HZ BOOKS
华章教育

高等院校精品课程系列教材·省级

32位汇编语言程序设计

钱晓捷 ◎编著

精品课程

32-bit Assembly Language Programming

机械工业出版社
China Machine Press

高等院校精品课程系列教材·省级

32位汇编语言程序设计

钱晓捷 ◎编著

32-bit Assembly Language Programming

机械工业出版社
China Machine Press

本书以32位Intel 80x86处理器和个人计算机为硬件平台，基于32位Windows操作系统软件平台，借助微软MASM汇编程序讲解汇编语言程序设计。本书内容包括基本的汇编语言基础、常用处理器指令和汇编语言伪指令以及顺序、分支、循环、子程序结构，还包括扩展的Windows和DOS编程、与C++语言的混合编程、输入输出指令及编程，并涉及浮点、多媒体及64位指令等方面。

本书可以作为普通高校“汇编语言程序设计”等课程的教材或参考书，适合计算机、电子、通信和自控等电类专业的本科学生以及软件学院、计算机等电类专业的高职学生、成教学生阅读，同时也适合作为计算机应用开发人员和希望深入学习汇编语言的读者的极佳参考书。

图书在版编目（CIP）数据

32位汇编语言程序设计/钱晓捷编著. —北京：机械工业出版社，2011.8
（高等院校精品课程系列教材）

ISBN 978-7-111-34750-7

Ⅰ.3…　Ⅱ.钱…　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（2011）第092472号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春
北京瑞德印刷有限公司印刷
2011年8月第1版第1次印刷
185mm×260mm · 17.25印张
标准书号：ISBN 978-7-111-34750-7
定价：35.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前 言

随着计算机技术的发展，国内高校师生希望能够在32位Windows操作系统平台学习汇编语言，但如何面向初学者实施教学却面临诸多难点。于是，我们结合近年来的32位汇编语言教学实践编写了本书。

本书具有以下特色。

1. 简单易用的开发环境

目前，32位Windows平台的汇编语言编程主要使用MASM32和Visual C++集成化开发系统，但它们都略显复杂和庞大，不适合初学者（本书将此内容安排在第6章和第7章）。为此，本书构建了一个简单易用的开发环境（详见第1章），无需安装和配置，直接复制就可使用。它支持32位Windows控制台和16位DOS环境，提供MASM汇编程序、连接程序、WinDbg和CodeView调试程序及其帮助文档、配套输入输出子程序库及方便操作的批处理文件等。

2. 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，同时让读者全面认识程序设计语言，体会低层编程特点，以便更好地应用高级语言。为此，本书不是详尽展开所有处理器指令、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，而是侧重编程思想和技术。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。

3. 突出实践的教学过程

本书以约70个示例程序和60个习题程序贯穿教学内容。第1章在介绍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，并利用简单易用的输入输出子程序编写具有显示结果的程序。第2章结合数据编码、常量定义和变量应用，自然地引出常用伪指令。第3章分类学习处理器基本指令，逐渐编写特定要求的程序片段。第4～9章以程序结构为主线，围绕数码转换子程序，结合Windows编程、混合编程、DOS和I/O编程、浮点指令，从简单到复杂逐步编写具有实用

价值的应用程序。

4. 循序渐进的教学原则

为了便于学生理解和掌握，且便于教师实施教学，本书以“循序渐进、难点分散、前后对照”为原则，努力做到“语言浅显、描述详尽、图表准确”。本书内容编排精彩纷呈，例如，将处理器指令和汇编伪指令分散于各个教学内容之中，引出列表文件暂时避开调试程序，用简单的子程序库化解系统调用的烦琐；程序具有交互性和趣味性，适当对比高级语言，并展示底层工作原理；每章都编制丰富的习题，满足课外练习、上机实践和试题组织的需要。

为了更好地服务于广大师生和读者，编者开辟了“大学微机技术系列课程教学辅助网站”（http://www2.zzu.edu.cn/qwfw）。该网站面向“汇编语言程序设计”和“微机原理及接口技术”课程，提供相关教学课件（电子教案）、教学大纲、教材勘误、疑难解答、输入输出子程序库、示例源程序文件等辅助资源，是本教材的动态延伸。

本书由郑州大学信息工程学院钱晓捷编写，并得到了穆玲玲、关国利、张青、张行进等人的帮助，衷心感谢他们，同时也感谢机械工业出版社华章公司的大力支持。

欢迎广大师生和读者通过电子邮件（qianxiaojie@zzu.edu.cn）与编者交流。

编　者

2011 年 5 月

教学建议

汇编语言课程主要有每周4授课学时（总学时68）和每周3授课学时（总学时51）两种教学方案，并配合足够的上机实践学时。结合本书内容，各章学时安排参见下表，表中各章内容简介用于提示教学重点。

目　录	内容简介	学时（68）	学时（51）
第1章 汇编语言基础	在了解软硬件开发环境的基础上，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	4	4
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，掌握利用处理器指令寻址数据的方式	8	6
第3章 通用数据处理指令	熟悉IA-32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	8	8
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的方法	12	10
第5章 模块化程序设计	以子程序结构为主体，围绕数码转换实现键盘输入和显示输出示例程序，掌握子程序、文件包含、宏汇编等多模块编程的方法	12	8
第6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。熟悉MASM的高级特性，理解Windows图形窗口程序的编写	10	6
第7章 与Visual C++混合编程	掌握嵌入汇编和模块连接进行混合编程的方法，理解堆栈帧的作用，熟悉汇编语言调用高级语言函数的方法和开发、调试过程	6	4
第8章 DOS环境程序设计	熟悉DOS应用程序的特点和DOS功能调用，掌握串操作指令和输入输出指令及应用，理解初始化编程、中断控制编程方法	6	4
第9章 浮点、多媒体及64位指令	熟悉浮点数据格式、多媒体数据格式及64位编程环境的特点，了解浮点操作、多媒体操作和64位指令	2	1

目 录

前言

教学建议

第 1 章 汇编语言基础 …… 1

1.1 Intel 80x86 系列处理器 …… 1

1.1.1 16 位 80x86 处理器 …… 1

1.1.2 IA-32 处理器 …… 2

1.1.3 Intel 64 处理器 …… 3

1.2 个人计算机系统 …… 4

1.2.1 硬件组成 …… 4

1.2.2 寄存器 …… 6

1.2.3 存储器组织 …… 9

1.2.4 程序设计语言 …… 13

1.2.5 软件系统 …… 15

1.3 汇编语言程序格式 …… 18

1.3.1 指令代码格式 …… 18

1.3.2 语句格式 …… 20

1.3.3 源程序框架 …… 21

1.3.4 开发过程 …… 25

第 1 章习题 …… 29

第 2 章 数据表示和寻址 …… 32

2.1 数据表示 …… 32

2.1.1 数制 …… 32

2.1.2 数值的编码 …… 35

2.1.3 字符的编码 …… 37

2.2 常量表达 …… 39

2.3 变量应用 …… 41

2.3.1 变量定义 …… 41

2.3.2 变量属性 …… 46

2.4 数据寻址方式 …… 49

2.4.1 立即数寻址方式 …… 49

2.4.2 寄存器寻址方式 …… 50

2.4.3 存储器寻址方式 …… 51

2.4.4 各种数据寻址方式的组合 …… 55

第 2 章习题 …… 56

第 3 章 通用数据处理指令 …… 59

3.1 数据传送类指令 …… 59

3.1.1 通用数据传送指令 …… 59

3.1.2 堆栈操作指令 …… 61

3.1.3 其他传送指令 …… 63

3.2 算术运算类指令 …… 66

3.2.1 状态标志 …… 66

3.2.2 加法指令 …… 68

3.2.3 减法指令 …… 69

3.2.4 乘法和除法指令 …… 71

3.2.5 其他运算指令 …… 72

3.3 位操作类指令 …… 74

3.3.1 逻辑运算指令 …… 74

3.3.2 移位指令 …… 77

第 3 章习题 …… 80

第 4 章 程序结构 …… 86

4.1 顺序程序结构 …… 86

4.2 分支程序结构 …… 87

4.2.1 无条件转移指令 …… 87

4.2.2 条件转移指令 …… 90

4.2.3 单分支程序结构 …… 94

4.2.4 双分支程序结构 …… 95

4.2.5 多分支程序结构 …… 97

4.3 循环程序结构 …… 99

4.3.1 循环指令 …… 99

4.3.2 计数控制循环 …… 100

4.3.3 条件控制循环…… 102
4.3.4 多重循环…… 103
第4章习题 …… 104
第5章 模块化程序设计 …… 109
5.1 子程序结构…… 109
5.1.1 子程序指令…… 109
5.1.2 子程序设计…… 112
5.2 参数传递…… 113
5.2.1 寄存器传递参数…… 113
5.2.2 共享变量传递参数…… 116
5.2.3 堆栈传递参数…… 119
5.3 多模块程序结构…… 121
5.3.1 源文件包含…… 122
5.3.2 模块连接…… 124
5.3.3 子程序库…… 125
5.4 宏结构…… 126
5.4.1 宏汇编…… 126
5.4.2 重复汇编…… 132
5.4.3 条件汇编…… 133
第5章习题 …… 136
第6章 Windows 编程 …… 140
6.1 操作系统函数调用…… 140
6.1.1 动态连接库…… 140
6.1.2 MASM 的过程声明和调用…… 141
6.1.3 程序退出函数…… 142
6.2 控制台应用程序…… 143
6.2.1 控制台输出…… 143
6.2.2 控制台输入…… 145
6.2.3 单字符输入…… 148
6.3 图形窗口应用程序…… 149
6.3.1 消息窗口…… 149
6.3.2 结构变量…… 150
6.3.3 MASM 的高级语言特性…… 152
6.3.4 简单窗口程序…… 159
第6章习题 …… 166
第7章 与 Visual C ++ 混合编程…… 169
7.1 嵌入汇编…… 169
7.2 模块连接…… 172
7.2.1 约定规则…… 172
7.2.2 堆栈帧…… 174
7.3 调用高级语言函数…… 181
7.3.1 嵌入汇编中调用高级语言函数…… 181
7.3.2 汇编语言中调用 C 库函数 … 181
7.4 使用 Visual C ++ 开发环境 …… 183
7.4.1 汇编语言程序的开发过程…… 183
7.4.2 汇编语言程序的调试过程…… 184
第7章习题 …… 187
第8章 DOS 环境程序设计 …… 191
8.1 DOS 编程 …… 191
8.1.1 实地址存储模型…… 191
8.1.2 DOS 应用程序框架…… 193
8.1.3 DOS 功能调用…… 195
8.2 串操作类指令…… 198
8.2.1 串传送指令…… 198
8.2.2 串检测指令…… 200
8.3 输入输出程序设计…… 203
8.3.1 输入输出指令…… 203
8.3.2 定时器初始化编程…… 205
8.3.3 扬声器控制编程…… 207
8.4 中断控制编程…… 208
8.4.1 中断控制系统…… 208
8.4.2 内部中断服务程序…… 211
8.4.3 驻留中断服务程序…… 212
第8章习题 …… 214
第9章 浮点、多媒体及 64 位指令…… 217
9.1 浮点指令…… 218
9.1.1 实数编码…… 218
9.1.2 浮点寄存器…… 221
9.1.3 浮点指令及其编程…… 224
9.2 多媒体指令…… 226
9.2.1 MMX 技术 …… 227
9.2.2 SSE 技术 …… 229
9.2.3 SSE2 技术 …… 230

9.2.4 SSE3 技术 …… 231
9.3 64 位指令 …… 232
9.3.1 64 位方式的运行环境 …… 233
9.3.2 64 位方式的指令 …… 234
第 9 章习题 …… 235
附录 A 调试程序 WinDbg …… 238
附录 B 输入输出子程序库 …… 252
附录 C 32 位通用指令列表 …… 254
附录 D MASM 伪指令和操作符列表 … 258
附录 E 列表文件符号说明 …… 260
附录 F 常见汇编错误信息 …… 261
参考文献 …… 264

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，如 C、C ++ 、Basic、Java 等。高级语言需要翻译为计算机能够识别的指令（机器语言），才能被计算机执行。机器语言是一串 0 和 1 组成的二进制代码，对程序员来说晦涩难懂，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，Mnemonic）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，也称为低层语言。

本章首先介绍汇编语言的硬件基础：Intel 80x86 系列处理器和个人计算机，然后是软件基础：Windows 操作系统和微软 MASM 汇编程序，接着讲解汇编语言的意义，最后学习汇编语言的程序格式，并编写第一个汇编语言程序。

1.1 Intel 80x86 系列处理器

汇编语言的主体是处理器指令。处理器（Processor）是计算机的运算和控制核心，也常称为中央处理单元（Central Processing Unit，CPU）。微型计算机中的处理器常采用一块大规模集成电路芯片，称之为微处理器（Microprocessor），它代表着整个微型计算机系统的性能。所以，通常将采用微处理器为核心构造的计算机称为微型计算机。

微型计算机（Microcomputer）是我们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微型计算机。

美国 Intel（英特尔）公司是目前世界上最有影响的处理器生产厂家，也是世界上第一个微处理器芯片的生产厂家，其生产的 Intel 80x86 系列处理器一直是个人计算机的主流处理器。

1.1.1 16 位 80x86 处理器

1971 年，Intel 公司生产的 4 位处理器芯片 4004 宣告了微型计算机时代的到来。1972 年，Intel 公司开发了 8 位处理器 8008 芯片；1974 年，生产了 Intel 8080；1977 年，Intel 公司将 8080 及其支持电路集成在一块集成电路芯片上，形成了性能更高的 8 位处理器 8085。从 1978 年开始，Intel 公司在其 8 位处理器基础上，陆续推出了 16 位结构的 8086、8088 和 80286（也可以表示成 Intel 286，本书采用 80286 这种形式）等处理器，它们在 IBM PC 系列机中获得广泛应用，被称为 16 位 80x86 处理器。

1. 8086

1978 年，Intel 公司推出 16 位 8086 处理器，这是该公司生产的第一个 16 位芯片。8086 的数据总线为 16 位，地址总线为 20 位，主存容量为 1MB，时钟频率为 5MHz。8086 支持的所有指令，即指令系统（Instruction Set）成为整个 Intel 80x86 系列处理器的 16 位基本指令集。

为了方便与当时的 8 位外部设备连接，1979 年，Intel 公司推出准 16 位处理器 8088。8088 只是将外部数据总线设计为 8 位，内部仍保持 16 位结构，指令系统等都与 8086 相同。随后的 80186 和 80188 则分别是以 8086 和 8088 为核心并配以支持电路构成的芯片，但它们在 8086 指令系统的基础上增加了若干条实用指令，涉及堆栈操作、输入输出指令、移位指令、乘法指令、支持高级语言的指令。

2. 80286

1982 年，Intel 公司推出仍为 16 位结构的 80286 处理器，但地址总线扩展为 24 位，即主存储器具有 16MB 容量。80286 设计了与 8086 工作方式一样的实方式（Real Mode），还新增了保护方式（Protected Mode）。在实方式下，80286 相当于一个快速 8086。在保护方式下，80286 提供了存储管理、保护机制和多任务管理的硬件支持。为支持保护方式，80286 引入了系统指令，为操作系统等核心程序提供处理器控制功能。

1.1.2 IA-32 处理器

IBM PC 系列机的广泛应用推动了处理器芯片的生产。Intel 公司在推出 32 位结构的 80386 处理器后，明确宣布 80386 芯片的指令集结构（Instruction Set Architecture，ISA）为以后开发的 80x86 系列处理器的标准，称为 Intel 32 位结构（Intel Architecture-32，IA-32）。现在，Intel 公司的 80386、80486 以及 Pentium 各代处理器统称为 IA-32 处理器或 32 位 80x86 处理器。

1. 80386

1985 年，Intel 80x86 处理器进入第三代 80386。80386 处理器采用 32 位结构，数据总线为 32 位，地址总线也是 32 位，可寻址 4GB（$1GB = 2^{30}B = 1024MB$）主存，时钟频率有 16MHz、25MHz 和 33MHz。IA-32 指令系统在兼容原 16 位 80286 指令系统的基础上，全面升级为 32 位，还新增了有关位操作、条件设置等指令。

80386 除保持与 80286 兼容外，又提供了虚拟 8086 工作方式（Virtual 8086 Mode）。虚拟 8086 方式是在保护方式下的一种特殊状态，类似于 8086 工作方式但又接受保护方式的管理，能够模拟多个 8086 处理器。32 位 PC 的 Windows 操作系统采用保护方式，其 MS-DOS 命令行（环境）就是虚拟 8086 方式，而早期采用的 DOS 操作系统是以实方式为基础建立的。

2. 80486

1989 年，Intel 公司推出 80486 处理器。它的内部集成了 120 万个晶体管，最初的时钟频率为 25MHz，很快发展到 33MHz 和 50MHz。从结构上来说，80486 = 80386 + 80387 + 8KB Cache，即 80486 把 80386 处理器与 80387 数学协处理器和 8KB 高速缓冲存储器（Cache）集成在一个芯片上，使处理器的性能大大提高。

传统上，中央处理单元 CPU 主要是整数处理器。为了协助处理器处理浮点数据（实数），Intel 公司设计了数学协处理器，后被称为浮点处理单元（Floating-point Processing Unit，FPU）。配合 8086 和 8088 整数处理器的数学协处理器是 8087，配合 80286 的是 80287，配合 80386 的是 80387。而从 80486 开始，FPU 已经被集成到处理器中。这样，IA-32 处理器的指令系统也就包含了浮点指令，能够直接支持对浮点数据的处理。

3. Pentium 系列

Pentium 芯片应该称为 80586 处理器，因为数字很难进行商标版权保护而特意取名。其实，

Pentium 是源于希腊文“pente”（数字 5），再加上后缀-ium（化学元素周期表中命名元素常用的后缀）变化而来的。同时，Intel 公司为其取了一个响亮的中文名称“奔腾”，并进行了商标注册。

Intel 公司于 1993 年制造成功 Pentium，于 1995 年正式推出 Pentium Pro（原来被称为 P6，中文名称为“高能奔腾”）。在处理器结构上，Pentium 主要引入了超标量（Superscalar）技术，Pentium Pro 主要采用了动态执行技术来提升处理器性能。它们增加了若干条整数指令，完善了浮点指令。

前面所述的各代 IA-32 处理器都新增了若干实用指令，但非常有限。为了顺应微机向多媒体和通信方向发展，Intel 公司及时在其处理器中加入了多媒体扩展（MultiMedia eXtension，MMX）技术。MMX 技术于 1996 年正式公布，它在 IA-32 指令系统中新增了 57 条整数运算多媒体指令，可以用这些指令对图像、音频、视频和通信方面的程序进行优化，使微机对多媒体的处理能力较原来有了大幅度提升。MMX 指令应用于 Pentium 处理器就是 Pentium MMX（多能奔腾）。MMX 指令应用于 Pentium Pro 处理器就是 Pentium Ⅱ，它于 1997 年推出。

1999 年，针对互联网和三维多媒体程序的应用要求，Intel 公司在 Pentium Ⅱ的基础上又新增了 70 条 SSE(Streaming SIMD Extensions）指令（原称为 MMX-2 指令），开发了 Pentium Ⅲ。SSE 指令侧重于浮点单精度多媒体运算，极大地提高了浮点 3D 数据的处理能力。SSE 指令类似于 AMD 公司发布的 3D Now！指令。由于这些多媒体指令具有显著的单指令多数据（Single Instruction Multiple Data，SIMD）处理能力，即一条指令可以同时进行多组数据的操作，所以现在统称为 SIMD 指令。

2000 年 11 月，Intel 公司推出 Pentium 4，它新增了 76 条 SSE2 指令集，侧重于增强浮点双精度多媒体运算能力。2003 年，新一代 Pentium 4 处理器又新增了 13 条 SSE3 指令，用于补充、完善 SIMD 指令集。

1.1.3　Intel 64 处理器

随着互联网、多媒体、3D 视频等的发展，信息时代的应用对计算机性能提出了越来越高的要求，32 位单核处理器已不能适应这一要求。

1. Intel 64 结构

一直以来，80x86 处理器的更新换代都保持与早期处理器的兼容，以便继续使用现有的软硬件资源。但是，Intel 公司迟迟不愿将 80x86 处理器扩展为 64 位，这给了 AMD 公司一个机会。AMD 公司是生产 IA-32 处理器兼容芯片的厂商，是 Intel 公司最主要的竞争对手。AMD 公司的 IA-32 兼容处理器，其价格低于 Intel，但性能却没有超越 Intel。于是，AMD 公司于 2003 年 9 月率先推出支持 64 位、兼容 80x86 指令集结构的 Athlon 64 处理器（K8 核心），将桌面 PC 引入了 64 位领域。

2005 年，在 PC 用户对 64 位技术的企盼和 AMD 公司 64 位处理器的压力下，Intel 公司推出了扩展存储器 64 位技术（Extended Memory 64 Technology，EM64T）。EM64T 技术是 IA-32 结构的 64 位扩展，首先应用于支持超线程技术的 Pentium 4 终极版（支持双核技术）和 6xx 系列 Pentium 4 处理器。随着 EM64T 技术的出现，IA-32 指令系统也扩展成为 64 位，称为 Intel 64 结构。这之后的 Pentium 4 处理器、Pentium E 系列多核处理器、酷睿（Core）2 和酷睿 i 系列多核处理器等都支持 Intel 64 结构。

Intel 64 结构为软件提供了 64 位线性地址空间，支持 40 位物理地址空间。IA-32 处理器支持保护方式（含虚拟 8086 方式）、实地址方式和系统管理 SMM 方式，Intel 64 结构则引入了一个新的工作方式：32 位扩展工作方式（IA-32e）。IA-32e 除有一个运行 32 位和 16 位软件的兼容方式

外，还有一个 64 位方式。在 64 位工作方式下，允许 64 位操作系统运行存取 64 位地址空间的应用程序，还可以存取 8 个附加的通用寄存器、8 个附加的 SIMD 多媒体寄存器、64 位通用寄存器和 64 位指令指针等。

2. 多核技术

单纯以提高时钟频率等增加处理器复杂度的传统方法已经很难提升处理器性能，传统方法还带来功耗剧增、发热量巨大的问题。于是，多核（Multi-core）技术应运而生。多核处理器是在一个集成电路芯片上制作了两个或多个处理器执行核心，依靠多个处理器核心相互协作同时执行多个程序线程提升性能。基于不同的处理器内部结构，Intel 公司推出了多款多核处理器，目前主要是 Intel 奔腾 E 系列多核处理器、酷睿 2 和酷睿 i 系列多核处理器。

另一方面，SSE 系列指令集继续丰富，酷睿 2 补充了 SSE3 指令（即 32 条 SSSE3 指令），又推出增加了 54 条指令的 SSE4 指令集。其中，47 条指令在 Intel 公司面向服务器领域的至强（Xeon）5400 系列和酷睿 2 至尊版 QX9650 中引入，称为 SSE4.1 指令，这些指令致力于提升多媒体、3D 处理等的性能；其余 7 条指令称为 SSE4.2 指令。

Intel 公司充分利用集成电路生产的先进技术和处理器结构的革新技术，推出了多种 Intel 80x86 系列处理器芯片。就目前的发展来看，Intel 公司正在利用单芯片多处理器技术生产双核、4 核等多核处理器，并逐渐推广支持 64 位处理器和 64 位软件的个人计算机。

1.2 个人计算机系统

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，就像人的躯体一样。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和有关的文档资料），是我们指示计算机工作的命令，就像人的思想一样。

1.2.1 硬件组成

源于冯·诺伊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；运算器是对信息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转换成计算机内部所能识别和接受的信息方式，并顺序地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·诺伊曼计算机结构进行了改进，例如，在现代计算机中，5 大部件成为 3 个硬件子系统：处理器、存储系统和输入输出系统。处理器（中央处理单元，CPU）包括运算器和控制器，是信息处理的中心部件。存储系统由寄存器、高速缓冲存储器、主存储器和辅助存储器构成层次结构。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，通常被合称为“主机”。输入（Input）设备和输出（Output）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设，还包括外设与主机之间相互连接的接口电路。

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的各个功能部件使得计算机系统具有组合灵活、扩展方便的特点。

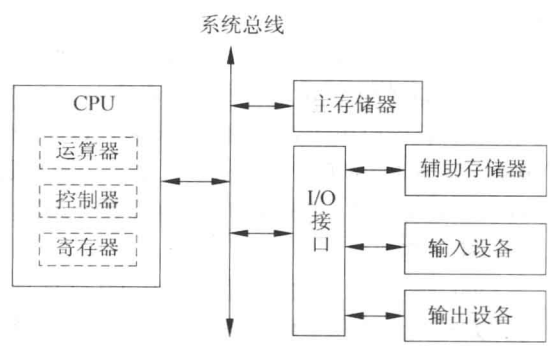

图 1-1 计算机系统的硬件组成

1. 处理器

计算机的核心是处理器（CPU），微型计算机中也常称为微处理器。它是采用大规模集成电路技术生产的半导体芯片，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。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。

16位IBM PC系列机采用16位80x86处理器，32位PC则采用IA-32处理器或其兼容芯片（如AMD公司的系列处理器）。

2. 存储器

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

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，主存储器造价高、速度快，但容量小，主要用来存放当前正在运行的程序和正待处理的数据。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，辅助存储器造价低、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。

从读写功能来区分，存储器分为可读可写的随机存取存储器（Random Access Memory，RAM）和只读存储器（Read Only Memory，ROM）。构成主存时既需要RAM也需要ROM，但注意半导体RAM芯片在断电后原存放信息将会丢失，而ROM芯片中的信息可在断电后长期保存。

个人计算机的主存由半导体存储芯片ROM和RAM构成。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，主要用来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空间执行。在16位PC系列机时代，RAM容量是64KB或1MB。32位PC RAM容量从最初的4MB，逐渐发展到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，俗称内存条。

3. 外部设备

外部设备是指计算机上配备的输入设备和输出设备，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，二者合称为控制台（Console）。个人计算机还可选择鼠标器、打印机、扫描仪等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上。这就需要有一个I/O接口来充当外设和主机间的桥梁，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接口电路常制成独立的电路板，也常称为接口卡（Card），使用时将其插在主板上。

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，物理上就是一组共用导线。例如，处理器芯片的对外引脚（Pin）常称为处理器总线。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（Industry Standard Architecture，ISA）系统总线连接各个功能部件。32 位 PC 使用外设部件互连（Peripheral Component Interconnect，PCI）总线连接I/O接口卡。系统总线除了作为主板上处理器、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，故被称为 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和处理等操作。下面具体学习 IA-32 处理器的寄存器和存储器组织，而输入输出地址将在 8.3 节“输入输出程序设计”中介绍。

1.2.2 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，这些存储单元称为寄存器（Register）。处理器内部设计有多种寄存器，每种寄存器又可能有多个，从应用的角度可以将这些寄存器分成两类：透明寄存器和可编程寄存器。

有些寄存器对应用人员来说不可见，不能直接控制，例如，保存指令代码的指令寄存器，所以称它们为透明寄存器。这里的“透明”（Transparency）是计算机学科中常用的一个专业术语，表示实际存在但从某个角度看好像没有，运用“透明”思想可以使我们抛开不必要的细节，而专注于关键问题。

底层语言程序员需要掌握可编程（Programmable）寄存器，它们具有引用名称，供编程使用，可编程寄存器还可以进一步分成通用寄存器和专用寄存器两类：

- 通用寄存器：这类寄存器在处理器中数量较多、使用频度较高，具有多种用途。例如，它们可用来存放指令需要的操作数据，又可用来存放地址以便在主存或 I/O 接口中指定操作数据的位置。
- 专用寄存器：这类寄存器只用于特定目的。例如，程序计数器（Program Counter，PC）用于记录将要执行指令的主存地址，标志寄存器用于保存指令执行的辅助信息。

IA-32 处理器通用指令（整数处理指令）的基本执行环境包括 8 个 32 位通用寄存器、6 个 16 位段寄存器、1 个 32 位标志寄存器和 1 个 32 位指令指针寄存器，如图 1-2 所示（图中数字 31、15、7、0 等依次用于表达二进制位 D_{31}、D_{15}、D_7、D_0）。

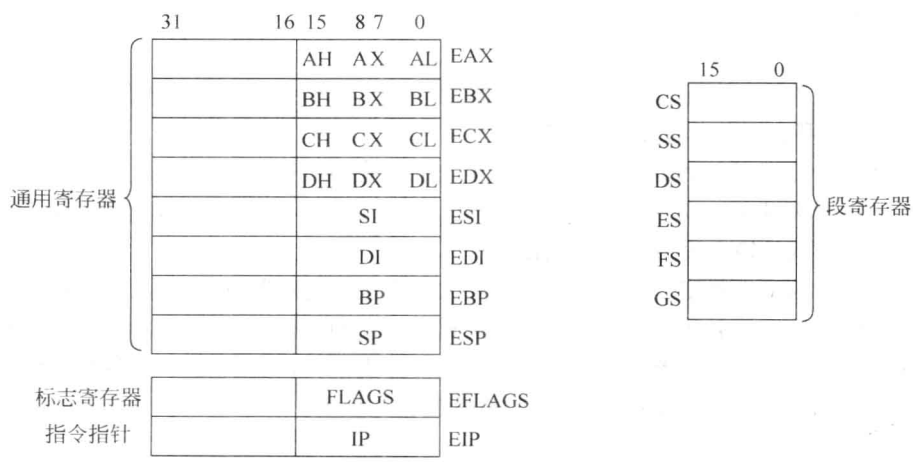

图 1-2 IA-32 常用寄存器

1. 通用寄存器

通用寄存器（General-Purpose Register）一般是指处理器最常使用的整数通用寄存器，可用

于保存整数数据、地址等。IA-32 处理器只有 8 个 32 位通用寄存器，数量有限。

IA-32 处理器的 8 个 32 位通用寄存器分别被命名为 EAX、EBX、ECX、EDX、ESI、EDI、EBP 和 ESP，它们是在原 8086 支持的 16 位通用寄存器的基础上扩展而得到的。上述 8 个名称中去掉表达扩展含义的字母 E（Extended），就是 8 个 16 位通用寄存器的名称：AX、BX、CX、DX、SI、DI、BP 和 SP，分别表示相应 32 位通用寄存器低 16 位部分。其中前 4 个通用寄存器 AX、BX、CX 和 DX 还可以进一步分成高字节 H（High）和低字节 L（Low）两部分，这样又有了 8 个 8 位通用寄存器：AH 和 AL、BH 和 BL、CH 和 CL、DH 和 DL。

编程中可以使用 32 位寄存器（如 ESI），也可以只使用其低 16 位部分（名称中去掉字母 E，如 SI，多用在 16 位平台和操作 16 位数据时）。对前 4 个 32 位通用寄存器（如 EAX），可以使用全部 32 位：$D_{31} \sim D_0$（EAX），可以使用低 16 位：$D_{15} \sim D_0$（AX），还可以将低 16 位再分成两个 8 位使用：$D_{15} \sim D_8$（AH）和 $D_7 \sim D_0$（AL）。注意，存取 16 位寄存器时，相应的 32 位寄存器的高 16 位不受影响；存取 8 位寄存器时，相应的 16 位和 32 位寄存器的其他位也不受影响。这样，Intel 80x86 处理器一方面保持了相互兼容，另一方面也可以方便地支持 8、16 和 32 位操作。

通用寄存器是多用途的，可以保存数据、暂存运算结果，也可以存放存储器地址、作为变量的指针。但在 IA-32 处理器中每个寄存器又有它们各自的特定作用，并因而得名。程序中通常也按照其含义使用它们，如表 1-1 所示。

表 1-1　IA-32 处理器的通用寄存器

寄存器名称	中英文含义	作用
EAX	累加器（Accumulator）	使用频度最高，用于算术运算、逻辑运算以及与外设传送信息等
EBX	基址寄存器（Base Address Register）	常用来存放存储器地址，以方便指向变量或数组中的元素
ECX	计数器（Counter）	常作为循环操作等指令中的计数器
EDX	数据寄存器（Data Register）	可用来存放数据，其中低 16 位 DX 常用来存放外设端口地址
ESI	源变址寄存器（Source Index Register）	用于指向字符串或数组的源操作数
EDI	目的变址寄存器（Destination Index Register）	用于指向字符串或数组的目的操作数
EBP	基址指针寄存器（Base Pointer Register）	默认情况下指向程序堆栈区域的数据，主要用于在子程序中访问通过堆栈传递的参数和局部变量
ESP	堆栈指针寄存器（Stack Pointer Register）	专用于指向程序堆栈区域顶部的数据，在涉及堆栈操作的指令中会自动增加或减少

许多指令有两个操作数：源操作数和目的操作数。

- 源操作数是指被传送或参与运算的操作数。
- 目的操作数是指保存传送结果或运算结果的操作数。

堆栈（Stack）是一个特殊的存储区域，它采用先进后出（First In Last Out，FILO）或称为后进先出（Last In First Out，LIFO）的操作方式存取数据。调用子程序时，它用于暂存数据、传递参数、存放局部变量，也可以用于临时保存数据。堆栈指针会随着处理器执行有关指令自动增大或减小，所以 ESP 不应该再用于其他目的，这样，ESP 可归类为专用寄存器，但是，ESP 又可以像其他通用寄存器一样灵活地改变。

2. 标志寄存器

标志（Flag）用于反映指令执行结果或控制指令执行形式。许多指令执行之后将影响有关的状态标志位，不少指令的执行要利用某些标志，当然，也有很多指令与标志无关。处理器中用一个或多个二进制位表示一种标志，其 0 或 1 的不同组合表示标志的不同状态。Intel 8086 支持的标志形成了一个 16 位的标志寄存器 FLAGS。以后各代 80x86 处理器有所增加，形成了 32 位的

EFLAGS 标志寄存器，如图 1-3 所示（图上方的数字表示该标志在标志寄存器中的位置）。EFLAGS 标志寄存器包含一组状态标志、一个控制标志和一组系统标志，其初始状态为 000000002H（也就是 D_1 位为 1，其他位全部为 0。H 表示这是用十六进制表达的数据），其中 1、3、5、15 和 22～31 位被保留，软件不应该使用它们或依赖于这些位的状态。

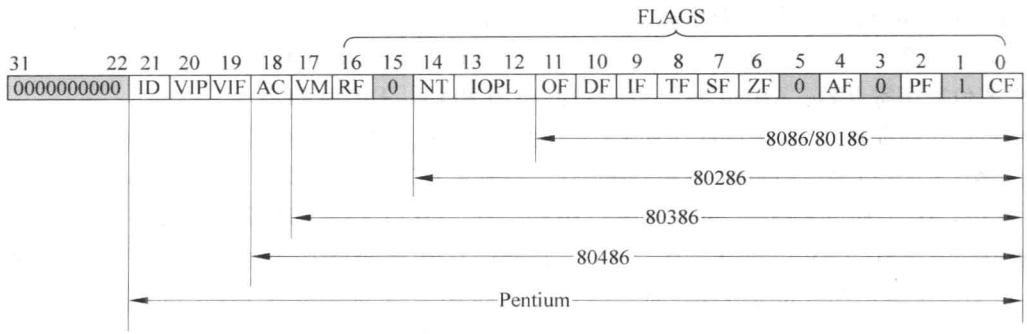

图 1-3 标志寄存器 EFLAGS

（1）状态标志

状态标志是最基本的标志，用来记录指令执行结果的辅助信息。加减运算和逻辑运算指令是主要设置状态标志的指令，其他有些指令的执行也会相应地设置它们。状态标志有 6 个，处理器主要使用其中 5 个构成各种条件，分支指令判断这些条件实现程序分支。从低位到高位依次是：进位标志 CF（Carry Flag）、奇偶标志 PF（Parity Flag）、调整标志 AF（Adjust Flag）、零标志 ZF（Zero Flag）、符号标志 SF（Sign Flag）、溢出标志 OF（Overflow Flag）。

（2）控制标志

IA-32 处理器只有一个控制标志：方向标志 DF（Direction Flag），该标志仅用于串操作指令中，控制地址的变化方向。

（3）系统标志

系统标志用于控制操作系统或核心管理程序的操作方式，应用程序不应该修改它们。例如，中断允许标志 IF（Interrupt-enable Flag）或简称中断标志，用于控制外部可屏蔽中断是否可以被处理器响应。再如，陷阱标志 TF（Trap Flag）也常称为单步标志，用于控制处理器是否进入单步操作方式。

8086 具有 9 个基本标志，后续处理器增加的标志主要用于处理器控制，由操作系统使用，在学习指令时将会涉及它们的具体用法。其中状态标志比较关键，它是汇编语言程序设计中必须特别注意的一个方面。

3. 指令指针寄存器

程序由指令组成，指令存放在主存储器中。处理器需要一个专用寄存器表示将要执行的指令在主存的位置，这个位置用存储器地址表示。在 IA-32 处理器中，存储器地址保存在 32 位指令指针寄存器 EIP 中。16 位 80x86 处理器使用低 16 位部分 IP。

EIP 具有自动增量的能力。处理器执行完一条指令，EIP 就加上该指令的字节数，指向下一条指令，实现程序的顺序执行。需要实现分支、调用等操作时要修改 EIP，其改变将引起程序转移到指定的指令执行。但 EIP 寄存器不能像通用寄存器那样直接赋值修改，是在执行控制转移指令（如跳转、分支、调用和返回指令）、出现中断或异常时被处理器赋值而相应改变。

4. 段寄存器

在程序中，有可以执行的指令代码，还有指令操作的各类数据等。遵循模块化程序设计思想，