计算机类本科规划教材
教育部－微软精品课程教学成果

汇编语言简明教程

◆ 钱晓捷　编著

- 基于80x86系列处理器的个人计算机
- 基于DOS/Windows操作系统
- MASM32和Visual C++集成化开发系统

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

计算机类本科规划教材

汇编语言简明教程

钱晓捷　编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书基于MS-DOS环境和MASM汇编程序讲解16位8086处理器基本指令及其汇编语言程序设计，内容包括汇编语言基础、处理器基本指令和汇编语言常用伪指令以及顺序、分支、循环、子程序结构，在此基础上，逐步展开32位指令编程、Windows编程、与C++语言的混合编程，并介绍浮点、多媒体及64位指令。

本书主要面向普通高等院校的计算机以及电子、通信和自控等电类专业，可用做“汇编语言程序设计”课程的教材或参考书。本书具有“重点明确、突出实践、深入浅出”等特色，使其还能很好地适合远程教育、成人教育、自学考试等本科或专科（含高职）学生，也适合计算机应用开发人员、希望深入学习汇编语言的普通读者作为入门教材。

图书在版编目（CIP）数据

汇编语言简明教程 / 钱晓捷编著. —北京：电子工业出版社，2013.5
计算机类本科规划教材
ISBN 978-7-121-20184-4

I. ①汇… II.①钱… 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（2013）第075970号

策划编辑：章海涛
责任编辑：章海涛　　特约编辑：何　雄
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编　100036
开　　本：787×1 092　1/16　印张：18.75　字数：480千字
印　　次：2013年5月第1次印刷
定　　价：39.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

汇编语言使用处理器指令编程，是一种底层程序设计语言。本书使用 80x86 系列处理器的个人计算机、基于 DOS/Windows 操作系统和 MASM 汇编程序，学习汇编语言程序设计，形成如下表所示章节。表中内容提要亦是教学重点，体现了本书的逻辑主线。

目　录	内容提要（教学重点）
第 1 章 汇编语言基础	在了解个人计算机软硬件系统的基础上，熟悉通用寄存器和存储器组织，掌握汇编语言的语句格式、程序框架和开发方法
第 2 章 数据表示和寻址	在理解计算机如何表达数据的基础上，熟悉汇编语言中如何使用常量和变量，掌握处理器指令寻址数据的方式
第 3 章 通用数据处理指令	熟悉处理器数据传送、算术运算、逻辑运算和移位操作等基本指令，通过程序片段掌握指令功能和编程应用
第 4 章 程序结构	以顺序、分支和循环程序结构为主线，结合数值运算、数组处理等示例程序，掌握控制转移指令以及编写基本程序结构的方法
第 5 章 模块化程序设计	以子程序结构为主体，围绕二进制、十六进制和十进制数码转换实现键盘输入和显示输出，熟悉子程序、文件包含、宏汇编等各种多模块编程的思想
第 6 章 32 位指令及其编程	了解 80x86 系列处理器发展概况，理解 32 位处理器的运行环境，熟悉新增的 32 位特色指令，掌握 DOS 环境使用 32 位指令的特点
第 7 章 Windows 编程	熟悉汇编语言调用 API 函数的方法，掌握控制台输入输出函数。了解 MASM 的高级特性，以及 Windows 图形窗口程序的编写
第 8 章 与 Visual C++的混合编程	了解嵌入汇编和模块连接进行混合编程方法，理解堆栈帧的作用，熟悉汇编语言调用高级语言函数和开发调试过程
第 9 章 浮点、多媒体及 64 位指令	理解浮点数据格式、多媒体数据格式及 64 位编程环境的特点，了解浮点、多媒体和 64 位等特色指令

汇编语言能够直接有效地控制硬件，便于编写代码量小、运行速度快的高效程序，在计算机及相关专业的教学和许多应用场合中具有不可或缺的作用。“汇编语言程序设计”也一直作为专业基础的必修教学内容。然而，随着计算机技术发展和高等教育教学改革，传统的教学理念、手段和方法都需要进步。基于“汇编语言在底层但不低级”的教学理念和“汇编、汇编一定会编”的精神，作者总结十多年课程主讲和教材编写的实践经验编写本书，并形成如下特色。

1. 删繁就简、重点明确的教学内容

汇编语言的教学目的是从软件角度理解计算机硬件工作原理，为相关课程提供基础知识，同时全面认识程序设计语言，体会底层编程特点，以便更好地应用高级语言。所以，本书没有将汇编语言作为一个通用程序设计语言那样展开，没有详尽论述所有处理器指令、全部汇编伪指令，而是选择处理器通用的基本指令和反映汇编语言特色的常用伪指令；没有引出复杂的程序格式、详述程序框架的每个细节，而是侧重指令功能和编程思想、特别介绍相关硬件工作原理。这样一方面能够降低教学难度、易于学生掌握，另一方面使得教学内容更加实用、便于学生实际应用。

2. 贯穿始终、突出实践的教学过程

本书将上机实践贯穿始终，教学内容融入了约 80 个例题程序和约 70 个习题程序。第 1 章在必要的寄存器和存储器知识后，就引出汇编语言开发环境，介绍汇编语言的语句格式、源程序框架和开发方法，并编写具有显示结果的程序。第 2 章结合数据编码、引出汇编语言的常量定义和变量应用，利用汇编过程生成的列表文件，自然掌握常用伪指令和数据寻址。第 3 章分类学习处理器基本指令，特别利用 DEBUG 调试程序直观体会指令功能，同时逐渐编写特定要求的程序片段。第 4 章以程序结构为主线，编写数值运算、数组处理、字符串操作等程序，同时使用 DEBUG 调试程序进行可执行程序的调试。第 5 章围绕二进制、十六进制和十进制数码转换子程序，掌握模块化编程方法。第 6 章 32 位编程、第 7 章 Windows 编程、第 8 章混合编程、第 9 章浮点指令，从不同方面强化程序设计能力。

为方便教学和初学者入门，本书构建了一个简单易用的 MASM 开发环境（详见本书第 1 章），无须安装和配置，直接复制就可使用。第 2 章到第 5 章还运用 DEBUG 调试程序，学习程序调试方法。第 7 章和第 8 章分别引出了 MASM32 和 Visual C++集成化开发系统，说明使用汇编语言开发 32 位 Windows 应用程序和混合编程的方法。

3. 循序渐进、深入浅出的教学原则

为了便于学生理解和掌握，也便于教师实施教学，本书以“循序渐进、难点分散、前后对照”为原则，做到“语言浅显、描述详尽、图表准确”，内容编排具有诸多特色。

例如，本书从简单的 16 位 8086 处理器入手，通过前 5 章掌握基本的，也是最核心的使用指令进行编程的方法，然后从第 6 章开始才引出相对复杂的 32 位处理器指令。高级编程技术如宏汇编、条件汇编等随着模块化编程引入，结构变量和 MASM 高级特性等也是随着 Windows 编程、结合应用展开，直到混合编程、浮点和多媒体指令等相对较为艰深的教学内容。

再如，将处理器指令和汇编伪指令分散于各章教学内容之中，引出列表文件暂时避开调试程序，配合屏幕截图详述 DEBUG 调试程序的操作过程。完整的程序尽可能具有交互性和趣味性，适当对比高级语言，并揭示底层工作原理、理解问题的来龙去脉。每章都编制丰富的习题，满足课外练习、上机任务和试题组织。

本书由郑州大学信息工程学院钱晓捷老师主编，同时衷心感谢关国利、张青、张行进、穆玲玲等老师多年来的合作和帮助，热诚期待广大师生和读者的反馈。

“大学微机技术系列课程教学辅助网站”（http://www2.zzu.edu.cn/qwfw）提供本书的教学课件（电子教案）、配套软件（含示例源程序文件）等辅助资源。

作　者

目　录

第 1 章　汇编语言基础

程序设计语言是人与计算机沟通的语言，程序员利用它进行软件开发。通常人们习惯使用类似自然语言的高级程序设计语言，例如 C、C++或者 Basic、Java 语言等。高级语言需要翻译为计算机能够识别的指令，即机器语言，才能被计算机直接执行。机器语言是一串由 0 和 1 组成的二进制代码，对程序员来说艰涩难懂，被称为低级语言。将二进制代码的指令和数据用便于记忆的符号——助记符——表示就形成汇编语言（Assembly），所以汇编语言是一种面向机器的低级程序设计语言，或称为低层语言。

本章首先理解汇编语言的硬件基础：个人计算机、处理器的寄存器和存储器组织，然后学习汇编语言的程序格式，接着编写出第一个汇编语言程序，最后掌握基于 DOS 操作系统和微软 MASM 汇编程序的程序开发方法。

1.1　个人计算机系统概述

计算机系统包括硬件和软件两大部分。硬件（Hardware）是指构成计算机的实在的物理设备，是我们看得见、摸得着的物体，就像人的躯体。软件（Software）一般是指在计算机上运行的程序（广义的软件还包括由计算机管理的数据以及有关的文档资料），是指示计算机工作的命令，就像人的思想。

微型计算机（Microcomputer），简称微机，是最常使用的一类计算机，在科学计算、信息管理、自动控制、人工智能等领域有着广泛的应用。工作、学习和娱乐中使用的个人计算机（PC）是我们最熟悉、也是最典型的通用微型计算机。

1.1.1　计算机的硬件

源于冯·诺依曼设计思想的计算机由 5 大部件组成：控制器、运算器、存储器、输入设备和输出设备。控制器是整个计算机的控制核心；运算器是对数据进行运算处理的部件；存储器是用来存放数据和程序的部件；输入设备将数据和程序变换成计算机内部所能识别和接受的信息方式，并把它们送入存储器中；输出设备将计算机处理的结果以人们能接受的或其他机器能接受的形式送出。

现代计算机在很多方面都对冯·诺依曼计算机结构进行了改进，5 大部件演变为 3 个硬件子系统：处理器、存储系统和输入输出系统。运算器和控制器被制作在一块大规模集成电路芯片上，称为处理器（Processor），也常被称为中央处理单元 CPU（Central Processing Unit）。传统的存储器也发展成为存储系统，由寄存器、高速缓冲存储器、主存储器及辅助存储器构成。处理器和存储系统在信息处理中起主要作用，是计算机硬件的主体部分，通常被称为“主机”。输入设备和输出设备统称为外部设备，简称为外设或 I/O 设备；输入输出系统的主体是外部设备，但还包括外设与主机之间相互连接的 I/O 接口电路。

为简化各个部件的相互连接，现代计算机广泛应用总线结构，参见图 1-1。采用总线连接系统中各个功能部件使得计算机系统具有了组合灵活、扩展方便的特点。

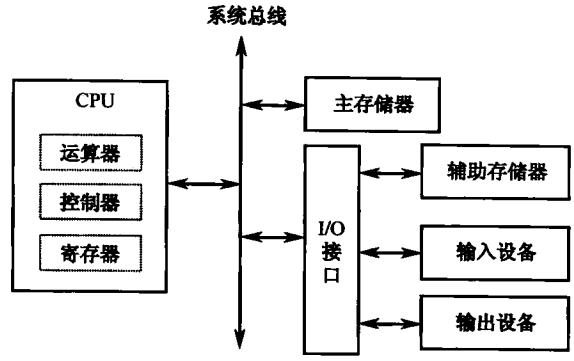

图 1-1　计算机系统的硬件组成

1．处理器

处理器是计算机的运算和控制核心，微机中可被称为微处理器（Microprocessor）。现代通用微处理器功能非常强大，人们已经习惯称之为处理器或者 CPU。

（微）处理器芯片内集成了控制器、运算器和若干高速存储单元（即寄存器）。高性能处理器内部非常复杂，例如运算器中不仅有基本的整数运算器、还有浮点处理单元甚至多媒体数据运算单元，控制器还会包括存储管理单元、代码保护机制等，为提高存储器的性能还会集成高速缓冲存储器。处理器及其支持电路构成了计算机系统的处理和控制中心，对系统的各个部件进行统一的协调和控制。

PC 采用美国英特尔（Intel）公司的 80x86 系列处理器或与其兼容的处理器，例如常用的奔腾系列处理器或者酷睿系列多核处理器。之所以称之为 Intel 80x86 系列处理器，是因为它们都源于 16 位结构的 Intel 8086 处理器，而 8086 具有的所有指令，即指令系统是整个 Intel 80x86 系列处理器的基本指令集。

本书前 5 章将介绍基于 8086 处理器的 16 位常用指令，从第 6 章开始介绍 32 位指令。

2．存储器

存储器（Memory）是计算机的记忆部件，存放程序和数据。存储系统由处理器内部的寄存器（Register）、高速缓冲存储器（Cache）、主板上的主存储器和以外设形式出现的辅助存储器构成。

按所起作用，存储器可分为主存储器和辅助存储器。主存储器（简称主存或内存）由半导体存储器芯片组成，安装在机器内部的电路板上，相对辅助存储器来说速度快，但容量小：造价高，主要用来存放当前正在运行的程序和等待处理的数据。辅助存储器（简称辅存或外存）主要由磁盘、光盘存储器等构成，以外设的形式安装在机器上，相对主存储器来说造价低、容量大、信息可长期保存，但速度慢，主要用来长久保存程序和数据。一般来说，程序和数据以文件形式保存在辅存上，只有使用它们时才读入主存。

按读写功能，存储器可分为可读可写存储器和只读存储器 ROM（Read Only Memory）。半导体存储器具有按指定位置访问，即随机存取的特点，所以可读可写的半导体存储器常被称为 RAM（Random Access Memory）。构成主存既需要 RAM，也需要 ROM，但需要注意的是，存放在 RAM 芯片上的信息断电后将会丢失，而 ROM 芯片中的信息则可在断电后保存。通常作为辅存的磁盘存储器和 CD-ROM 光盘都可以在断电后长期保存信息，它们二者的不

同在于，CD-ROM 光盘是只读的，而作为辅存的磁盘存储器是可读可写的。不过，由于读写时涉及磁头或光头的移动、磁盘或光盘的旋转，它们的存取性能低于半导体存储器。

个人计算机的主存由半导体存储芯片 RAM 和 ROM 构成。在 16 位 PC 系列机时代，RAM 容量不过是 64KB 或 1MB。32 位 PC 的 RAM 容量从最初的 4MB，逐渐发展直到 2010 年的 2GB 或 4GB。由于大量应用程序都需要 RAM 主存空间，因此 PC 的主存主要由 RAM 构成，俗称主存条（内存条）。

个人计算机的 ROM 部分主要是固化的 ROM-BIOS。BIOS（Basic Input/Output System）表示“基本输入输出系统”，是 PC 软件系统最底层的程序。它由诸多子程序组成，主要用于驱动和管理诸如键盘、显示器、打印机、磁盘、时钟、串行通信接口等基本的输入输出设备。操作系统通过对 BIOS 的调用驱动各硬件设备，用户也可以在应用程序中调用 BIOS 中的许多功能。ROM 空间还包含机器复位后初始化系统的程序，它将操作系统引导到 RAM 存储器中执行。

3. 外部设备

外部设备是指计算机上配备的输入（Input）设备和输出（Output）设备，也称 I/O 设备或外围设备，简称外设（Peripheral），其作用是让用户与计算机实现交互。

个人计算机上配置的标准输入设备是键盘、标准输出设备是显示器，二者又合称为控制台（Console）。个人计算机还可使用鼠标、打印机等 I/O 设备。作为外部存储器驱动装置的磁盘驱动器，既是输出设备，又是输入设备。

由于各种外设的工作速度、驱动方法差别很大，无法与处理器直接匹配，所以不可能将它们直接连接到主机。这里就需要有一个 I/O 接口来充当外设和主机之间的桥梁，通过该接口电路来完成信号变换、数据缓冲、联络控制等工作。在个人计算机中，较复杂的 I/O 接口电路通常制成独立的电路板，也常被称为接口卡（Card），例如显示卡，使用时需要将其插在主板的总线插槽上。

4. 系统总线

总线（Bus）是用于多个部件相互连接、传递信息的公共通道，物理上就是一组公用导线。例如，处理器芯片的对外引脚（Pin）常被称为处理器总线。这里的系统总线（System Bus）是指计算机系统中主要的总线，例如处理器与存储器和 I/O 设备进行信息交换的公共通道。

16 位 PC 采用 16 位工业标准结构 ISA（Industry Standard Architecture）系统总线连接各个功能部件。32 位 PC 上使用外设部件互连 PCI（Peripheral Component Interconnect）总线连接 I/O 接口卡。系统总线除了作为主机板上处理器、主存和 I/O 接口的公共通道外，主板上还设置有许多系统总线插槽，主要用于插接 I/O 接口电路以扩充系统连接的外设，故也被称作 I/O 通道。

对汇编语言程序员来说，处理器、存储器和外部设备依次被抽象为寄存器、存储器地址和输入输出地址，因为编程过程中将只能通过寄存器和地址实现处理器控制、存储器和外设的数据存取及处理等操作。

1.1.2 计算机的软件

完整的计算机系统包括硬件和软件，软件又分为系统软件和应用软件。

1. 系统软件

系统软件是为了方便使用、维护和管理计算机系统的程序及其文档，通常由计算机生产厂商或者软件公司提供，其中最重要的是操作系统。

操作系统 OS（Operating System）管理着系统的软硬件资源，为用户提供使用机器的交互界面，为程序员使用资源提供可供调用的驱动程序，为其他程序构建稳定的运行平台。

程序员采用某种程序设计语言编写源程序，利用语言翻译程序将源程序转变成可运行的程序。例如，本书介绍用汇编语言编写源程序的方法，但源程序必须利用“汇编程序”完成翻译才可被计算机执行。高级语言则采用编译类或解释类程序来完成这个工作。辅助程序开发的程序设计语言的翻译程序等一般也归类为系统软件。

2. 应用软件

应用软件是解决某个问题的程序及其文档，大到用于处理某专业领域问题的程序，小到完成一个非常具体工作的程序。它覆盖了计算机应用的所有方面，每个应用都需要相应的应用程序。

个人计算机系统具有多种多样的应用软件。例如，进行程序设计时要采用文本编辑软件编写源程序，带有丰富格式的字处理软件帮助你书写文章，排版软件则用于书刊出版。

大型的程序设计项目往往要借助软件开发工具（包）。这个开发工具是进行程序设计所用到的各种程序的有机集合，所以也被称为集成开发环境，包括文本编辑器、语言翻译程序，有用于形成可执行文件的连接程序，以及进行程序排错的调试程序等。

1.1.3 程序设计语言

利用计算机解决实际问题，一般要编制程序。程序设计语言就是程序员用来编写程序的语言，它是人与计算机之间交流的工具。程序设计语言可以分为机器语言、汇编语言和高级语言。

1. 机器语言

计算机能够直接识别的是由二进制数 0 和 1 组成的代码。机器指令（Instruction）就是用二进制编码的指令，指令是控制计算机操作的命令，是处理器不需要翻译就能识别（直接执行）的“母语”，通常一条机器指令控制计算机完成一个操作。每种处理器都有各自的机器指令，某处理器支持的所有指令的集合就是该处理器的指令集（Instruction Set）或指令系统。指令集及使用它们编写程序的规则被称作机器语言（Machine Language）。

用机器语言形成的程序是计算机唯一能够直接识别、并执行的程序，而用其他语言编写的程序必须经过翻译、变换成机器语言程序；所以，机器语言程序常称为目标程序（或目的程序）。

例如，完成两个数据 100 和 256 相加的功能，在 8086 处理器的二进制代码序列如下：

```
10111000 01100100 00000000
00000101 00000000 00000001
```

几乎没有人能够直接读懂该程序段的功能，因为机器语言就是看起来毫无意义的一串代码。用机器语言编写程序的最大缺点是难以理解，因而极易出错，也难以发现错误。所以，只是在计算机发展的早期或不得已的情况下，才采用机器语言编写程序。现在，除了有时在

程序某处需要直接采用机器指令填充外，几乎没有人采用机器语言编写程序了。

二进制虽然只有 0 和 1 两个数码，但表达信息时会很长。为了简化表达，常用到十六进制。因为一个十六进制位就可以表达 4 位二进制数，并且易于相互转换，即二进制数 0000、0001、0010、……、1001、1010、1011、1100、1110、1111 用十六进制表达依次是 0、1、2、……、9、A、B、C、D、E、F，其中 A～F 依次表示十进制 10～15。这样，上述二进制代码序列用十六进制代码表示为：

```
B8 64 00
05 00 01
```

汇编语言中，习惯使用后缀字母区别不同进制的数据。例如，使用字母 B（或小写字母形式 b，来自二进制的英文单词 Binary）表示二进制数，使用字母 H（或小写字母形式 h，来自十六进制的英文单词 Hexadecimal）表示十六进制数，而十进制数通常不需要特别说明（或者用后缀字母 D 或 d，以示强调）。另外，涉及计算机硬件原理的技术文献中，所谓的“位”常指二进制位，也会表示十六进制位、或者十进制位，根据上下文予以分辨，否则可能产生误解。

2．汇编语言

为了克服机器语言的缺点，人们采用便于记忆并能描述指令功能的符号来表示机器指令。表示指令功能的符号称为指令助记符，或简称助记符（Mnemonic）；助记符一般采用表明指令功能的英语单词或其缩写。指令操作数同样也可以用易于记忆的符号表示。用助记符表示的指令就是汇编格式指令。汇编格式指令以及使用它们编写程序的规则形成汇编语言（Assembly Language）。用汇编语言书写的程序就是汇编语言程序，或称汇编语言源程序。

例如，实现 100 与 256 相加的 MASM 汇编语言程序片段如下：

```
mov   ax,100
add   ax,256
```

第一条指令的功能将数据 100 传送给名为 AX 的寄存器，MOV 是传送指令的助记符，实现赋值功能。该指令对应的机器代码就是机器语言举例的第一个二进制串。

第二条指令实现加法操作，ADD 是加法指令的助记符，对应机器语言举例的第二个二进制串。

如果熟悉有关助记符及对应指令的功能，就可以读懂上述程序片段了。

汇编语言是一种符号语言，它用助记符表示操作码，比机器语言容易理解和掌握、也容易调试和维护。但是，汇编语言源程序要翻译成机器语言程序才可以由处理器执行。这个翻译的过程称为“汇编”，完成汇编工作的程序就是汇编程序（Assembler）。

3．高级语言

汇编语言虽然较机器语言直观一些，但仍然烦琐难记。于是在 20 世纪 50 年代，人们研制出了高级程序设计语言（High Level Language）。高级语言比较接近于人类自然语言的语法习惯及数学表达形式，它与具体的计算机硬件无关，更容易被广大计算机工作者掌握和使用。利用高级语言，即使一般的计算机用户也可以编写程序，而不必懂得计算机的结构和工作原理。

目前，计算机高级语言已有上百种之多，得到广泛应用的有十几种，每种高级语言都有其最适用的领域。用任何一种高级语言编写的程序都要通过编译程序（Compiler）翻译成机器语言程序（称为目标程序）后计算机才能执行，或者通过解释程序边解释边执行。

例如，用高级语言表达 100 与 256 相加，就是通常的数学表达形式：100 + 256。

4. 学习汇编语言的意义

高级语言简单、易用，而汇编语言复杂、难懂，是否就没有必要再采用汇编语言了呢？让我们首先列表 1-1 简单比较一下汇编语言和高级语言的特点。

表 1-1　汇编语言和高级语言的对比

汇编语言	高级语言
与处理器密切相关。每种处理器都有自己的指令系统，相应的汇编语言各不相同。所以，汇编语言程序的通用性、可移植性较差	与具体计算机无关，高级语言程序可以在多种计算机上编译后执行
功能有限，又涉及寄存器、主存单元等硬件细节。所以，编写程序比较烦琐，调试起来也比较困难	采用类似自然语言的语法，不必关心诸如标志、堆栈等琐碎问题。容易被掌握和应用
本质上就是机器语言，可以直接、有效地控制计算机硬件，因而容易产生运行速度快、指令序列短小的高效率目标程序	不易直接控制计算机的各种操作，编译程序产生的目标程序往往比较庞大，运行速度较慢

通过对比，高级语言的优势明显。很自然地人们称机器语言和汇编语言为低级语言。但事实上，汇编语言被称为低层语言（Low Level Language）更合适。因为，程序设计语言是按照计算机系统的层次结构区分的，本没有“高低贵贱”之分，只是某种语言更适合某种应用层面（或说场合）而已。我们看到，汇编语言便于直接控制计算机硬件电路，可以编写在“时间”和“空间”两方面最有效，即执行速度快和目标代码小的程序。这些优点使得汇编语言在程序设计中占有重要的位置，是不可被取代的。下面罗列了汇编语言的主要应用场合：

- ⊙ 程序要具有较快的执行时间，或者只能占用较小的存储容量。例如，操作系统的核心程序段，实时控制系统的软件，智能仪器仪表的控制程序等。
- ⊙ 程序与计算机硬件密切相关，程序要直接、有效地控制硬件。例如，I/O 接口电路的初始化程序段，外部设备的低层驱动程序等。
- ⊙ 大型软件需要提高性能、优化处理的部分。例如，计算机系统频繁调用的子程序、动态连接库等。
- ⊙ 没有合适的高级语言、或只能采用汇编语言的时候。例如，开发最新的处理器程序时，暂时没有支持新指令的编译程序。
- ⊙ 许多实际应用的情况，例如分析具体系统尤其是该系统的低层软件、加密解密软件、分析和防治计算机病毒等。

当然，无须回避的事实是，随着各种编程技术的发展，单独使用汇编语言开发程序、尤其是应用程序的情况越来越少。所以，在实际的程序开发过程中，可以采用高级语言和汇编语言混合编程的方法，互相取长补短，更好地解决实际问题。

另外，编写汇编语言程序，需要使用处理器指令解决应用问题，而指令只是完成诸如将一个数据从存储器传送到寄存器、对两个寄存器值求和、指针增量指向下一个地址等简单的功能。所以，从教学角度来说，汇编语言程序员在将复杂的应用问题翻译成简单指令的过程中，就是从处理器角度解决问题，自然就容易理解计算机的工作原理了。

1.2 8086 处理器

处理器是计算机系统的硬件核心部件，决定着计算机的关键性能，常被称为中央处理单元，简称 CPU。了解处理器的基本结构、熟悉其寄存器作用以及存储器的组织是学习处理器指令的基础。

1.2.1 8086 的功能结构

处理器由多个功能部件组成。Intel 公司按两个功能模块描绘了 Intel 8086 处理器的内部结构，如图 1-2 所示。

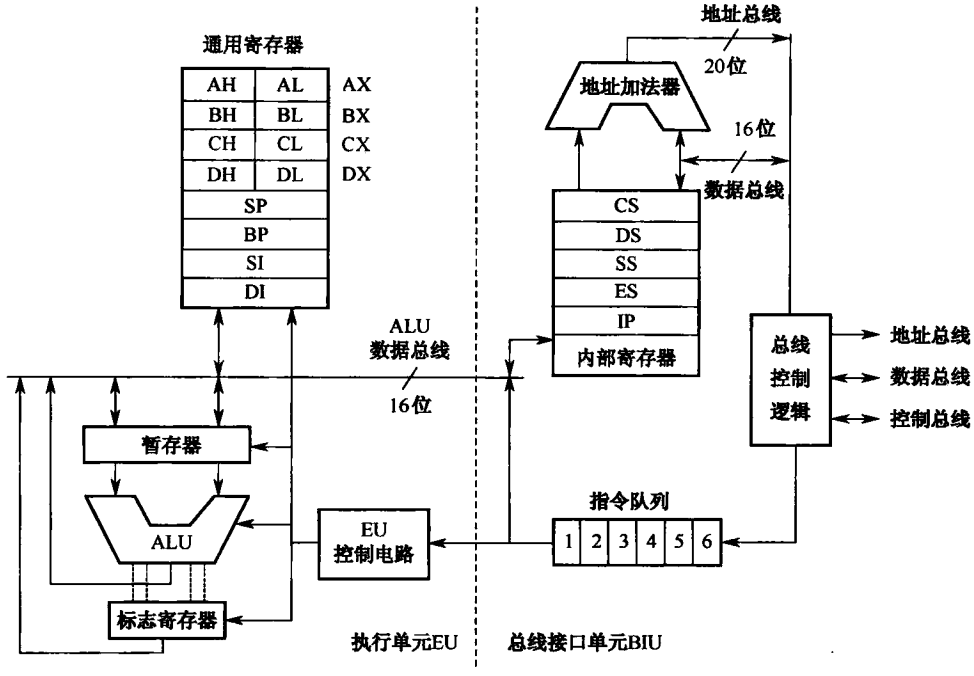

图 1-2 8086 的内部结构

1. 总线接口单元和执行单元

图 1-2 虚线右边部分是总线接口单元 BIU（Bus Interface Unit）。它由 6 字节的指令队列（即指令寄存器）、指令指针 IP（等同于程序计数器的功能）、段寄存器（CS、DS、SS 和 ES）、地址加法器和总线控制逻辑等构成。该单元管理着 8086 与系统总线的接口，负责处理器对存储器和外设进行访问。对外的 8086 处理器引脚由 16 位双向数据总线、20 位地址总线和若干控制总线组成。8086 所有对外操作必须通过 BIU 和这些总线进行，例如从主存中读取指令、从主存或外设读取数据、向主存或外设写出数据等操作。

图 1-2 虚线左边部分是执行单元 EU(Execution Unit)。它由算术逻辑单元 ALU(Arithmetic Logic Unit)、标志寄存器、通用寄存器和进行指令译码的 EU 控制电路等构成，负责执行指令的功能。

源程序由一条条语句组成，而可执行程序则由一条条指令组成，运行程序就是执行一条条

指令的过程。一条指令的整个执行过程又可以分成两个主要阶段：取指和执行，如图 1-3（a）所示。

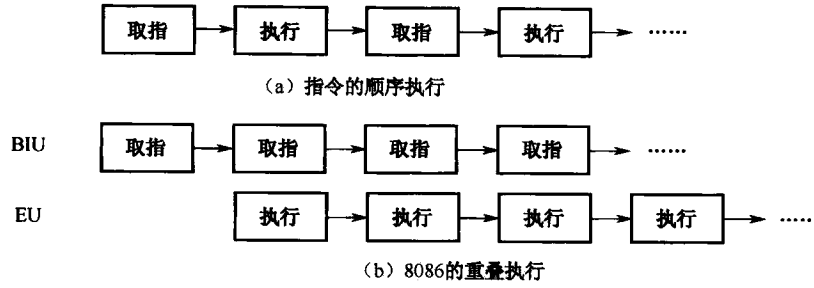

图 1-3　指令的执行过程

取指阶段是处理器将指令代码从主存储器中取出并进入处理器内部的过程。8086 处理器中，指令在存储器中的地址（即位置编号）由代码段寄存器 CS 和指令指针寄存器 IP 共同提供，再由地址加法器得到 20 位存储器地址。总线接口单元 BIU 负责从存储器取出这个指令代码，送入指令队列。

执行阶段是处理器将指令代码翻译成它代表的功能（被称为译码）、并发出有关控制信号实现这个功能的过程。8086 处理器中，执行单元 EU 从指令队列中获得预先取出的指令代码，在 EU 控制电路中进行译码，然后发出控制信号由算术逻辑单元 ALU 进行数据运算、数据传送等操作。指令执行阶段需要的操作数据有些来自处理器内部的寄存器，有些来自指令队列，还有些来自存储器和外设。如果需要来自存储器或外设的数据，则控制单元 EU 控制总线接口单元 BIU 从外部获取这些数据。

2. 指令预取

8086 处理器维护着长度为 6 字节的指令队列，该队列按照“先进先出”FIFO（First In First Out）的方式进行工作。当指令队列中出现空缺时，BIU 会自动取指填补这一空缺；而当程序不能按顺序执行，即发生转移（出现分支）时，BIU 又会废除已经取出的指令，重新取指形成新的指令队列。

8086 处理器中，指令的读取操作在 BIU 单元实现，而指令的执行阶段在 EU 单元完成。因为 BIU 和 EU 两个单元相互独立、分别完成各自操作，所以可以并行操作。换句话说，也就是在 EU 单元对一个指令进行译码执行时，BIU 单元可以同时对后续指令进行读取；所以，8086 处理器的指令读取，实际上是指令预取（Prefetch），如图 1-3（b）所示。

由于要译码执行的指令已经预取到了处理器内部的指令队列，所以 8086 不需要等待取指操作就可以从指令队列获得指令进行译码执行。而对于简单的处理器来说，在指令译码前必须等待取指操作的完成。取指是处理器最频繁的操作，每条指令都要读取指令代码一到数次（与指令代码的长度有关），所以 8086 的这种结构和操作方式节省了处理器的许多取指时间，提高了工作效率。这就是最简单的指令流水线技术。同时也看到，程序转移将使预取指令作废，从而降低了流水线效率。

1.2.2　8086 的寄存器

处理器内部需要高速存储单元，用于暂时存放程序执行过程中的代码和数据，这些存储