)
ISBN 7-302-11082-4

Ⅰ.D…　Ⅱ.①赵…②杨…　Ⅲ. 汇编语言—程序设计—教材　Ⅳ. TP313

中国版本图书馆 CIP 数据核字(2005)第 050986 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：胡辰浩
文稿编辑：鲍　芳
封面设计：王　永
版式设计：康　博
印 刷 者：北京市昌平环球印刷厂
装 订 者：三河市化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：23.5　**字数**：543 千字
版　　次：2005 年 6 月第 1 版　2005 年 6 月第1 次印刷
书　　号：ISBN 7-302-11082-4/TP·7339
印　　数：1～5000
定　　价：32.00 元

前　言

本书以 Intel 80X86 指令系统和汇编语言开发工具 Masm611 与 Masm32 为主体，在 PC 机的 MS-DOS 和 Windows 环境中，较为系统地介绍了进行汇编语言程序设计所需要的指令、语法以及调试工具 Debug 和 W32Dasm。

按照我国《普通高等学校本科专业目录和专业介绍》的规定，“汇编语言”是计算机科学与技术专业的主要课程，也是其他相关专业的基础课程。该课程的教学已经有 20 多年的历史，教材品种繁多。但是，由于计算机技术的飞速发展，以前的教材和教学方法已经不能完全适应今天的人才培养需要。在 10 多年使用汇编语言工具和 3 年教学经历后，本人编写了本书。

本书具有以下特点。

面向实用：学习汇编语言主要是为了应用，而不是研究汇编语言本身。因此，本书重点讲解了常用指令、语法的应用。本书专门使用两章内容分别讲述了 MS-DOS 和 Windows 的应用实例，着重强调了 Masm611 与 Masm32、Debug 与 W32Dasm 的应用。

面向发展：目前汇编语言更多使用于 Windows 环境下，例如病毒分析、软件调试和软件加密。并且学习完 Windows 下的汇编语言，对面向对象程序设计会有较为深入的了解。因此，本书使用 1 章的篇幅介绍了 Windows 下的汇编程序设计和如何使用 Windows 下的调试工具 W32Dasm。

注重趣味性：汇编语言已经不像十几年前普遍作为工具用于程序开发，目前更多地是作为专业基础课程。如果继续把一些算法放到书中，用汇编去实现，学生学习起来会索然寡味。因此，本书实例提供了怎样做钥匙盘、怎样对软件加密、怎样调试修改程序、怎样与硬盘、键盘进行低级交互等，学生会非常有兴趣。

全书共分 6 章。第 1、2、3 章主要由杨建军老师编写，第 4、5、6 章由郑州大学升达经贸管理学院赵树升老师编写。最后由赵树升老师进行统稿。

限于作者的学术水平，本书难免存在疏漏和不当之处，敬请广大同行和读者指正。我们的邮箱是：huchenhao@263.net。

赵树升

2005 年 3 月

目　录

第 1 章　汇编语言基础知识……1
1.1　汇编语言的由来与特点……1
1.1.1　机器语言……1
1.1.2　汇编语言……1
1.1.3　汇编语言的应用领域与地位……2
1.2　系统结构……2
1.2.1　微型计算机的系统结构……2
1.2.2　INTEL 8086/8088 16 位机的系统结构……4
1.2.3　INTEL 80386 32 位机的系统结构……9
1.3　数据表示……17
1.3.1　数制及其转换……17
1.3.2　有符号数的表示法……19
1.3.3　BCD 码和字符编码……20
1.4　小结……21
1.5　习题……21

第 2 章　16 位汇编程序设计……24
2.1　16 位汇编程序设计概述……24
2.1.1　汇编语言程序的特点……24
2.1.2　指令寻址方式……26
2.2　16 位指令系统……32
2.2.1　数据传送类指令……32
2.2.2　算术运算类指令……44
2.2.3　位运算类指令……56
2.2.4　串操作类指令……61
2.2.5　控制转移类指令……67
2.2.6　处理机控制类指令……77
2.3　伪指令……79
2.3.1　常量、变量、标号和表达式……79
2.3.2　伪指令语句……84
2.3.3　结构……89
2.3.4　记录……91
2.4　宏汇编程序格式……93
2.4.1　完整段定义格式……94
2.4.2　简化段定义格式……97
2.4.3　与完整段定义有关的伪指令……98
2.4.4　与简化段定义有关的伪指令……100
2.5　汇编程序设计过程……103
2.5.1　汇编程序的设计步骤……103
2.5.2　程序流程图……104
2.5.3　宏汇编 MASM 611 命令行方式上机操作……106
2.5.4　DEBUG 及 CV 调试程序的使用……109
2.6　小结……117
2.7　习题……117

第 3 章　程序设计技巧……125
3.1　顺序程序结构形式……125
3.2　分支程序……128
3.2.1　分支程序结构……128
3.2.2　条件控制伪指令……129
3.2.3　双分支程序设计……132
3.2.4　多分支程序设计……134
3.3　循环程序……139
3.3.1　循环程序概述……139

3.3.2 循环控制指令及伪指令……141
3.3.3 循环程序设计方法……142
3.3.4 多重循环程序设计方法……150
3.4 子程序与扩展子程序……151
3.4.1 一般过程定义(子程序)伪指令……151
3.4.2 子程序参数传递方法……154
3.4.3 扩展过程定义(扩展子程序)伪指令……162
3.5 中断程序……167
3.5.1 中断的有关概念……167
3.5.2 8086/8088 的中断源及其优先级……168
3.5.3 中断向量表……171
3.5.4 中断处理的基本过程……172
3.6 输入输出程序……176
3.6.1 输入输出原理……176
3.6.2 I/O 程序设计方法……178
3.7 宏结构程序……183
3.7.1 宏汇编……183
3.7.2 重复汇编……192
3.7.3 条件汇编……194
3.8 模块化程序……197
3.8.1 模块化的特点……198
3.8.2 源程序文件包含……198
3.8.3 目标代码文件包含……200
3.8.4 子程序库……209
3.9 小结……210
3.10 习题……210

第 4 章 16 位汇编应用举例……215

4.1 模拟 C 语言函数实现……215
4.1.1 数据转换……215
4.1.2 字符串操作……219
4.2 键盘中断……222
4.2.1 键盘控制原理……222
4.2.2 通过中断获取键盘信息……224
4.3 视频控制程序……226
4.3.1 直接控制显存……226
4.3.2 使用 BIOS……228
4.3.3 使用 DOS 功能……234
4.4 磁盘控制程序……235
4.4.1 常用的 INT 13H 功能……235
4.4.2 设计一个简单钥匙软盘程序……236
4.4.3 设计软盘扫描程序……239
4.4.4 读写大硬盘扇区数据……243
4.5 中断程序设计……248
4.5.1 设计自己使用的中断程序……249
4.5.2 设计驻留内存的中断程序……251
4.6 引导程序设计……254
4.6.1 引导程序原理……254
4.6.2 引导程序的编写与安装……256
4.7 定时器应用程序……261
4.7.1 产生时钟原理……261
4.7.2 定时器代码实现……262
4.8 用 Debug 修改程序结构……266
4.8.1 修改代码原理……267
4.8.2 修改程序代码……269
4.9 加密一个 EXE 文件……270
4.9.1 加密前的代码识别……271
4.9.2 加密方法的实现……271
4.10 小结……274
4.11 习题……274

第 5 章 32 位汇编程序设计……276

5.1 32 位指令系统……276
5.1.1 32 位的寻址方式……276

5.1.2 32 位扩展指令……278
5.1.3 80386 新增指令……281
5.1.4 80486 新增指令……285
5.1.5 Pentium 新增指令……287
5.1.6 Pentium Pro 新增指令……290
5.2 DOS 下 32 位汇编程序……290
5.2.1 32 位程序编写规范……291
5.2.2 DOS 32 位程序举例……292
5.3 Windows 下 32 位汇编程序…295
5.3.1 Windows 汇编语言特点……296
5.3.2 Masm32 的使用服务……298
5.4 Windows 程序的反汇编……302
5.4.1 W32Dasm 的使用……302
5.4.2 W32Dasm 的反汇编代码阅读……304
5.5 Windows 程序的调试……307
5.6 小结……311
5.7 习题……311

第 6 章 Windows 32 位汇编应用举例·313

6.1 窗口设计程序……313
6.1.1 窗口程序的运行过程……313
6.1.2 在窗口上添加子窗口……320
6.1.3 子窗口的控制……322
6.1.4 复杂形状的窗口……324
6.1.5 资源……325
6.1.6 一个简单对话框……329
6.1.7 菜单应用举例……331
6.2 键盘控制程序……335
6.2.1 获取输入字符……335
6.2.2 虚拟键盘信息……335
6.3 鼠标控制程序……337
6.3.1 获取鼠标信息……337
6.3.2 鼠标的控制……338
6.4 视频控制程序……339
6.4.1 在窗口中绘图……339
6.4.2 位图按钮……343
6.5 磁盘文件操作与内存操作……344
6.5.1 文件操作……345
6.5.2 内存操作……347
6.5.3 内存与文件使用举例……349
6.6 定时器程序……353
6.6.1 建立定时器……353
6.6.2 定时器应用举例……354
6.7 进程控制……355
6.7.1 获取命令行参数……356
6.7.2 建立进程……356
6.8 控制台程序……358
6.9 动态链接库……359
6.9.1 动态链接库的概念……359
6.9.2 动态链接库的建立……360
6.9.3 动态链接库的使用……362
6.10 小结……364
6.11 习题……364

参考文献……366

第1章　汇编语言基础知识

程序设计语言是开发软件的工具，它的发展经历了由低级语言到高级语言的过程。汇编语言相对于中级语言 C 和高级语言 VB、VC，是一种面向机器的低级程序设计语言。

1.1　汇编语言的由来与特点

汇编语言是在机器语言之上出现的，是高级语言的基础。

1.1.1　机器语言

机器指令指的是 CPU 能直接识别并执行的指令，它的表现形式是二进制编码。机器指令通常由操作码和操作数两部分组成。操作码指出该指令所要完成的操作，即指令的功能;操作数指出参与运算的对象，以及运算结果所存放的位置等。

机器语言直接用机器指令来设计程序，是 CPU 能直接识别的惟一的语言。也就是说，CPU 能直接执行用机器语言描述的程序。在计算机诞生的早期，程序员就是用机器语言编写程序的。由于用机器语言编写的程序不易读，出错率高，难以维护，不能直观地反映计算机解决问题的基本思路，所以现在很少用了。

1.1.2　汇编语言

虽然用机器语言编写程序有很高的要求和许多不便，但编写出来的程序执行效率高。CPU 严格按照程序员的要求去做，没有多余的操作。所以，在保留程序执行效率高的前提下，人们就开始着手研究一种能大大改善程序可读性的编程方法。

为了改善机器指令的可读性，人们选用了一些能反映机器指令功能的单词或词组来代表该机器指令，而不再关心机器指令的具体二进制编码。与此同时，也把 CPU 内部的各种资源符号化，使用该符号名等于引用了具体的物理资源。这样，令人难懂的二进制机器指令就可以用通俗易懂、具有一定含义的符号指令来表示了。于是，汇编语言就有了雏型。现在，我们称这些具有一定含义的符号为助记符。用指令助记符、符号地址等组成的符号指令称为汇编格式指令。

汇编语言是汇编指令集、伪指令集和使用它们的规则的统称。伪指令是在程序设计时所需要的一些辅助性说明指令，它不对应具体的机器指令，有关内容在以后各章节中会有详细叙述，在此不展开介绍。

用汇编语言编写的程序称为汇编语言程序，也可简称为源程序。汇编语言程序要比用机器指令编写的程序容易理解和维护。

1.1.3 汇编语言的应用领域与地位

汇编语言的优点在于与机器相关且具备执行的高效率。但也导致了其可移植性差和调试难的缺点。所以，我们在选用汇编语言时要根据实际的应用环境，尽可能避免其缺点对整个应用系统的影响。汇编语言主要适用于下列领域：

- 要求执行效率高，反应快的领域，如操作系统内核，工业控制，实时系统等。
- 系统性能的瓶颈，或频繁被使用的子程序或程序段。
- 与硬件资源密切相关的软件开发，如设备驱动程序。
- 受存储容量限制的应用领域，如家用电器的计算机控制功能等。

汇编语言也有很多不宜使用的领域，尤其是在大型应用软件如信息管理系统等的整体开发中。

作为计算机科学与技术专业的专业必修课程，学好汇编语言具有下列意义。

- 理解硬件的控制原理：掌握如磁盘驱动器、键盘、鼠标等的控制原理。在学过汇编语言以后，可以和以前学习过的《计算机原理》结合起来，对硬件的控制，硬件与软件的交互机理有深入的认识。
- 深入理解高级语言：学过 C 语言后，知道 C 语言用 cin 和 cout 怎么输入输出字符，在汇编中可以知道它们是怎么实现的。其他语句如 switch、if...else、goto 等，其他函数如 strcmp、strlen、strcmp，也可以模拟它们的实现。
- 软件的加密解密：可使用汇编语言对软件进行加密解密。
- 信息安全：汇编语言适合分析病毒为什么能感染可执行文件的问题，假设一个文件感染了病毒，可以用汇编语言去分析它，来祛除病毒代码。

1.2 系 统 结 构

本节首先介绍微型计算机系统的基本结构，然后分别以 8086 和 80386 为例介绍 16 位微处理器的结构和 32 位微处理器的结构。

1.2.1 微型计算机的系统结构

微型计算机的系统包括硬件系统和软件系统。

1. 硬件系统

硬件是指构成计算机物理设备的实体(如 CPU、显卡、外部设备等)，一台计算机所有硬件的集合构成了计算机的硬件系统。一般将其分为微处理器、主存储器、接口电路、外部设备和系统总线等，如图 1-1 所示。

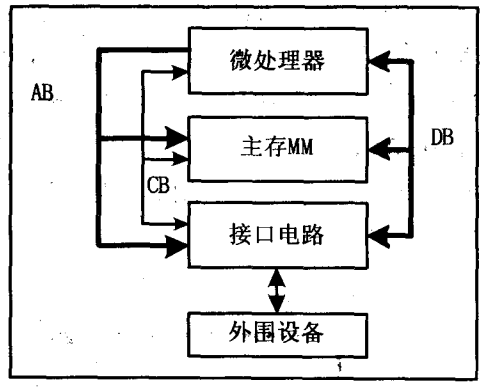


图 1-1　微型计算机的系统结构

微处理器就是由控制器和算术逻辑部件(ALU)组成的中央处理器(即 CPU)。它的作用是自动地执行各条指令，协调整个系统的工作。

主存储器是计算机的记忆装置，用于存储计算机当前正在执行的程序和数据。人们通常接触的是 RAM，它是一种随机存取存储器。它存储的数据在计算机重启或关机后会丢失，而且在计算机运行时还需不断刷新。

系统总线是将 CPU 与存储器及外部设备连接起来的总线，用来传输信息。按传输信息的种类可把系统总线分为地址总线(AB)、数据总线(DB)和控制总线(CB)。

外部设备按功能可分为两类：一类是与计算机进行通讯的设备如键盘、打印机和显示器等，这些设备与计算机的通讯是通过 I/O 接口实现的。另一类是用来存储信息的设备如磁盘、光盘等。

2. 软件系统

软件是所有程序和数据的总称。通常我们将它分为系统软件和应用软件两大类。

系统软件用来对计算机系统的实际运行进行控制、管理和服务。它主要分为操作系统(如 DOS、Unix、Windows、Macintosh 和 Linux 等)、诊断程序、调试程序和语言处理程序(如编译程序、解释程序和汇编程序等)。

应用软件指的是用户自己编写的各类应用程序。

1.2.2　INTEL 8086/8088 16 位机的系统结构

1. 8086/8088 CPU 的功能结构

Intel 8086/8088 是第三代微处理器。它们与第二代微处理器 8080/8085 是兼容的。8086 有 20 条地址线，16 条数据线，直接寻址的内存空间可达 1MB(2^{20})。8088 和 8086 的内部组成完全相同，不同的是 8088 外部数据总线只有 8 条。因此 8088 被称为准 16 机。IBM PC 机及其兼容机上广泛采用了 8088 CPU。

8086/8088 CPU 按功能可分为两个部分：总线接口单元(BIU: Bus Interface Unit)和执行单元(EU: Execute Unit)。如图 1-2 所示。

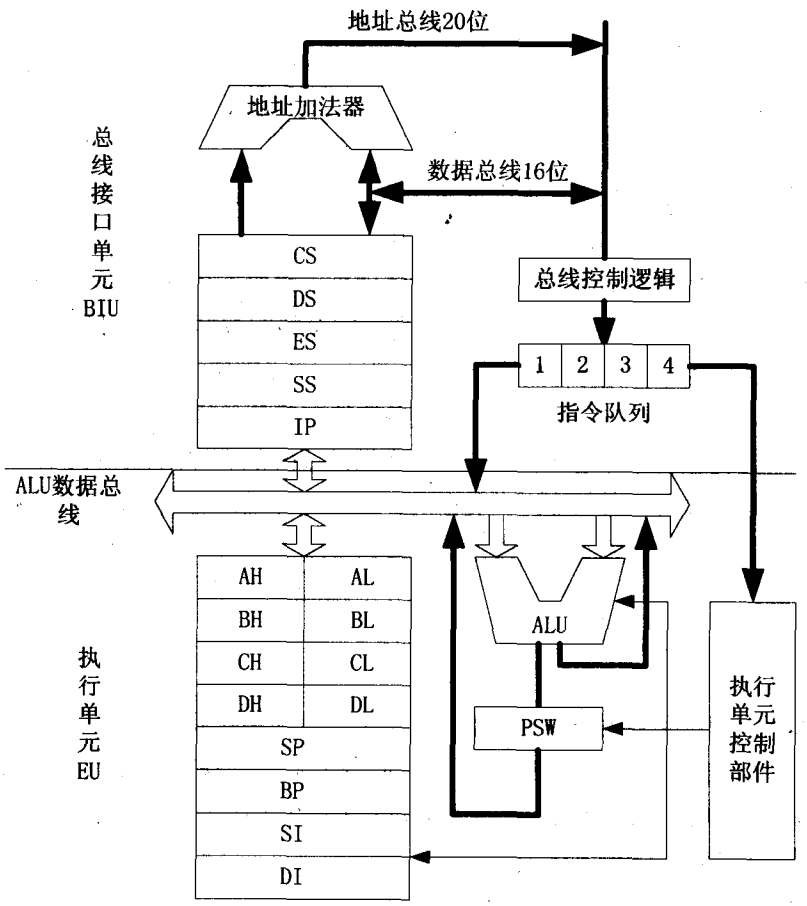


图 1-2　8086/8088 CPU 的功能结构图

BIU 是由地址加法器，指令指针寄存器 IP，指令流字节队列和 4 个段寄存器(ES、CS、SS、DS)所组成的，它主要负责 CPU 与存储器及外部设备之间的信息传输。

EU 由算术逻辑单元 ALU(Arithmetic and Logic Unit)，执行单元控制部件，8 个 16 位寄存器和 1 个标志寄存器 PSW 组成，它负责全部指令的执行，即负责向总线接口单元提

供数据和地址，并对通用寄存器和标志寄存器进行管理，在 ALU 中进行算术运算和逻辑运算。

BIU 和 EU 是分开的，而且是并行工作的，取指令和执行指令同时进行，从而提高了处理器的工作效率。

2. 8086/8088 CPU 的寄存器

这里主要介绍和汇编语言编程有关的寄存器，如图 1-3 所示。我们应该从 3 方面去掌握它们：名称、代号、作用。

(1) 数据寄存器

数据寄存器包括 AX，BX，CX 和 DX4 个 16 位寄存器。其中任意一个数据寄存器均可以分为两个 8 位寄存器来使用，如 AX 可分解为 AH 和 AL，BX 可分解为 BH 和 BL。它们的作用如表 1-1 所示。

(2) 指针和变址寄存器

它们包括 SP、BP、SI、DI 这 4 个 16 位寄存器。这 4 个寄存器只能以字(16 位)为单位使用。编程时可以用它们存放操作数，但它们更常用于存放存储器操作数的偏移地址。

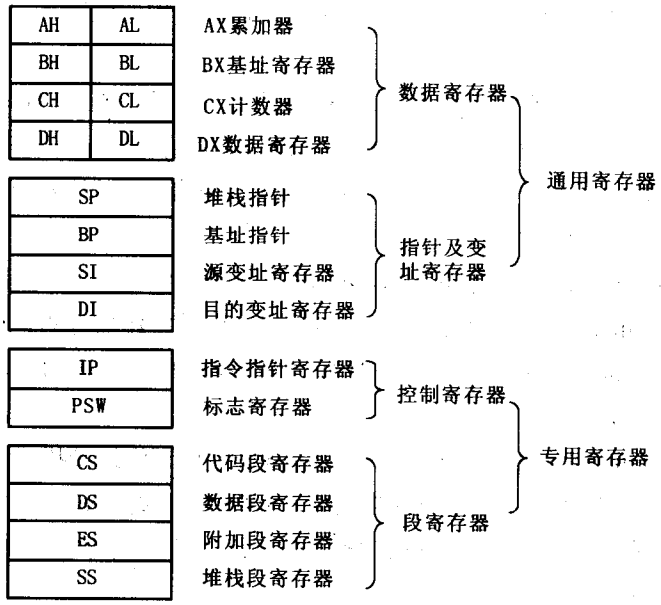


图 1-3　8086/8088 CPU 的寄存器结构

表 1-1　通用寄存器的作用

寄　存　器	功　　能
AX	字乘法(存放乘积的高 16 位)，字除法(存放被除数及余数)
AL	字节乘法，字节除法，字节 I/O，换码，十进制调整指令
AH	字节乘法(存放乘积的高 8 位)，字节除法(存放余数)

(续表)

寄 存 器	功　能
BX	换码指令 XLAT 的表首地址
CX	循环与字符串指令
CL	移位指令与循环控制
DX	字乘除指令，间接 I/O

SP(Stack Pointer)是堆栈指针寄存器，用来指示堆栈栈顶单元的偏移地址，SP 和堆栈段寄存器 SS 一起确定栈顶存储单元的物理地址。

BP(Base Pointer)是基址指针寄存器，用来指示堆栈区中任意一个单元的偏移地址或偏移地址的基地址。

SI(Source Index)是源变址寄存器。DI(Destination Index)是目的变址寄存器。这两个寄存器一般用来指示数据段中任意一个存储单元的偏移地址。在串处理指令中，SI 和 DI 作为隐含的源变址寄存器和目的变址寄存器使用，这时指令对 SI 和 DI 具有自动增量或减量的功能(若方向标志位 DF=0，则 SI、DI 自动增量；若 DF=1，则 SI、DI 自动减量)。SI 和 DS 确定源操作数的物理地址，DI 和 ES 确定目的操作数的物理地址。

(3) 段寄存器

有 4 个 16 位段寄存器 CS、DS、ES、SS，它们只能以字(16 位)为单位使用。由于内存地址是 20 位，而寄存器是 16 位的。为了读写 20 位的内存数据，用两个寄存器组成逻辑地址。一个叫段地址，另一个叫偏移地址。物理地址由段地址乘以 16 变成 20 位，再加上偏移地址得到。4 个段寄存器如下。

- CS(Code Segment)：代码段寄存器，和 IP 寄存器组成逻辑地址描述代码段。
- DS(Data Segment)：数据段寄存器，和 BX、立即数等组成逻辑地址描述数据段。
- ES(Extra Segment)：附加数据段寄存器，在数据块操作时和 DI 寄存器组成逻辑地址描述数据段。
- SS(Stack Segment)堆栈段寄存器，和 SP 或 BP 组成逻辑地址描述堆栈段。

8086/8088 内存管理采用分段的方法来实现。8086/8088 把 1MB 的内存空间分成若干个段，CS 存放代码段的段地址，DS 存放数据段的段地址，ES 存放附加数据段段地址，SS 存放堆栈段的段地址。

(4) 控制寄存器

控制寄存器有两个 16 位寄存器，分别用 IP 和 PSW(即 FLAGS)表示。

IP(Instruction pointer)指令指针寄存器：存放的是代码段的偏移地址。程序执行时用来指示将要执行的下一条指令的偏移地址，它和 CS 一起确定下一条指令的物理地址。计算机就是用 IP 来控制程序的执行流程，改变了 IP 的值也就改变了程序执行的顺序。

PSW(Program Status Word)程序状态字寄存器(或称标志寄存器 FR)：是一个 16 位寄存器，但只用了 9 位来分别表示状态和控制标志。这些状态信息常常作为条件转移指令的转移控制条件，所以又可称为条件码。各位如图 1-4 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

图 1-4　标志寄存器

各位状态标志如下。

- X：表示该位无定义。
- CF(Carry Flag)进位标志：用来记录运算时最高位产生的进位值或借位值。运算结果最高位有进位或有借位，该位置 1；否则置 0。
- ZF(Zero Flag)零结果标志：运算结果为零，该位置 1；否则置 0。
- AF(Auxiliary carry Flag)：辅助进位标志。该位用来记录运算时第 3 位(或说成低 4 位)产生的进位值或借位值。运算结果第 3 位(或低 4 位)有进位或有借位，该位置 1；否则置 0。AF 主要用于十进制调整指令 DAA、DAS、AAA、AAS。
- SF(Sign Flag)符号标志：用来记录运算的符号位。运算结果为负，该位置 1；否则置 0。
- OF(Overflow Flag)溢出标志：运算结果超出计算机所能表示的范围，该位置 1；否则置 0。
- PF(Parity Flag)奇偶标志：运算结果低 8 位中 1 的个数为偶数，该位置 1；否则置 0。PF 主要在计算机串行传送时判断是否出错提供校验条件。

还有 3 个控制标志位。

- DF(Direction Flag)方向标志：用在串处理指令中控制 SI 和 DI 是增量还是减量。DF=0，每次操作后 SI 和 DI 自动增量；DF=1，每次操作后 SI 和 DI 自动减量。
- IF(Interrupt Flag)中断标志：CPU 总的中断允许控制位。IF=1 时，允许中断(开中断)；IF=0 时，禁止中断(关中断)。
- TF(Trap Flag)跟踪标志或单步运行标志：TF=1 时，程序将单步运行，每条指令执行完产生一个内部中断，允许程序员检查指令执行结果，TF=0 时，程序将连续执行。

其中状态标志是根据指令执行结果自动置 0 或置 1 的。而对控制标志位，程序员可以根据程序设计的需要置 0 或置 1。8086/8088 指令系统提供了设置部分状态标志和控制标志的指令。例如。

```
CLI    ; IF 置 0
STI    ; IF 清 0
```

3. 8086/8088 存储器结构

存储器是计算机的重要组成部分。当要访问某个存储单元时，首先必须获得该单元的物理地址。IBM PC 微机系统可以直接读写 1MB 的内存单元，地址从 00000H 到 FFFFFH，所以需要有 20 位寄存器才能遍寻 1MB 内存。然而 CPU 中与地址有关的寄存器，如 IP、

SP、BP、SI 和 DI 都是 16 位的，它们能访问的地址空间只能达到 64KB。为了解决这一矛盾，采取了分段技术。

(1) 分段的方法

段长不超过 64KB；段的起始地址低 4 位为 0000B(即 16 的倍数)。在 1MB 的存储器中，每一个存储单元都有惟一的 20 位的地址，称为存贮单元的物理地址。8086/8088 均把 1MB 的存储空间划分成若干个段，每段 64KB。段的起始单元地址叫做段基地址(简称段地址)，它为 16 的整数倍。段地址存储在相应的段寄存器中。每段存储空间为 64KB，则段内可用 16 位来表示其相对地址，这 16 位的数值称为偏移地址。编程时偏移地址可由 BX、BP、SI、DI 来存贮它称为逻辑地址 LA(Logical Address)。逻辑地址由段地址和偏移地址两部分组成，表示为段地址:偏移地址。

(2) 物理地址的计算

一个 20 位的物理地址是由段寄存器的内容和偏移地址形成的，其关系式为：

物理地址(PA)=段寄存器的内容*16D+偏移地址

(3) 段的种类

在 8086/8088 微处理器中，有 4 个段寄存器专门用来存放段地址，它们分别是代码段寄存器 CS、数据段寄存器 DS、附加数据段寄存器 ES、堆栈段寄存器 SS。每个段寄存器可以规定一个段的起始地址，每个段都有各自的用途。代码段用来存放正在执行的程序，数据段和附加数据段存放当前运行的程序所需数据。堆栈段则定义了堆栈所在的区域。一般来说，各段在存贮器中的分配是由操作系统负责的。

如果程序中 4 个段都在 64KB 范围内，则可在程序开始时，分别给各个段寄存器赋值。如果程序中的某一段在程序运行过程中超过 64KB，或者程序中要访问除本身 4 个段以外其他区段的信息，则要在程序中动态修改有关段寄存器的值，以保证获得信息的正确性。

8086/8088 这种存储器分段的方法，扩大了可访问的存储空间也方便了，程序在内存中的再定位。

(4) IBM PC 存储空间的布局

IBM PC 机对存储空间作了固定的安排：其中 RAM 为 64KB，可扩充到 640KB；显示缓冲区有 128KB；只读存储器 ROM 位于主存的尾部一般只有 40KB，但可使用 ROM 选件板扩充到 256KB。

4. 堆栈结构

堆栈是一个很重要的概念，也是编程的基本手法。这里主要介绍堆栈的概念和堆栈的操作。

(1) 堆栈概念

堆栈是按先进后出的原则在内存中组织的一个特殊的存储区域。该区域一端固定一端活动，固定的一端称为栈底，而活动的一端称为栈顶。往堆栈中存入或取出信息总是在堆栈的栈顶单元进行的。CPU 中的堆栈指针指示器 SP 总是指向堆栈的栈顶，而堆栈段寄存器 SS 则指明了堆栈段的起始位置。

堆栈的设置主要用来解决子程序嵌套，递归子程序及多级中断中一些比较难以处理的问题，还可用来保护现场，保存中间结果以及子程序和中断服务程序的调用和返回等(这些在以后章节将逐步学到)。

(2) 堆栈操作

它可分为两类操作，分别为进栈和出栈。进栈时是从高地址到低地址，出栈时是从低地址到高地址。进栈操作指令为 PUSH，出栈操作为 POP，后面章节会详细介绍。

1.2.3　INTEL 80386 32 位机的系统结构

Intel 公司于 1984 年底推出了与 8088/8086/80286 相兼容的高性能的 32 位微处理器—Intel 80386。它是为满足高性能的应用领域与多用户、多任务的操作系统的需要而设计的，它最大的特点是在 CPU 芯片上集成了一个存储器管理部件(MMU)，可对 2^{46} 字节的虚拟存储器和 4 千兆字节的物理存储器进行分段和分页管理，段的最大长度为 4 千兆字节。

80386 采用 CHMOS 工艺，陶瓷网格阵列(Plastic Grid　Array，PGA)封装，全 32 位结构，它的时钟频率有 16MHZ 和 20MHZ 两种，总线周期只有 2 个时钟周期。

1. 80386 的体系结构

80386 的内部结构如图 1-5 所示，由 6 个功能部件组成：总线接口部件 BIU(Bus Interface)、指令预取部件 IPU(Instruction Prefetch Unit)、指令译码部件 IDU(Instruction Decode Unit)、存储器管理部件 MMU(Memory Management Unit)、执行部件 EU(Execution Unit)和控制部件(Control Unit)。这 6 个部件可以并行工作，构成一个 6 级流水线体系结构。这样，可以同时处理多条指令以减少程序实际执行时间。

总线接口部件 BIU 在总线周期内对必要的信号进行控制。在 80386 内部，指令预取部件从存储器取指令，执行部件在执行指令时访问存储器以读/写数据，分页部件将线形地址转换成物理地址后，都会发出总线周期请求。总线接口部件会根据优先级对这些请求进行仲裁，从而有条不紊地服务于多个请求，并产生相应总线操作所需信号。这些信号包括地址信号。数据信号、读/写控制信号等。此外，总线接口部件也实现 80386 和协处理器之间的协调控制。

指令预取部件 IPU 从存储器中以 4 个字节为单位预取指令，按顺序存放到 16 个字节的预取指令队列中，以便在 CPU 执行当前指令时，指令译码部件对下一条指令进行译码。如果指令队列向指令译码部件输送一条指令，指令队列会有部分空字节，指令预取部件就会向总线接口部件发出总线请求，如总线接口部件此时处于空闲状态，则会响应此请求，从存储器取指令填充预取指令队列。

指令译码部件 IDU 从指令预取部件中的预取指令队列里按顺序取出指令并译码。只要译码指令队列有剩余空间，译码部件就会从预取指令队列取下一条指令进行译码。

在 INTEL 80386 中为了加快 CPU 的速度，采用指令重叠执行技术。具体地说，就是将一条访问存储器的指令和前一条指令的执行重叠起来，使两条指令并行执行。指令预取

队列和译码指令队列是这种功能实现的前提。

执行部件包括 8 个 32 位的通用寄存器组、1 个 32 位的算术逻辑运算单元 ALU、1 个 64 位的桶形移位寄存器和一个乘法除法器，它们共同执行各种数据处理和运算。此外，执行部件中还包括 ALU 控制部件和保护测试部件，前者实现有效地址的计算、乘除法的加速等功能，后者检验指令执行中是否符合存储器分段规则。

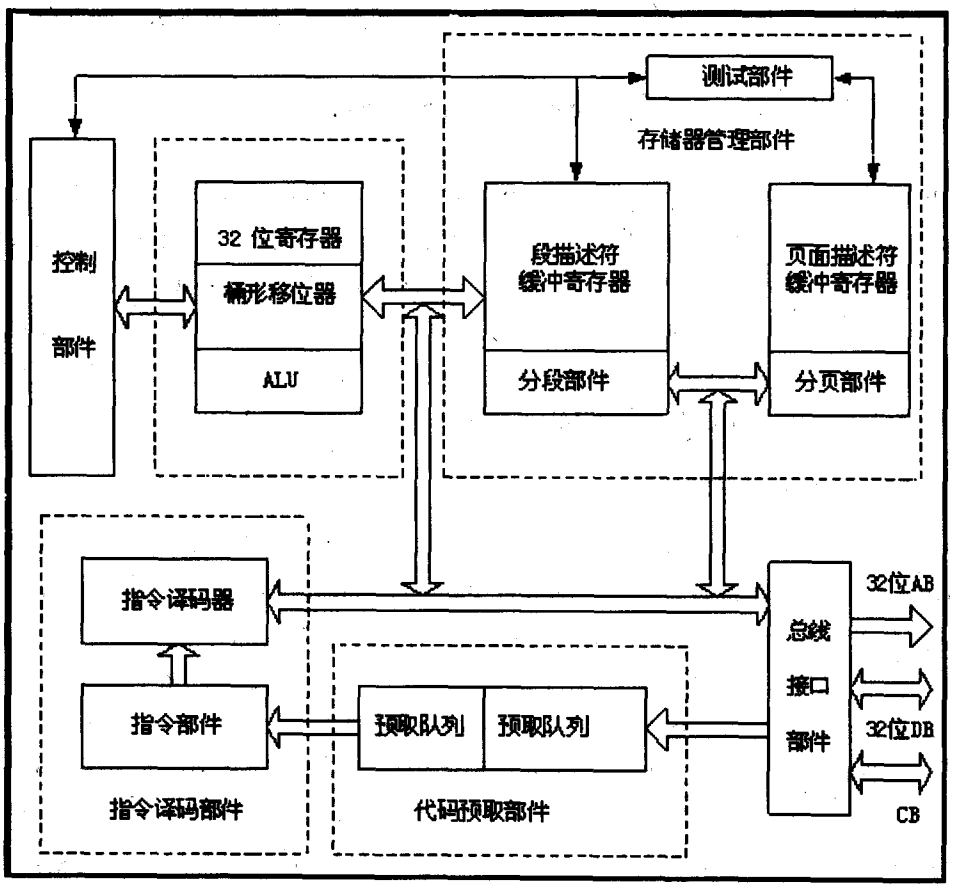


图 1-5 80386 的内部结构

80386 允许使用虚拟存储器。所谓虚拟存储器就是系统中有一个速度较快的，容量比较小的内部主存储器，还有一个速度较慢但容量较大的外部存储器，通过存储管理机制，使二者有机地、灵活地组合在一起。这样，从程序员的角度看，系统中似乎有一个容量很大，速度也相当快的主存储器 MM(Main Memory)。但它并不是真正的物理上的主存，所以称为虚拟存储器 VM(Virtual Memory)。80386 的虚拟存储器容量可高达 64TB(2^{46} 字节)。它就可以运行要求存储器容量比实际主存储器大得多的程序 。

在 INTEI 80386 系统中，存储体按段划分，每个段的容量可变，最大可达 4GB(2^{32} 字节)。分段的作用是可以对容量可变的代码存储块或数据存储块提供模块性和保护性。80386 在运行时，可以同时执行多个任务，即进行多任务操作。对每个任务来讲，可以拥有多达 16384 个段，即 64TB($2^{32}*2^{14}$)。每个段又划分为多个页面，一个页面可为 1 到 4K 字节。但