汇编语言程序设计B
(Assembly Language Programming)

李显巨

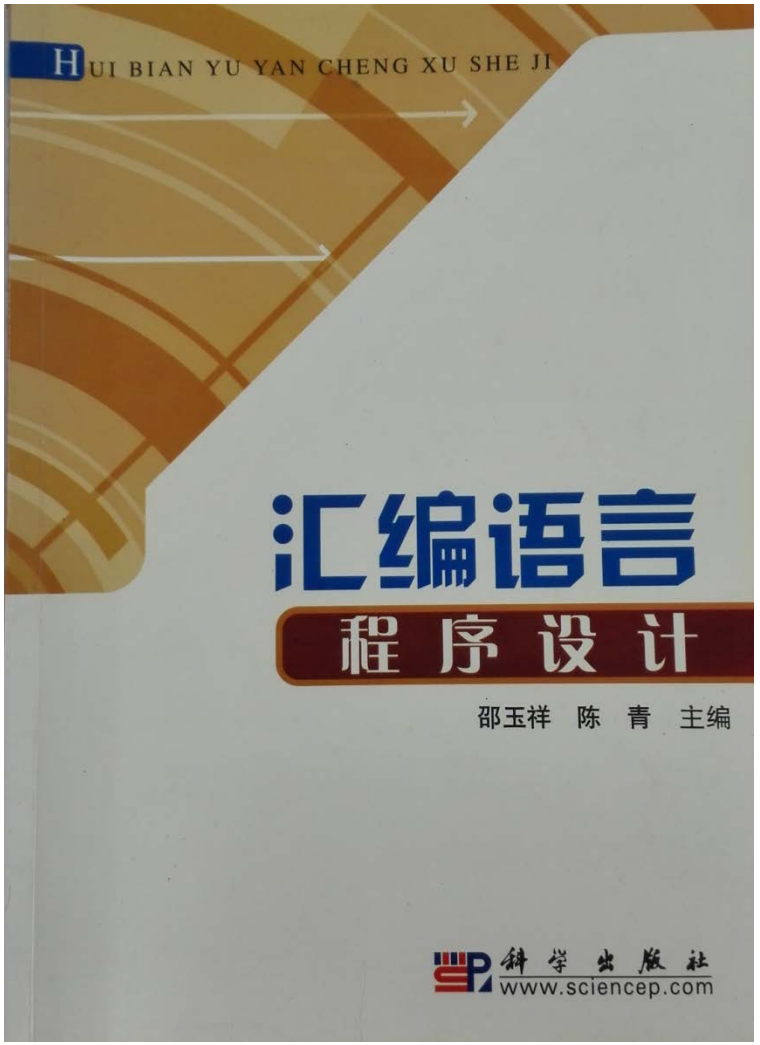
15171468643

ddwhlxj@cug.edu.cn

使用教材及参考书

使用教材

✓邵玉祥主编，**汇编语言程序设计**，科学出版社



参考书

✓王庆生主编，**汇编语言程序设计教程**，人民邮电出版社

✓王爽著，**汇编语言(第三版)**，清华大学出版社

使用教材及参考书


21世纪高等教育计算机
汇编语
教程
A Guide for As
Programming
王庆生 主编
刘维平 刘政怡 徐怡 陈洁 张燕
从指令执行看计算机工作原理
用程序实例训练和提高编程能力
以注重基础结合实验为编排指导
人民邮电出版社
POSTS & TELECOM PRESS
王爽 著
汇编语言(第3版)
Assembly L
清华大学出版社
计
PEARSON
汇编语言 基于x86处理器
Assembly
Win3
汇编
Win32汇编语言程序设计
高等院校精品课程系列教材·省级
32位汇编语言
程序设计
第2版
钱晓捷 编著
32位汇编语言程序设计 第2版
32-bit Assembly
Language Programming
Second Edition

考核方式

总成绩

✓**期末考查成绩(70%)+平时成绩(30%)**

平时成绩

✓**考勤**：16次课，**随机考勤10次**(时间、点名人数随机)

✓**课堂提问**：鼓励主动回答问题(**加分，主动算两次**)

✓**作业**：4次

✓**上机实习**：上机考勤、程序检查

某学期平均分(67人)：79，3不及格

14人加分绩点突破(80以上)

课程引言

已学计算机语言

✓C++…

找工作需掌握的语言

✓C++、Java、Python…

为什么还要学汇编？？

汇编语言的独特优势

✓控制计算机的硬件，充分发挥硬件性能

✓目标代码简短，占用内存少，执行速度快

课程引言

汇编语言的用途

✓存储器容量有限，但需要快速和实时响应的场合，比如**仪器仪表，家用电器和工业控制**设备中

行业性单位

课程引言

汇编语言的用途

✓在系统程序的核心部分，以及与系统硬件频繁打交道的部分。比如**操作系统的核心程序段**、**I/O接口电路的初始化程序**、**外部设备的底层驱动程序**，以及**频繁调用的子程序**、**动态连接库**、**某些高级绘图程序**、**视频游戏程序**等等。

✓还可以用于**软件的加密和解密**、**计算机病毒的分析和防治**，以及**程序的调试和错误分析**等

课程引言

课程目的

✓通过学习汇编语言，能够**加深对计算机原理和操作系统等课程的理解**。通过学习和使用汇编语言，能够**感知、体会和理解机器的逻辑功能**，向上为理解**各种软件系统的原理**，打下技术理论基础；向下为掌握**硬件系统**的原理，打下实践应用基础。

总结

✓要**学深计算机专业**，请学汇编语言。

要想**开发系统底层**，请学好汇编语言。

其掌握程度是**衡量计算机专业学生水平的标准**。

第1章 汇编语言基础

- ✓1.1 机器语言与**汇编语言**
- ✓1.2 **数制**与数制之间的转换
- ✓1.3 **有符号数**与无符号数
- ✓1.4 原码、反码、**补码**
- ✓1.5 ASCII码

1.1 机器语言与汇编语言

计算机语言 交流信息的工具

✓机器语言、汇编语言、**高级语言** 优缺点

机器语言

✓**机器指令**：是指
每一条机器指令

✓**机器指令格式**：
进制代码

CPU指令集

型号	Intel 酷睿i7 8700K	Intel 酷睿i7 8700	Intel 酷睿i7 8086K（限量版）
价格/商家	¥2599 2018-05-10 70商家在售	¥2399 2018-05-10 71商家在售	¥3699 2018-08-07 37商家在售
技术参数			

Baidu百科 指令集 进入词条

指令集是存储在CPU内部，对CPU运算进行指导和优化的硬程序。拥有这些指令集，CPU就可以更高效地运行。Intel主要有x86，EM64T，MMX，SSE，SSE2，SSE3，SSSE3 (Super SSE3)，SSE4A，SSE4.1，SSE4.2，AVX，AVX2，AVX-512，VMX等指令集。AMD主要是x86，x86-64，3D-Now!指令集。

1.1 机器语言与汇编语言

提问：计算机字长、操作系统位数是否相等？
4GB/16TB

- ✓**机器指令长度**：计算机字长。
- ✓**机器指令**也常常被称为**硬指令**，它是面向机器硬件的，即每台计算机都规定了自己所特有的、一定数量的基本指令，这批指令的全体即为计算机的指令系统，这种机器指令的集合就是**机器语言**。
- ✓**机器语言**是最低级的语言，是用**二进制代码**表示的计算机能**直接识别和执行**的一种机器指令的集合。用机器语言编写的、计算机能直接执行的程序称为机器语言程序。

1.1 机器语言与汇编语言

机器指令举例

✓将变量x的内容加2，结果仍保留在x存储单元中，其中变量x的**偏移地址**为1000H，且为**字类型**存储单元。指令码如下：

✓10000011
00000110
00000000
00010000
00000010

第1、2行中的两个8位二进制数是**操作码**，表示要进行“加”操作；

第3、4行中的两个8位二进制数指出了第一个加数（称目的操作数）所存放的偏移地址**1000H**，相加的结果也送入该存储单元中。

第5个字节的8位二进制数指出了第二个加数（称源操作数）是2。

机器语言的优缺点

优点

✓ 执行速度快，直接执行指令码。

缺点

✓机器指令是用许多二进制数表示的，用机器语言编程必然**很繁琐**，非常消耗精力和时间，**难记忆，易弄错**，并且**难以检查程序和调试程序**，工作效率低。

作为用户该怎么办？

汇编语言的产生

- ✓因为机器指令是用二进制表示的，编写程序相当麻烦，而且写出的程序也难以阅读和调试，所以为了克服这些缺点，人们就想出了用“助记符”表示机器指令的**操作码**，用“变量”代替操作数的存放地址，另外还可以在指令前加上**标号**，用来代表该**指令的存放地址**等。

- ✓这种用符号书写的、其主要操作与机器指令**基本一一对应的**、并遵循一定语法规则的计算机语言就是汇编语言，用汇编语言编写的程序称为**汇编源程序**。

汇编语言的实质

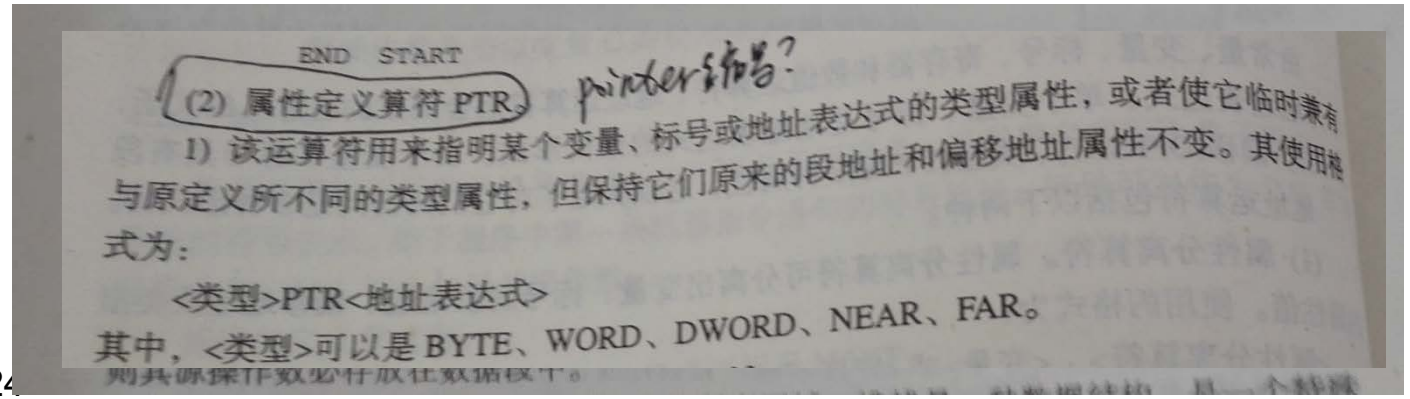
- ✓汇编语言也是低级语言，
- ✓是面向机器的语言，
- ✓实质是机器语言的符号化。

例题改写

- ✓前面的例题用**汇编语言**来写：

  ADD WORD PTR DS：[1000]，2

- ✓ADD为加指令的**助记符**
- ✓DS：[1000]表示在当前**数据段中**、**偏移地址**为1000H存储单元中的内容，是目的操作数
- ✓WORD PTR说明了这个目的操作数是**字类型**，而源操作数是2，相加的结果送入目的操作数所在的原存储单元中



END START

(2) 属性定义算符 PTR

1) 该运算符用来指明某个变量、标号或地址表达式的类型属性，或者使它临时兼有与原定义所不同的类型属性，但保持它们原来的段地址和偏移地址属性不变。其使用格式为：

<类型>PTR<地址表达式>

其中，<类型>可以是 BYTE、WORD、DWORD、NEAR、FAR。

汇编与编译

- ✓由于汇编语言是为了方便用户而设计的一种符号语言，因此，用它编写的源程序并不能直接被计算机所识别，必须将它“翻译”成由机器指令组成的机器语言程序后，计算机才能执行。

- ✓这种由汇编源程序经过“翻译”转换成的机器语言程序也称为目标程序，目标程序中的二进制代码（即机器指令）称为目标代码，一般以OBJ作为文件扩展名。

- ✓这个“翻译”工作又称为“**汇编**”，在高级语言中又称为“编译”。

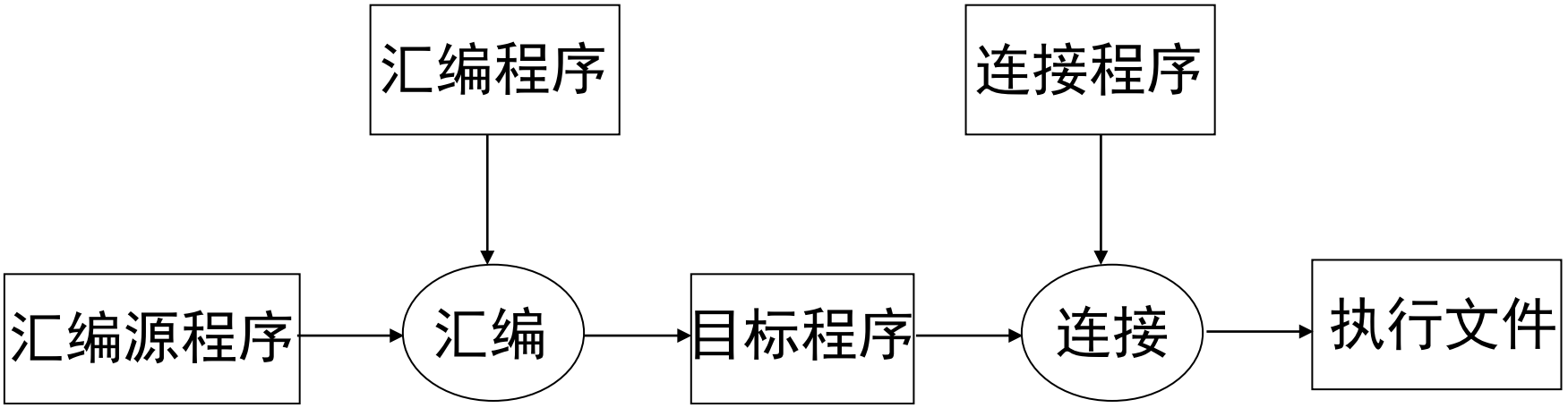
装配与连接

- ✓汇编源程序经汇编后生成的目标代码，还不能直接交给计算机去执行，还需要通过连接程序的装配才具备可执行性，装配结果称为“执行文件”，一般以EXE作为文件的扩展名。

- ✓同时，连接程序还具有把多个目标程序装配在一起的功能，或者把目标程序与预先编写好的子程序库中的子程序连接在一起，构成较大的执行文件。

程序调试

✓汇编语言源程序、汇编程序、目标程序、连接程序、执行文件关系图



汇编语言

- ✓在汇编程序过程中，为了让汇编程序正确地完成翻译工作，必须告诉汇编程序，**源程序从什么位置开始存放**，汇编到**什么位置结束**，**数据放在什么位置**，**数据的类型**是什么，**留多少内存单元作为临时存储区**等。

- ✓这就需要源程序中有一套告诉汇编程序如何进行汇编工作的命令，成为**伪指令**（或汇编控制命令）。

- ✓指令助记符、语句标号、数据变量和伪指令及它们的使用规则构成了**整个汇编语言的内容**。

宏汇编

- ✓由于汇编语句与机器指令相对应，编写也相当麻烦。为提高编程效率，现代计算机提供了宏汇编程序。
- ✓**宏汇编程序**：允许程序员用一个名字（宏指令）来代替程序中重复出现的一组语句。然后用**宏指令名**和**不同参数**进行调用。
- ✓Intel 8086宏汇编程序：MASM.EXE

  Microsoft Macro Assembler

汇编语言的优点

- ✓能够最大限度地**发挥硬件的功能**。
- ✓**高速度、高效率**：能够直接访问与硬件相关的存储器或 I/O 端口；
- ✓能够对**关键代码进行更准确的控制**，避免因线程共同访问或者硬件设备共享引起的死锁；
- ✓能够根据特定的应用**对代码做最佳的优化**，提高运行速度；
- ✓能够**不受编译器的限制**，对生成的二进制代码进行完全的控制；

汇编语言的缺点

- ✓机器相关性，可移植性差
- ✓可维护性差，难以调试
- ✓必须对机器非常了解

汇编有16位汇编，32位汇编，64位汇编；

本课程重点介绍的是Intel 8086/8088宏汇编语言，因此以16位为主，32位为辅。

1.2 数制与数制之间的转换

- ✓ **数制**是人们常用的一种计数方法，任何一种数制都涉及以下三个问题。

计数符号

- ✓**二进制**数符集中有2个符号：0和1；
- ✓**八进制**数符集中有8个符号：0，1，2，3，4，5，6，7；
- ✓**十进制**数符集中有10个符号：0，1，2，3，4，5，6，7，8，9；
- ✓**十六进制**数符集中有16个符号：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。

基数和权

✓如果把用K进制书写的一个整数从右往左依次记做第0位、第1位、…、第n位，则第i位上的数符a_i所代表的含义是a_i*k^i。在此，我们把k称为一个数制的**基数**，而把k^i称为k进制数第i位的**权**。

计数规则

✓简单的说，就是“**逢k进1，借1当k**”。

不难看出，一个任意k进制数可以写成以下形式：

$$a_n a_{n-1} a_{n-2} \ldots a_1 a_0 = a_n*k^n + a_{n-1}*k^{n-1} + a_{n-2}*k^{n-2} \ldots\ldots + a_1*k^1 + a_0*k^0$$

数制之间的转换

- ✓把非十进制数转换成十进制数
- ✓把十进制数转换成非十进制数(除k取余）
- ✓二进制、八进制、十六进制之间的转换

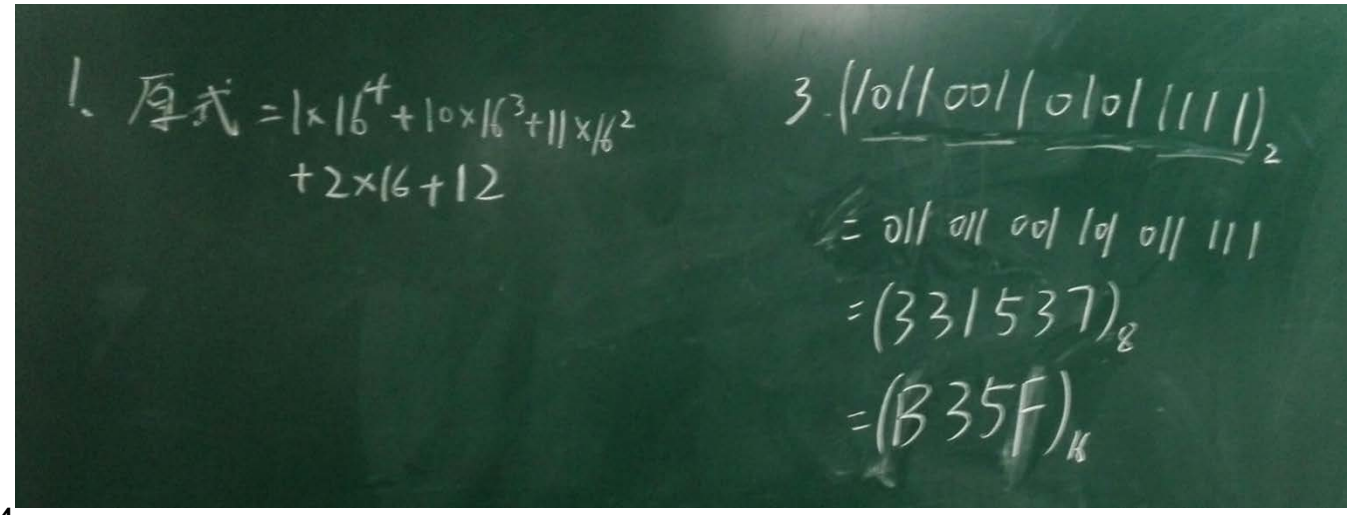
切记比例系数？（3:1,4:1）

例题

1．把十六进制数1AB2C转为十进制

2．把十进制数15370转换为十六进制

3．将二进制数1011001101011111转换为八进制和十六进制



数的书写方法

下标法

✓$(1001)_2$表示二进制数1001；
✓$(377)_8$表示八进制数378；
✓$(377)_{16}$表示十六进制数377；

前导法

✓十六进制数123在C语言中写作0x123。

后缀法

✓汇编语言中就采取这种写法。二进制、八进制、十进制和十六进制的后缀符号分别是B、Q、D和H。比如：

1011B表示二进制数1011；
1011H表示十六进制数1011。

0ABCH
578

1.3 有符号数与无符号数

- ✓负数是程序设计中必须面对的问题。但是计算机内部并没有正负号，只有0和1。计算机内部区分正负数的方法是，在存放数的若干个二进制位（一般是8位、16位或32位）中，用最高位作为符号位，这一位是1表示该数是负数，而这一位是0则表示该数非负。
- ✓8位补码所能表示的有符号数范围为80H～7FH，即－128<=N<=127；
- ✓16位补码所能表示的有符号数范围为8000H～7FFFH，即－32768<=N<=32767；
- ✓32位有符号数的表示范围为80000000H～7FFFFFFFH。

1.3 有符号数与无符号数

但是，在某些情况下，要处理的数全是正数，此时再保留符号位就没有意义了，为此可以把最高有效位作为数值位处理，这样的数称为无符号数。其中：

8位无符号数的表示范围为00H～0FFH，即0<=N<=255；

16位无符号数的表示范围为0000H～0FFFFH，即0<=N<=65535；

32位无符号数的表示范围为00000000H～0FFFFFFFFH，0<=N<=2^32-1。

重点强调

- ✓Intel 8086微处理器是定点机，没有处理浮点数的专门指令，对于浮点数的运算只能用程序来实现，而且是将小数点位固定在第0位的后面，所以针对8086宏汇编语言而言处理的数据都是有符号定点整数，无符号整数通常用来表示地址和进行逻辑运算。

- ✓ *存放在计算机内部的数是否有符号是人为看待的问题，而不是数据本身所具备的属性。*

1.4 原码、反码、补码

- ✓计算机中的**数**是用二进制数来表示的，数的**符号**也是用二进制表示，一般用最高有效位来表示机器数的符号（“0”表示正数，“1”表示负数）。
- ✓一个数连同其符号在内均用二进制表示，这样的数称为机器数。
- ✓机器数原来的实际值（符号仍用“＋”和“—”）称为真值。
- ✓机器数可以用不同的编码来表示，常用的有原码、反码及补码。

1.4 原码、反码、补码

- ✓*原码的编码规则为*：保持真值的数值部分不变，最高位为符号（0或1）的数值化表示。
- ✓*反码的编码规则为*：对于正数，保持真值的数值部分不变，最高位加符号位“0”；对于负数，对其真值的数值部分按位取反（将1变为0，0变为1），最高位加符号位“1”。
- ✓*补码的编码规则为*：对于正数，保持真值的数值部分不变，最高位加符号位“0”；对于负数，对其真值的数值部分按位取反后加1，最高位加符号位“1”。

例题：求解原码、反码、补码

例1：设M=＋61=＋3DH

则n＝8时

M的8位原码为：[M]原=3DH

M的8位反码为：[M]反=3DH

M的8位补码为：[M]补=3DH

而n=16时

M的16位原码：[M]原=003DH

M的16位反码：[M]反=003DH

M的16位补码：[M]补=003DH

例2：设M=－61＝－3DH

则n＝8时，M的8位原码为：[M]原=0BDH

M的8位反码为：[M]反=0C2H

M的8位补码为：[M]补=0C3H

而n＝16时，M的16位原码：[M]原=803DH

M的16位反码：[M]反=0FFC2H

M的16位补码：[M]补=0FFC3H

三种编码之间有如下关系：

正数：[M]原= [M]反= [M]补

负数：反码是原码除符号位外的按位取反，补码是原码的“取反加1”，保留符号位。

符号扩展

✓**符号扩展**是指一个数从较少位数增加到较多位数的过程，但增加的空位均由最高符号位填充。一般是指一个数由8位扩展到16位，或从16位扩展到32位。从本节以上的两个实例中可以看出，M的16位补码实际上是其8位补码的符号扩展。

✓***由此可以得出以下重要结论：***一个二进制数的补码表示中的*最高位（即符号位）向左扩展*若干位（即符号扩展）后，所得到的数仍是该数的补码。

错误？

课前回顾

- ✓汇编语言的本质？
- ✓汇编语言有哪4个组分及它们的使用规则构成？

1.5 ASCII码

- ✓计算机中所处理的数据包括数值数据和字符数据，数据在计算机内的存储都以二进制的形式。对于字符数据而言，是用一个字节表示一个字符的，其字符的数值表示规律是以美国信息标准交换代码ASCII码为标准的。

1.5 ASCII码

- ✓在8086/8088中，从键盘输入的任意字符， 在计算机内的表示都是ASCII码的形式。例如，当从键盘输入字符串 ‘1234ABCD’时，它们立即被转换成与之对应的ASCII 31H，32H，33H，…，44H，如果在程序中需要用到数字1234时，必须把相应的ASCII码转换成原来数字。
- ✓同样，当用户需要在显示器上显示自己程序的运行结果1234时，只要将它们逐一转换成与其对应的ASCII码后，存放在主存中，然后送显示器显示即可。

1.5 ASCII码

- ✓在高级语言中，这些转换工作由系统独立完成，而不需用户作处理，但在汇编语言中，这些工作只能由用户自己完成。
- ✓为了区别数值数据，程序中的字符数据全部以单引号或双引号括起来，常用字符的ASCII码如下表所示。

ASCII码表

字符	ASCII码	字符	ASCII码	字符	ASCII码	字符	ASCII码
0	30H	a	61H	k	6BH	u	75H
1	31H	b	62H	l	6CH	v	76H
2	32H	c	63H	m	6DH	w	77H
3	33H	d	64H	n	6EH	x	78H
4	34H	e	65H	o	6FH	y	79H
5	35H	f	66H	p	70H	z	7AH
6	36H	g	67H	q	71H	A	41H
7	37H	h	68H	r	72H	B	42H
8	38H	i	69H	s	73H	C	43H
9	39H	j	6AH	t	74H	D	44H
E	45H	J	4AH	O	4FH	T	54H
F	46H	K	4BH	P	50H	U	55H
G	47H	L	4CH	Q	51H	V	56H
H	48H	M	4DH	R	52H	W	57H
I	49H	N	4EH	S	53H	X	58H
Y	59H	Z	5AH	回车	0DH	换行	0AH