汇编语言程序设计

电子科技大学·计算机学院

邢建川

E-mail: xingjianchuan@sina.com

教学计划

1. 总学时数为32。

2. 教学方式：大型开放式网络授课（MOOC）

3. 最后成绩评定办法：平时作业占65%，期末考试占35%。

4. 教材：《汇编语言程序设计》廖建明主编　清华大学出版社 2009.10

5. 参考资料：

《8086/8088宏汇编语言程序设计教程》（第二版）王正智 编著电子工业出版社 2002.3

《微机原理与接口技术》陆鑫等编著 机械工业出版社2005.9

《IBM-PC汇编语言程序设计》（第2版）沈美明等编著 清华大学出版社2001.8

《Win32汇编语言程序设计教程》 严义等编著 机械工业出版社 2005.8

《汇编语言程序设计》 殷肖川 主编 清华大学出版社 2005.1

第一章　基础知识

本章主要学习内容：

1.汇编语言的一般概念

2.学习和使用汇编语言的目的

3.进位计数制及其相互转换

4.带符号数的表示

5.字符的表示

6.基本逻辑运算

§1.1 汇编语言的一般概念

计算机程序设计语言可分为机器语言、高级语言和汇编语言三类。

1. 机器语言

机器语言就是把控制计算机的命令和各种数据直接用二进制数码表示的一种程序设计语言。

例如，要实现将寄存器AH的内容与数10相加，结果再送回到寄存器AH中。

用机器语言实现上述操作的代码：

1011 0100 0000 1010

为了书写和记忆方便可用十六进制数表示：**B40A**

又如，要让计算机完成4×6+40的算式运算。假设参加运算的数据事先分别存放在寄存器AL、BL和CL中，并要求将运算结果存放到寄存器AL中。

用机器指令来实现的代码为：

1111 0110 1110 0011	十六进制数：F6E3
0000 0000 1100 1000	十六进制数：00C8

在32位二进制数表示的机器语言程序代码中，包含了乘法和加法运算操作，其中前16位代码表示了乘法运算，后16位代码表示了加法运算。

机器指令中既包含了指示运算功能的代码，又给出了参加运算的操作数据，表示非常详细。

优点：机器语言最直接地表示了计算机内部的基本操作，用它编制的程序在计算机中运行的效率最高。即运行速度最快，程序长度最短。

缺点：用二进制数表示的内容既不便于记忆又难于阅读。

2. 高级语言

高级语言将计算机内部的操作细节屏蔽起来，用户不需要知道计算机内部数据的传送和处理的细节，使用类似于自然语言的一些语句来编制程序，完成指定的任务。

特点：程序设计简单，但程序效率较机器语言低。

3. 汇编语言

（1）定义

虽然高级语言方便了人对计算机的使用，但其运行效率较低。在一些应用场合，如系统管理,实时控制等，难于满足要求。因此又希望使用机器语言。

为了便于记忆和阅读，使用字母和符号来表示机器语言的命令，用十进制数或十六进制数来表示数据，这样的计算机程序设计语言就称为汇编语言。

（2）汇编语言程序与机器语言程序的关系

一条汇编语言的语句与一条机器语言指令对应，汇编语言程序与机器语言程序效率相同.

例如,对于前述的4×6+40算式运算，如果把机器语言程序改写为汇编语言程序，则为以下两条汇编指令：

```
MUL  BL
ADD  AL, CL
```

（3）不同类型计算机有不同的机器指令系统和汇编语言描述

为了学习和使用某种计算机的汇编语言，必须熟悉计算机的内部组成结构。但并非要掌握计算机系统的全部硬件组成，只需掌握用汇编语言编制程序时所涉及到的那些硬件的结构和功能。

对一台计算机来说，机器语言的执行主要取决于该计算机的中央处理器CPU。因此熟悉计算机内部结构主要是指CPU的功能结构。它包括：

- CPU中有多少个寄存器及其作用？
- CPU是如何访问存储器？
- 输入输出操作的方式有哪些？

在本课程中，将以IBM-PC系列微型计算机及其汇编语言为例，学习汇编语言程序设计的基本原理、方法和技巧。

§ 1.2 学习和使用汇编语言的目的

1.学习和使用汇编语言可以从根本上认识、理解计算机的工作过程。

通过用汇编语言编制程序可以更清楚地了解计算机是如何完成各种复杂的工作。在此基础上，程序设计人员能更充分地利用机器硬件的全部功能，发挥机器的长处。

2. 在计算机系统中，某些功能必须用汇编语言程序来实现。

如：机器自检、系统初始化、实际的输入输出设备的操作等。

3. 汇编语言程序的效率高于高级语言程序

“效率”有两个方面的含义：程序的目标代码长度和运行的速度。

在某些要求节省内存空间和提高程序运行速度的应用场合，如实时过程控制、智能仪器仪表等，常常用汇编语言来编制程序。

§ 1.3 进位计数制及其相互转换

一. 进位计数制

使用一定个数的数码的组合来表示数字，这种表示方法称为进位计数制。根据所使用的数码的个数，就产生了不同的进位计数制。

如十进制数，用0、1～9十个数码的组合来表示数字。每个数码排在不同位置，所表示的数值大小不相同。

例如：222从右边开始,第一个2表示2个1，第二个2表示有2个10，第三个2表示有2个100.

将各个位置上所表示的基本数值称为位权，简称权。

不同的进位制和不同的位置其位权是不同的。位权乘以对应位置上的数码就等于该数位上数值的大小。

每个数位上能使用不同数码的个数称为基数。

例如十进制有十个数码0～9，基数为10，二进制基数为2。

每个数位能取的最大数码值=基数-1。如十进制为10-1=9

在计算机中数据表示一般采用二进制数，因为它在计算机中最容易表示和存储，且适合于逻辑值的表达与运算。

对人来说二进制不便于书写和阅读，因此书写时常使用8进制和16进制。

二进制与8进制、16进制之间有非常简单的转换关系：3位二进制数与一位8进制数对应，4位二进制数与一位16进制数对应。

在书写不同进位计数制数时，为了区别，常在数的尾部用一个字母来表示。

B（Binary）——二进制数
O (Octal)或Q——八进制数
D (Decimal)——十进制数
H (Hexadecimal)——十六进制数。
如未使用任何字母，则默认表示是十进制数。

例如，10B, 10Q, 10D,10H

二. 各种数制间的相互转换

由于二进制与八进制和十六进制间的转换很简单，下面主要讨论二进制与十进制之间的相互转换。

1. 十进制整数转换为二进制数

有两种转换方法：

（1）减权定位法

- 从二进制数高位起，依次用待转换的十进制数与各位权值进行比较；
- 如够减，则该数位系数**Ki=1**，同时减去该位权值，余数作为下一次比较的值；
- 如不够减，则**Ki=0** 。

例：将325转换为二进制数，直到余数为0。
首先确定二进制数的最高位
因为2^9(512)>325>2^8(256)。因此从K_8位开始比较。

减数比较	Ki	对应二进制数
325-256=69	K8	1
69<128	K7	0
69-64=5	K6	1
5<32	K5	0
5<16	K4	0
5<8	K3	0
5-4=1	K2	1
1<2	K1	0
1-1=0	K0	1

所以　325D=101000101B

(2) 除基取余法

将十进制数除以基数2，其余数为二进制数的最低位，再用其商除2，其余数为次低位，反复做下去，直到商0.

除基	余数	Ki
2 | 325		
2 | 162	1	**k0**
2 | 81	0	**k1**
2 | 40	1	**k2**
2 | 20	0	**k3**
2 | 10	0	**k4**
2 | 5	0	**k5**
2 | 2	1	**k6**
2 | 1	0	**k7**
0	1	**k8**

这种转换方法同样适合于其它进制数之间的转换。

2.十进制小数转换为二进制数

(1) 减权定位法

例 将十进制数0.645转换为二进制数

减权比较	Ki	对应二进制数
0.645-0.5 =0.145	k-1	1
0.145<0.25	k-2	0
0.145-0.125=0.02	k-3	1
0.02<0.0625	k-4	0
0.02<0.03125	k-5	0
0.02-0.015625	k-6	1

所以 0.645D=0.101001B

转换时应根据程序要求的精度或计算机的字长来确定二进制的位数.

（2）乘基取整法

例 将**0.8125D**转换为二进制数

乘以基数	**Ki**	整数部分
0.8125×2＝1.625	K-1	1
0.625×2＝1.25	K-2	1
0.25×2＝0.5	K-3	0
0.5×2＝1.	K-4	1

所以 **0.8125D＝0.1101B**

3. 二进制整数转换为十进制数

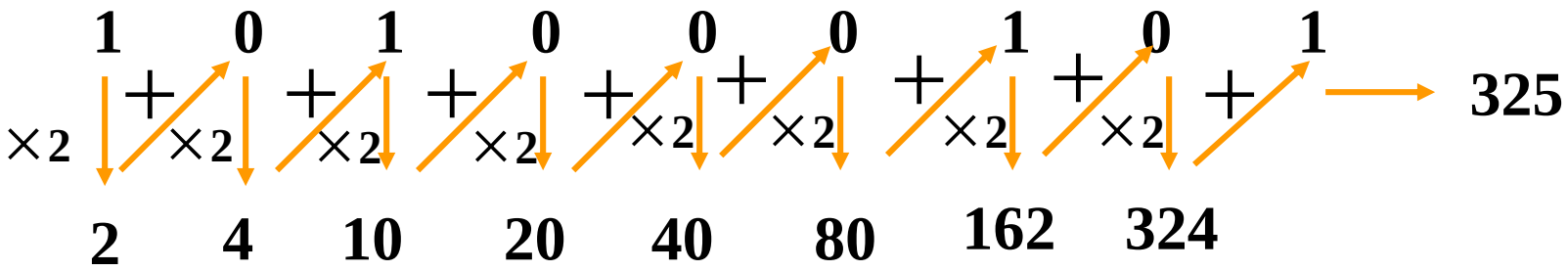
（1）按权相加法

例 $101000101B = 1\times2^8+1\times2^6+1\times2^2+1\times2^0$

$= 256+64+4+1$

$= 325$

（2）逐次乘基相加法



4. 二进制小数转换为十进制数

（1）按权相加法

例　$0.101001B = 1\times2^{-1}+1\times2^{-3}+1\times2^{-6}$

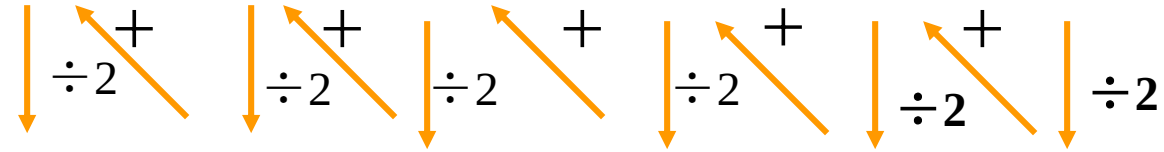
$=0.5+0.125+0.0156$

$=0.640625D$

（2）逐次除基相加法

转换从最低位开始

例　0.　1　0　1　0　0　1



0.640625　0.28125　0.5625　0.125　0.25　0.5

0.101001B＝0.640625D

5. 二进制与八进制和十六进制间的转换

二进制与八进制和十六进制之间的对应关系很简单：

三位二进制数对应一位八进制数，四位二进制数对应一位十六进制数。

例如：10100010B =10 100 010

2 4 2

所以 10100010B=242Q

10100010B = 1010 0010

A 2

所以 10100010B = A2 H

§ 1.4 带符号数的表示

在一般算术表示中使用”“+”和“-”来表示正数与负数，而在计算机中使用“0”和“1”来表示正数和负数。

用“＋”或“－”表示正负的数叫真值
用“0”或“1”表示正负的数叫机器数

带符号的机器数可以用原码、反码和补码三种不同码制来表示。一般计算机中大多采用补码表示。

一、原码表示

二进制数的最高位表示符号，0表示正，1表示负。数值 部分用二进制数绝对值表示

8位二进制数原码的最大数为01111111（+127）

最小数为11111111 （-127）

8位二进制数表示范围：－127≤X≤+127

0的原码有两种表示形式：00000000和10000000（+0和-0）

二、补码的表示

1.补码的定义

带符号数X的补码表示$[X]_{补}$定义为：

$$[X]_{补}=M+X \quad (\text{Mod } M)$$

其中模数M根据机器数的位数而定，如位数为8则$M=2^8$

用补码表示的机器数，符号位仍然表示数的符号：0为正,1为负。对于正数，补码与原码相同，对于负数需要进行变换。

2.由真值、原码变换为补码

由于正数的原码与补码相同，下面讨论负数的变换方法。

负数的真值变换为补码的方法：将各位变反（0变1，1变0）然后在最低位加1.

负数的原码变换为补码：保持符号位不变,其余各位变反，最低位加1。

例 将－59变换为补码

真值	－00111011	原码	10111011
变反	11000100	变反	11000100
加1	11000101	加1	11000101

所以 [-59]补＝11000101

3.补码数的表示范围

当位数为8时，最大补码为01111111=$[+127]_{补}$

最小补码为10000000=$[-128]_{补}$

0的补码只有一个，$[0]_{补}$=00000000，而10000000是$[-128]_{补}$

11111111=$[-1]_{补}$

对于16位数，则补码表示范围为-32768~+32767

4. 补码的加减运算

规则：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$

$[X-Y]_{补}=[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$

其中$[-Y]_{补}$是对$[Y]_{补}$执行一次求补运算

求补运算是将原数连同符号位一起（不管是正还是负）按位求反，再在最低位加1。

（1）加法运算：X+Y

例1 X=74D　Y=41D

$[X]_{补}$＝01001010　$[Y]_{补}$＝00101001

$$\begin{array}{r} 01001010 \\ + \quad 00101001 \\ \hline 01110011 \end{array}$$

所以 $[X]_{补}+[Y]_{补}=01110011=[115]_{补}$

例2 X=74D　Y=-41D

$[X]_{补}$＝01001010　$[Y]_{补}$＝11010111

$$\begin{array}{r} 01001010 \\ + \quad 11010111 \\ \hline 1 \quad 00100001 \end{array}$$

自动舍去

所以 $[X]_{补}+[Y]_{补}=00100001=[33]_{补}$

例 3　X= -74D　　Y=41D

$[X]_{补}$＝10110110　$[Y]_{补}$＝00101001

```
  10110110
+ 00101001
----------
  11011111
```

所以 $[X]_{补}+[Y]_{补}=11011111=[-33]_{补}$

例4　X= -74D　Y= -41D

$[X]_{补}$＝10110110　$[Y]_{补}$＝11010111

```
  10110110
+ 11010111
----------
1 10001101
```

自动舍去

所以 $[X]_{补}+[Y]_{补}=10001101=[-115]_{补}$

(2) 减法运算

例5　X=74D　　Y=41D

$[X]_{补}$＝01001010　　$[Y]_{补}$＝00101001　$[-Y]_{补}$=11010111

```
   01001010
 + 11010111
-----------
 1 00100001
```

自动舍去

所以 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=00100001=[33]_{补}$

例6　X=74D　Y=－41D

$[X]_{补}$＝01001010　　$[Y]_{补}$＝11010111　$[-Y]_{补}$=00101001

```
   01001010
 + 00101001
-----------
 1 01110011
```

自动舍去

所以 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=01110011=[115]_{补}$

例7　X= -74D　　Y=41D

$[X]_{补}$＝10110110　　$[Y]_{补}$＝00101001　$[-Y]_{补}$=11010111

自动舍去

```
  10110110
+ 11010111
-----------
1 10001101
```

所以 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$＝10001101＝$[-115]_{补}$

例8　X= -74D　　Y= -41D

$[X]_{补}$＝10110110　　$[Y]_{补}$＝11010111　$[-Y]_{补}$=00101001

自动舍去

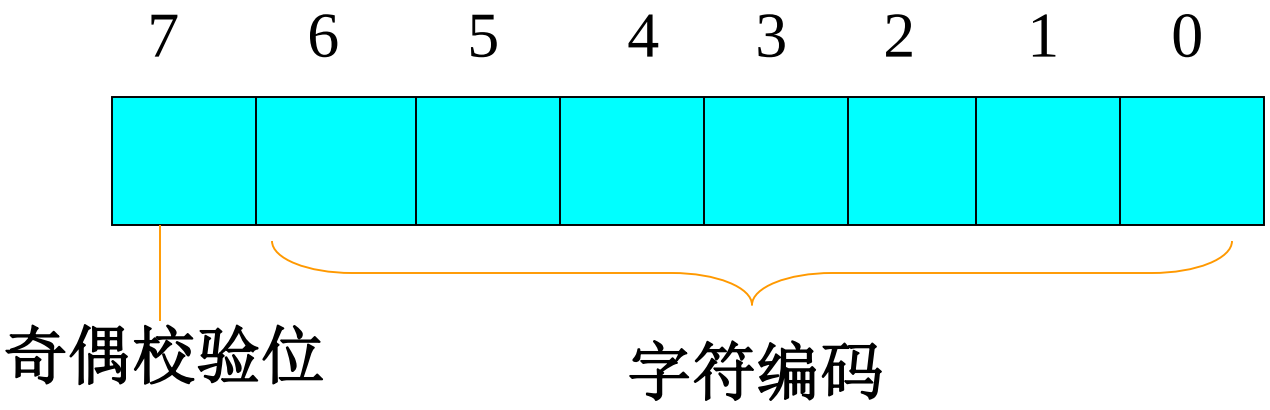
```
  10110110
+ 00101001
-----------
1 11011111
```

所以 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}$＝11011111＝$[-33]_{补}$

§ 1.5 字符的表示

在计算机内部，各种字符（字母、符号、数字码）都是按一定的方式编写成二进制信息。不同的计算机以及不同的场合所采用的编码形式可能不同。目前最广泛采用的是ASCII码（American Standard Code for Information Interchange）

标准ASCII码为一字节，其中用低七位表示字符编码(见附录A),用最高位表示奇偶数验位。



标准ASCII码共有128个，可分为两类：

非打印ASCII码,共33个,用于控制操作, 如BEL(响铃07H), DEL(删除7FH),CR(回车,0DH), LF(换行,0AH).

可打印ASCII码共有95个，如数字符0~9，大小写字母等。

§ 1.6 基本逻辑运算

计算机内部采用二进制数表示信息，具有物理实现容易、可靠性高的优点，且由于状态“0”和“1”正好与逻辑运算中的逻辑“真”和“假”对应，因此可以用“0”和“1”来表示逻辑变量的取值，很容易地实现各种复杂的逻辑运算。

在计算机的指令系统中，一般都有逻辑运算指令。下面介绍几种常见的基本逻辑运算。

1.“与”运算（AND）

“与”运算也叫逻辑乘，常用 ∧ 或 · 表示。

设有逻辑变量A和B，则“与”运算为：

F=A ∧ B 或 F=A · B

“与”运算是指：仅当逻辑变量A与B都是1时，运算结果F才为1。其它情况F为0，

即有：

0 ∧ 0 =0　0 ∧ 1 = 0　1 ∧ 0 = 0　1 ∧ 1 =1

2. “或”运算（OR）

“或”运算也叫逻辑加，用∨或＋表示。即有：

F=A ∨B　　F=A +B

“或”运算是指当逻辑变量A与B中，至少有一个为1时，结果F为1，其他情况F为0。

即有：0 ∨0＝0　　0 ∨1＝1　　1 ∨0＝1　　1 ∨1＝1

3. “非”运算

“非”运算是指对逻辑变量取相反的一个逻辑值。

逻辑“非”运算通常是在逻辑变量上方加一横线表示。

如A为1，则 $\overline{A}=0$，若A为0，则 $\overline{A}=1$

“非”运算规则为：

$\overline{1} = 0 \qquad \overline{0} = 1$

4. “异或”运算（XOR）

通常用 ⊕表示，即 F =A ⊕ B

“异或”运算是指：当A和B相同时（同时为1或同时为0），运算结果F为0，而不同时，F为1。

运算规则为：

1 ⊕ 1=0　　1 ⊕ 0=1　　0 ⊕ 1=1　　0 ⊕ 0=0

上述四种基本逻辑运算规则用真值表表示为：

A	B	A ∧ B	A ∨ B	$\overline{A}$	$\overline{B}$	A ⊕ B
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	1	0	0	0