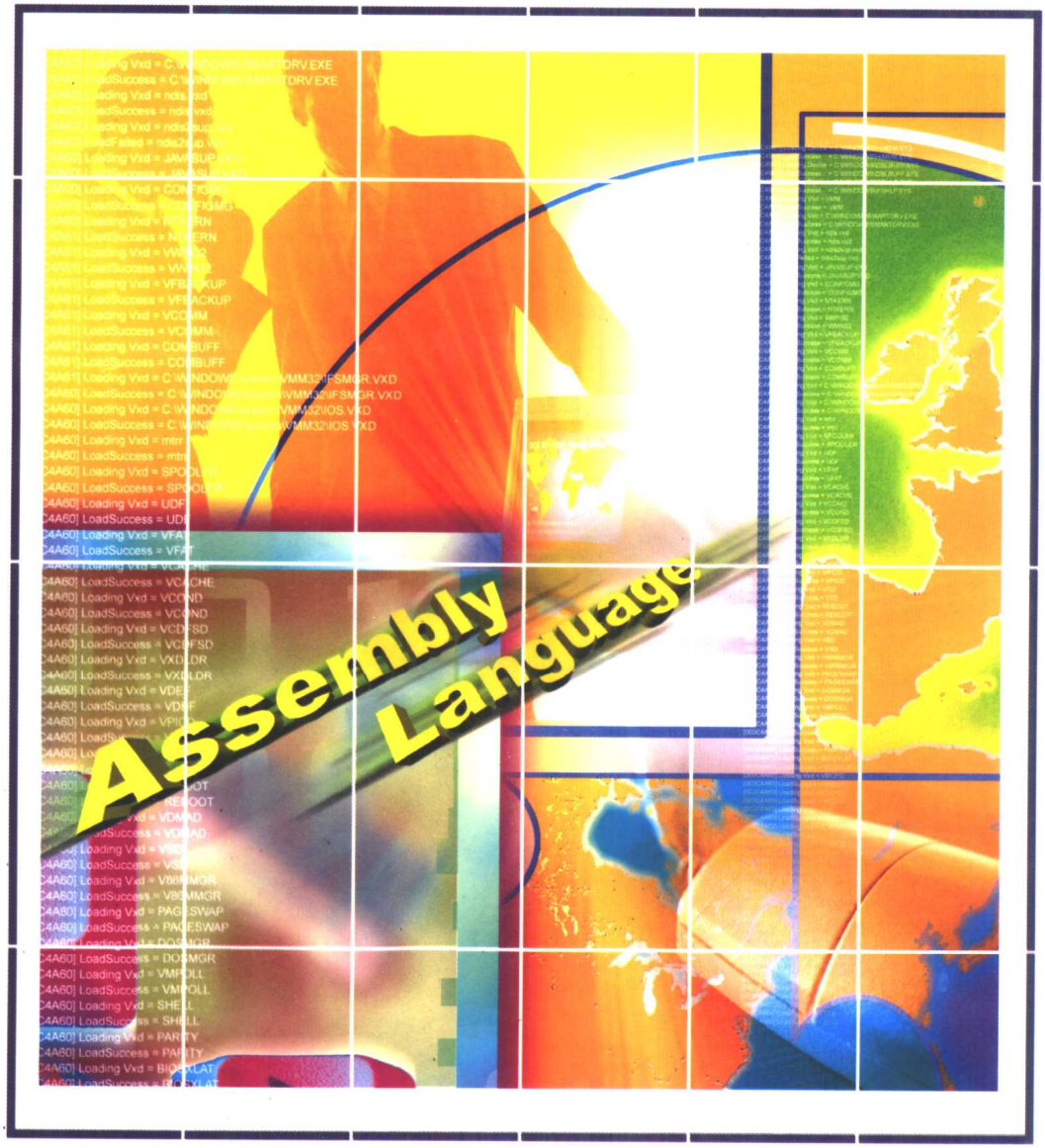
新世纪计算机类本科系列教材



汇编语言实践教程

胡元义　邓亚玲　江　峰　编著

西安电子科技大学出版社
http://www.xduph.com

★新世纪计算机类本科系列教材

汇编语言实践教程

胡元义　邓亚玲　江　峰　编著

西安电子科技大学出版社

内 容 简 介

本书力求以实践的角度来达到学习和掌握汇编语言的目的。本书剖析了汇编语言程序设计学习中的要点与难点，同时给出了大量与实践有关的实验程序。此外，书中还介绍了与实践环节有关的编程与调试工具，并给出了汇编语言应用的有关实例，以开拓学习视野。

本书主要涉及汇编语言程序设计的实践内容，它可配合目前各类汇编语言程序设计教材使用，起到衔接教学与实践的作用。本书还可作为计算机应用的参考资料。

图书在版编目（CIP）数据

汇编语言实践教程 / 胡元义等编著.

—西安：西安电子科技大学出版社，2002.1

新世纪计算机类本科系列教材

ISBN 7-5606-1057-9

Ⅰ.汇… Ⅱ.胡… Ⅲ.汇编语言-程序设计-高等学校-教材 Ⅳ.TP313

中国版本图书馆 CIP 数据核字(2001)第 091139 号

策　　划　陈宇光

责任编辑　杨宗周

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)8227828　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　西安文化彩印厂

版　　次　2002 年 1 月第 1 版　2002 年 1 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 11.875

字　　数　275 千字

印　　数　1～4 000 册

定　　价　14.00 元

ISBN 7-5606-1057-9/TP・0550

如有印装问题可调换

XDUP 1328001-1

前　言

汇编语言是计算机能够提供给用户最快而又最为有效的语言，同时也是利用计算机所有硬件特性并能直接控制硬件的唯一语言。使用汇编语言编写程序能够充分发挥计算机硬件的功能，并具有占用存储空间少、运行速度快以及编程质量高等优点。也正是由于汇编语言具有这样的优点，所以，那些需要对计算机硬件进行控制或者对运行时间和效率有要求的应用软件或系统软件，都是用汇编语言编写的。高级语言为了获得汇编语言的这些优点，通常都增加了调用汇编语言程序的接口或与汇编语言混合编程的功能。

任何事件都有两面性，汇编语言依赖具体机器的这一特性反过来又决定了它的使用不如高级语言直观和方便，而这一点正是汇编语言难以学习和掌握的症结所在。本书针对这一问题，从实际使用汇编语言的角度出发，剖析了有关汇编语言程序设计的要点与难点，并给出了大量的有关实验程序，以提高读者对汇编语言的运用能力。

本书主要涉及汇编语言程序设计的实践内容，它可配合目前各类汇编语言程序设计教材使用，起到衔接教学与实践的作用。全书分为七章：第一章汇编语言概论，简要介绍了汇编语言的概念；第二章汇编语言程序设计基础，介绍了与汇编语言程序设计有关的软、硬件知识；第三章汇编语言高级编程技术，介绍了深入汇编语言程序设计有关的中断与内存驻留技术；第四章汇编语言编程工具软件及使用，介绍了汇编与连接工具软件；第五章汇编语言程序调试及工具软件 DEBUG，讲述了汇编语言程序的调试并介绍了调试汇编语言程序特别是调试较大汇编应用程序必不可少的工具——DEBUG；第六章汇编语言程序设计实践，这是本书的重点，通过这一章的实践练习力图提高读者对汇编语言的运用能力；第七章汇编语言程序设计应用，给出了汇编语言不同应用的实例。

由于编者水平所限，书中难免存在错误和不妥之处，敬请广大读者批评指正。

作　者

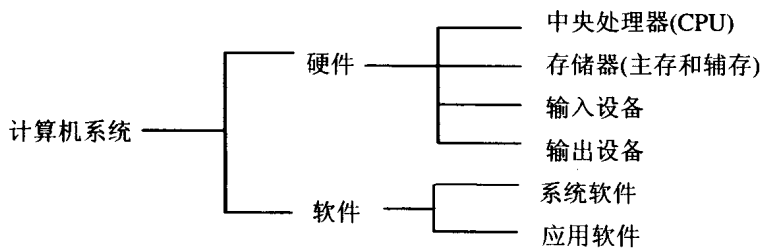
2001 年 11 月 10 日

目　录

第一章

汇编语言概论

计算机是一种能够自动、高速处理数据的工具，是20世纪人类最杰出的科技成就之一。一个完整的计算机系统应包括硬件和软件两大部分，它的组成如下：



硬件是指计算机的机器部分，即我们所见到的物理设备和器件的总称。软件通常指计算机系统中的程序和数据，并按功能分为系统软件和应用软件两类。系统软件指的是计算机系统管理和使用必须配置的那部分软件，如：操作系统、汇编程序、编译程序等。应用软件是指针对某类专门应用的需要而编写或配置的软件。硬件是计算机的物质基础，软件是建立在硬件基础之上的，对硬件功能的扩充与完善，两者缺一不可。没有软件，计算机硬件难以工作；没有硬件，软件的功能也不能实现。

1.1 汇编语言简介

为了处理和解决实际问题，每一种计算机都具有其特定的功能，而这些功能是通过计算机执行一系列相应的操作来实现的。计算机所能执行的每一种操作，称为一条指令。计算机能够执行的全部指令集合，就是该计算机的指令系统。由于计算机硬件的器件特性，决定了计算机本身只能直接接受由0和1编码的二进制指令和数据，这种二进制形式的指令集合就称之为该计算机的机器语言，它是计算机唯一能够直接识别并接受的语言。而计算机程序就是人们编写的指挥和控制计算机完成某一任务的指令序列，这个指令序列就是对要解决问题的各个对象和处理规则的描述。用机器语言编写程序很不方便并容易出错，编写出来的程序也难以调试、阅读和交流。为此，出现了用助记符代替机器语言二进制编码的另外一种语言，这就是汇编语言。

汇编语言是建立在机器语言之上的，因为它是机器语言的符号化形式，所以较机器语言直观，但是计算机并不能够直接识别这种符号化语言，用汇编语言编写的程序必须翻译成机器语言之后才能执行，这种“翻译”是通过专门的软件——汇编程序实现的。因此，

汇编语言程序的执行需要分成两个阶段：汇编阶段和运行阶段。汇编阶段仅把汇编源程序汇编成为机器语言程序，运行阶段才真正执行这个机器语言程序(见图 1 - 1)。

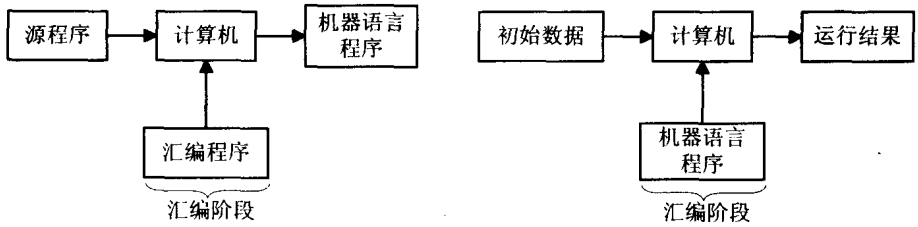


图 1 - 1　汇编过程

早期的计算机其功能相对较弱，指令系统也比较简单，有些计算机甚至没有乘、除法指令，而是通过加法和减法编制乘、除法子程序来实现的。并且，与之对应的汇编语言也比较简单，几乎完全成了机器语言的翻版。这时的汇编语言，一方面显示出了它日益强大的生命力和不可替代的重要性；另一方面又暴露出不少问题。第一，汇编语言的程序环境较差。如汇编语言程序和数据在内存中的分配和管理，都需要编程人员事先予以安排，因此要了解内存的使用情况，无疑给编程人员带来很大的麻烦。第二，汇编语言指令与机器语言指令一一对应的方式使得汇编语言不够灵活。例如，对经常使用的同样一组汇编指令是否能用一条宏指令代替?使得这条宏指令对应几条甚至几十条机器指令，以降低编程强度。第三，汇编语言是面向具体计算机的，这种依赖具体机器的性质决定了各种不同计算机上的汇编语言很难相同。所以，一种计算机上的汇编语言程序难以移到另外一种计算机上运行，这种面向具体机器的特性给汇编语言的学习和交流带来许多不便。

随着计算机的不断发展，计算机系统日趋成熟和完善，指令系统也不断扩充。汇编语言一方面指令随之增加，一方面性能也得到改善， 现今的汇编语言已普遍使用了伪指令和宏定义功能。伪指令与一般的汇编指令不同，它并不产生可执行的机器指令，而只是协助完成诸如数据定义、存储器分配、确定程序开始或结束地址等任务，从而减轻了编程人员的负担。宏定义则是利用计算机指令系统现有的指令按照一定的规则来定义新的指令的，这一新指令称为宏指令。宏指令的功能是根据需要由编程人员自己确定的；宏指令一经定义，就可以像其它汇编指令一样使用，这相当于扩充了计算机的指令系统；不过，一条宏指令不是对应一条机器指令，而是对应几条甚至几十条机器指令。宏指令的出现，增加了汇编语言的灵活性，方便了汇编语言程序的编制。

尽管汇编语言的功能有了很大的改善，但是它依赖具体机器的特性是无法改变的，这也决定了汇编语言没有高级语言那样方便和直观。高级语言已经从具体机器中抽象出来，摆脱了依赖具体机器的问题，用高级语言编制的程序几乎在不改动的情况下能够在其它计算机上运行，而这是汇编语言难以做到的。正是依赖具体机器的特性，决定了汇编语言是计算机能够提供给用户最快而又最为有效的语言，同时它又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。使用汇编语言编写程序可以充分发挥计算机硬件的功能，并且还具有占用存储空间少、运行速度快以及编程质量高等优点。也正是由于汇编语言具有这样的优点，那些需要对计算机硬件进行控制或者对运行时间和效率有要求的应用软件或系统软件，都是用汇编语言编制的。高级语言为了获得汇编语言的这些优点，通常都增

加了调用汇编语言程序的接口或与汇编语言混合编程的能力。

目前，已经有越来越多的人在学习和应用汇编语言，除了用汇编语言编写各种系统软件外，还广泛地用于各种应用软件的开发与研制。汇编语言已经成为计算机不可缺少的、最有影响的几个语言之一。

1.2　汇编语言程序设计过程

汇编语言程序设计过程分为两个阶段：　编写程序阶段和上机调试阶段。

1.2.1　编写程序阶段

1. 分析问题

对需要解决的问题进行全面的了解和分析，明确求解问题的内容和任务。如：解决问题需要做哪些工作，对给定的条件和数据需要进行哪些处理，输入信息的形式和种类的规定，输出的要求是什么以及输出什么样的结果等等。经过详细分析，把一个实际问题转化成为计算机可以进行处理的问题。

2. 确定算法

所谓算法，通俗地讲就是计算机能够实现的有限解题步骤。我们知道，计算机只能进行最基本的算术运算和逻辑运算，要完成较为复杂的运算和操作，就必须采用合适的算法，这样才能正确的求解问题。因此，选择恰当的算法是编制正确解题程序的基础。

算法可以用文字或者流程图进行描述。流程图中较为通用的几种符号见图 1－2。起始和终止框表示程序的开始和结束；执行框表示要完成的某项功能，可以是一条指令或一段程序，无论哪种情况，该框只能有一个入口和一个出口；判断框是用来表示程序在此处需要根据不同的情况形成分支，框内要写明比较的条件，此框只有一个入口和两个出口；连接线是用来连接两个流程框图并指明框图的流向。

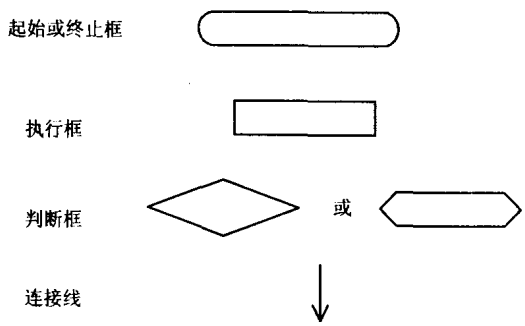


图 1－2　流程图的常用符号

3. 编写程序

采用汇编语言编写程序来实现已经确定的算法。这时应注意以下几个问题：

(1) 详细了解 CPU、指令系统、寻址方式和有关的伪指令；

(2) 注意存储空间和工作单元的合理分配；

(3) 多次使用的程序段可以采用子程序或者宏指令；

(4) 程序中尽可能用标号或变量代替绝对地址和常数。

1.2.2 上机调试阶段

程序编写好后，还要经过上机调试，以便检验程序是否正确并得到程序的运行结果。8086/8088 汇编语言的上机过程是：首先通过编辑程序，如 EDIT 或高级语言集成环境编辑器(如 TC)将手工编写的程序输入到计算机中产生汇编语言的源文件(后缀为.ASM)。由于源文件中的汇编指令不能直接为计算机所识别，因此还要经过汇编程序，如 MASM 把它翻译、转换成用二进制代码表示的目标文件(后缀为.OBJ)。在这个转换过程中，汇编程序对源文件进行两遍扫描来寻找源文件中存在的语法错误，如果确有语法错误存在，则扫描结束后，汇编程序将给出源文件中的错误信息而不产生.OBJ 文件。此时，用户仍需返回到编辑程序对源文件中的错误进行修改，直到汇编程序扫描源文件不再出现错误时，才真正形成.OBJ 文件。.OBJ 文件意为地址浮动的二进制文件，即该文件涉及地址的部分不是“真实”的，因此它还不能直接投入运行。这时，必须使用连接装配程序，如 LINK 将这个.OBJ 文件(或者与库文件及其它.OBJ 文件) 连接装配成为可执行文件(后缀为.EXE)。至此，就可以在 DOS 环境下直接键入文件名来运行这个可执行文件了。我们将在计算机上运行一个汇编语言程序的步骤归纳如下：

(1) 用编辑程序 EDIT 建立一个.ASM 源文件；

(2) 用汇编程序 MASM 把.ASM 文件汇编成.OBJ 文件；

(3) 用连接程序 LINK 把.OBJ 文件连接装配成.EXE 文件；

(4) 在 DOS 环境下运行可执行文件。

汇编语言程序经过编辑、汇编和连接之后就形成了没有语法错误的可执行文件，但是并不能保正它不存在算法上的错误。所以，还需要对可执行文件进行调试，以检查其算法的正确性，即是否能够达到并实现设计时的要求。调试程序 DEBUG 就是专门用于调试汇编语言程序的一种工具，它可以逐段或逐条地调试运行可执行文件并同时给出有关结果信息，以便对该文件的功能进行检查，如果发现算法有错，就要回到编制程序阶段对程序重新进行设计。汇编语言上机的全过程可示意为图 1－3。

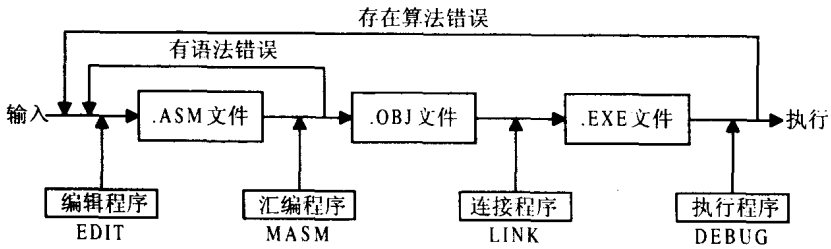


图 1－3　汇编语言上机过程示意

1.3 汇编语言编程工具与环境

我们从汇编语言程序的设计过程可以了解到，用于汇编语言的编程工具(系统软件)主要有编辑程序、汇编程序、连接程序和调试程序。

1. **文本编辑程序**(Text Editor)

在文本编辑程序的帮助下，用户可以通过键盘输入汇编语言源程序(当然也可以输入其它语言的源程序)，对源程序进行编辑、修改，并把源程序作为文件保存在磁盘上。早期的文本编辑程序是行编辑程序 EDLIN，由于 EDLIN 操作复杂、直观性差，现在已经较少使用。目前使用较多的编辑程序是编辑器 EDIT 以及高级语言中所使用的集成环境编辑器等。它们共同的特点是易于编辑、操作灵活、直观性强。

2. **汇编程序**(Assembler)

汇编程序用于将用户编制的源程序文件翻译成机器语言文件。汇编程序的主要功能是：

(1) 检查源程序文件；

(2) 检测源程序中的语法错误并给出错误信息；

(3) 产生源程序的目标文件(.OBJ)；若需要的话，也同时给出列表文件(后缀为.LST 的汇编语言和机器语言对照文件)；

(4) 展开宏指令。

目前 IBM PC 系列机使用最多的汇编程序有两种：一种是 Microsoft 公司研制的宏汇编程序 MASM；一种是 Borland 公司研制的 Turbo 系列汇编程序 TASM。MASM 和 TASM 两种汇编程序基本相同。

3. **连接程序**(Linker)

汇编语言程序经过汇编后生成目标程序，但是这个目标程序中的地址是“浮动”的，它只是一种逻辑地址，所以称为浮动二进制文件(.OBJ)。连接程序的功能就是将浮动二进制文件中的逻辑地址转变成能够在计算机上直接运行的物理地址，即浮动二进制文件只有经过连接程序的连接装配后才能成为可以在计算机上直接执行的文件(.EXE)。注意，不同种类、版本的汇编程序有着与其相配的连接程序，与 MASM 配合使用的是连接程序 LINK，而与 TASM 配合使用的是连接程序 TLINK。

4. **调试程序**(Debuger)

调试程序 DEBUG 用于在 DOS 环境下调试运行一个可执行文件(.EXE 或.COM)。虽然高级语言程序经过编译生成可执行文件后也可以用 DEBUG 程序进行调试，但由于高级语言的编译特点，这种调试极少使用。因此，DEBUG 程序主要用于汇编语言程序的调试。DEBUG 程序除了调试、运行可执行文件外，还具有在 DEBUG 环境下直接建立、修改、运行一个小汇编语言源程序(不能使用伪指令、宏指令及符号地址)。初学者往往认识不到 DEBUG 程序的重要性， 在程序通过汇编、连接生成了可执行文件后，总是寄希望于程序一次运行成功，这往往是不现实的。我们知道，汇编和连接过程只能查出源程序中的语法错误以及生成可执行文件，而根本无法检查程序的算法是否有错或者不完善。只有通过 DEBUG 程序调试可执行文件，逐段甚至是逐条指令地调试执行，从中观察是否达到预期的功能或得出预测的结果，特别是转移指令是否按设想进行转向等，从而发现程序在设计上的缺陷和错误。

第二章

汇编语言程序设计基础

2.1 概　　述

计算机系统的组成包括硬件(Hardware)和软件(Software)两个部分。硬件是组成计算机系统的物理实体，微型计算机系统的硬件包括主机箱(内有主机板以及软、硬盘驱动器和电源等)、键盘、显示器和打印机等。软件则是为了运行、管理和维护计算机而编制的各种程序的总和。计算机系统中的硬件和软件互为基础、互相支持，它们共同完成计算机的各种功能。

典型的计算机结构可用图 2 - 1 表示，即由中央处理器 CPU(Central Processing Unit)、存储器(Memory)和输入/输出(I/O)子系统三个主要组成部分构成，并用系统总线将它们连接在一起。

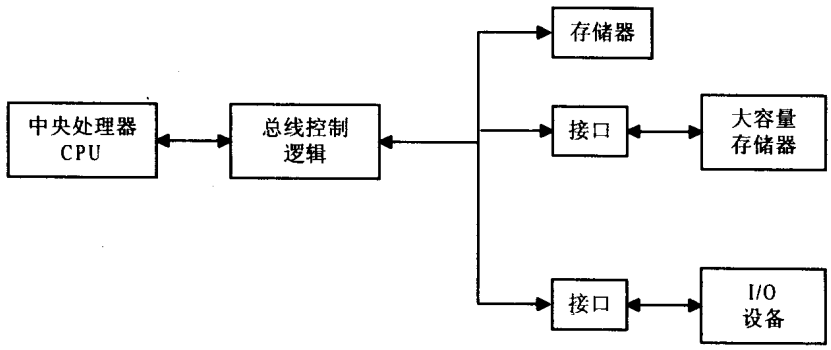


图 2 - 1　计算机系统结构

中央处理器包括运算器和控制器两部分。运算器执行所有的算术和逻辑运算指令；控制器则负责全机的控制工作——把指令逐条从内存中取出，经译码分析后向全机发出取数、执行、存数等控制命令信号，正确地完成程序所要求的功能。

存储器是计算机的记忆部件。编写的求解问题程序(指令序列)、程序运行中要处理的数据、信息以及中间结果都存放在存储器中。存储器分为两类：内存储器和外存储器。机器内部的存储器称为内存储器(简称内存)，内存的容量有限但速度较快，只有在内存的程序才能够执行。大容量存储器称为外存储器(简称外存)，如磁盘、磁带、光盘等。外存的容量比内存大得多，但存取信息的速度较慢。所以程序通常是存放在外存中，只有在运行时才从外存调入内存并由中央处理器控制执行。

I/O 子系统通常包括 I/O 设备和大容量存储器这两类外部设备。I/O 设备是指负责与计算机外部进行通信的输入、输出设备，如键盘、显示器、打印机等各类外部设备。大容量存储器即存储器中提到的外存。

系统总线把 CPU、存储器和 I/O 设备连接起来进行各部分信息的传送。系统总线包括数据线、地址线和控制线三类。数据线用来传送数据信息；地址线指出数据信息来源的地址或传送的目的地址；控制线规定总线的动作，如方向等。系统总线是在总线控制逻辑的指挥下进行工作的。

中央处理器(CPU)中有一个以全加器为核心的算术逻辑单元 ALU。在控制信号的控制下，ALU 可以完成两个数的相加、相减、逻辑加以及对一个数取反、取负等算术或逻辑运算。除了这些基本算术逻辑运算外，CPU 还可以完成数据在寄存器之间，或在寄存器与存储器之间，或在寄存器与外部设备之间的传送操作。因此，计算机能够直接完成两数的相加、相减、逻辑乘、逻辑或以及数的取反、取负、传输等多种基本运算或操作，每一种基本运算或操作称为一条指令。一个 CPU 所能执行的全部指令集合就是这个 CPU 的指令系统(Instruction System)。

一个 CPU 的指令系统是在设计 CPU 时确定的，它决定了这个 CPU 功能的强弱。指令在 CPU 内是以二进制代码形式出现并实施操作的。任何一条指令都有着和其它指令不同的代码表示。我们按完成运算功能的要求把指令排列起来，这就是程序。指令是构成程序的基本单位，编写程序时不能使用指令系统中并不存在的“指令”。

直接以指令的二进制代码进行编程称为机器语言编程。为了便于记忆和交流，机器语言每条指令的二进制代码可以用一串字母或符号来表示。这种用助记符表示的指令称之为汇编语言。用汇编语言编写的程序仍需变换成机器语言程序才能够在计算机内执行(计算机内的任何信息都是以二进制代码表示的，机器码指令程序只有调入内存后方可执行)。计算机的工作就是运行程序，即逐条地从内存中取出程序指令并执行该指令所规定的操作。

计算机硬件本身的功能是有限的，软件是硬件功能的扩展，即编制具有特定功能的程序在计算机上运行时就使计算机具有了新的功能。从计算机功能的角度来看，硬件和软件是相同的。对微型计算机来说，没有软件是无法工作的。所以，软件是组成微型计算机系统不可缺少的组成部分。

计算机软件分为系统软件和用户软件两大类。系统软件是由计算机生产厂家或计算机软件公司提供的。系统软件面向机器本身，其算法和功能不依赖于特定的用户，它的主要任务是使机器硬件的功能得以充分的利用，支持用户软件的运行并提供恰当的服务。用户软件则是用户为解决自己特定的问题而自行或委托他人编制的各种程序。

操作系统(Operating System)是最基本的系统软件，它负责管理计算机系统的硬、软件资源，为用户提供方便、可靠的人机环境。操作系统是人与计算机交往的接口或界面。

2.2　PC 微机系统软硬件环境

2.2.1　8086/8088 中央处理机

1．8086/8088 CPU 结构

8086/8088 CPU 有 20 条地址线，直接寻址能力达 1 MB。在结构上 8086/8088 CPU 采用

了流水指令队列以及地址计算和算术逻辑运算分布处理的先进技术。8086/8088 CPU 由总线接口部件 BIU(Bus Interface Unit)和执行部件 EU(Execution Unit)组成，其功能框如图 2－2 所示。

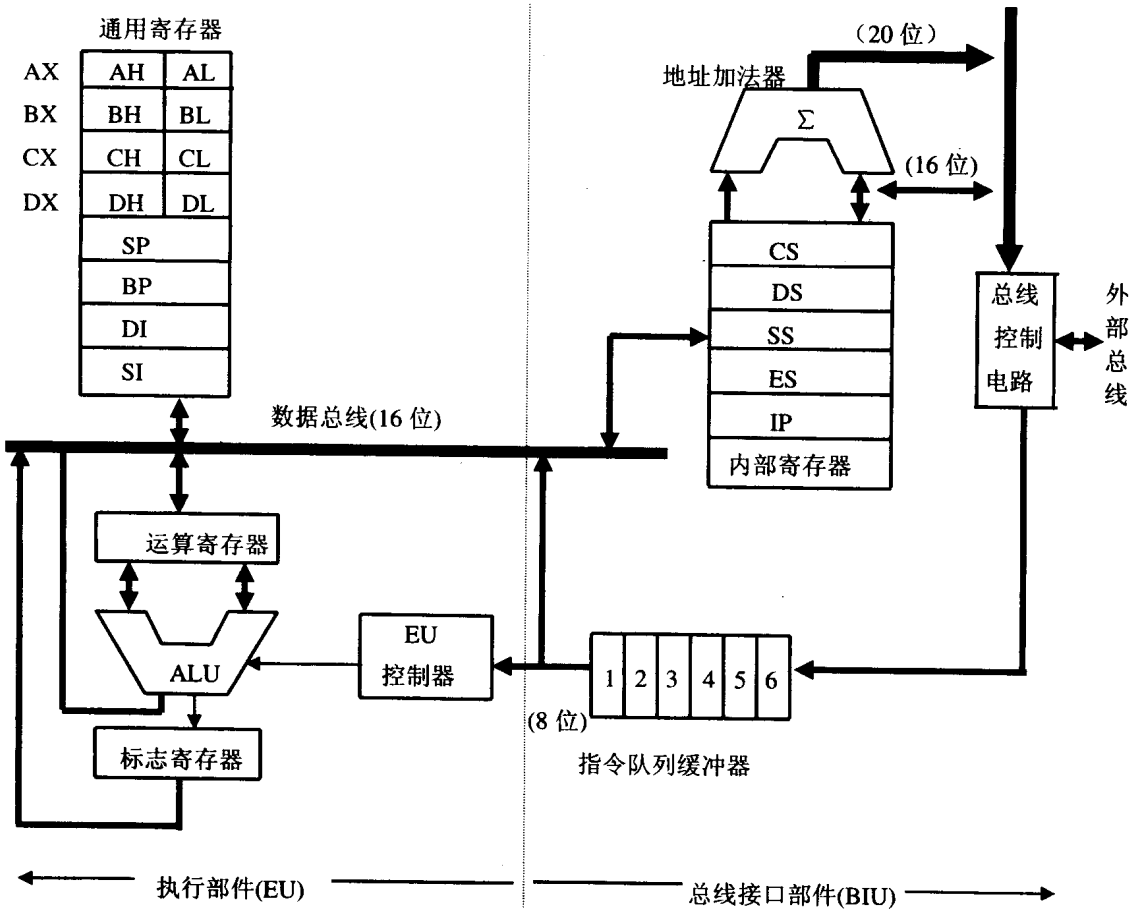


图 2－2　8086/8088 CPU 组成框图

执行部件 EU 由算术逻辑部件 ALU、通用寄存器组和标志寄存器(PSW)组成。EU 解释、执行来自指令队列中的指令，完成算术逻辑运算动作。EU 产生数据在内存中的相对地址，接受来自内部总线的数据。

总线接口部件 BIU 包括段寄存器(4 个)、指令指针寄存器、6B 指令队列、地址计算逻辑和执行机构等。BIU 专门用于计算实际内存地址，不仅把来自 EU 的数据相对地址转换为实际地址，还负责提供当前取指令的内存地址，即取指令时，从存储器指定地址取出指令送入指令队列排队；执行指令时，根据 EU 命令对指定存储器单元或 I/O 端口存取数据。

CPU 与内存之间的数据传送是经过总线接口部件 BIU 的。从内存中取出的指令存放在指令队列中，执行部件控制逻辑从这个队列中取出指令并把它送到执行部件去解释和执行。在指令的解释执行过程中，总线接口部件就继续由内存取出指令装满指令字节队列，这样就保证在 CPU 结束当前正在执行的指令时有另一条指令在等待执行(除了转移类指令之外)。这种流水线的处理方式提高了 8086/8088 CPU 执行指令的效率。

2．8086/8088 的寄存器

8086/8088 CPU 内部具有 14 个 16 位寄存器，这些寄存器在计算机中起着重要的作用，每一个寄存器相当于运算器中的一个存储单元，但它的存取速度比存储器要快得多。寄存器用来存放计算过程中所需要或所得到的各种信息，包括操作数地址、操作数及运算的中间结果等。8086/8088 CPU 的寄存器结构见图 2－3。

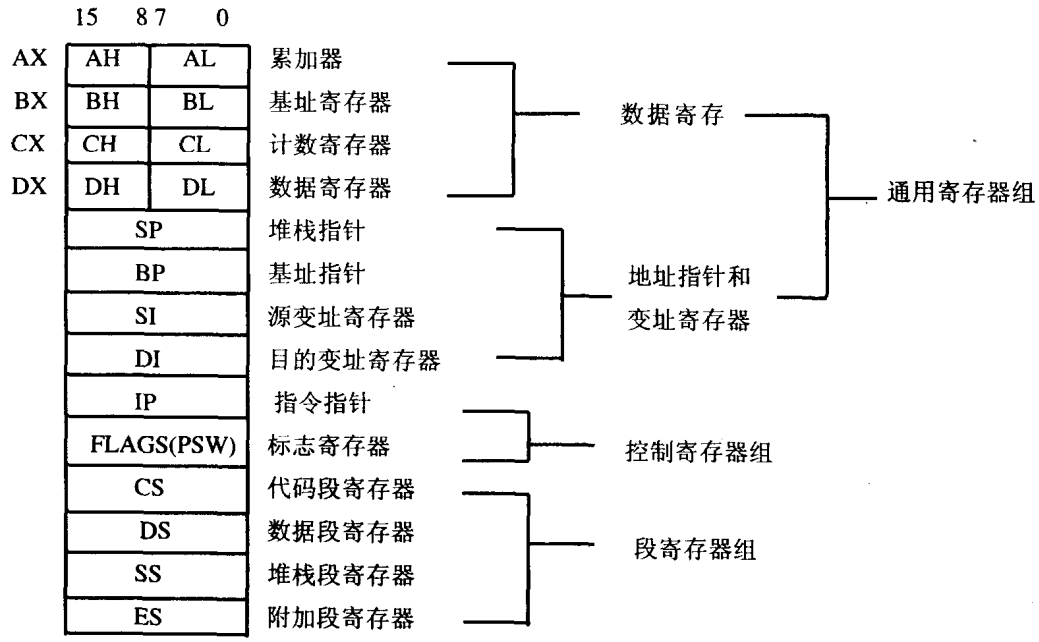


图 2－3　8086/8088 CPU 的寄存器组

1) 通用寄存器组

(1) 数据寄存器。数据寄存器包括 AX、BX、CX 和 DX，用于暂时存放计算过程中所用到的操作数、结果和其它信息。这四个寄存器的每一个都可以以字(16 位)的形式访问，也可以以字节(8 位)的形式访问。当以字节方式访问时，则每一个寄存器分为高 8 位和低 8 位两个 8 位寄存器，如 AX 寄存器可以分为高 8 位的 AH 寄存器和低 8 位的 AL 寄存器。16 位数据寄存器可以用来存放数据和地址，而 8 位寄存器只能存放数据。

(2) 地址指针和变址寄存器。地址指针和变址寄存器包括 SP、BP、SI 和 DI。它们可以像通用寄存器一样，在运算过程中存放操作数，但只能以字方式访问。这组寄存器更为常用的是在存储器操作数寻址时形成 20 位物理地址码的一部分。由于访问存储器的地址码是由段地址(存放在段寄存器中)和段内的偏移地址两部分构成的，因此上述 4 个寄存器只用于存放段内偏移地址的全部或部分。在任何情况下，它们都无法单独形成访问存储器的 20 位物理地址。其中：

SP 为堆栈指针寄存器，用于存放堆栈操作(压入或弹出)地址中的段内偏移地址，其段地址由段寄存器 SS 提供。

BP 为基址指针寄存器，在某些间接寻址方式中，BP 用来存放段内偏移地址的一部分。应注意的是：若无特别说明，凡含有 BP 的寻址，其段地址由段寄存器 SS 提供，而即该寻址方式是对堆栈区的存储单元进行的。

SI 和 DI 为变址寄存器。在某些间接寻址方式中，SI 和 DI 用于存放段内偏移地址一部分或全部。在字符串操作指令中，SI 用作源变址寄存器，DI 用作目的变址寄存器。这种情况下它们具有自动增量或自动减量的功能，并且此时 SI 和 DS 联用、DI 和 ES 联用来确定数据段和附加段中存储单元的物理地址。

(3) 段寄存器。由于访问存储器的地址由段地址和段内偏移地址两部分组成，段寄存器用来存放段地址。段寄存器共有 4 个，CPU 可以通过这 4 个段寄存器访问存储器中的 4 个不同的段，且每个段地址空间为 64 KB。4 个段寄存器分别是：

代码段寄存器 CS：用于存放当前执行程序所在段的段地址。下一条要执行指令的代码存放地址是由 CS 内容左移 4 位再加上指令指针寄存器 IP 的内容来确定的。

数据段寄存器 DS：存放当前数据段的段地址。数据段通常用来存放程序中使用的数据和变量。DS 的内容左移 4 位再加上在存储器寻址方式下计算出来的偏移地址，就是访问数据段中存储单元的物理地址。

堆栈段寄存器 SS：用于存放当前堆栈段的段地址。堆栈是一种数据结构，它在存储器中开辟了一个特殊的存储区，并按后进先出的方式访问该存储区。堆栈段主要用于在调用子程序时，保存返回主程序的地址和那些在进入子程序后可能改变内容的寄存器值。堆栈操作(压入或弹出)的物理地址由 SS 的内容左移 4 位加上 SP 的内容确定。

附加段寄存器 ES：用于存放附加数据段的段地址。ES 通常是在字符串操作中用来指定目的字符串的段地址，此时寄存器 DI 则用来存放目的字符串的偏移地址。

DS、ES、SS 都需要在程序中设置初值，如果 DS 和 ES 设置的初值相同，则 DS 与 ES 所指的为同一个数据段。CS 值的设置将影响到程序是否能正确执行，因此只能通过伪指令 ASSUME 并最终由系统确定，而不能通过其它指令来更改 CS 的值。

2) 控制寄存器组

(1) 指令指针寄存器。指令指针寄存器 IP 用来存放代码段中指令的偏移地址。在程序运行中，它始终指向下一条指令的首地址，并与 CS 一起确定这条指令的物理地址。IP 内容的修改是自动完成的，即控制器取得一条要执行的指令时就马上修改 IP 的内容，使它指向下一条指令的首地址。因此，IP 寄存器是用来控制指令序列执行流程的，用户程序中不能使用 IP 寄存器。IP 的内容在执行转移指令、过程调用指令和返回指令时将被改变。

(2) 标志寄存器。标志寄存器 FLAGS(又称程序状态字寄存器 PSW)由条件码标志和控制标志构成(见图 2 - 4)。

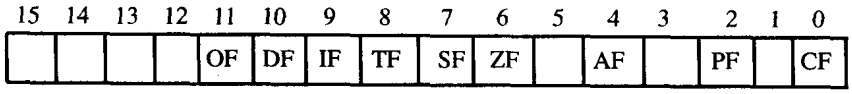


图 2 - 4　标志寄存器各标志位

其中，条件码标志是反映刚执行的指令涉及 ALU 操作结果的状态信息。由于这些状态信息往往作为后继的条件转移指令中的转移控制条件，故称为条件码。它包括以下 6 位：

OF：溢出标志。在运算过程中，如果操作数超出了机器能够表示的范围称为溢出，此时 OF 位置 1，否则置 0。

SF：符号标志。记录运算结果的符号，结果为负时置 1，否则置 0 。

ZF：零标志。运算结果为 0 时，ZF 位置 1，否则置 0。