高等学校21世纪计算机教材

IBM PC 80X86

汇编语言程序设计

姜媛媛　任卓谊　编著

冶金工业出版社

IBM PC 8086汇编语言程序设计

高等学校21世纪计算机教材

IBM PC 80X86 汇编语言程序设计

姜媛媛　任卓谊　编著

北　京

冶 金 工 业 出 版 社

2004

内 容 简 介

现代计算机的编程语言发展很快，高级语言的更新换代更是日新月异，本书结合汇编语言自身的优点，以目前广泛运用的IBM PC为背景，全面系统地介绍了80X86计算机汇编语言程序设计的基础知识、基本原理和程序设计的基本方法。

本书语言通俗易懂、概念精确、实例丰富、章节安排合理，既可作为高等院校计算机及相关专业的本、专科教材，也可以作为从事软件工程技术编程人员的参考书。

图书在版编目（CIP）数据

IBM PC 80X86汇编语言程序设计 / 姜媛媛等编著.
北京：冶金工业出版社，2004.6
ISBN 7-5024-3534-4

I. I... II. 姜... III. 汇编语言—程序设计
IV. TP313

中国版本图书馆CIP数据核字（2004）第045904号

出版人　曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑　戈兰
湛江蓝星南华印务公司印刷；冶金工业出版社发行；各地新华书店经销
2004年8月第1版，2004年8月第1次印刷
787mm×1092mm　1/16；　23.5印张；　545千字；　368页；　1-3500册
39.00元
冶金工业出版社发行部　电话：（010）64044283　**传真：**（010）64027893
冶金书店　地址：北京东四西大街46号（100711）　电话：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

汇编语言程序设计是计算机相关专业本、专科和研究生都必须学习和掌握的一门核心课程，也是学习操作系统、计算机接口技术和单片机技术的基础课程。

现代计算机的编程语言发展很快，高级语言的更新换代更是日新月异。尽管如此，许多计算机专业人士还是青睐于汇编语言。这是因为汇编语言是计算机系统提供给用户的最快、最有效且能对硬件直接编程的语言，而且它的编程模型是 CPU 内部的寄存器、内存单元、CPU 的芯片端口等。因此，掌握了汇编语言，就了解了计算机的内部结构和 CPU 的工作流程。另外，在运用汇编语言进行程序设计的过程中，需要用到大量的计算机原理知识，这样就在学习和运用过程中使专业技能得到了相应地提高。

本书以 Intel 80X86 系列微处理器为基础，系统地介绍了汇编语言的基本概念、基本原理和程序设计的基本方法。另外本书还采用了实例教学的方法，通过大量实例讲述了如何用汇编语言开发应用程序，并介绍了上机调试运行汇编源程序的方法。

二、本书结构

全书由 12 章和 5 个附录组成，内容如下：

第 1 章：概述。主要介绍了计算机系统的结构、汇编语言的相关知识和计算机中数据表示的方法及数据类型。

第 2 章：IBM PC 微处理器的结构及存储器组成。主要介绍了 IBM PC 微处理器基本结构、IBM PC 的存储器组织和堆栈。

第 3 章：寻址方式和指令系统。主要介绍了寻址方式和指令系统。

第 4 章：汇编语言程序结构。主要介绍了汇编语言的语句格式、伪指令、汇编语言的源程序结构和汇编语言的上机过程。

第 5 章：基本结构程序设计。主要介绍了顺序结构程序设计、分支结构程序设计和循环结构程序设计。

第 6 章：子程序设计及应用。主要介绍了子程序设计、主程序与子程序的参数传递、子程序的嵌套和递归、基本 DOS 功能子程序调用、多模块程序设计及汇编语言和高级语言的混合编程。

第 7 章：简单程序设计应用。主要介绍了串操作程序、代码与数制之间的转换、冒泡排序举例和多精度数运算举例。

第 8 章：高级汇编语言技术。主要介绍了宏汇编、重复汇编和条件汇编。

第 9 章：输入/输出和中断程序设计。主要介绍了输入/输出指令、查询传送方式、中断的相关知识、中断处理程序的设计和 BIOS 功能调用。

第 10 章：磁盘文件存取技术。主要介绍了传统文件管理方式和扩充文件管理方式。

第 11 章：彩色图形程序设计。主要介绍了单色与彩色显示控制、彩色显示适配器、

EGA/VGA 图形程序设计、计算机动画。

第 12 章：发声系统的程序设计。主要介绍了可编程内部定时器 8253/54、通用发声程序设计、乐曲程序设计。

附录 A：ASCII 字符表。

附录 B：DOS 功能调用。

附录 C：BIOS 功能调用。

附录 D：80X86/Pentium 指令系统。

附录 E：出错信息。

三、本书特点

本书的实用性很强，其语言叙述通俗易懂、概念准确、结构严谨、实例丰富，且课程编排由浅入深、由易到难、循序渐进，章节安排合理、前后衔接紧密、难易过渡自然，每章后均配有综合练习，书末还附有参考答案，以便于读者复习和检查学习效果。

四、适用对象

本书可作为高等院校计算机及相关专业的本、专科教材，也可以作为从事软件工程技术编程人员的参考书。

由于编写时间仓促、编者水平有限，书中不足之处在所难免，恳请广大读者批评指正。

虽然经过严格的审核、精细的编辑，本书在质量上有了一定的保障，但我们的目标是力求尽善尽美，欢迎广大读者和专家对我们的工作提出宝贵建议，联系方法如下：

电子邮件：service@cnbook.net

网址：www.cnbook.net

此外，**本书附送的电子教案可从该网站免费下载**，同时该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2004 年 3 月

目　录

第1章 概 述

学习汇编语言之前，首先应该对计算机系统及汇编语言的一些基础知识有所认识，熟悉和理解计算机系统软硬件的一些基本概念，了解和掌握计算机的硬件环境（即硬件编程模型）。本章先介绍计算机的系统结构，然后阐述汇编语言的一些基本概念，最后介绍计算机中的数据表示方法和数据类型。

1.1 计算机系统概述

1.1.1 硬件系统结构

计算机本质上是一种能按照程序对各种数据信息进行自动加工和处理的电子设备。计算机依靠硬件和软件的协同工作来执行给定的工作任务。一台完整的计算机系统由硬件系统和软件系统两大部分组成。计算机的硬件是人们所见到的物理实体，包括主机板、各种板卡、显示器等。软件是运行、开发、管理和维护计算机所需的各种程序的总和。

人们通常所说的微机系统，无论是以前的 8086、8088 和 80X86 还是现在的 Pentium Ⅲ、Pentium IV 微机，从硬件体系结构来看，采用的都是计算机的经典结构——冯·诺依曼结构。冯·诺依曼结构的主要特点包括以下 3 个方面：

（1）硬件系统由 5 部分组成，它们分别是运算器、控制器、存储器、输入设备和输出设备，其结构如图 1-1 所示。5 个部分之间既有分工又有合作，共同完成处理任务。

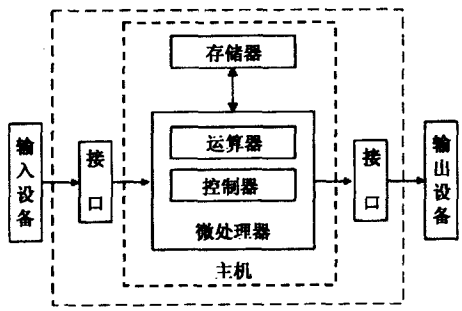

图 1-1 微型计算机硬件系统组成示意图

（2）计算机所处理的数据，包括数值、字符、声音、图形、图像和视频等，都是以二进制形式进行存储和处理的，描述反映数据存储位置的地址也是二进制形式。

（3）计算机以运算器和控制器（合称为中央处理单元）为中心，在控制器的作用下从存储器中依次读出指令和数据，在运算器中执行相应操作，最终结果也将在控制器的作用下写回存储器中。中央处理单元的工作过程就是反复地执行以下 3 步操作的过程。

① 取指令：从存储器中取出下一条要执行的指令。

② 指令译码：解释这条指令，确定这条指令所对应的操作，并向各部件发出相应的操作指令。

③ 执行指令：各部件实际执行相应的操作。

整个程序由一个程序计数器（即指令计数器）来控制指令的执行。每执行一条指令，程序计数器加 1。同时，控制器也具有判断能力，能根据程序运行的不同状态选择不同的操作流程。

运算器（ALU——Arithmetic Logical Unit）：运算器又称为算术逻辑单元，用来进行算术或逻辑运算以及位移循环等操作。通常情况下，参加运算的两个操作数，一个来自累加器 A（Accumulator），另一个来自内部数据总线，可以是数据寄存器 DR（Data Register）中的内容，也可以是寄存器阵列 RA 中某个寄存器的内容。运算结果往往也送回累加器 A 中暂存。

控制器（CU——Control Unit）：控制器负责统一指挥计算机各部分协调工作，能根据事先安排好的指令发出各种控制信号，以控制计算机各个部分的工作。例如：控制从存储器中读出数据，或将数据写入存储器中，按照程序规定的步骤进行各种运算和处理等，使计算机按照预定的工作顺序高速工作。控制器包括以下 3 个部分：

（1）指令寄存器 IR（Instruction Register）。

指令寄存器 IR 用来存放从存储器取出的将要执行的指令（实为其操作码）。

（2）指令译码器 ID（Instruction Decoder）。

指令译码器 ID 用来对指令寄存器 IR 中的指令进行译码，以确定该指令应执行什么操作。

（3）可编程逻辑阵列 PLA（Programmable Logic Array）（也称为定时与控制电路）。

可编程逻辑阵列用来产生取指令和执行指令所需的各种微操作控制信号。由于每条指令所执行的具体操作不同，所以每条指令将对应控制信号的某一种组合，以确定相应的操作序列。

运算器与控制器组成计算机的中央处理单元（CPU——Central Processing Unit）。CPU 是整个计算机系统的核心部件。它主要负责通过总线在自己和存储器及外部 I/O 之间传递数据，进行简单的算术或逻辑运算，如加、减、乘、除及与、或、非、取反、移位、循环等，通过简单的判断来控制程序的流程。

存储器（Memory）：是计算机的记忆部件，负责存储处理器要运行的程序和数据，这些程序和数据由指令序列组成，顺序存放在这里。通常存储器还用来存放一些程序中用到的数据、信息和中间结果。从存储器读出数据后，该单元的内容仍保持不变，可重复读取，直到存入新的数据。

输入/输出设备：一般包括一些 I/O 设备和存储器两类。其中 I/O 设备是负责计算机与外部传递信息的输入、输出设备，如显示器、键盘、打印机及数码设备等，而存储器主要是硬盘、光盘和磁带等可存储大容量信息的设备，通常称为外存。对外部设备的编程是汇编语言的重要应用之一。

根据冯·诺依曼体系结构，微机的工作过程可描述为：通过输入设备将程序和数据输入到微机中，并且转换成二进制数，然后存放在具有地址编号的存储器中。在运算器和控制器的作用下，存储在存储器中的指令被一条一条的读取出来，然后经过译码形成一系列的控制命令，在控制器的指挥下向各相关部件发出控制命令并执行相应操作。操作过程产生的结果又存储在存储器中，并可通过输出设备将存储在存储器中的运算结果以直观的形式输出。

在微机的工作过程中，实际上有3种信息在5个部分之间流动。

（1）数据信息：包括程序和数据，即事先准备好的，希望微机遵照一步步执行的指令及其相关的原始数据。

（2）控制信息：由控制器发出，控制微机各个部件按程序中各指令的要求完成预定的工作。

（3）地址信息：存放在存储器中的原始程序和数据，其每一单元必须有一个惟一的地址；运算的结果数据存储时也需要一个惟一的地址，这样才能保证微机正常工作。

这三种信息都由专门的通道（总线）来负责传送。总线分为以下3种：

① 数据总线（Data Bus）：用于传送程序等数据信息。

② 控制总线（Control Bus）：用于传送控制信息。

③ 地址总线（Address Bus）：用于传送数据的地址信息。

处理器读取一个存储单元的内容时，先通过地址总线发出一个地址，接下来发出存储器读控制信号到目标存储器读数据，最后将从存储器读出的数据，通过数据总线送入运算器中进行运算。运算结果在内存储器中，再通过写控制信号，将结果写回外存储器。在数据总线、控制总线、地址总线的共同作用下，微机系统的5个模块被联结成为一个有机的整体，它们之间相互协作，共同完成复杂的运算控制工作。工作原理如图1-2所示。

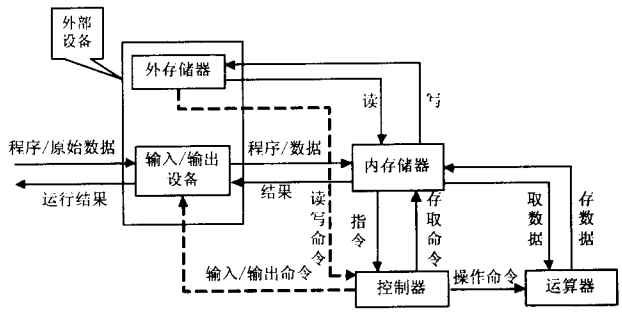

图1-2 计算机工作原理示意图

1.1.2 软件系统结构

计算机软件主要分为系统软件和应用软件两大类。其中系统软件又包括操作系统和应用软件开发服务程序。

应用软件是用户为解决科学计算、办公自动化、检测与实时控制等不同领域的任务所编制的应用程序。

系统软件是指不需用户干预的能生成、准备和执行其他程序所需的一组程序。

操作系统是一套复杂的系统程序，用于提供人机接口和管理、调度计算机的所有硬件与软件资源。它所包含的系统程序的具体分类尚不完全统一。其中，最为重要的核心部分是常驻监控程序。计算机开机后，常驻监控程序始终存放在内存中，它通过接收用户命令，并启动操作系统执行相应的操作。

操作系统还包括I/O驱动程序和文件管理程序，前者用于执行I/O操作；后者用于管理存放在外存中的大量数据集合。每当用户程序或其他系统程序需要使用I/O设备时，通常并不是由该程序执行操作，而是由操作系统利用I/O驱动程序来执行任务。文件管理程

序与 I/O 驱动程序配合使用，用于文件的存取、复制和其他处理。

应用软件开发服务程序包括各种高级语言翻译程序、汇编程序、文本编辑程序、系统程序库、连接程序和装入程序以及辅助编写其他程序的程序。

人是通过软件系统与硬件系统与计算机打交道的。通常，由人使用程序设计语言编制应用程序，在系统软件的干预下使用硬件系统。

1.2 汇编语言概述

本节介绍汇编语言的一些基本概念及用汇编语言编程所需要的基本知识。

1.2.1 汇编语言程序设计的一般概念

在引入汇编语言之前，先了解一下什么是机器语言，这有助于对汇编语言的理解。

1. 机器语言

机器指令是 CPU 能直接识别并执行的指令，它的表现形式为由 0、1 构成的二进制编码。从 1946 年到现在计算机的基本工作方式都采用冯·诺依曼的“存储程序”的工作原理，这里的“程序”指的就是机器指令。机器指令通常由操作码和操作数（即操作对象）两部分组成，操作码在前，操作数在后。操作码指出该指令所要完成的操作，即指令的功能。一台计算机可能有几十甚至上百条指令，每一条指令都有一个相应的操作码，计算机就是通过识别操作码来完成不同的操作。操作数指出参与运算的对象，以及运算结果所存放的位置等。

一台计算机的全部机器指令就是常说的计算机的指令系统。指令系统反映了这台计算机的基本功能。由于机器指令与 CPU 紧密相关，所以不同种类的 CPU 所对应的机器指令也就不同，而且它们的指令系统往往相差很大。但对同一系列的 CPU 来说，为了让各型号之间具有良好的兼容性，新一代 CPU 的指令系统必须包括先前同系列 CPU 的指令系统，即向下兼容，比如 80386 的指令系统就应兼容 8086/8088 指令集。只有这样，先前开发出来的各类程序在新一代 CPU 上才能正常运行，从而也保证了程序在同一系列 CPU 上的可移植性。

本书是以 Intel 80X86 为硬件基础介绍指令的，重点介绍 8086/8088 的汇编语言，由于指令系统的向下兼容性，有了 8086/8088 汇编语言程序设计的基础，在学习 80386 及以上汇编语言程序设计就非常容易了。

机器语言是用来直接描述机器指令、使用机器指令的规则及用二进制代码来表示机器指令的一种程序设计语言。机器语言是 CPU 能直接识别的惟一语言，即 CPU 能直接执行用机器语言描述的程序，而无需任何翻译程序。用机器语言描述的程序称为目的程序或目标程序。在本书中，为了书写方便，通常将机器语言用十六进制形式表示。

下面来看一个简单的机器语言程序片段，它的功能是计算存储器内两个数的和，把结果存入存储器的另一个单元，程序如下：

```
A1 00 00      ----------  取第一个加数放入累加器
03 06 02 00   ----------  将两个数相加，结果放入累加器
A3 04 00      ----------  将和放入目的存储单元
```

由此可看出，机器语言程序虽然执行效率高，但却难理解和读懂。用机器语言编写程

序是早期经过严格训练的专业技术人员的工作，普通的程序员一般难以胜任。而且用机器语言编写的程序难记忆、开发和调试，易出错和难维护，且不能直观地反映用计算机解决问题的基本思路。另外，不同型号 CPU 的指令集有较大差异，对应的机器指令也不同，所以难于移植，通用性不高。

正是由于机器语言编写程序有以上诸多的不便，现在几乎没有程序员用它编写程序了。

2. 汇编语言（Assembly Language）

虽然用机器语言编写程序有以上诸多的困难与不便，但编写出来的程序执行效率却高。由于 CPU 严格按照程序员的要求去做，无需任何的额外操作。所以，在保留“程序执行效率高”的前提下，人们就开始着手研究一种能大大改善程序可读性的编程方法。

为了改善机器指令的可读性，选用了一些便于记忆并能反映机器指令功能的符号来代表该机器指令的操作码，而不再关心机器指令的具体二进制编码。与此同时，也把 CPU 内部的各种资源（如 CPU 的寄存器，存储单元的地址等）符号化，使用该符号名也等于引用了该具体的物理资源。CPU 在执行时只需要编译程序将其编译成机器语言即可。

如此一来，令人难懂的二进制机器指令就可以用通俗易懂的、具有一定含义的符号指令来表示了，于是汇编语言就有了雏型。现在，这些具有一定含义的符号被称为助记符。助记符一般都是能够说明指令功能的英语词汇或词汇的缩写。如：传送指令 MOV，加法指令 ADD，调用指令 CALL 等。用指令助记符、符号地址等组成的符号指令称为汇编格式指令（或汇编指令）。

汇编语言是汇编指令集、伪指令集和使用它们规则的统称。伪指令是在程序设计时所需要的一些辅助性说明指令，它不对应具体的机器指令，有关内容在以后的各章节中会有详细叙述，在此不展开介绍。

用汇编语言编写的程序称为汇编语言程序，或汇编语言源程序，在本教材中或特定的环境下，也可简称为源程序。汇编语言程序要比用机器指令编写的程序容易理解和维护。

就上面的程序段而言，用汇编语言书写如下：

```
MOV  AX, VAR1          A1 00 00
ADD  AX, VAR2   ⟹      03 06 02 00
MOV  VAR3, AX          A3 04 00
```

因此说汇编语言是符号化的机器语言，也就是说汇编语言的每一条指令都与一条机器指令相对应。

1.2.2 汇编程序

用汇编语言编写的程序大大提高了程序的可读性，但失去了 CPU 能直接识别的特性。例如用汇编语言书写的指令：

```
MOV AX, BX
```

CPU 不会知道这几个字符所表达出来的功能，但程序员一看就知道即要求 CPU 把寄存器 BX 的值传送给寄存器 AX。

把机器指令符号化增加了程序的可读性，但存在如何让 CPU 知道程序员的用意，并按其要求完成相应操作的问题。解决问题的办法是翻译程序，它能把汇编语言编写的源程序翻译成 CPU 能识别的机器指令序列。这里，该翻译程序被称为汇编程序，而翻译过程被称

之为汇编，如图 1-3 所示。

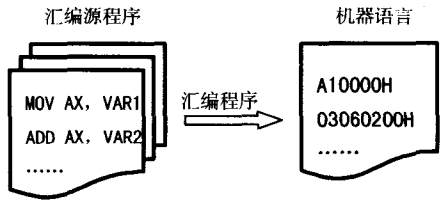

图 1-3 汇编程序翻译过程的示意图

从图中可看出汇编程序能把左边汇编语言源程序翻译成右边的机器指令序列。其中，把汇编语言指令“MOV AX, VAR1”和“ADD AX, VAR2”分别转换成机器指令 A10000H 和 03060200H，而后者都是 CPU 能直接识别的，所以可执行它们。

目前，常用的汇编程序有：Microsoft 公司的 MASM、Borland 公司的 TASM 和 DEBUG 等。

1.2.3 汇编语言的特点

一方面，汇编语言指令是用一些具有相应含义的助记符来表达的，所以，它要比机器语言容易掌握和运用。另一方面，它要直接使用 CPU 的资源，相对高级程序设计语言来说，它又显得难掌握。

汇编语言程序归纳起来主要有以下几个特性。

1. 与机器的相关性

汇编语言指令是机器指令的符号表示，而不同类型的 CPU 有不同的机器指令系统，也就有不同的汇编语言。所以，汇编语言程序与机器有着密切的关系，如图 1-4 所示。

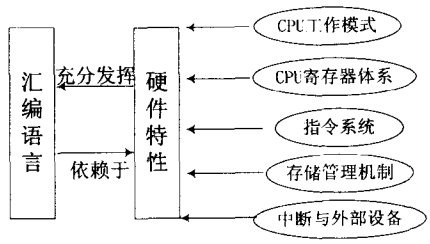

图 1-4 汇编语言与硬件的关系示意图

由于汇编语言程序与机器的相关性，所以，除了同系列、不同型号 CPU 之间的汇编语言程序有一定程度的可移植性之外，其他不同类型（如：单片机和微机等）CPU 之间的汇编语言程序是无法移植的，也就是说，汇编语言程序的通用性和可移植性要比高级语言程序要低。

2. 执行的高效率

正因为汇编语言有“与机器的相关性”的特性，程序员用汇编语言编写程序时，可充分发挥自己的聪明才智，对机器内部的各种资源进行合理的安排，让它们始终处于最佳的使用状态。这样做的最终效果就是：程序的执行代码短，执行速度快。

现在，高级语言的编译程序在进行寄存器分配和目标代码生成时，也都有一定程度的优化（在后续课程《编译原理》的有关章节会有详细介绍）。但由于所使用的“优化策略”

要适应各种不同的情况，所以这些优化策略只能在宏观上，不可能在微观上、细节上进行优化。而用汇编语言编写程序几乎是程序员直接在写执行代码，程序员可以在程序的每个具体细节上进行优化，这也是汇编语言程序执行效率高的原因之一。

3. 编写程序的复杂性

汇编语言是一种面向机器的语言，其汇编指令与机器指令基本上一一对应，所以，汇编指令也同机器指令一样具有功能单一、具体的特点。要想完成某件工作（如计算：A+B+C等），就必须安排CPU的每步工作（如：先计算A+B，再把C加到前者的结果上）。另外，在编写汇编语言程序时，还要考虑到机器资源的限制、汇编指令的细节等等。

由于汇编语言程序要安排运算的每一个细节，这就使得编写汇编语言程序比较繁琐、复杂。一个简单的计算公式或计算方法，也要用一系列汇编指令一步一步来实现。

4. 调试的复杂性

在通常情况下，调试汇编语言程序要比调试高级语言程序困难，其主要原因有四点：

（1）汇编语言指令涉及到机器资源的细节，在调试过程中，要清楚每个资源的变化情况。

（2）程序员在编写汇编语言程序时，为了提高资源的利用率，可以使用各种实现技巧，而这些技巧完全有可能破坏程序的可读性。这样，在调试过程中，除了要知道每条指令的执行功能，还要清楚它在整个解题过程中的作用。

（3）高级语言程序几乎不显式地使用“转移语句”，但汇编语言程序要用到大量的、各类转移指令，这些跳转指令大大地增加了调试程序的难度。如果在汇编语言程序中也强调不使用“转移指令”，那么，汇编语言程序就会变成功能单调的顺序程序，这显然是不现实的。

（4）调试工具落后，高级语言程序可以在源程序级进行符号跟踪，而汇编语言程序只能跟踪机器指令。不过，现在这方面也有所改善，TD（Turbo Debug）等软件也可在源程序级进行符号跟踪了。

1.2.4 汇编语言的使用场合

综上所述，正因为汇编语言有如上的特点，所以，我们在选用时应扬长避短，充分发挥其优点并根据实际的应用环境来选用。

下面简单列举几个汇编语言适用的领域：在要求执行效率高、反应快的领域，如：操作系统内核，工业控制，实时控制等；系统性能的瓶颈或在大程序中被频繁使用的子程序或程序段；在软件与硬件资源密切，软件要直接和有效控制硬件的场合，如：外部设备驱动程序等；在对执行时间和存储容量要求较高的场合，如：家用电器的计算机控制功能等；最后是在没有合适的高级语言的场合也可以考虑使用汇编语言。

1.2.5 汇编语言的学习

由于汇编语言是介于高级语言和低级语言之间的一种语言，是面向机器的，学习起来不易掌握，所以要学好汇编语言应做到以下几点：

（1）对CPU内部结构和工作原理应非常熟悉，应熟练地掌握CPU的指令系统（包括伪指令），数据存储组织方式，这是学好汇编语言的前提。