汇编语言程序设计

HUIBIAN YUYAN CHENGXU SHEJI

樊景博　田祎　主编

天津大学出版社
TIANJIN UNIVERSITY PRESS

汇编语言程序设计

主　编　樊景博　田　祎

天津大学出版社
TIANJIN UNIVERSITY PRESS

内容提要

本书以8086/8088指令为主，系统地介绍了汇编语言的基础理论知识和程序设计方法。主要内容包括：基础知识、寻址方式、基本指令、数据的表示和常用伪指令、顺序程序设计、分支程序设计、循环程序设计、子程序、字符串处理技术、宏、输入输出和中断、文件操作与终端控制、汇编语言和C语言。各章节内容循序渐进，并有侧重地与C语言进行对比，重点突出，结构清晰，简洁易懂。

本书可作为本、专科院校计算机及相关专业的教材，也可供科研及软件开发人员自学参考。

图书在版编目（CIP）数据

汇编语言程序设计 / 樊景博，田祎主编. — 天津：天津大学出版社，2016.8

ISBN 978-7-5618-5600-0

Ⅰ.①汇… Ⅱ.①樊… ②田… Ⅲ.①汇编语言-程序设计-高等学校-教材 Ⅳ.①TP313

中国版本图书馆CIP数据核字（2016）第171087号

出版发行	天津大学出版社
地　　址	天津市卫津路92号天津大学内（邮编：300072）
电　　话	发行部：022-27403647
网　　址	publish.tju.edu.cn
印　　刷	天津泰宇印务有限公司
经　　销	全国各地新华书店
开　　本	185mm×260mm
印　　张	14
字　　数	349千
版　　次	2016年8月第1版
印　　次	2016年8月第1次
定　　价	35.00元

前　言

汇编语言是一门低级语言，它能充分利用计算机的硬件资源且能进行有效的直接控制，同时利用它来深刻理解和掌握计算机深层结构和许多专业应用知识，即使在计算机业高速发展的今天，也起着独特的甚至不可替代的作用。汇编语言程序设计是计算机专业一门重要的基础课程，是必修的核心课程之一，是“数据结构”“操作系统”和“微机原理与接口技术”等其他核心课程的必要先修课。

目前市场上的微型计算机大都采用 Intel 公司的 CPU 或 AMD 等公司的其他兼容产品，因此，本书也将围绕 Intel CPU 来介绍汇编语言，这些内容不仅适用于 Intel 系列的 CPU，对 AMD 公司的 CPU 也同样适用。除了某些特殊功能外，AMD 等公司的 CPU 和 Intel CPU 是完全兼容的，从程序员的角度看没有太多差别。汇编语言的主要应用领域是工业控制，现在工业控制中使用的计算机、单片机，有很多与 8086/8088 有相似结构，原理也大体相同，因此本教材以 8086/8088 作为主讲 CPU，采用循序渐进的教学思想，按照结构化程序设计流程，将各指令分散到不同的章节进行讲述，有效避免学生学习过程中由于知识的遗忘导致后续课程进行困难的问题。通过循序渐进的学习系统理论，让学生理解与掌握程序设计技术，培养学生综合分析问题、解决问题的能力，也为掌握工控机铺平道路，成为读者进入硬件领域的一块铺路石。

本教材共分为 13 章，其中第 6、7、9、11、12 章由樊景博编写，第 1、2、3、4、5、8、10、13 章和附录由田祎编写。全书内容由田祎统稿，樊景博定稿。本书在编写过程中参考了大量的文献资料，在此对这些文献资料的所有者表示衷心的感谢。由于计算机技术发展很快，加上编者水平由限，书中难免有不妥之处，恳请广大读者批评指正，编者不胜感激。本书的出版受到了商洛学院教材基金项目的大力支持，在此一并表示感谢。

编　者

2016 年 3 月

目 录

第1章　基础知识

汇编语言和计算机的机器语言有着直接的联系，要理解计算机的工作原理与工作过程，学习汇编语言很有必要。要学习计算机的程序设计，懂一点汇编语言也是很有好处的，如果进行涉及计算机控制、通信、动画、虚拟现实程序设计及许多对速度要求较高的软件设计，都常要求使用汇编语言设计；汇编语言程序经常被用来改进软件和硬件控制系统的效率，即使是使用如 C、C++、JAVA 等高级语言进行应用系统的开发，也常要求嵌入汇编语言程序模块，以提高软件或硬件控制系统的运行速度；汇编语言还常用于高级语言的程序调试，解决一些涉及机器底层的问题。

汇编语言是一种面向机器的语言，不同 CPU 的计算机，其汇编语言都不相同。要学习某种汇编语言，就必须首先了解应用该汇编语言的计算机的硬件结构、数据类型及其在机内的表示方法。本书围绕 8086/8088 CPU 展开学习，纯粹的 8086 PC 机已经不存在了，但现在的任何一台 PC 机中的微处理器，只要是和 Intel 兼容的系列，都能够以 8086 的方式进行工作。学习 8086/8088 CPU 可以方便地进行实践，较容易理解并掌握其精髓。

汇编语言是一种可以直接控制计算机硬件设备的计算机语言，掌握一些计算机硬件知识是学习汇编语言的必要前提，其中最重要的是了解计算机各部件的基本结构和逻辑连接关系，尤其是核心部件 CPU 的内部结构。掌握内部结构的目的在于了解计算机的各主要部件分别能完成什么样的功能，也就是它们能做什么，然后才可以通过计算机语言编程告诉计算机一个个基本步骤，从而完成复杂的任务。

本章介绍汇编语言的最基本的概念及学习汇编语言所必须具备的计算机系统的相关知识。

1.1　汇编语言简介

从 1946 年第一台可编程计算机 ENIAC 诞生至今，计算机经历了电子管、晶体管、集成电路和超大规模集成电路四个发展阶段，现正朝着巨型化、微型化、网络化和智能化的第五代计算机发展，已渗透到社会和生活的各个领域。人们与计算机进行交流的“语言”也从机器语言发展到汇编语言与高级语言，现正朝着“自然语言”的方向发展。

1.1.1　机器语言

计算机的所有操作都是在指令的控制下进行的。能够直接控制计算机完成指定动作的是机器指令。一条机器指令是一个由 0 和 1 组成的二进制代码序列，不同的机器指令对应的二进制代码序列也各不相同。一条机器指令通常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	操作数

操作码部分用来指出这条指令要求计算机做什么样的操作，是做加法，做减法，还是完成数据传送，抑或是其他的操作；操作数部分给出参与操作的数据值，或者指出操作对象在什么地方。下面的二进制代码序列就是一条 8086/8088 的机器指令：

10000000 00000110	01100100 00000000 00010010

这条指令的前 16 位是操作码部分，含义是要求计算机做两个数的加法操作；后 24 位是操作数部分，第 17 位至第 32 位指出第一个加数在内部存储器的编号为 100 的那个字节中，最后 8 位指出另一个加数就在指令中，是 18。

对于同样的二进制序列，不同型号的 CPU 对它的“理解”是不一样的，比如上面的那一串二进制代码在 8086/8088 看来是要求做加法，换到另一种 CPU 中完全可能被当作是另一种操作，甚至是错误的指令，所以机器指令与机器本身有着紧密的联系。不同型号的计算机（准确地说是不同型号的 CPU）都有自己的一套指令，一种机型的所有机器指令的集合就是它的指令系统。指令系统及其使用规则构成这种计算机的机器语言。选择指令系统中的指令并排列起来，可以构成一个指令序列，用以告诉计算机完成一连串的动作，就是一个机器语言程序。

1.1.2 汇编语言

早期的程序员们很快就发现了使用机器语言带来的麻烦，它是如此难于辨别和记忆，给整个产业的发展带来了障碍，于是产生了汇编语言。汇编语言是一种采用指令助记符、符号地址、标号等符号书写程序的语言，它便于人们书写、阅读和检查。汇编语言指令与计算机指令基本上是一一对应的，汇编语言与计算机有着密不可分的关系，处理器不同，汇编语言就不同，因此它是一种低级语言，同时它也是唯一能够充分利用计算机硬件特性并直接控制硬件设备的语言。利用汇编语言进行程序设计体现了计算机硬件和软件的结合。

1.1.3 高级语言

高级语言是一种与具体的计算机硬件无关，独立于计算机类型的通用语言，比较接近人类自然语言的语法，用高级语言编程不必了解和熟悉计算机的指令系统，更容易掌握和使用。高级语言采用接近自然语言的词汇，其程序的通用性强，易学易用，这些语言面向求解问题的过程，不依赖具体计算机。高级语言也要翻译成机器语言才能在计算机上执行。其翻译有两种方式，一种是把高级语言程序翻译成机器语言程序，然后经过连接程序连接成可执行文件，再在计算机上执行，这种翻译方式称为编译方式，大多数高级语言如 JAVA、C、C++、C#等都是采用这种方式；另一种是直接把高级语言程序在计算机上运行，一边解释一边执行，这种翻译方式称为解释方式，如 BASIC 语言就采用这种方式。

高级语言源程序是在未考虑计算机结构特点情况下编写的，经过翻译后的目标程序往往不够精练，过于冗长，加大了目标程序的长度，占用较大存储空间，执行时间较长。

1.1.4　自然语言与汇编语言的对比

机器语言是计算机的“母语”，这是一种绝大多数人都不懂也很难学会的语言，正如前面给出的一条机器指令的例子会令试图学习机器语言的人望而生畏。另一方面，人类自己使用汉语、英语、法语等自然语言进行交流。任何一种自然语言对于当今的计算机来说都是无法领会的，而且，目前的技术还无法把人的自然语言直接翻译成机器语言。因而，人与计算机之间进行交流就存在一定的困难。比较好的解决方法是找一种双方都能够学会也容易学会的语言作为中间媒介，汇编语言以及后来的高级语言、第四代语言都扮演着这样的中介角色。

一个已掌握了自己的母语的人，如果要学习一种新的语言，他该学些什么呢？不妨想象一下中国人学英语的过程：大概所有把英语作为外语来学习的人都是从字母开始的，以后是单词、简单的句子，再发展到用若干连贯的句子描述一件简单的事情，最后是熟练地写英语文章。在学习过程中，从单词的拼写到句子的组织，再到文章的连贯，都会穿插着相应的语法知识。汇编语言既然是一种语言，学习过程也大致如此。表 1.1 中列举了自然语言与汇编语言的对照关系，一方面说明在学习这两种语言时有很多共同之处，另一方面也表明汇编语言需要学习的主要内容。

表 1.1　自然语言与汇编语言的对照

语言 / 对比项目	自然语言（英语）	汇编语言
基本符号	字母表	字母、专用符号
词	单词	保留字、标识符
句	句子	完整的指令、伪指令
段	段落	子程序
章	文章	程序
语法	拼写、句法、文法	指令、子程序、程序的格式及其使用规则
技巧	句子正确、文理通顺	指令正确、程序精简、易读性好、结构化好

汇编语言是介于自然语言和机器语言之间的一种人机交流媒介。人可以发挥自己的聪明才智学会这一类新的语言，但计算机又如何去“学会”呢？这是利用汇编语言到机器语言的固定翻译机制实现的。编写好的汇编语言程序可以通过一种固定的模式翻译成机器语言。这种翻译工作如果由人来完成同样是非常困难的，而且出错的可能性很大；再说，这种翻译很枯燥、很机械，倒是非常适合由计算机按人们指定的方法自动进行，因此计算机专家们已编制了一些翻译程序供汇编语言编程人员使用，这种翻译程序被称为“汇编程序”。

1.1.5　汇编程序和连接程序

汇编程序是一种计算机软件，属于系统软件部分，它能够把人们编写的汇编语言程序（称为源程序，一般以. ASM 作为文件扩展名）翻译成机器语言，这种翻译操作称为“汇编”。由于不同的计算机有不同的机器语言，因而也需要有不同的翻译器——汇编程序。

MASM. EXE 是一种专门用于把 Intel 8086/8088 的汇编语言源程序翻译成相应的机器语言程序的翻译器，是 8086/8088 汇编语言编程人员必备的基本工具之一。

汇编程序还具有语法检查的功能，交给汇编程序进行处理的源程序在翻译之前都必须经过语法检查这一关。如果汇编程序发现源程序中有违背汇编语言语法的地方，将不进行翻译工作，而是指出错误的位置以及类型。从这个角度来说，汇编程序决定了汇编语言的语法，不同厂家、不同版本的汇编程序在语法规定上可能有细微的差别。

汇编程序翻译的结果已具备机器语言的形式，称为“目标程序”，一般以. OBJ 作为文件扩展名。但是，目标程序还不能直接交给计算机去执行，它还需要通过连接程序（LINK. EXE）的装配才具备可执行的形式，装配结果称为“执行文件”，一般以. EXE 作为文件扩展名。另一方面，连接程序还具有把多个目标程序装配在一起的功能，也可以把目标程序与预先编写好的一些放在子程序库中的子程序连接在一起，构成较大的执行文件。汇编语言的源程序、汇编程序、目标程序、连接程序、执行文件的关系如图 1.1 所示。

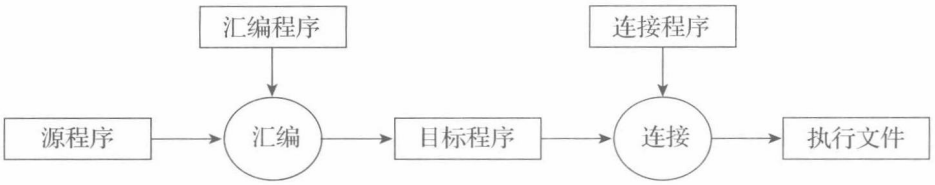

图 1.1　由汇编语言源程序到执行文件的处理过程

1.1.6　汇编语言的构成

汇编语言是较早发明的一种程序设计语言。为了使汇编语言到机器语言的翻译比较简单，汇编语言用大量的语法规则对从指令到程序的书写加以限制。与后来的高级语言、第四代语言相比，汇编语言更接近于机器语言，用汇编语言编写的源程序还保留了很多机器语言的影子。比如，机器指令中的操作码部分在汇编语言中表现为与该指令的功能相关的英文单词或其缩写，例如加法指令用 ADD 表示，数据传送指令用 MOV 表示，这类符号称为“助记符”。汇编语言指令的格式是助记符在前，参与操作的数据在后。

对于存放在内存当中的一批数据，如果要求编程人员记住每一个数据在内存中的具体位置是不现实的（在最早的计算机中，这个工作却是必需的）。在汇编语言中，每存放一个数据都可以为它起一个名字——变量名，程序员只要记住变量的名字即可。

跳转是程序设计中不可避免的问题。在机器语言中，跳转的目的地是用指令所在的位置（即在内存的哪一个字节）来表示的，而汇编语言中的跳转则是在目的地做一个称为“标号”的标记。

除了与机器语言有直接对应关系的助记符、变量、标号外，为了能让汇编程序正确地完成翻译工作，必须要告诉汇编程序变量需要占据多少字节的内存、程序到何处结束、整个程序的第一条指令在什么地方等问题。因此，源程序中需要有一些告诉汇编程序如何进行翻译操作的“说明”，这类说明在翻译结果中没有对应的机器代码，所以称为“伪指令”。

指令助记符、数据和存放数据的变量、标号、伪指令以及相应的使用规则构成了汇编语言的全部内容。

1.1.7　汇编语言的特点

汇编语言使用助记符和符号地址，所以它要比机器语言易于掌握，与高级语言相比较，汇编语言有以下特点。

1. 汇编语言与计算机关系密切

汇编语言中的指令是机器指令的符号表示，与机器指令是一一对应的，因此它与计算机有着密切的关系，不同类型的 CPU 有不同的汇编语言，也就有各种不同的汇编程序。汇编语言源程序与高级语言源程序相比，其通用性和可移植性要差得多。

2. 汇编语言程序效率高

由于构成汇编语言主体的指令是用机器指令的符号表示的，每一条指令都对应一条机器指令，且汇编语言程序能直接利用计算机硬件系统的许多特性，如它允许程序员利用寄存器、标志位等编程。用汇编语言编写的源程序在编译后得到的目标程序效率高，主要体现在空间效率和时间效率上，即目标程序短、运行速度快这两个方面，在采用相同算法的前提下，任何高级语言程序在这两个方面的效率与汇编语言相比都望尘莫及。

3. 特殊的使用场合

汇编语言可以实现高级语言难以胜任甚至不能完成的任务。汇编语言具有直接和简捷的特点，用它编制程序能精确地描述算法，充分发挥计算机硬件的功能。在过程控制、多媒体接口、设备通信、内存管理、硬件控制等方面的程序设计中，用汇编语言直接方便，执行速度快，效率高。

汇编语言提供了一些模块间相互连接的方法，一个大的任务可以分解成若干模块，将其中执行频率高的模块用汇编语言编写，可以大大提高大型软件的性能。

高级语言和第四代语言在科学计算、事务处理等方面比汇编语言有巨大的优势，但用高级语言编写的程序，在翻译成机器语言后，程序代码冗长，占用存储空间大，执行速度慢。如果用高级语言来编写接口控制、设备通讯等方面的程序则不太合适，相反这样的情况下汇编语言更容易发挥其长处：最终的执行代码简短，执行速度快，效率高，特别是汇编语言能直接控制计算机的外设，这些特点是高级语言和第四代语言望尘莫及的。

因此高级语言、第四代语言适合于编写应用软件，而对于系统软件，尤其是涉及内存管理、硬件控制方面问题时汇编语言更合适。可以说，汇编语言程序设计是从事计算机研究与应用的重要手段，是软件与硬件相结合的基础。

1.2　微型计算机概述

微型计算机由中央处理器（Central Processing Unit，CPU）、存储器、输入输出接口电路和总线构成。CPU 如同微型计算机的心脏，它的性能决定了整个微型计算机的各项关键指标。存储器包括随机存取存储器（Random Access Memory，RAM）和只读存储器（Read Only Memory，ROM）。输入输出接口电路用来连接外部设备和微型计算机。总线为 CPU 和其他部件之间提供数据、地址和控制信息的传输通道。如图 1.2 所示为微型计算机的基本结构。

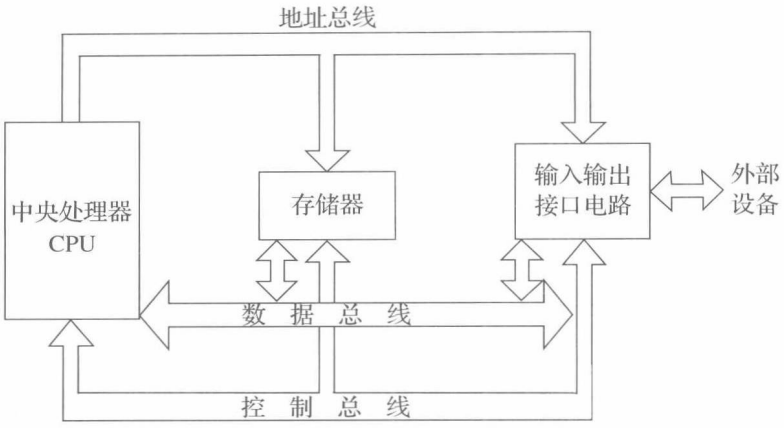

图 1.2　微型计算机基本结构

特别要提到的是，微型计算机的总线结构，使系统中各功能部件之间的相互关系变为各个部件面向总线的单一关系。一个部件只要符合总线结构标准，就可以连接到采用这种总线结构的系统中，使系统功能得到扩展。

数据总线用来在 CPU 与内存或其他部件之间进行数据传送。它是双向的，数据总线的位宽决定了 CPU 和外界的数据传送速度，8 位数据总线一次可传送一个 8 位二进制数据（即一个字节），16 位数据总线一次可传送两个字节。在微型计算机中，数据的含义是广义的，数据总线上传送的不一定是真正的数据，还可能是指令代码、状态量或控制量。

地址总线专门用来传送地址信息，它是单向的。地址总线的位数决定了 CPU 可以直接寻址的内存范围。如 CPU 的地址总线的宽度为 N，则 CPU 最多可以寻找 2^N 个内存单元。

控制总线用来传输控制信号，其中包括 CPU 送往存储器和输入输出接口电路的控制信号，如读信号、写信号和中断响应信号等；也包括其他部件送到 CPU 的信号，如时钟信号、中断请求信号和准备就绪信号等。

1.2.1　Intel 公司微处理器简介

自 20 世纪 70 年代开始出现微型计算机以来，CPU 经历了飞速的发展。1971 年，Intel 设计成功了第一片 4 位微处理器 Intel 4004；随之又设计生产了 8 位微处理器 8008；1973 年推出了 8080；1974 年基于 8080 的个人计算机（Personal Computer，PC）问世，Microsoft 公司的创始人 Bill Gates 为 PC 开发了 BASIC 语言解释程序；1977 年 Intel 推出了 8085。自此之后，Intel 又陆续推出了 8086、80386、Pentium 等 80×86 系列微处理器。各种微处理器的主要区别在于处理速度、寄存器位数、数据总线宽度和地址总线宽度。下面简要介绍不同时期 Intel 公司制造的几种主要型号的微处理器，这些微处理器都曾经或正在广为流行。

1. 80×86 系列微处理器

1）8086 微处理器

指令系统与 8088 完全相同，具有多个 16 位的寄存器、16 位数据总线和 20 位地址总线，可以寻址 1MB 的内存，一次可以传送 2 个字节。该处理器只能工作在实模式。

2）8088 微处理器

具有多个 16 位的寄存器、8 位数据总线和 20 位地址总线，可以寻址 1MB 的内存。虽然这些寄存器一次可以处理 2 个字节，但数据总线一次只能传送 1 个字节。该处理器只能工作在实模式。

3）80286 微处理器

比 8086 运行更快，具有多个 16 位的寄存器、16 位数据总线和 24 位地址总线，可以寻址 16MB 内存。它既可以工作在实模式，也可以工作在保护模式。

4）80386 微处理器

具有多个 32 位的寄存器、32 位数据总线和 32 位地址总线，可以寻址 4GB 内存。它提供了较高的时钟速度，增加了存储器管理和相应的硬件电路，减少了软件开销，提高了效率。它既可以工作在实模式，也可以工作在保护模式。

5）80486 微处理器

具有多个 32 位的寄存器、32 位数据总线和 32 位地址总线。它比 80386 增加了数字协处理器和 8kB 的高速缓存，提高了处理速度。它既可以工作在实模式，也可以工作在保护模式。

6）Pentium（奔腾）

具有多个 32 位的寄存器、64 位数据总线和 36 位地址总线。因为它采用了超标量体系结构，所以每个时钟周期允许同时执行两条指令，处理速度得到了进一步提高，性能比 80486 优越得多。它既可以工作在实模式，也可以工作在保护模式。

以上介绍了 Intel 80×86 系列的一些主要微处理器，表 1.2 给出了该系列部分微处理器的数据总线和地址总线宽度。实际上 80×86 系列的功能还在不断改进和增强，它们的速度将会更快，性能将会更优越。但无论怎样变化，它们总会被设计成是完全向下兼容的，就像在 8086 上设计和运行的软件可以不加任何改变地在 Pentium 4 机上运行一样。对于汇编语言编程人员来讲，掌握 16 位计算机的编程十分重要，它是学习高档计算机及保护模式编程的基础，也是掌握实模式程序设计的唯一方法。

表 1.2　Intel 80×86 系列微处理器总线宽度

CPU	数据总线宽度	地址总线宽度	CPU	数据总线宽度	地址总线宽度
8086	16	20	Pentium	64	36
8088	8	20	Pentium Ⅱ	64	36
80286	16	24	Pentium Ⅲ	64	36
80386SX	16	24	Pentium 4	64	36
80386DX	32	32	Itanium	64	44
80486	32	32			

2. CPU 的主要性能指标

1）机器字长

机器字长和 CPU 内部寄存器、运算器、内部数据总线的位宽相一致。如 8086CPU，它的内部寄存器是 16 位的、运算器能完成两个 16 位二进制数的并行运算、数据总线的位宽为 16 位，则它的机器字长为 16 位，也称其为 16 位计算机。通常，机器字长越长，计算机的运算能力越强，其运算精度也越高。

2）速度

CPU 的速度是指单位时间内能够执行指令的条数。速度的计算单位不一，若以单字长定点指令的平均执行时间计算，用每秒百万条指令（Million Instructions Per Second，MIPS）作为单位；若以单字长浮点指令的平均执行时间计算，则用每秒百万条浮点运算指令（Million Floating - point Operations Per Second，MFLOPS）表示。现在，采用计算机中各种指令的平均执行时间和相应的指令运行权重的加权平均法求出等效速度作为计算机的运算速度。

3）主频

主频又称为主时钟频率，是指 CPU 在单位时间内产生的时钟脉冲数，以 MHz/s（兆赫兹每秒）为单位。由于计算机中的一切操作都是在时钟控制下完成的，因此，对于机器结构相同或相近的计算机，CPU 的时钟频率越高，运算速度越快。

1.3 程序可见寄存器组

80386 以上型号（含 80386）的 CPU 能够处理 32 位数据，其寄存器长度是 32 位的，但为了与早期的 8086 等 16 位机 CPU 保持良好的兼容性，80386 以上型号的 CPU 中程序可见寄存器组包括多个 8 位、16 位和 32 位寄存器，如图 1.3 所示。

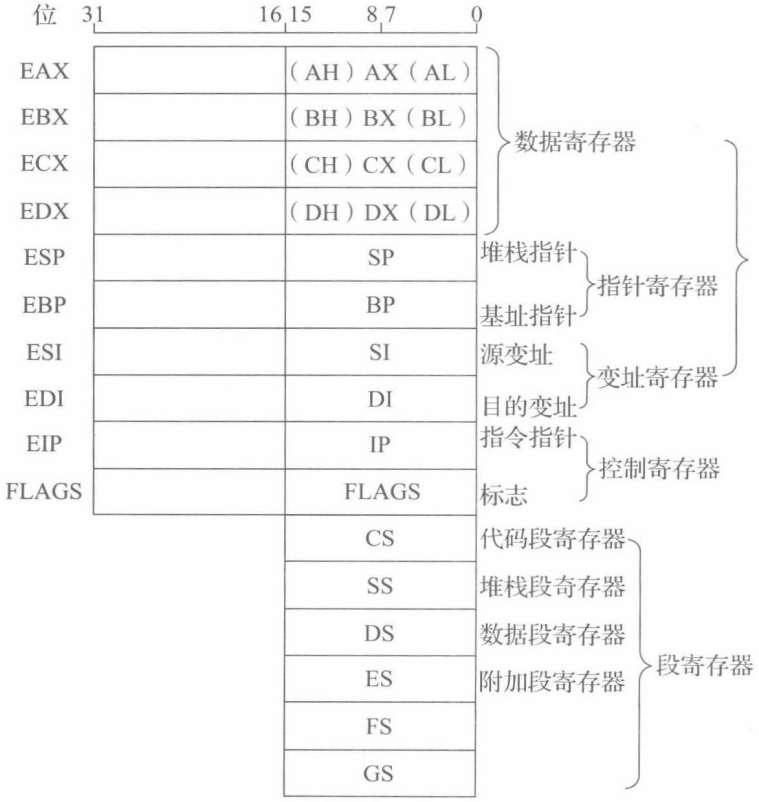

图 1.3　8086 ~ Pentium CPU 程序可见寄存器组

1. 通用寄存器

8086 ~ 80286 CPU 各有 8 个 16 位通用寄存器 AX、BX、CX、DX、SP、BP、SI、DI。对于 4 个 16 位数据寄存器 AX、BX、CX、DX，其每个又可以作为 2 个独立的 8 位寄存器使用，它们被分别命名为 AH、AL、BH、BL、CH、CL、DH、DL。80386 以上型号的 CPU 各有 8 个 32 位通用寄存器，它们是相应 16 位寄存器的扩展，被分别命名为 EAX、EBX、ECX、EDX、ESP、EBP、ESI、EDI。在程序中每个 8 位、16 位、32 位寄存器都可以独立使用。

SP 叫做堆栈指针寄存器，其中存放当前堆栈段栈顶的偏移量，它们总是与 SS 堆栈段寄存器配合存取堆栈中的数据。

除 SP 堆栈指针不能随意修改、需要慎用外，其他通用寄存器都可以直接在指令中使用，用以存放操作数，这是它们的通用之处。在后边讨论指令系统时，可以看到某些通用寄存器在具体的指令中还有其他用途，例如 AX、AL（通常分别被称为 16 位、8 位累加器），它们在乘除法、十进制运算、输入输出指令中有专门用途。另外有些通用寄存器也可以存放地址用以间接寻址内存单元，例如在实模式中 BX、BP、SI、DI 可以作为间接寻址的寄存器，用以寻址 64KB 以内的内存单元。

2. 段寄存器

在 IBM PC 机中存储器采用分段管理的方法，因此一个物理地址需要用段基地址和偏移量表示。一个程序可以由多个段组成，但对于 8086 ~ 80286 CPU，由于只有 4 个段寄存器，所以在某一时刻正在运行的程序只可以访问 4 个当前段，而对于 80386 及其以上的计算机，由于有 6 个段寄存器，则可以访问 6 个当前段。在实模式下段寄存器存放当前正在运行程序的段基地址的高 16 位，在保护模式下存放当前正在运行程序的段选择子，段选择子用以选择描述符表中的一个描述符，描述符描述段的基地址、长度和访问权限等，显然在保护模式下段寄存器仍然是选择一个内存段，只是不像实模式那样直接存放段基址罢了。

代码段寄存器 CS 指定当前代码段，代码段中存放当前正在运行的程序段。堆栈段寄存器 SS 指定当前堆栈段，堆栈段是在内存开辟的一块特殊区域，其中的数据访问按照后进先出（Last in First Out，LIFO）的原则进行，允许插入和删除的一端叫做栈顶。IBM PC 机中 SP（或 ESP）指向栈顶，SS 指向堆栈段基地址。数据段寄存器 DS 指定当前运行程序所使用的数据段。附加数据段寄存器 ES 指定当前运行程序所使用的附加数据段。段寄存器 FS 和 GS 只对 80386 以上 CPU 有效，它们没有对应的中文名称，用于指定当前运行程序的另外两个存放数据的存储段。虽然 DS、ES、FS、GS（甚至于 CS、SS）所指定的段中都可以存放数据，但 DS 是主数据段寄存器，在默认情况下使用 DS 所指向段的数据。若要引用其他段中的数据，需要显式地说明。

3. 控制寄存器

控制寄存器包括指令指针寄存器和标志寄存器。在程序中不能直接引用控制寄存器名。

1）IP

IP 叫做指令指针寄存器，它总是与 CS 段寄存器配合指出下一条要执行指令的地址，其中存放偏移量部分。

2）标志寄存器（FLAGS）

标志寄存器也被称为状态寄存器，由运算结果特征标志和控制标志组成。8086～80286 CPU 为 16 位，80386 及以上为 32 位。如图 1.4 所示，可以看出它们完全向下兼容。空白位为将来保留，暂未定义。

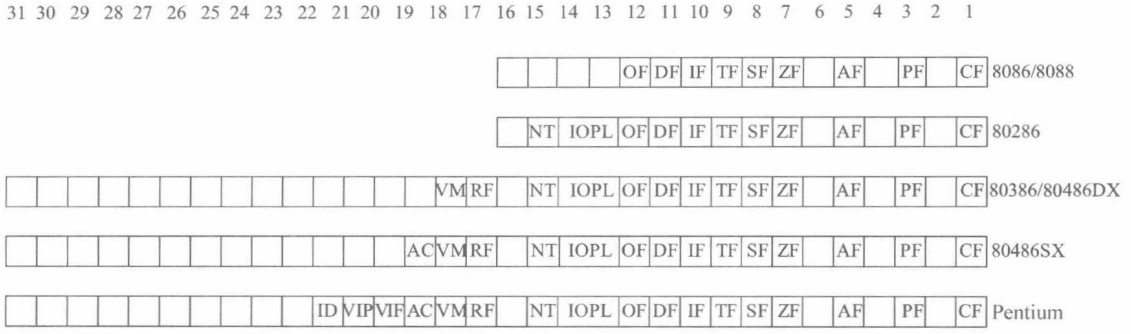

图 1.4　标志寄存器

（1）运算结果特征标志：用于记录程序中运行结果的特征，8086～Pentium CPU 的标志寄存器均含有这 6 位标志。

CF（Carry Flag）：进位标志，记录运算结果的最高位向前产生的进位或借位。若有进位或借位则置 CF = 1，否则清零。可用于检测无符号数二进制加减法运算时是否发生溢出（溢出时 CF = 1）。

PF（Parity Flag）：奇偶标志，记录运算结果中含 1 的个数。若个数为偶数则置 PF = 1，否则清零。可用于检测数据传送过程中是否发生错误。

AF（Assistant Carry Flag）：辅助进位标志，记录运算结果最低 4 位（低半字节）向前产生的进位或借位。若有进位或借位则置 AF = 1，否则清零。只有在执行十进制运算指令时才关心此位。

ZF（Zero Flag）：零标志，记录运算结果是否为零，若结果为零则置 1，否则清零。

SF（Sign Flag）：符号标志，记录运算结果的符号，若结果为负则置 1，否则清零。

OF（Overflow Flag）：溢出标志，记录运算结果是否超出了操作数所能表示的范围。若超出则置 1，否则清零。可用于检测带符号数运算时是否发生溢出。

（2）控制标志：控制标志控制处理器的操作，要通过专门的指令才能使控制标志发生变化。

① 以下控制标志对 8086～Pentium CPU 均有效。

IF（Interrupt Flag）：中断允许标志，当 IF = 1 时允许 CPU 响应外部可屏蔽中断请求（INTR）；当 IF = 0 时禁止响应 INTR。IF 的控制只对 INTR 起作用。

DF（Direction Flag）：方向标志，专门服务于字符串操作指令。当 DF = 1 时，表示串操作指令中操作数地址为自动减量，这样使得对字符串的处理是从高地址向低地址方向进行的；当 DF = 0 时，表示串操作指令中操作数地址为自动增量。

TF（Trap Flag）：陷阱标志，用于程序调试。当 TF = 1 时，CPU 处于单步方式；当 TF = 0 时，CPU 处于连续方式。状态标志位的符号表示见表 1.3。

表 1.3　状态标志位的符号表示

标志位	标志为 1	标志为 0
CF 进位（有/否）	CY	NC
PF 奇偶（偶/奇）	PE	PO
AF 半进位	AC	NA
ZF 全零（是/否）	ZR	NZ
SF 符号（负/正）	NG	PL
IF 中断（允许/禁止）	EI	DI
DF 方向（增量/减量）	DN	UP
OF 溢出（是/否）	OV	NV

1.4　存储器

1.4.1　基本概念

计算机中存储信息的基本单位是 1 个二进制位，简称位（bit），可用小写字母 b 表示，一位可存储一位二进制数。

IBM PC 机中常用的数据类型如下。

字节（byte）：IBM PC 机中存取信息的基本单位，可用大写字母 B 表示。1 个字节由 8 位二进制数组成，其位编号自左至右为 b_7、b_6、b_5、b_4、b_3、b_2、b_1、b_0。1 个字节占用 1 个存储单元。

字：1 个字 16 位，其位编号为 $b_{15} \sim b_0$。1 个字占用 2 个存储单元。

双字：1 个双字 32 位，其位编号为 $b_{31} \sim b_0$。1 个双字占用 4 个存储单元。

四字：1 个四字 64 位，其位编号为 $b_{63} \sim b_0$。1 个四字占用 8 个存储单元。

为了正确区分不同的内存单元，给每个单元分配一个存储器地址，地址从 0 开始编号，顺序递增 1。在计算机中地址用无符号二进制数表示，可简写为十六进制数形式。一个存储单元中存放的信息称为该单元的内容。例如 2 号单元中存放了一个数字 8，则表示为：(2) =8。

对于字、双字、四字数据类型，由于它们每个数据都要占用多个单元，访问时只需给出最低单元的地址号即可，然后依次存取后续字节。

需要注意的是，按照 Intel 公司的习惯，对于字、双字、四字数据类型，其低地址中存放低位字节数据，高地址中存放高位字节数据，这就是有些资料中称为“逆序存放”的含义。

例如内存现有以下数据（后缀 H 表示是十六进制数）。

地址：0　　1　　2　　3　　4　　5…

内容：12H　34H　45H　67H　89H　0AH…

存储情况如图 1.5 所示，则对于不同的数据类型，从 1 号单元取到的数据是：

$(1)_{字节}$ =34H

$(1)_{字}$ =4534H

$(1)_{双字}$ =89674534H

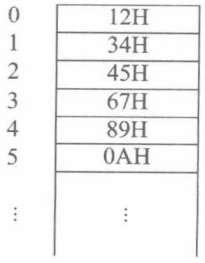

图 1.5　存储单元的地址和内容