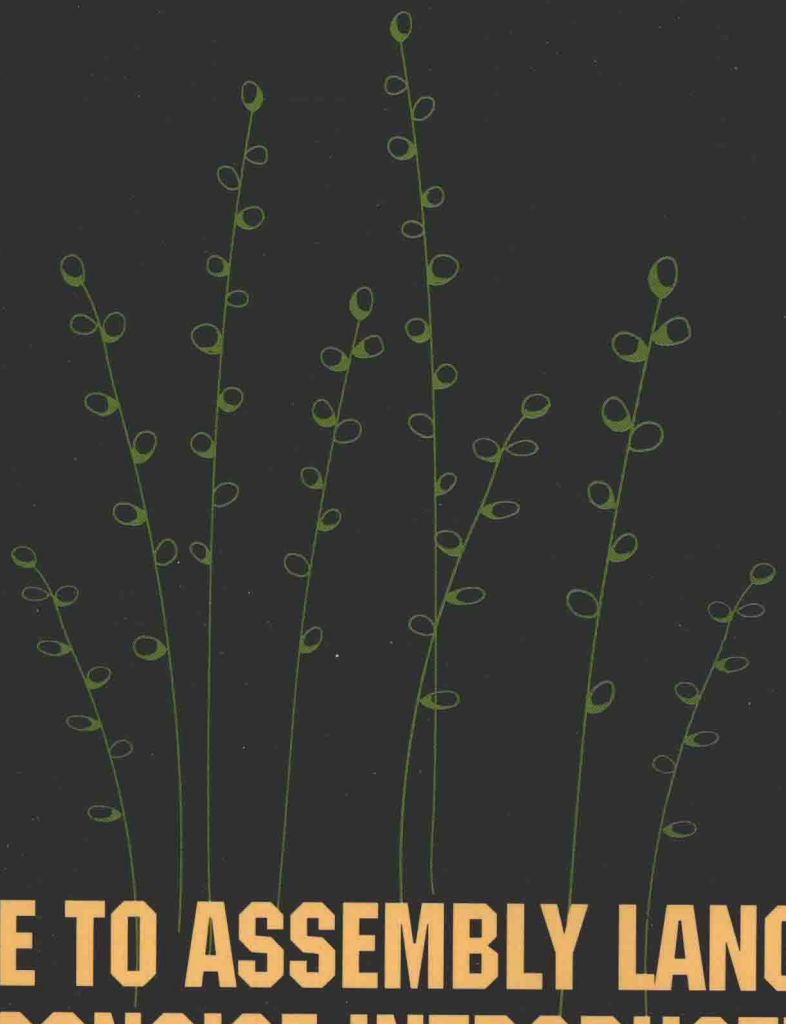
Springer

世界著名计算机教材精选

汇编语言基础教程

James T. Streib 著

远红亮 等译



GUIDE TO ASSEMBLY LANGUAGE A CONCISE INTRODUCTION

清华大学出版社

世界著名计算机教材精选

汇编语言基础教程

James T. Streib 著
远红亮 等译

清华大学出版社
北京

图书在版编目（CIP）数据

汇编语言基础教程 /（美）斯特雷布（Streib, J. T.）著；远红亮等译. —北京：清华大学出版社，2014
书名原文：Guide to assembly language: A concise introduction
世界著名计算机教材精选
ISBN 978-7-302-37058-*

Ⅰ. ①汇… Ⅱ. ①斯…②远… Ⅲ. ①汇编语言－程序设计－教材 Ⅳ. ①TP313

中国版本图书馆 CIP 数据核字（2014）第 145049 号

责任编辑：龙启铭
封面设计：傅瑞学
责任校对：梁　毅
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：13.75　　字　　数：341 千字
版　　次：2014 年 12 月第 1 版　　印　　次：2014 年 12 月第 1 次印刷
印　　数：1～3000
定　　价：29.00 元

产品编号：043109-01

译 者 序

汇编语言是计算机科学技术专业的专业基础课程，同时也是电子、通信及自动控制等相关专业的计算机技术课程的内容。汇编语言是很多相关课程（如数据结构、操作系统、微机原理等）的重要基础。汇编语言是面向机器的程序设计语言，它是机器语言便于记忆和理解的符号形式（又称为助记符）。它是一种功能很强的程序设计语言，也是利用计算机所有硬件特性并能直接控制硬件的语言。汇编语言是一种与硬件紧密相关的程序设计低级语言，在不同的设备中，汇编语言对应着不同的机器语言指令集。一种汇编语言专用于某种计算机系统结构，而不像许多高级语言，可以在不同系统平台之间移植。本书选择了广泛应用的 Intel 架构，希望读者花最少的时间学会使用 Intel 汇编语言来编写程序。通过编写程序让读者了解和掌握更多的基于 Intel 32 位处理器的计算机体系结构方面的知识，更好地理解高级程序设计语言与低级程序设计语言之间的关系。

本书涵盖了汇编语言的基础知识，它既可以作为汇编语言单学期课程的独立教材来使用，也可以作为计算机组成原理或计算机体系结构课程的辅助教材。本书作者的基本想法是，希望读者能够更加快速地学会如何使用汇编语言进行程序编写，希望读者在最短的时间内，学会如何编写出逻辑上正确的程序来。因此，本书在第 1 章概要介绍了汇编语言及寄存器方面的内容，后续章节主要介绍如何使用汇编语言进行输入输出操作、如何进行算术运算、如何实现选择结构、迭代结构、逻辑运算、移位运算、堆栈、过程、宏、数组和字符串处理，章节中贯穿了大量示例，帮助读者理解学习相关概念及用法；同时，在大多数汇编语言代码段的前面，会先给出相应的 C 语言代码，这样可以帮助读者更好地理解高级语言与低级语言之间的相互关系。同时，本书对寄存器的用法进行了精简，使用了类 C 的语句简化输入/输出操作，同时使用高级控制语句。这些特色能够帮助读者快速地开始程序编写。本书中很多控制语句没有使用高级结构，这有助于读者更好地理解这些控制语句是如何被实现的。另外，每章结尾包含了一个或多个完整的程序示例来说明和应用该章节所介绍的相关知识和概念。每章小结作为各章内容的复习素材，帮助读者梳理总结相关知识点。全书共有 10 章和 5 个附录，非常全面地介绍了汇编语言程序设计基础内容，认真阅读本书，你一定会收获颇丰。

本书主要由远红亮翻译，张艳艳、康文萍、李建伟、林杰、蔡玲玲、刘燕、王亚红、王翔、李克新、王岚、邓勇、朱园园、张鹏、林青和王立等也参与了部分章节内容的翻译，在此对大家表示衷心感谢。

由于译者知识有限，书中的翻译错误和不妥之处在所难免，读者若发现翻译处理不当之处，欢迎批评指正。

前　言

目标

本书希望达到的目标是让读者花最少的时间学会使用Intel汇编语言来编写程序。除此之外，希望读者能够通过编写程序来了解和掌握更多的基于Intel 32位处理器的计算机体系结构方面的知识，以及高级程序设计语言与低级程序设计语言之间的关系。

需求

在过去，许多系会开设两门独立的课程：一门是汇编语言程序设计（有时称为计算机系统原理），另一门是计算机组成原理与体系结构。在今天繁重的课程体系之下，有时在计算机科学课程体系中只有一门关于计算机组成原理与体系结构的课程，因此过去两门课程的内容会包含在今天一门课程中来介绍。这样导致的不幸结果是，不能充分学习关于汇编语言程序设计方面的知识。

汇编语言的重要性

虽然社会对于汇编语言程序员的需求在不断减少，但是对于程序员来说，理解汇编语言的需求却从来没有削减过，之所以需要学习使用汇编语言编写程序，原因如下：

- 有时候单纯阅读汇编语言本身是不够的，最好动手使用汇编语言编写一段程序才能更好地理解相关内容（编写的代码不需要非常复杂或者难度系数特别高，简单的程序就可以达到帮助学习相关知识的目的）。
- 虽然某些高级程序设计语言包含有低级语言的特性，但是有时候使用汇编语言编写的程序才能在运行时间和内存占用上实现更高的效率。
- 使用汇编语言编写的程序同机器语言程序优点相同，而且汇编语言程序更容易编写。另外，程序员能够通过编写汇编语言程序更加直接地学习到关于计算机系统、组成原理和体系结构方面的相关知识。
- 低级语言程序设计的相关知识能够帮助程序员更好地理解高级程序设计语言是如何被设计实现的以及各种相关编译器的构造概念。

与其他计算机组成原理和汇编语言教程的对比

大多数计算机组成原理方面的教程只有很少的章节介绍汇编语言，这种情况会导致这些教程难以充分地覆盖汇编语言的多方面内容。有时候这些教程还会使用非真实的汇编语言进行概念的讲解，也就是说，它们使用虚构的汇编机器语言进行相关概念的介绍，虽然这种做法在理解一些基本概念的时候还是可以的，但是对于学生来说，很难从中学习并领会到真实汇编语言的重要概念。

另外，还有一部分汇编语言教程着重介绍语言细节方面的内容，这些内容很容易填满整个学期的课程，而且几乎都可以连续安排两个学期来学习了。基于同样的原因，不幸的是，其中一些容易理解的汇编语言教程可能也不是学习汇编语言编写程序的最佳选择，因

此，这些种类的教程都不是最佳的学习教程。

本书不打算去满足上述两类教程中任何一类的需求，因为这样会使得本书与它们归于同一类别。本书着重于介绍汇编语言编程的基础内容，这使得本书既可以作为独立的教程供学生学习，也可以作为热门的计算机组成原理课程的辅助教程供学生参考。

本书的特色

本书的主要目的是让学生能够更加快速地学会如何使用汇编语言进行程序编写。为实现这个目的，本书的特色之一是简化寄存器的用法，使用类 C 的语句简化输入输出的用法以及使用高级控制语句。这些特色能够帮助读者快速地开始程序编写工作，且能够将读者先前从其他计算机科学课程中学过的相关概念用于其中。另外，很多控制语句没有使用高级结构，这会让读者更好地理解这些控制语句是如何实现的。再者，在大多数汇编语言代码前面先给出相应的 C 语言代码，这样可以帮助读者更好地理解高级语言与低级语言之间的相互关系。每章结尾的特色内容如下：

- 使用一个或多个完整的程序示例来说明和应用该章节所介绍的相关知识和概念。
- 每章小结虽然不能作为读者阅读各个章节内容的替代读物，但是它们可以用作为每章内容的复习素材，帮助读者筹备相关的小测验或考试。
- 复习题包含有各种类型的练习题，从简答题到程序设计题都有。标记有*号的练习题，它们的答案可以在附录 E 中找到。

各章与附录内容简介

如果把本书作为计算机组成原理课程的辅助教材，其内容可能会与计算机组成原理教程的内容有一些重复的部分。例如，大多数汇编语言教程都会先开始介绍二进制运算，该内容在低级语言课程中是相当重要的部分。但是，假如本书与计算机组成原理教程一起使用的话，可以肯定的是很多相关概念在计算机组成原理教程中都已经介绍过了。因此，本书直接从教授学生如何编程开始，二进制的相关内容作为必要的基础知识进行介绍与回顾。但是，如果把本书作为单独的教程进行学习的时候，附录 B 中介绍的二进制、十六进制、进制之间的转换、逻辑运算与算数运算的详细内容可能就需要首先学习一下了。如下是各章与附录内容的概要：

第 1 章概要介绍汇编语言以及通用寄存器方面的内容。

第 2 章特别使用 C 语言中的 scanf 和 printf 指令来介绍汇编语言中的输入与输出操作。

第 3 章介绍汇编语言中的算数运算，包括加法、减法、乘法、除法以及运算优先级。

第 4 章介绍在汇编语言中的如何实现选择结构，比如 if-then 结构、if-then-else 结构、嵌套 if 结构以及 case（switch）结构。

第 5 章介绍迭代结构，包括 pre-test 结构、post-test 结构，以及迭代 loop 结构与嵌套 loop 结构。

第 6 章介绍逻辑、移位、算数移位、循环移位以及堆栈指令。

第 7 章将讨论过程，并介绍宏，以及条件汇编。

第 8 章介绍数组、顺序检索和选择排序。

第 9 章介绍字符串、字符串指令、字符串数组以及字符串比较。

第 10 章以探索的视角来观察学习机器语言，也可以将这部分内容作为计算机组成原理

某些知识的入门学习或者作为计算机组成原理教程的补充材料进行学习。

附录 A 介绍如何安装 Visual C++和 MASM 并使用它们汇编程序。

附录 B 介绍二进制与十六进制之间的相互转换，以及逻辑运算和算术运算。本书的前三章用到了一些二进制和十六进制的知识，因此可以等后续课程中用到相关内容的时候再来阅读本附录即可。不过，第 6 章需要用到二进制数值和逻辑运算方面的知识。如果读者之前没有学习过这方面的知识，本附录的内容应该在学习第 6 章之前先读一读，本附录的内容是一个很好的入门材料；如果读者之前已经学习过数值系统的内容，那么可以把这里的内容作为复习材料来参考。另外，如果读者在其他先修课程、同期课程或本课程的其他参考文献中接触过这方面的知识，那么本附录的内容可以跳过不读。

附录 C 是术语表。术语表中关于术语的描述不能替代本书中关于术语的完整全面描述，术语表的内容主要是为了方便快速查阅相关术语以及提醒该术语的基本意义。如果需要了解该术语的完整描述，索引部分标注了该术语在本书中的具体页码，读者可以到书中查阅该术语的详细解释。

附录 D 总结了本书中介绍过的汇编语言指令。

附录 E 给出了每章末尾以及附录 B 末尾练习题中带有*号练习题的参考答案。

内容范围

本书涵盖了汇编语言的基础知识，它既可以作为汇编语言单学期课程的独立教材来使用，也可以作为计算机组成原理或计算机体系结构课程的辅助教材。在编写任何一本教材的时候，都会遇到这样的问题，必须要决定这本教材需要包含哪些内容、不需要包含哪些内容、需要强调哪部分内容以及需要弱化哪部分内容。本书也不例外，本书并不会包含汇编语言内容的方方面面，也不会把所有指令的每一个子主题都介绍一遍。比如，16 位运算、浮点数运算、Windows 编程等等，这一类内容可由教师自行掌握是否需要补充讲授。不过，对读者来说，最需要的是能够在最短的时间内，学会如何编写出逻辑上正确的程序来，这也是本书的基本想法。

之所以选择 Intel 架构是基于它被广泛应用的特点，而选择 MASM（Microsoft Assembler，微软汇编程序）是因为在该汇编程序中可以使用大量的高级控制结构。注意 Java 是 Oracle 公司的注册商标或旗下产品，Intel 386 与奔腾是 Intel 公司的注册商标，而 Visual Studio、Visual C++与 MASM 是微软公司的注册商标。

读者

希望本书的读者已经学完了两个学期的高级程序设计语言的入门课程，比如 C、C++或 Java。尽管有些读者可能在一个学期之后就会用到本书，但是多一个学期高级程序设计课程的学习会很好地帮助理解本书的内容，因为这里的内容涉及到高级程序设计技能。

致谢

本书作者非常感谢本书的编辑 Wayne Wheeler 的帮助；感谢本书的评审，Albion 大学的 Mark E. Bollman，Iowa 大学的 James W. Chaffee，Loras 大学的 Brenda Tuomi Litka，Illinoins 大学的 Takako Soma，DePaul 大学的 Curt M. White，感谢他们提出的宝贵意见；感谢 Illinions 大学计算机科学与技术专业的学生，感谢他们在课堂学习中对本书所有内容的细致检查；

特别感谢我的妻子 Kimberly A. Streib 及其儿子 Daniel M. Streib，感谢他们的耐心；最后谨以此书献给我的母亲 Doris G. Streib 以及我的姐姐 Lynn A. Streib。

反馈

由于时间有限，错误之处在所难免。欢迎任何形式的评论、指正和建议，可以联系下面的邮箱进行反馈。另外，每章末尾程序的完整复制以及所有重要的修正内容都可以到如下的网站上查找。

Illinois 大学
Jacksonville, Illinois
October 2010

James T. Streib
E-mail: jtstreib@ic.edu
Web 网站: http://www2/jtstreib/guide

目　录

第 1 章　变量、寄存器与数据移动

1.1　引　　言

高级程序设计语言，比如 C、C++和 Java，它们更类似于自然语言，因此使用高级程序设计语言能够更容易地编写和阅读程序。而低级程序设计语言更接近于机器语言，且高级程序设计语言与低级程序设计语言之间存在着一对多的关系，而语言转换软件，比如编译软件或解释软件，可以将一条高级语言指令转换成多条低级语言指令。对于一台具体的计算机来说，它的本机语言是一套低级程序设计语言，称它为机器语言。机器语言是用多个 0 或 1 编写的。具体来说，Intel 微处理器上的机器语言肯定与其他厂商微处理器上的机器语言是不一样的，因此不同厂商之间的机器语言是不能够相互转换使用的。

使用机器语言编写程序绝对是一件令人头疼且容易出错的事情。为了避免使用 0 和 1 的组合来编写程序，取而代之的汇编语言体现出了它的优势，汇编语言使用短语（缩写）来表示指令，使用变量名称来表示存储空间的位置。汇编语言指令与机器语言指令之间存在着一对一的关系。使用汇编语言可以比较容易地编写程序，可以避免很多种由于使用机器语言编写程序带来的麻烦。而使用汇编语言编写程序与使用高级语言编写程序相比，它的优势体现在，可以更加清楚地了解到计算机的体系结构，编写出更加高效的程序，运行速度更快而且更加节省存储空间。

正如编译程序将高级语言程序转换成低级语言程序一样，汇编程序将把由汇编语言编写的程序转换为机器语言程序。而某些新型编译程序可以把高级语言程序（比如 Java 程序）转换成中间语言程序（比如字节码），这些中间语言程序可以被解释成为机器语言程序，最终的结果都是机器语言代码在计算机上面运行。图 1.1 说明了机器语言是如何产生的。

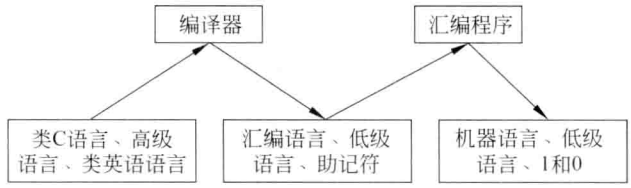


图 1.1　机器语言程序由高级语言程序与汇编语言程序转换而来

有很多种汇编程序可以完成将汇编语言程序转换为 Intel 机器语言程序的工作，不过，本书选用了 MASM（Microsoft Assembler，微软汇编程序）。如何安装汇编程序、如何编辑汇编程序以及如何汇编及运行程序可能一部分读者已经在学校学习过，或者由同事或朋友演示过；如果读者在学习本书的时候希望在个人计算机上面安装学习汇编程序，而又没有这方面的知识和经验的话，可以参见附录 A 的相关说明和指导。

在学习一种新的程序设计语言来编写程序的时候，无论要学习的语言是高级语言还是低级语言，最好的方式都是由一个简单的程序来开始。通常在学习一门高级语言的时候，

第一个学习的程序都是人尽皆知的“Hello World”程序，通过这个程序，程序员可以学会基本的编辑输入、正确的编译程序和运行程序。不幸的是，在学习低级语言编写程序的时候，编写“Hello World”程序不是一个好的开始，因为低级语言涉及的输入输出（I/O）操作要复杂得多。因此，本书先从基本的汇编语言知识开始学习，然后再学习涉及输入输出操作的程序编写，以检验前面学习到的知识以及程序实现方面的正确性。

1.2 第一个程序

下面第一个要实现的汇编语言程序在功能上同如下的 C 语言程序是一样的，它定义了两个变量，并为第一个变量赋值，然后将第一个变量的内容赋值给第二个变量：

```
int main(){
    int num1,num2;
    num1=5;
    num2=num1;
    return 0;
}
```

下面的汇编语言程序在逻辑上等价于如上所示的 C 语言程序。尽管乍一看，下面的程序比较难懂，但是它很好地体现出了汇编语言程序的基本结构和格式，是一个好的学习开始点：

```
        .386
        .model flat, c
        .stack 100 h
        .data
num1    sdword ?        ; 第一个数值
num2    sdword ?        ; 第二个数值
      .code
main    proc
        mov num1,5      ; 将 num1 的值初始化为 5
        mov eax,num1    ; 将 num1 的值装载到寄存器 eax 中
        mov num2,eax    ; 将寄存器 eax 的值存储到 num2 中
        ret
main    endp
        end
```

在上面的代码段中，首先需要弄清楚的问题是，其中一部分代码是汇编指令，其余的代码是程序指令。后面将会对这两种指令进行详细的介绍，这里可以简单地理解为，程序指令告诉中央处理单元（CPU）该做什么，而汇编指令告诉编译程序该做什么。运算符类似于汇编指令，它告诉汇编程序具体的程序指令该做什么。

上述程序段开头的.386 是一个汇编指令，它表明该程序段将被汇编成能够在 Intel 386 系列或更高级的计算机（比如奔腾计算机或 64 位计算机）上运行的程序。尽管这里可以指定更老旧的处理器型号，但是.286 或更老旧的 16 位处理器会缺少许多 32 位.386 处理器所

拥有的特性。另外也可以指定更加新型的处理器，不过由于本书中并没有大量介绍新的程序指令，而且使用.386 汇编指令可以让汇编后的程序运行在更老的处理器上。

汇编指令.model flat 表明程序使用保护模式，即程序将使用 32 位的地址，它能够使用 4GB 的内存空间。虽然有几种过去常用的地址解析格式，但是目前这种保护模式已经应用的相当普及，它更加容易理解，能够表示更多的内存地址。汇编指令.model 中的 C 表示该程序可以与 C 或 C++程序进行连接，且需要运行在 Visual C++环境中。

汇编指令.stack 以十六进制（参见附录 B）的形式表示堆栈的大小，这里的汇编指令表示堆栈的大小应该达到十六进制数 100 个字节大小或者十进制数 256 个字节大小。第 6 章将详细介绍堆栈的知识。汇编指令.data 和.code 稍后将进行介绍，而 proc 汇编指令表示一个过程，这里的过程名称为 main。虽然过程名称可以使用其他的名字，但是这里的 main 类似于 C、C++或 Java 程序中的 main，它能够让该汇编程序独立于其他程序而运行。指令 ret 与 C 和 C++语言中的 return 0 语句是一样的。main endp 标号与指令表示一个过程在这里结束，end 指令表示程序结束。

在过去，不同的汇编语言每一列所表示的汇编语言指令的字段是各式各样的。尽管字段放在哪一列的规则没有进行严格的限制，但是按照习惯进行按列对齐会使最终的程序代码便于阅读。

按照从左向右的顺序，一条指令的四列或四个字段分别是标号、操作字段(opcodes)、操作数字段和注释字段。第一个字段通常是预留给变量名或用于跳转到其他指令的标号用的。

第二个字段通常用于操作字段（opcodes），它表示了可执行的程序指令或汇编指令。第三个字段通常与第二个字段之间有一个空格，后面会接上 0～3 个操作数。最后一个字段是一个可选字段，用于添加注释的。请注意并没有限制注释必须写在一行的第四个字段，注释可在一行的任何地方开始，但是必须要注释前面加分号。

举例来说，图 1.2 给出了上一个程序的部分代码段。请注意标号字段、操作字段和注释字段通常排成 1 列，而操作数字段通常紧接着操作字段后一个空格处开始，不需要特别严格按照列来对齐。

```
标记    操作码 操作数     注释
        .data
num1    sdword ?          ; 第一个数值
num2    sdword ?          ; 第二个数值
        .code
main    proc
        mov num1,5        ; 将num1的值初始化为5
        mov eax,num1      ; 将num1的值装载到寄存器eax中
        mov num2,eax      ; 将寄存器eax的值存储到num2中
        ret
```

图 1.2　标号、操作字段、操作数字段、与注释字段

如图 1.2 所示，汇编语言程序主要由两个部分组成，数据段和代码段，分别由.data 和.code 汇编指令标注。1.3 节将介绍数据段，然后再介绍代码段。

1.3 变量声明

在上面程序的数据段中，声明了两个变量，分别为 num1 和 num2，变量名分别在两行的标号字段给出。变量名的命名规则同高级程序设计语言基本一样，略微有点差别。同高级语言类似，变量名必须以字母打头，后面可以跟着字母或数字。另外，也可以在变量名的任意位置包含特殊字符_、@或$，但是通常情况下要避免使用这三个字符。与 C、C++和 Java 语言不一样的地方是，汇编语言的变量名是不区分大小写的，比如变量 cat 和 CAT 所表示的是同样的内存空间位置。变量名的最大长度为 247 个字符，但是通常情况下变量名由 1～10 个字符组成。表 1.1 给出了一些合规的与不合规的变量名。

在声明变量的时候，操作字段上写的汇编指令是 sdword，它表示带符号双字，长度为 32 位，与 Visual C++中 int 变量是一样的。字的比特位表示二进制数字，其中 1 个比特位可以存储 1 个二进制数字 0 或 1，8 个比特位称为 1 个字节。在 Intel 处理器中，一个字由两个字节或 16 个比特位组成，双字由 4 个字节或 32 个比特位组成。如果读者之前没有接触过比特位、字节或二进制数字的相关概念，或者需要重新熟悉一下这方面的知识，可以参阅附录 B。

表 1.1 有效与无效变量名

有效的、合法的变量名	无效的、不合法的变量名
auto	1num
num1	7eleven
z28	57chevy

表 1.2 给出了变量声明中的一些类型，以及这些类型对应的比特位大小，另外还给出了这些比特位中能够表示的数值范围。为了简单起见，本书中只使用符号双字类型来表示正整数和负整数，以及字节类型来表示字符。

表 1.2 类型、比特位数、数值范围

类型	所占比特位数	表示范围（包含）
sdword	32	–2147483648～+2147483647
dword	32	0～+4294967295
sword	16	–32768～+32768
word	16	0～+65535
sbyte	8	–128～+127
byte	8	0～+255

上面程序中两个变量声明部分的第三个字段（或操作数字段）是一个问号，它表示该变量在声明的时候汇编程序不对其进行初始化操作。当然，也可以使用一个数值来代替问号，这样汇编程序在进行汇编的时候将对该变量进行初始化操作。如下是 C 语言程序中变量的初始化：

```
int num3 = 5;
```

与上面 C 语言程序对应的汇编语言程序如下：

```
num3 sdword 5 ; num3 将被初始化为 5
```

最后，在第四个字段是程序注释，程序注释也可以出现在一行的前面部分，不管哪种情况，注释部分要由分号开始。两种类型的注释方法都可以在汇编语言中使用，而出现在代码行前面的注释方法比较常见，行右边的注释通常只对该行程序进行解释。由于高级程序设计语言中的选择结构和迭代结构的缩进问题，注释通常不会写在一行代码的右边部分。而对于汇编语言来说，它通常不缩进，因此一行代码的右边有足够的空间来放置程序注释。

字符数据变量的声明与数值数据变量的声明类似。要声明两个变量，第 1 个变量名称为 grade1，它不需要初始化；第 2 个变量名称为 grade2，它需要初始化为'A'，使用 C/C++/Java 语言可以这样写：

```
char grade1;
char grade2='A';
```

在汇编语言中，类似于前面使用 sdword 类型来声明数值变量，这里使用 byte 来声明字符变量。注意在声明字符数据的时候，既可以使用单引号，也可以使用双引号。

```
grade1 byte ?
grade2 byte 'A'
```

另外，也可以把字符串声明为字节数组。虽然关于字符串处理的指令将在第 9 章进行介绍，但是有时候需要输出程序消息或提示用户程序输入信息。如下所示，字符串可以声明为单独的字符：

```
grades byte 'A','B','C'
```

除非字符串的每一个字符需要单独进行处理，否则通常情况下，为了简单起见，把字符声明为一个整体字符串：

```
name byte 'Abe'
```

1.4 节将了解到，字符串结尾通常带有二进制的 0，它占用 1 个字节，用于表示字符串结束。通常在输出语句中会用到这种情况，声明如下所示：

```
name byte 'Abe',0
```

字符串声明还有很多种方式，如前所说，字符串处理指令也有很多种，但是它们都比较复杂，不过上面的相关知识足以满足阅读第 2 章内容的需求了。另外，也可以声明整型数组或 sdword 数组，不过处理数组的相关指令暂时不介绍，第 8 章将对其进行详细介绍。