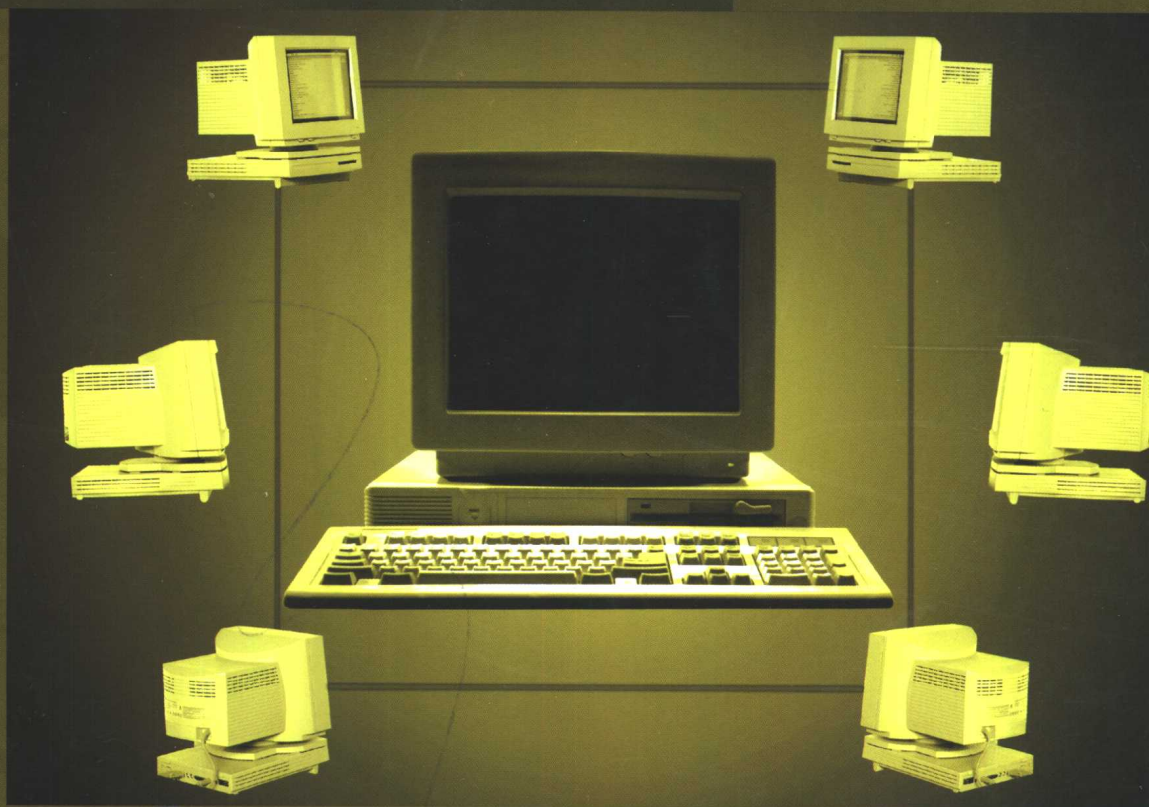


21世纪高等学校教材

汇编语言与接口技术

HUI BIAN YU YAN YU JIE KOU JI SHU

冯萍 史新福 编



 机械工业出版社
CHINA MACHINE PRESS



21 世纪高等学校教材

汇编语言与接口技术

冯 萍 史新福 编

机械工业出版社

本书以 Intel 80X86 为背景机介绍汇编语言与接口技术的基础知识、原理和使用方法。全书分为两部分:第一部分是汇编语言部分,以 MASM6.11 的 Programmer's Work Bench 为平台,介绍 80X86 指令系统及汇编语言程序设计技术基础,并通过典型应用帮助读者深入学习和掌握汇编语言程序设计的方法。第二部分是接口技术部分,首先引入微机基本接口技术,系统和详细地介绍了中断、串行和并行通信、时钟以及总线等技术的基本原理和应用方法,进一步讲述了 Pentium PC 发展的部分接口技术。

本书可以作为本科计算机专业、自动控制类专业“汇编语言与接口技术”课程的教材,亦可供从事系统开发的工程技术人员学习使用。

图书在版编目(CIP)数据

汇编语言与接口技术/冯萍,史新福编. —北京:机械工业出版社,2002.8
21 世纪高等学校教材
ISBN 7-111-10685-7

I. 汇... II. ①冯... ②史... III. ①汇编语言-程序设计-高等学校-教材②电子计算机-接口-高等学校-教材 IV. TP3

中国版本图书馆 CIP 数据核字(2002)第 053756 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑:王小东 版式设计:霍永明 责任校对:张媛

封面设计:张静 责任印制:路琳

北京机工印刷厂印刷·新华书店北京发行所发行

2002 年 9 月第 1 版·第 1 次印刷

787mm × 1092mm ¹/₁₆ · 30.25 印张 · 752 千字

0 001—4 000 册

定价:42.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换
本社购书热线电话(010)68993821、68326677-2527
封面防伪标均为盗版

前 言

目前,国内最广泛使用的微机系统采用 Intel 的 80X86 系列微处理器或者兼容的微处理器。在 Intel 的 80X86 家族中,16 位的 8086/8088 微处理器,实现了以分段方式管理存储器;32 位的 80X86 高档微处理器,实现了支持多任务的保护工作方式;基于 MMX 技术的 Pentium 是新一代的微处理器,实现了对多媒体处理的支持。因此,本书以 80X86 作为教材背景机。

汇编语言面向机器,能够为程序员提供最直接操纵机器硬件系统的途径,利用它可以编写出在“时间”和“空间”两个方面最具效率的程序。随着微型计算机应用的日益广泛,微机接口技术的重要性也日益明显地表现出来。“汇编语言与接口技术”是计算机各专业的一门重要技术基础课程,是必修的核心课程之一,该课程对于训练学生掌握汇编语言程序编程和进行微机接口设计都有重要作用。此外,“汇编语言与接口技术”也是其他相关专业的必修或选修课。

为了适应教学需要,我们编写了《汇编语言与接口技术》教材。在汇编语言程序设计部分,讲解了汇编语言程序设计的一般概念、基本技术和常用技巧,模块化程序设计及保护方式程序设计的方法,应用编程技术及实现细节。在微机接口技术部分介绍了接口技术的基本概念、总线连接技术、中断技术、并行通信和串行通信接口技术等,首先讲述基本原理,然后介绍实现这些原理的典型芯片,并且给出这些原理或芯片的应用实例,力求理论联系实际,做到原理、技术和应用并重,硬件和软件结合。

全书共分 8 章。第 1 章绪论,简要介绍有关汇编语言与微机接口技术的特点及应用、80X86 微型计算机的特点和基本结构。第 2 章 80X86 寻址方式和指令系统,对 32 位机在 16 位模式或 32 位模式中工作的寻址方式及操作都做了较详细的阐述,同时列举了一些程序,帮助读者深入理解其指令的功能。第 3 章汇编语言及程序设计,论述了汇编语言源程序的设计方法,常用的伪指令使用技巧,宏和模块化程序设计的方法以及汇编语言程序和高级语言程序的相互调用。第 4 章应用程序设计,介绍了 DOS 应用程序设计方法,实模式和保护模式切换的接口技术,以及 Windows 环境编程方法。第 5 章微机接口技术,讲述了 I/O 接口技术,DMA 技术,传统中断及高级中断控制技术等。第 6 章总线技术,主要描述总线的有关概念及功能,总线的握手技术和判决技术,常用总线的有关规范等。第 7 章可编程接口芯片,介绍了常用并行和串行接口芯片的结构及其与 MPU 接口方式和编程。为了与当前各大、专院校普遍使用的 16 位微型计算机实验设备相衔接,主要以 8 位接口芯片为例。第 8 章数/模与模/数转换,主要讲述了数-模转换器和模-数转换器的一般工作原理,重点介绍与 MPU 的接口技术及其编程。

本书每章都有习题与思考题,以便帮助读者理解和掌握有关内容。

本书第 1 章~第 5 章、第 7 章由冯萍编写,第 6 章和第 8 章由史新福编写,全书由冯萍统稿。西安交通大学冯博琴教授、吴宁高工审阅了全书,金翊副教授、吴晓副教授对本书提出了很多有益的意见,编者在此表示感谢。

由于水平有限,书中不妥和谬误之处在所难免,恳请读者批评指正。

编 者
2002 年 3 月

目 录

前言	
第 1 章 绪论	1
1.1 概述	1
1.2 微处理器基本结构	3
思考题与习题	16
第 2 章 80X86 寻址方式和指令系统	17
2.1 80X86 寻址方式	17
2.2 80X86 指令格式	26
2.3 80X86 指令系统	27
思考题与习题	57
第 3 章 汇编语言及程序设计	59
3.1 汇编语言	59
3.2 分支程序设计	69
3.3 循环程序设计	74
3.4 子程序设计	80
3.5 控制汇编语言程序语句	94
3.6 结构和记录	96
3.7 宏	101
3.8 源程序的结合	105
3.9 PUBLIC 和 EXTRN 伪指令	107
3.10 库文件	109
3.11 汇编语言与高级语言的接口	110
3.12 条件汇编和条件错误汇编	114
3.13 汇编和连接程序	115
3.14 汇编语言程序上机过程	116
思考题与习题	117
第 4 章 应用程序设计	120
4.1 保护方式编程	120
4.2 中断程序和中断拦截	134
4.3 EXEC 功能和程序段前缀	143
4.4 Windows 环境下汇编语言程序设计	150
4.5 设备驱动程序	168
4.6 图形显示	178
4.7 鼠标器	187
思考题与习题	189
第 5 章 微机接口技术	192
5.1 I/O 控制	192
5.2 中断控制	194
5.3 8259A 可编程中断控制器	201
5.4 APIC 技术	217
5.5 直接存储器存取控制	234
思考题与习题	251
第 6 章 总线技术	253
6.1 概述	253
6.2 总线数据传输	255
6.3 局部总线	269
6.4 系统总线	315
6.5 通信总线	321
思考题与习题	336
第 7 章 可编程接口芯片	338
7.1 可编程并行输入输出接口 8255A	338
7.2 可编程定时器/计数器 8254	352
7.3 可编程串行通信接口 16C550C	370
7.4 可编程网络接口控制器 DP83905	397
思考题与习题	418
第 8 章 数/模转换与模/数转换	420
8.1 信号转换技术	421
8.2 数/模转换原理	424
8.3 数/模转换芯片及接口技术	429
8.4 模/数转换原理	440
8.5 模/数转换芯片及接口技术	448
8.6 D/A 和 A/D 器件的选择	465
思考题与习题	466
附表	468
附表 A ASCII 码表	468
附表 B INT 2FH DPMI 功能调用	468
附表 C INT 31H DPMI 功能调用	469
参考文献	480

第 1 章 绪 论

1.1 概述

本节介绍汇编语言与接口技术的特点和使用场合。

1.1.1 汇编语言的特点和使用场合

汇编格式指令是机器指令符号化的指令。机器语言是用二进制编码的机器指令的集合及一组使用机器指令的规则。它是 MPU 能直接识别的唯一语言。只有用机器语言描述的程序，MPU 才能直接执行。用机器语言描述的程序称为目的程序或目标程序。机器指令在形式上表现为二进制编码。机器指令一般由操作码和操作数两部分构成，操作码在前，操作数在后。操作码指出要进行的操作或运算，如加、减、传送等；操作数指出参与操作或运算的对象，也指出操作或运算结果存放的位置，如 MPU 的寄存器、存储单元和数据等。机器指令与 MPU 有着密切的关系。通常，MPU 的种类不同，对应的机器指令也就不同。同一个系列 MPU 的指令集常常具有良好的向下兼容性，例如，Pentium 指令集包含了 8086 指令集。

机器语言用于描述数据在机器中的流动，要求使用者熟悉机器内部结构，由于采用人们所不熟悉的形式来描述计算机要执行的任务，所以用机器语言编写程序十分繁杂，而且需采用编码的方式表示机器要执行的任务，因此用机器语言编制出的程序不易为人们理解、记忆和交流，而且极易出错，一旦有错，难于检查。为了克服机器语言的上述缺点，人们采用便于记忆、并能描述指令功能的符号来表示指令的操作码。这些符号被称为指令助记符。指令中一般包括指令功能和操作数，说明指令功能的助记符采用英文缩写及指令执行的操作数用符号地址来表示，例如 MPU 的寄存器、存储单元地址等。汇编语言是指令和伪指令的集合。伪指令主要用于解释和说明指令中操作数的存放形式、指令和数据的分段和指令段之间的关系等。用汇编语言书写的程序称为汇编语言程序，或称为汇编语言源程序。用汇编语言编写的程序要比用机器语言编写的程序容易理解、调试和维护。

由于 MPU 能直接识别的唯一语言是机器语言，所以用汇编语言编写的源程序必须被翻译成用机器语言表示的目标程序后才能由 MPU 执行。把汇编语言源程序翻译成目标程序的过程称为汇编。完成汇编任务的程序叫做汇编程序。由于汇编语言使用指令助记符和符号地址，所以它要比机器语言容易掌握。与高级语言相比较，汇编语言有如下特征：

- 1) 汇编语言与机器关系密切。因为汇编格式指令是机器指令的符号表示，所以汇编语言与微处理器有着十分密切的关系。对于各种不同类型的微处理器，有各种不同的汇编程序。由于汇编语言与机器关系十分密切，所以汇编语言源程序与高级语言源程序相比，它的通用性和可移植性要差得多。但通过汇编语言可最直接和最有效地控制机器，而这是大多数高级语言难以做到的。

- 2) 汇编语言程序效率高。用汇编语言编写的源程序在汇编后所得的目标程序效率高。这种目标程序的高效率反映在时间和空间两个方面：其一是运行速度快；其二是目标程序短。在采用相同算法的前提下，任何高级语言程序在这两方面的效率都不如汇编语言程序。

汇编语言程序能获得“时空”高效率的主要原因是：构成汇编语言主体的指令是机器指令的符号表示，每一条汇编格式指令都有其对应的某条机器指令；另一个重要原因是汇编语言程序能直接并充分利用机器硬件系统的许多特性。高级语言程序在上述两点上要逊色得多。

3) 编写汇编语言源程序繁琐。编写汇编语言源程序要比编写高级语言源程序繁琐得多。汇编语言是面向机器的语言，它要求程序员比较熟悉机器硬件系统，要考虑许多细节问题，最终导致程序员编写程序繁琐。对于汇编语言来说，作为机器指令符号化的每条指令所能完成的操作是有限的。例如，80X86 指令集中没有同时完成多个算术运算的指令。程序员在利用汇编语言编写程序时，必须考虑如何使用寄存器、存储单元，采用何种寻址方式，还有指令执行的细节问题。例如，指令执行后对标志的影响，堆栈设置的位置等。而高级语言是面向过程或面向目标、对象的语言。在使用高级语言编写程序时，程序员不会遇到这些琐碎却重要的问题。

4) 汇编语言程序调试困难，维护、交流和移植程序更困难。调试汇编语言程序往往要比调试高级语言程序困难。汇编格式指令的功能有限，程序员在考虑功能逻辑实现的同时，还要考虑数据在机器中流动的物理路径，需要注意太多的细节问题是造成这种困难的客观原因；汇编语言提供给程序员一个充分发挥的“舞台”，程序员为了追求“时空”上的高效率往往编制出的程序结构不清晰，这是造成调试困难的主观原因。

汇编语言面向机器的特点，从编程和调试的角度看是缺点，但从机器效率来看，它又具有其他高级语言所不具有的优点。利用汇编语言可以编写出在“时空”两个方面最有效率的程序。另外，通过它可最直接和最有效地操纵机器硬件系统。

汇编语言的使用场合，要充分考虑软件的开发时间和软件的质量等诸多方面的因素。在下列应用场合，可考虑使用汇编语言编写程序。

1) 对软件的执行时间或存储容量有较高要求的场合，例如：系统程序的关键核心，智能化仪器仪表的控制系统，实时控制系统等。

2) 需要提高大型软件性能的场合。通常把大型软件中执行频率高的子程序（过程）用汇编语言编写，然后把它们与其他程序一起连接。

3) 软件与硬件关系密切，软件要有直接和有效控制硬件的场合，如设备驱动程序等。

4) 没有合适的高级语言的场合。

编写汇编语言程序，要求适度地追求“时空”效率。在编写汇编语言程序时，要尽量利用最恰当的指令，以便节约一个字节或节省几个机器周期。但时至今日，计算机硬件系统的整体性能已极大地提高，所以，除非不得已，不要为节约少量字节或机器周期而影响程序的结构性、健壮性和可读性等。要在确保汇编语言程序上述性能良好的前提下追求“时空”性能。

1.1.2 接口技术的特点

自从 20 世纪 70 年代初第一个微处理器 Intel 4004 问世以来，微型计算机的发展迅速。在短短 20 多年内，微处理器由 4 位机、8 位机、16 位机、32 位机发展到 64 位机。微处理器和微型计算机的性能提高了成百上千倍，在很多应用领域微型计算机的使用占了主导地位。作为一个微型计算机系统，除了有微处理器外，还包括一定的输入输出外围设备。例如，输入设备有键盘、鼠标、扫描仪、光笔等；输出设备有 CRT 显示终端、打印机、绘图仪等。

这些外围设备（简称外设）都需要连接到微处理器。在数据采集、参数检测和实时控制等领域，微处理器往往需要采集测控对象的状态和变化的信息，经过微处理器处理后，再向控制对象输出控制信息。这些输入信息和输出信息的表现形式具有多样性，除了开关量、数字量外，更多是各种模拟量。例如，温度、压力、流量、长度、浓度等等，各种传感器和执行机构都需要连接到微处理器。由于设备的多样性决定了连接方式的多样性，而微机采用处理机总线形式，不提供和任意设备连接时的特殊控制，因此微处理器必须通过一定的“接口”才能对外围设备进行检测、控制及交换信息。

微机接口是指微型计算机与外围设备之间的必经之地，是把微机与外部各种控制对象联系起来的桥梁。所以，将微机组成实际应用系统的一个关键技术是接口技术。在任何一个微机应用系统的研制和设计中，硬件方面设计主要是微机接口电路的研制和设计，软件方面设计主要是控制这些电路的驱动程序。由于微机应用的范围涉及科学计算、实时控制、数字通信、数据采集、仪器仪表、图像处理、语音合成等各个领域，在应用中生产厂家提供的产品，很难满足各种应用技术的要求，因此，把微机应用于不同领域、不同对象时，几乎都要对接口硬件和软件进行专门设计或程度不同的二次开发。正是微机应用的这一特点，推动了微机接口技术的迅速发展，使之成为微型计算机应用技术的一个重要分支。

从应用技术的角度看，微机接口技术有两个特征：

1) 微机接口技术具有综合性的特征，为了开发应用系统所需要的特定接口功能，往往采用对具有接口功能的通用产品进行二次开发。而功能较强的接口产品，一般具备可编程性，所以，接口设计涉及软件硬件综合技术。

2) 微机接口技术具有复杂性的特征。接口技术的基本任务是要把微处理机和外围设备连接起来，使两者之间正确地交换和传递信息，这就决定了接口技术总体上的复杂性。一方面必须对微型计算机的硬件和软件，包括它的总线标准、组织结构及其指令系统和汇编语言编程等知识有较深入的了解，另一方面，还必须对被接口的外围设备的原理和特性，特别是电气工作特性、信息类型和格式、控制信号及其相互之间的定时关系等有较准确的认识。

综上所述，微机接口技术是计算机应用领域中一种软硬结合、以硬为主，机电结合、以电为主的综合技术，是多种技术交叉的学科，具有较强的理论性和工程实践性。

1.2 微处理器基本结构

1.2.1 80X86 内部结构

由于指令直接涉及到微处理器内部部件的使用，在学习指令之前，必须了解微处理器的内部结构。80X86 微处理器内部包括 8 位、16 位、32 位的通用寄存器和一些专用寄存器。其中 8086、8088、80286 为 16 位结构，80386、80486 和 Pentium 为 32 位结构。80X86 可编程寄存器结构如图 1.1 所示，图中阴影部分表示不属于 8086、8088 和 80286 的寄存器。

1. 通用寄存器

通用寄存器名及功能见表 1.1，当使用 32 位寄存器时，寄存器为 EAX、EBX、ECX、EDX、ESP、EBP、EDI、ESI；当使用 16 位寄存器时，寄存器为 AX、BX、CX、DX、SP、BP、DI、SI；当使用 8 位寄存器时，寄存器为 AH、AL、BH、BL、CH、CL、DH、DL。

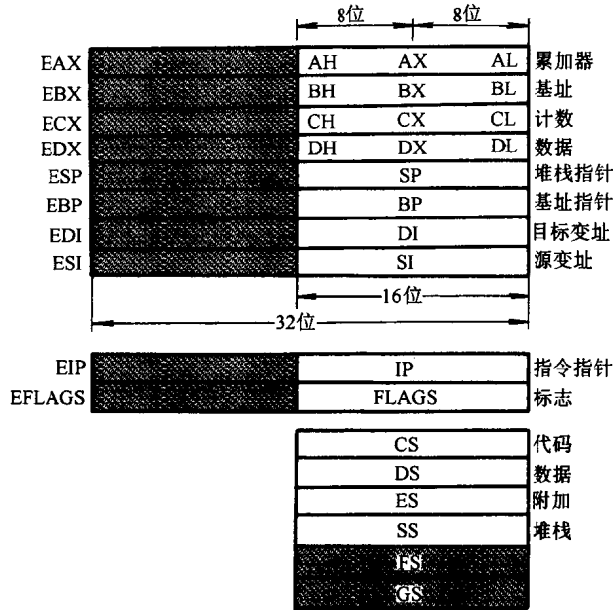


图 1.1 微处理器内部寄存器

表 1.1 通用寄存器表

寄存器定义	32 位寄存器	16 位寄存器	8 位寄存器	功 能
累加器	EAX	AX	AH AL	保存数据和作为某些指令的专用寄存器
基址寄存器	EBX	BX	BH BL	保存数据, 16/32 位可保存地址
计数器	ECX	CX	CH CL	保存数据和用于指令计数
数据寄存器	EDX	DX	DH DL	保存数据, 乘/除法专用寄存器
基址指针	EBP	BP	—	保存地址, 16 位默认保存堆栈地址
堆栈指针	ESP	SP	—	保存堆栈栈顶地址
源变址寄存器	ESI	SI	—	保存数据, 串操作时保存源数据地址
目标变址寄存器	EDI	DI	—	保存数据, 串操作时保存目标数据地址

注: 32 位模式下, 32 位寄存器均可以作为地址寄存器使用。

2. 段寄存器

在 8086 微处理器中, 有 4 个 16 位的段寄存器, 把 1M 字节内存空间分成若干独立的逻辑地址空间, 每个程序同时可以有 4 个逻辑地址空间 (四个不同类型的段) 来使用。在 80386 以上微处理器中, 均有 6 个 16 位的段寄存器, 用于实现存储空间的分段, 即把系统 64T 字节的内存空间, 分成各自独立的逻辑地址空间, 每个程序均同时可以用 6 个逻辑地址空间 (六个不同类型的段)。在实模式下, 段寄存器直接存放某个段的段基地址, 段长在 1~64K 字节内。在保护模式下, 系统采用描述符表来描述不同段的起始地址和长度, 段长在 1~4G 字节内, 而段寄存器存放选择符, 该选择符可以在描述符表中选择一个描述段起始地址和长度的描述符。段以及段寄存器定义见表 1.2。

3. 专用寄存器

(1) 指令指针 EIP/IP 存放存储器中代码段的下一条指令的地址。在实模式下使用 IP, 在保护模式下使用 EIP, 指令指针可以用跳转指令或调用指令修改。

表 1.2 段寄存器

段名	功能	段寄存器
代码段	用于存放程序地址	CS
数据段	用于存放程序数据	DS
堆栈段	用于存放程序执行的中间结果	SS
附加段	用于存放数据	ES
附加段	用于存放数据	FS
附加段	用于存放数据	GS

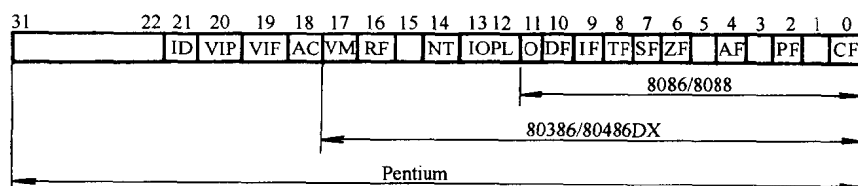


图 1.2 标志寄存器

(2) 标志寄存器 EFLAGS/FLAGS 指示微处理器的控制状态和运算状态。其格式如图 1.2 所示。

- CF: 进位标志, 该位保存加法运算后的进位和减法运算后的借位。例如执行加法运算后, 最高有效位有进位部分时, CF 置 1, 否则置 0。

- PF: 奇偶标志, 用于数据传送检查传送错误。当该位置 1 时, 表示 AL 中数据位有偶数个 1; 当该位置 0 时, 表示 AL 中数据位有奇数个 1。

- AF: 辅助进位标志, 该位保存加法或减法运算后, 最低有效 4 位向高位产生的进位或借位。

- ZF: 零标志。当该位置 1 时, 表示算术或逻辑运算操作的结果为零; 当该位置 0 时, 表示运算结果不为 0。

- SF: 符号标志, 保存算术或逻辑运算后的符号位 (最左边一位)。当符号位为 1 时, SF 为 1; 当符号位为 0 时, SF 为 0。

- OF: 溢出标志。在加减法运算中, 当运算结果超出规定的机器数 (例如 8 位、16 位或 32 位数据) 可以表示的范围时, OF 置 1, 否则为 0。

(3) 控制状态标志

- IF: 中断标志。当该位置 1 时, 允许响应外部可屏蔽中断 INTR; 当该位置 0 时, 禁止响应外部可屏蔽中断。IF 不影响外部非屏蔽中断或内部产生的中断。

- TF: 陷阱标志。当该位置 1 时, 将处理器设置成供调试的单步方式, 在这种方式下, 每条指令执行后 MPU 自动产生一个内部中断, 代码调试程序可以利用陷阱特性和调试寄存器调试软件。

- DF: 方向标志, 用于控制串操作指令中, 地址递增/递减方式。当该位置 0 时, SI/ESI、DI/EDI 中的地址自动增量; 当该位置 1 时, SI/ESI、DI/EDI 中的地址自动减量。

- IOPL: 输入/输出特权标志, 用 2 位表示 0~3 级, 0 级最高, 用于定义允许执行 I/O 指令的特权级, 如果当前任务的特权级高于或等于 IOPL 中规定的级别, 则可以执行该 I/O 指令, 否则发生保护异常中断, 执行程序被挂起。

- NT: 嵌套任务标志, 表示当前的任务是否嵌套在另一个任务内, 该标志控制中断返回指令的执行。当该位置 1 时, 表示当前的任务有一个有效的链连接到前一个任务 (被嵌

套), 则通过任务转换, 实现中断返回; 当该位置 0 时, 执行常规中断返回过程。

- RF: 重启动标志, 用于控制是否接受调试故障。标志与调试寄存器的断点一起使用。当该位置 0 时, 接收调试故障; 当该位置 1 时, 忽略调试故障, 即使遇到断点或调试故障, 也不产生异常中断。在成功执行每条指令后, RF 将自动复位, 而当接收一个非故障调试时, 处理器把 RF 置 1。

- VM: 虚拟 8086 方式标志。当该位置 1 时, MPU 由保护模式切换到虚拟 8086 方式; 当该位置 0 时, MPU 工作在保护模式下。

- AC: 对准检查标志。当该位置 1 时, 如果进行非对准字或者双字地址访问, 则产生异常中断。非对准字地址访问是指访问字数据地址为奇地址, 或者访问双字数据的地址不为 4 的倍数地址, 它主要用于和协处理器同步工作。

- VIF: 虚拟中断标志。它为虚拟方式下, 中断标志位的拷贝, 用于多任务环境下, 给操作系统提供中断的信息。

- VIP: 虚拟中断暂挂标志, 用于多任务环境下, 给操作系统提供中断暂挂信息。

- ID: 标识标志, 指示 Pentium 微处理器支持 CPUID 指令。该 CPUID 指令给系统提供有关该微处理器的各种信息, 例如版本号和制造商等。

1.2.2 实模式存储器寻址

8086 工作实模式下, 80286 以上微处理器可工作于实模式或者保护模式下, 实模式下一个段的长度为 1~64K 字节内, 段寄存器直接存放某一段的段基地址 (20 位段起始地址的高 16 位, 低 4 位默认为 0000B), 编制程序时, 程序给出指令或者数据的逻辑地址, 包括段基地址 (16 位) 和相对于段基地址的段内偏移量 (16 位), 又称偏移地址。在存储器寻址时, 将逻辑地址转换为存储器存储单元的物理地址 (实际地址或者绝对地址), 1M 存储空间物理地址为 20 位。地址转换规则为:

物理地址 = 段基地址 × 10H + 偏移地址

例如, 某个程序定义 CS = 1000H, DS = 2000H, SS = 3000H, ES = 4000H, 4 个段长度均为 64K 字节, 则偏移地址为 0000H ~ FFFFH, 4 个段在内存的分布如图 1.3 所示。

在寻址代码段时, 段基地址在 CS 中, 段内偏移地址在 IP/EIP 中, 在寻址其他段时, 段基地址在 DS、ES、SS、FS、GS 中, 段内偏移地址由指令指定寻址方式, 或者由指令直接定义, 或存放在通用寄存器中, 或者存放在存储器单元中。

这种分段寻址方式, 允许程序在存储空间内重定位, 允许在实模式下编写的程序在保护模式下运行, 因为这种寻址方式允许在不改变任何偏移地址的情况下, 只修改段寄存器的内容, 就可以将整段程序移到内存空间任何地方, 而不用修改任何程序和数据, 实现了程序重定位。

在实模式下, 8086~80286 微处理器允许四种存储器分段, 段寄存器为 CS、DS、SS、ES。80386 以上微处理器允许六种存储器分段, 段寄存器为 CS、DS、SS、ES、FS、GS。当段的长度小于 64K 字节时, 重定位方式允许段重叠。

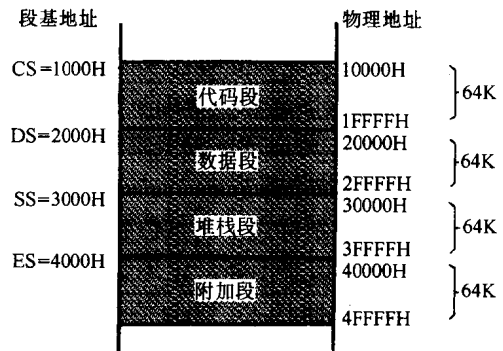


图 1.3 实模式存储器寻址示意图

某个段寄存器时，微处理器自动从描述符表中取出相应的描述符，把描述符中的信息保存到对应的描述符高速缓冲寄存器中，此后在对该段访问时，微处理器直接使用描述符高速缓冲寄存器中的描述符信息，而不用从描述符表中取出描述符（注：系统地址寄存器中 GDTR、IDTR 直接存放段基址和界限）。系统段寄存器和段描述符高速缓冲寄存器结构如图 1.6 所示。其中只有段寄存器是编程可见的，段描述符高速缓冲寄存器是编程不可见的。每个描述符由 8 个字节组成，描述符的基地址指针指示存储器段的起始单元，允许段在其 4G 字节存储器的任何单元开始，段界限包含该存储段最大偏移地址。例如，某存储器起始单元地址为 10000000H，存储单元末地址为 1000FFFFH，则物理基地址为 10000000H，段界限为 FFFFH。描述符格式如图 1.7 所示。

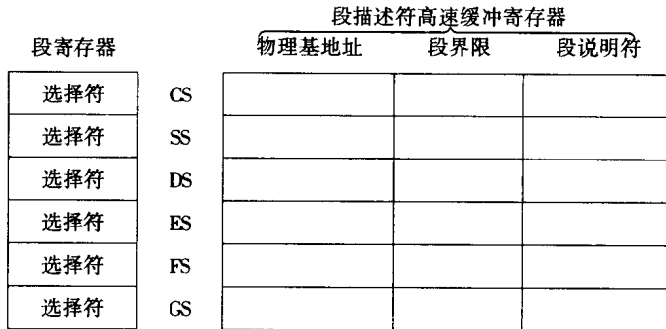


图 1.6 系统段寄存器和段描述符高速缓冲寄存器结构

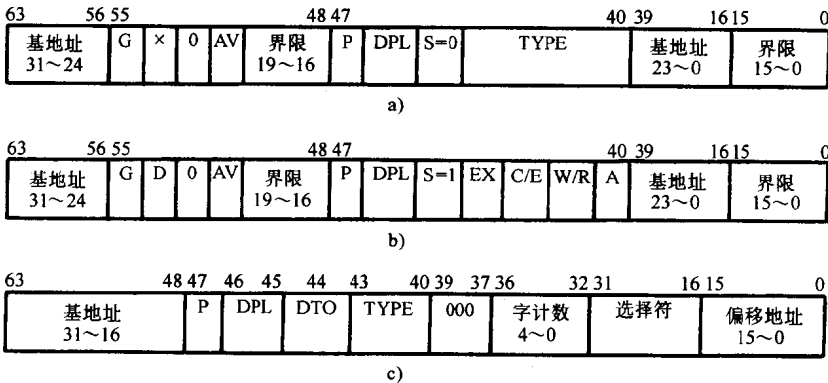


图 1.7 80386/80486/PENP IUM 描述符格式

a) 系统段描述符 b) 程序段描述符 c) 门描述符

- AV 为段有效指示位。置 1，指示段有效；置 0，指示段无效。
 - G 为段界限长度单位标志。置 1，指示界限长度以 4K 字节为单位，段长从 4K 字节开始，最大到 4G 字节；置 0，指示界限长度以字节为单位，段长从 1 字节开始，最大到 1M 字节。
 - D 为指令模式标志。置 1，为 32 位指令模式，使用 32 位偏移地址和缺省的 32 位寄存器；置 0，为 16 位指令模式，使用 16 位偏移地址和默认的 16 位寄存器。
- 访问权限包括五个标志位：
- P 为选择符有效指示位。置 1，表示选择符定义的段在存储器中存在，描述符包含有效基地址和界限值；置 0，表示选择符定义的描述符没有意义。

- DPL 设定描述符特权级 0~3, 0 级最高, 3 级最低。一个任务可以访问相同或者低特权级的数据段; 可以调用同特权级的程序, 可以经过门间接访问更高特权级的段, 但不可以执行低特权级的程序段。
- S 定义描述符。置 1, 表示代码或数据段描述符; 置 0, 表示系统描述符。
- E、ED/C、R/W、A 位功能如图 1.8 所示。

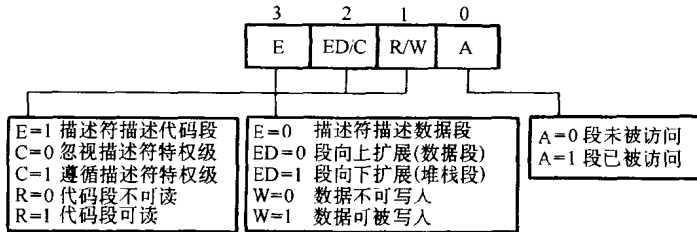


图 1.8 E、ED/C、R/W 和 A 位说明

- TYPE 字段功能如图 1.9 所示。

序号	功能	序号	功能
0	保留	8	保留
1	可用 80286TSS	9	可用 80386 以上 TSS
2	LDT	10	保留
3	忙 80286TSS	11	忙 80386 以上 TSS
4	80286 调用门	12	80386 以上调用门
5	任务门 (286/386)	13	保留
6	80286 中断门	14	80386 以上中断门
7	80286 陷阱门	15	80386 以上陷阱门

图 1.9 TYPE 字段功能

3. 描述符表

由描述符构成描述符表, 系统有 3 种描述符表, 它们是全球描述符表、局部描述符表和中断描述符表。描述符表界限值采用 16 位二进制表示, 表长可以达到 64K 字节, 每个描述符占 8 个字节, 所以全局描述符表和局部描述符表最多可以存放 8K 个描述符。由于 80X86 只能够支持 256 个中断和异常, 所以中断描述符表最多存放 256 个中断描述符。

描述符表存放在内存中, 由 MPU 中相应的系统地址寄存器寻址。系统地址寄存器包括全局描述符表寄存器、局部描述符表寄存器、中断描述符表寄存器和任务状态段寄存器。

全局描述符表 GDT 用来定义全局存储器地址空间, 全局存储器是一种可能被许多或者所有软件任务共享的通用系统资源。整个系统只设一个 GDT。GDT 中包含每个任务都可能或者可以访问的段描述符, 全局描述符表在存储空间的位置由全局描述符表寄存器说明, GDTR 中的高四个字节指示 GDT 在物理存储器中的起始地址。全局描述符表寻址结构如图 1.10 所示。

局部描述符表 LDT 用于描述局部存储器地址空间, 包括与给定任务相关联的描述符, 每个任务都有自己独立的局部描述符表。LDT 中包括该任务自己的代码段、数据段和堆栈段的描述符, 也包括该任务所使用的一些门描述符, 例如任务门和调用门描述符等。随着任务

切换，系统当前的局部描述符表 LDT 也随之切换。局部描述符表在存储空间的位置由局部描述符表寄存器间接说明。局部描述符表寄存器中存放的是选择符，根据选择符从全局描述符表中取出某任务局部描述符表 (LDT_i) 的描述符，存入局部描述符高速缓冲寄存器中。该 LDT 描述符用于描述 LDT 所在存储空间的基地址和界限值，根据该描述符表 (LDT_i) 的描述符可以选择局部描述符表中的某个局部描述符。局部描述符表寻址结构如图 1.11 所示。

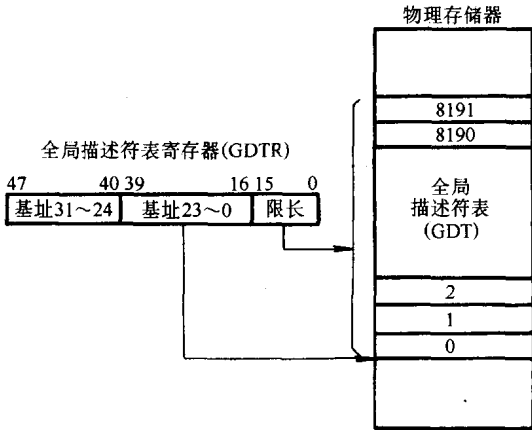


图 1.10 全局描述符表寻址结构

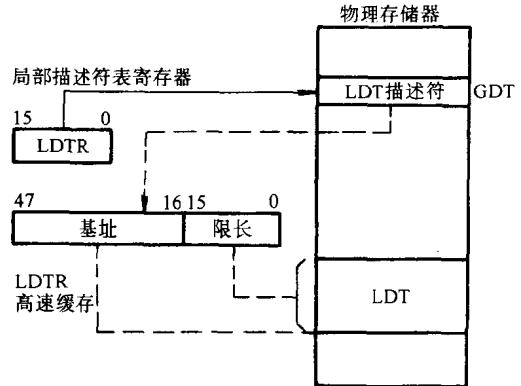


图 1.11 局部描述符表寻址结构

中断描述符表 IDT 用来定义中断服务程序地址空间，它包含中断描述符，整个系统只设一个。中断描述符在中断描述符表存储空间中的位置由中断描述符表寄存器说明，IDTR 中的高四个字节指示 IDT 在物理存储器中的起始地址。中断描述符表寻址结构如图 1.12 所示。

IDT 中用到的描述符类型称为中断门，中断门提供中断服务程序的起始地址及属性，可以将程序控制转移到中断服务程序去执行。

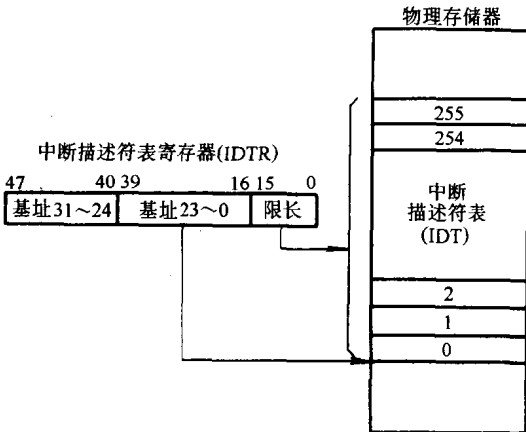


图 1.12 中断描述符表寻址结构

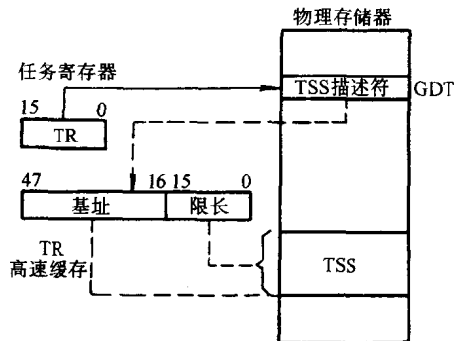


图 1.13 任务状态段寻址结构

任务寄存器 TR 的功能是支持任务切换，允许多任务系统以简单而有序的方式从一个任务切换到另一个任务。任务常常是一个进程或者应用程序。任务寄存器 TR 存放的是选择符，该选择符用来指示全局描述符表中任务状态段描述符的位置，通过它可以间接访问一个确定任务的描述符。当选择符装入 TR 中的时候，相应的任务状态段 (TSS) 描述符被从存

存储器中自动读出并且装入任务状态段描述符高速缓冲寄存器中。TSS 在任务切换过程中起着重要作用，通过它实现任务的挂起和恢复。在任务切换过程中，①处理器中各寄存器的当前值被自动地保存到 TR 所指定的 TSS 中；②下一任务 TSS 的选择符被装入 TR；③从 TR 所指定的 TSS 中取出各个寄存器的值送到处理器的各寄存器中，实现了任务切换。任务状态段寻址结构如图 1.13 所示。任务状态段基本格式如图 1.14 所示。任务状态段描述符，包括段起始地址和段界限。每个任务都有自己的任务状态段，任务状态段包括了任务所必需的信息，例如访问某个寄存器的初始值等。

31	0000000000000000	链接字段	0
	ESP ₀		4
	0000000000000000	SS ₀	8
	ESP ₁		0CH
	0000000000000000	SS ₁	10H
	ESP ₂		14H
	0000000000000000	SS ₂	18H
	CR ₃		1CH
	EIP		20H
	EFLAGS		24H
	EAX		28H
	ECX		2CH
	EDX		30H
	EBX		34H
	ESP		38H
	EBP		3CH
	ESI		40H
	EDI		44H
	0000000000000000	ES	48H
	0000000000000000	CS	4CH
	0000000000000000	SS	50H
	0000000000000000	DS	54H
	0000000000000000	FS	58H
	0000000000000000	GS	5CH
	0000000000000000	LDT	60H
	I/O 许可位图 I 偏移	0000000000000000	68H

图 1.14 任务状态段基本格式

4. 虚拟空间

32 位机通过全局描述符表、局部描述符表实现对程序的地址变换，把程序的顺序关系（编程地址）映射到 4GB 的线性地址空间。显然 32 位机能提供给编程者的最大编程空间就是它能完成变换的那些空间，称为虚拟空间。虚拟空间在物理上是由外存与内存结合提供的，它的最大值可以按以下过程计算：

1) 段寄存器中的 T_1 位决定一个任务（程序）可以使用一个全局描述符表和一个局部描述符表。

2) 13 位选择码决定了每个描述符表中最多有 2^{13} 个描述符。因此一个任务可拥有的描述符数目最多为 2×2^{13} 个。

3) 描述符中的 G 位为 1 时，相应的段长度为 $2^{20} \times 2^{12}$ 字节。

所以，一个任务最多可拥有的编程用字节数为 $2 \times 2^{13} \times 2^{20} \times 2^{12} = 2^{46} = 64\text{TB}$ ，即编程空间——虚拟空间为 64TB。

5. 特权级与特权保护

特权级与特权保护是为了支持多用户多任务操作系统，使系统程序和用户的任务程序之间、各任务程序之间互不干扰而采取的保护措施。32 位微处理器提供了一个 4 级特权管理系统，也就是 4 级保护系统。这样可为不同程序规定一个权限，控制特权指令和 I/O 指令的使用，控制对段和段描述符的访问，从而有效地防止不同程序执行时的相互干扰或非法访问、非法改写 GDT 和 LDT。为了使程序和数据安全可靠，80X86 除了提供 4 级特权保护之外还采取了页面保护措施。页面保护将在后面介绍。

特权级用 PL 表示，分为 0、1、2 和 3 级。0 级特权最高，一般赋给操作系统的核心程序；1 级赋给来自操作系统的服务程序；2 级赋给操作系统扩展程序；3 级赋给用户程序，级别最低。在实施管理中使用了 3 种形式的特权管理：当前任务特权（CPL）、选择符特权（RPL）和描述符特权（DPL），并且规定某一特权级的段中的数据只能由同级或高特权级中运行的程序使用，某一特权级的代码段或过程只能由同级或低特权级中运行的程序调用。

(1) 选择符特权 RPL 选择符特权（又称请求特权），是指选择符的请求特权级，由段寄存器中选择符的低 2 位 RPL 确定。它规定了访问该描述符表的任务的最低级别。只有当前任务的 CPL 等于或高于 RPL 时，才能访问该描述符表，同时等于高于所选择描述符中的 DPL 时才能访问其中的数据。代码段的 RPL 是指被调用程序段的特权级。

(2) 描述符特权 DPL 描述符特权是由段描述符或者门描述符中的 DPL 确定，它规定了访问该描述符所确定段的任务的最低级别。如果所描述的段是数据段，则规定了访问该段的最低特权级；如果所描述的段是代码段，则规定了执行该段所需的 CPL。只有当 CPL 等于或高于 DPL 时，当前任务才能访问描述符所确定的段中的数据。

(3) 任务特权 CPL 任务特权（又称当前特权级）是指当前任务中正在运行的代码段或者访问的数据段的特权级。每一项任务执行时都是在其代码段描述符所确定的特权级中运行。一个任务根据其特定时刻所执行的不同段，可运行于四种不同的特权级之一。

任务执行时当前特权级一般不能改变，如果必须改变，只能通过代码段的门描述符的控制转换才能实现。当任务通过一个任务转换操作启动后，该任务就在代码段寄存器的低 2 位 RPL 指定的特权级中执行。

对于数据段的访问规定为，根据 CPL 代码段可以访问同级或者低特权级 GDT 和 LDT 中定义的数据段，不允许访问高特权级的数据段。例如，CPL = 1，则只能访问 DPL 为 1、2 或 3 的数据段。

对于代码段转移的规定是：新代码段的特权级必须更高，即控制转移的条件是：CPL = DPL，CPL 的特权级小于等于 RPL 的特权级。如果 CPL = DPL，则属于同一级的转移；如果 CPL 的特权级比 DPL 高，则程序执行在 CPL 级。任务的特权级从 CPL 到更高特权级的改变是由调用门的间接控制转移来实现的。调用门的内容包括控制转移入口点的虚拟地址。

对于输入/输出有两个特权级，第一个是 I/O 驱动程序，属于系统程序，规定特权级为 0；第二个是 IOPL，用于限制 I/O 指令的执行，IN、INS、OUT、OUTS、CLI 和 STI 只能在大于和等于 IOPL 规定的特权级上执行。

6. 段地址转换