图书在版编目（CIP）数据

汇编语言程序设计/吴向军，罗源明，刘广旋编著.
北京：高等教育出版社，2002.7
计算机专业本专科通用
ISBN 7-04-010674-4

Ⅰ.汇… Ⅱ.①吴…②罗…③刘… Ⅲ.汇编语言-程序设计-高等学校-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2002）第 023121 号

责任编辑 董建波 **封面设计** 王凌波 **版式设计** 胡志萍 **责任校对** 陈 荣
责任印制 宋克学

汇编语言程序设计
吴向军 罗源明 刘广旋 编著

出版发行	高等教育出版社	**购书热线**	010-64054588
社 址	北京市东城区沙滩后街 55 号	**免费咨询**	800-810-0598
邮政编码	100009	**网 址**	http://www.hep.edu.cn
传 真	010-64014048		http://www.hep.com.cn
经 销	新华书店北京发行所		
排 版	高等教育出版社照排中心		
印 刷	北京地质印刷厂		
开 本	787×1092 1/16	**版 次**	2002 年 7 月第 1 版
印 张	23.75	**印 次**	2002 年 7 月第 1 次印刷
字 数	490 000	**定 价**	27.30 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

汇编语言程序设计是计算机及电子信息类有关专业一门重要的基础课程，是培养学生直接使用计算机硬件资源能力的一门课程。它不仅能帮助学生进一步理解计算机组成原理课程中的各种概念，而且还为其他课程如：操作系统、接口与通信技术和计算机控制技术等课程提供必要的预备知识。该课程在计算机学科课程设置中起着承上启下的作用。

本书以 Intel 80X86CPU 的指令系统为介绍对象，以并行推进的方式介绍其 16 位和 32 位 CPU 中的指令功能。选用这种方式，不仅希望利用 16 位指令系统简单好学的特点来加深对 32 位指令的理解，而且也希望读者在学习过程中能达到相互参考、相互对照的目的。

本书在内容的组织上做了大胆的尝试，把学习高级语言的习惯和汇编语言的特点结合起来，按“硬件资源、变量定义、指令系统、编程”的顺序来安排相应的内容。另外，还结合 MASM V6.11 的编程环境介绍了一些新的伪指令，运用这些伪指令可使汇编语言源程序具有良好的结构化特征。

本书在介绍完汇编语言编程的基本方法和基本技巧后，安排了一章的篇幅来介绍浮点数的定义格式、浮点指令的功能和编程方法。该部分的内容不仅使计算机组成原理课程中对浮点数格式的描述更加具体化，而且也使读者对高级语言中的各种数据类型与低层实现之间的联系有一个更清晰的认识。

本书是新世纪网络课程建设工程课题——“汇编语言程序设计”的文字教材，用于网上本科教学活动是它的任务之一，所以在线学习者的接受能力是编者取舍教学内容的因素之一。再加上目前的教学改革，在授课总学时数缩减的大环境下，汇编语言课程的周学时已减少为 3 学时，甚至更少。出于这两方面因素的考虑，在本书中，编者没有对 32 位的编程技术给予详细的讲述，但编者相信读者在掌握了本书的基本内容后，对自学 32 位的编程技术会有很大的帮助。

在本书的编写安排上，罗源明副教授编写了第一～三章，吴向军副教授编写了第四～十三章以及附录等内容，刘广旋助工对每章的配套练习做了大量的工作。本书的统筹安排和最后定稿均由吴向军副教授负责。

在编写本书的初期，姜丽帆副教授对编写计划给予了积极的肯定，也提出了一些建设性的意见；初稿完成后，李宏新高级工程师审阅了全部内容，并在内容的取舍和一些重要内容的组织安排上给出了修改意见。在此，向他们表示衷心的感谢。

由于编者能力和知识不断更新等因素，书中不足和谬误之处在所难免，恳请广大读者批评指正（联系 E-Mail:issxjwu@zsu.edu.cn）。

编 者

2001 年 12 月

目　录

第一章 预备知识

本章介绍汇编语言的一些基本概念，给出一些用汇编语言编程所需要的基本知识。

1.1 汇编语言的由来及其特点

1.1.1 机器语言

机器指令是CPU能直接识别并执行的指令，它以二进制编码的形式来表示。机器指令通常由操作码和操作数两部分组成，操作码指出该指令所要完成的操作，即指令的功能；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。

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。对同一系列的CPU而言，为了使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。只有这样，先前开发出来的各类程序才能在新一代CPU上正常运行。

机器语言是用来直接描述机器指令和使用机器指令规则等的语言。它是CPU能直接识别的惟一一种语言，即CPU能直接执行用机器语言描述的程序。

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不易读、出错率高、难以维护，也不能直观地反映用计算机解决问题的基本思路。

鉴于用机器语言编写程序有以上诸多的不便，现在已很少有程序员这样编写程序了。

1.1.2 汇编语言

虽然用机器语言编写程序对程序员有很高的要求并存在着许多不便，但编写出来的程序执行效率高，CPU严格按照程序员的要求去做，没有多余的额外操作。所以，在确保“程序执行效率高”的前提下，人们开始着手研究一种能大大改善程序可读性的编程语言。

为了改善机器指令的可读性，选用了一些能反映机器指令功能的单词或词组来代表该机器指令，而不再关心机器指令的具体二进制编码。与此同时，也把CPU内部的各种资源符号化，在编写程序时，使用该符号名相当于引用了该具体的物理资源。

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，这就是汇编语言的雏形。这些具有一定含义的符号现在称为助记符，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的规则的统称。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,在本书中简称为源程序。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。

1.1.3 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: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,但引起了如何让 CPU 知道程序员的用意并按照其要求完成相应操作的问题。解决该问题就需要一个翻译程序,它能把用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,称该翻译程序为汇编程序。图 1.1 是一次翻译过程的示意图。

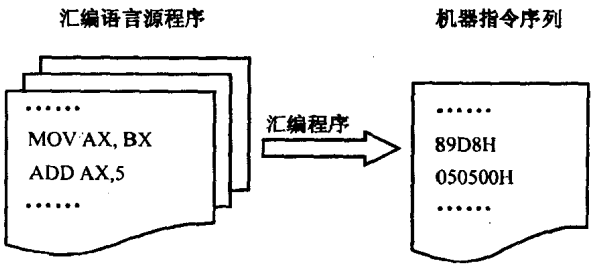

图 1.1 汇编语言指令翻译为机器指令的示意图

从图中不难看出,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, BX"和"ADD AX, 5"分别转换成机器指令 89D8H 和 050500H,而后者 CPU 都能直接识别并执行。

目前,常用的汇编程序有 MASM、TASM 和 DEBUG 等。

1.1.4 汇编语言程序的主要特点

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;但另一方面,它要直接使用 CPU 的资源,相对高级程序设计语言而言,它又显得较难掌握。

汇编语言程序归纳起来大概有以下几个主要特性。

1．与机器相关性

汇编语言指令是机器指令的一种符号表示，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。

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外，其他不同类型（如，小型机和微机等）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低。

2．执行的高效率

正因为汇编语言有“与机器相关性”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，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可执行代码，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。要想完成某项工作（如计算 A + B + C 等），就必须安排 CPU 的每一步工作（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，还要考虑机器资源的限制、汇编指令的细节和限制等。

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 4 点：

1）汇编语言指令涉及机器资源的细节，调试时要清楚机器资源的变化情况；

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，还要清楚它在整个程序中的作用；

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，这些转移指令大大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）汇编语言的调试工具落后。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，CV（CodeView）、TD（Turbo Debug）等软件也可在源程序级进行符号跟踪了。

1.1.5 汇编语言的应用领域

综上所述，汇编语言的特点明显，其诱人的优点直接导致其严重的缺点："与机器相关"和"执行的高效率"导致其可移植性差和调试困难。所以，在选用汇编语言时要根据实际的应用环境，尽可能避免其缺点对整个应用系统的影响。

下面简单列举几个领域以示说明，但不要把它们绝对化。

1. 适用的领域

- 要求执行效率高、反应快的领域，如操作系统内核、工业控制、实时系统等；
- 系统性能的瓶颈或频繁被使用的子程序或程序段；
- 与硬件资源密切相关的软件开发，如设备驱动程序等；
- 受存储容量限制的应用领域，如家用电器的计算机控制功能等；
- 没有适当的高级语言开发环境。

2. 不宜使用的领域

- 大型软件的整体开发；
- 没有特殊要求的一般应用系统的开发等。

1.2 数据的表示和类型

用汇编语言进行程序设计时，程序员可以直接访问内存，因此对数据在存储器内的表示形式要有清晰的认识。下面，只介绍最基本的数据表示知识，为本课程的学习做必要的知识准备。

1.2.1 数值数据的表示

1. 二进制

在计算机内，数值是用二进制来表示的。在书写二进制时，为了区别，在数据后面紧跟一个字母 B。

二进制的一般表示形式为 $b_{n-1}\cdots b_0B$，其代表的十进制数值为 $b_{n-1}2^{n-1}+\cdots+b_02^0$。

数据的二进制表示形式简单、明了，但它书写起来比较长，所以，通常情况下，在程序中不直接用二进制来书写具体的数值，而改用八进制、十进制或十六进制。

2. 八进制

八进制是一种二进制的变形，3 位二进制数可变为 1 位八进制数，反之亦然。八进制的表示元素为 0、1、…、7。在书写时，为了区别，在数据后面紧跟一个字母 Q，如 1234Q、7654Q、54Q 等都是八进制数。

八进制数在程序中的使用频率不高。

3. 十进制

十进制是最常见的一种数据表示形式,它的基本元素为 0、1、…、9。在书写时,为了区别,在数据后面紧跟一个字母 D。在程序中经常用十进制来表示数据。

4. 十六进制

十六进制是另一种二进制的变形,4 位二进制数可变为 1 位十六进制数,反之亦然。十六进制的基本元素为 0、1、…、9、A、B、…、F(字母小写也可以),其中,字母 A、B、…、F 依次代表十进制数的 10、11、…、15。

在书写时,为了区别,在数据后面紧跟一个字母 H(表 1.1 给出了进制及其字符表示)。当十六进制数的第一个字符是字母时,在第一个字符之前必须添加一个"0",如 100H、56EFH、0FFH、0ABCDH 等都是十六进制数。

十六进制数在程序中的使用频率很高。

表 1.1 进制及其字符表示

进 制	字 符	例 子
二进制	B/Y①	1010B、1011B
八进制	Q/O	1234Q、311Q
十进制	D/T	1234D、512D
十六进制	H	1234H、1011H

5. 数的补码表示法

在计算机内,为了表示正、负数,并便于进行各种算术运算,对有符号数采用二进制的补码表示形式。补码的最高位用来表示正、负数:0 表示正数,1 表示负数。正数的补码是其自身的二进制形式,负数的补码是把其正数的二进制编码变"反",再加 1 而得。

6. 二进制数的符号扩展

在汇编语言中,经常要对字/字节的数据进行操作。当把"字节"转换成"字",或把"字"转换成"双字"时,就需要进行符号扩展。符号扩展的具体操作就是把已知信息的最高位扩展到所有更高位。

例 1.1 把 8 位补码 01011010、10101100 分别扩展成 16 位补码。

解:根据符号扩展的含义,"字节→字"的具体扩展结果如图 1.2 所示:

例 1.2 把 16 位补码 0101101111001010、1010111101011011 别扩展成 32 位补码。

解:根据符号扩展的含义,"字→双字"的具体扩展结果如图 1.3 所示:

① 字符 Y、O 和 T 是宏汇编 MASM 系统所增加的进制表示符。

图 1.2 8 位扩展为 16 位的示意图

图 1.3 16 位扩展为 32 位的示意图

7. *n* 位二进制的表示范围

n 位二进制所能表示的无符号整数的范围：$0 \leqslant x \leqslant 2^n - 1$。

n 位二进制所能表示的有符号整数(补码表示)的范围：$-2^{n-1} \leqslant x \leqslant 2^{n-1} - 1$。

在汇编语言中，常用到 *n* 为 8 和 16 时的数值范围：

- $n=8$ 时，无符号整数的范围：0～255，有符号整数的范围：－128～127；
- $n=16$ 时，无符号整数的范围：0～65 535，有符号整数的范围：－32 768～32 767。

8. BCD 码

通常习惯用十进制表示数据，但计算机是用二进制来表示数据的，这就需要进行数值进制之间的转换。把每位十进制数转换成二进制的编码，简称为 BCD 码(Binary Coded Decimal)。

BCD 码是用 4 位二进制编码来表示 1 位十进制数。这种编码方法有多种，但常用的编码是 8421BCD 编码，如表 1.2 所示。这种 BCD 编码实际上就是 0～9 的“等值”二进制数。

用 BCD 码进行进制的转换时，是要求在两种进制的表现形式上快速转换，而不是要求在“数值相等”的含义上快速转换。

表 1.2 8421BCD 编码列表

十进制数字	8421BCD 码	十进制数字	8421BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

例 1.3 求十进制数 2 000 的 BCD 编码和其二进制数。

解：2 000 的 BCD 编码是把每位上的数 2、0、0、0 分别转换为其对应的 BCD 编码：0010、0000、0000 和 0000，把它们合在一起就是 2 000 的 BCD 编码：0010 0000 0000 0000。

十进制数 2 000 的二进制数是 11111010000，它们在数值上是相等的。

1.2.2 非数值数据的表示

计算机除了具有进行数值计算能力之外，还具有进行非数值计算的能力。现在，后者的应用领域已远远超过了前者，如文字处理、图形图像处理、信息检索、日常的办公管理等。所以，对非数值信息的编码就显得更加重要。

1. ASCII 码

ASCII 码(American Standard Code for Information Interchange)是目前应用极其广泛的一种信息编码，许多计算机系统都是采用它为字符进行编码。它是一种 7 位二进制编码，如表 1.3 所示是 ASCII 码的具体编码方案。

表 1.3 ASCII 码的编码方案

高位 / 低位	000	001	010	011	100	101	110	111
0000	NUL	DEL	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

在上表中,对学习本课程有用的主要信息有:

• 字符0～9是连续编码的,其ASCII码的低4位就是该字符在十进制数中对应的数值;

• 小写字母的ASCII码比大写字母的ASCII码大,对应字母的编码之间相差20H。

当然从ASCII码表中还可看出其他有用信息,还有扩展的ASCII码等知识,但这些内容对学习本课程的帮助不明显,故不再叙述。

2. 汉字编码

ASCII码是针对英文字母、数字和其他特殊字符进行编码的,它不能用于对汉字的编码。要想用计算机来处理汉字,也必须先对汉字进行适当的编码。我国在1981年5月对6 000多个常用的汉字制定了交换码的国家标准,即GB2312－80。该标准规定了汉字交换用的基本汉字字符和一些图形字符,它们共计7 445个,其中汉字有6 763个。该标准给定了每个字符的二进制编码,即国标码。

有关汉字编码的详细信息,可参阅其他有关书籍,在此不再介绍。

1.2.3 基本的数据类型

汇编语言所用到的基本数据类型有字节、字、双字等,这些数据类型在以后的章节中都有相应的类型说明符。下面对它们进行最基本的描述。

1. 字节

一个字节由8位二进制组成,其最高位是第7位,最低位是第0位,如图1.4(a)所示。在表示有符号数时,最高位就是符号位。

通常情况下,存储器按字节编址,读写存储器的最小信息单位就是一个字节。

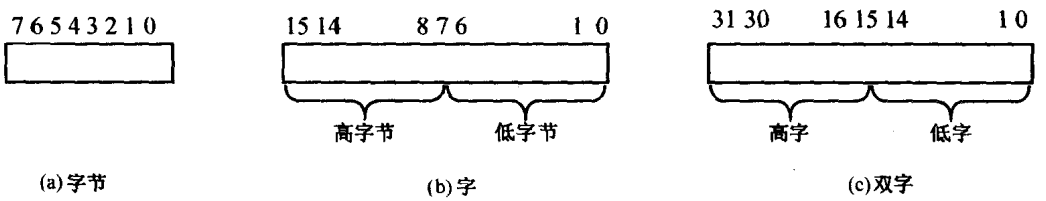

图1.2 各类基本数据类型示意图

2. 字

由2个字节组成一个字,其最高位是第15位,最低位是第0位。高8位称为高字节,低8位称为低字节,如图1.4(b)所示。

字节和字是汇编语言程序中最常用的两种数据类型,也是最容易出错的数据类型。

3. 双字

用2个字(4个字节)来组成一个双字,其高16位称为高字,低16位称为低字,如图

1.4(c)所示。双字有较大的数据表示范围,它通常是为了满足数据的表示范围而选用的数据类型,也可用于存储远指针。

4. 四字

由 4 个字(8 个字节)组成一个四字类型,它总共有 64 个二进制位,当然,也就有更大的数据表示范围,但在汇编语言中很少使用该数据类型。

5. 十字节

由 10 个字节组成一个十字节类型,它总共有 80 个二进制位。在汇编语言中很少使用该数据类型。

6. 字符串

字符串是由若干个字节组成的,字节数不定,通常每个字节存储一个字符。该数据形式是汇编语言程序中经常使用的一种数据形式,它并没有 C 语言中的规定:以 ASCII 码 0 为字符串的结束符。

习　题

1. 汇编语言的主要特点有哪些?

2. 汇编语言适用于哪些领域? 在哪些领域使用不合适?

3. 在汇编语言中,如何表示二进制、八进制、十进制和十六进制的数值?

4. 在计算机中,如何表示正、负数? 在保持数值大小不变的情况下,如何把位数少的二进制数值扩展成位数较多的二进制数值?

5. 在 ASCII 表,字符"0"～"9"与数值 0～9 之间编码规律是什么? 大写字母和小写字母之间的编码规律是什么?

6. 汇编语言中的基本数据类型有哪些? 它与高级程序设计语言(如 C 语言)中的数据类型的对应关系是什么?

第二章　CPU 资源和存储器

计算机的硬件资源是用汇编语言编程所必须要了解的重要内容，因为汇编语言允许、也需要程序员直接使用这些硬件资源，只有这样才能编写出高效的源程序。

在汇编语言中，需要访问的硬件资源主要有 CPU 内部资源、存储器和 I/O 端口。本章着重讲解 CPU 内部寄存器的命名、功能及其常见的用途，还要介绍存储器的分段管理模式、存储单元地址的表示法以及其物理地址的形成方式。

2.1　寄 存 器 组

寄存器是 CPU 内部重要的数据存储资源，是汇编程序员能直接使用的硬件资源之一。由于寄存器的存取速度比内存快，所以在用汇编语言编写程序时，要尽可能充分利用寄存器的存储功能。

寄存器一般用来保存程序的中间结果，为随后的指令快速提供操作数，从而避免把中间结果存入内存再进行存取的操作。在某些高级语言（如 C 语言）中，也可以定义变量为寄存器类型，这是提高寄存器利用率的一种可行的方法。

另外，由于寄存器的个数和容量都有限，不可能把所有中间结果都存储在寄存器中，所以要对寄存器进行适当的调度。根据指令的要求，如何安排适当的寄存器，避免操作数过多的传送操作是一项细致而又周密的工作。

由于 16 位/32 位 CPU 是微机 CPU 的两个重要代表，所以在此只介绍它们内部寄存器的名称及其主要功能。

2.1.1　寄存器组

1. 16 位寄存器组

在 16 位 CPU 系统中，汇编语言程序可访问的寄存器有（如图 2.1 所示）：

4 个数据寄存器：AX、BX、CX 和 DX，每个 16 位寄存器又可分为 2 个 8 位寄存器；

2 个变址寄存器：DI 和 SI；

2 个指针寄存器：SP 和 BP；

4 个段寄存器：ES、CS、SS 和 DS；

1 个标志寄存器：FLAG；

1 个指令指针寄存器：IP。