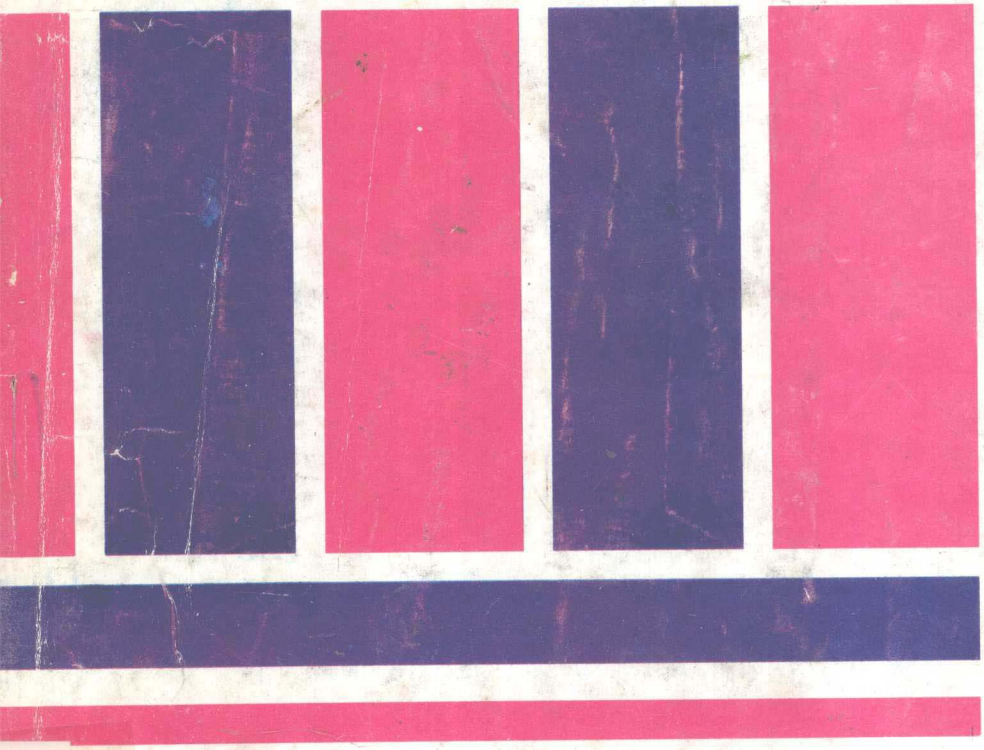
高等学校
工科电子类 规划教材

微型计算机原理

季维发 王典金 编



电子科技大学出版社

微型计算机原理

季维发　王典金　编

电子科技大学出版社
·1994·

[川] 新登字 016 号

内容提要

本书以 Intel 公司 8086/8088 微处理器为核心，介绍 16 位微处理器的内部结构、指令系统和微型计算机的工作原理，并以 ASM86 汇编程序来叙述汇编语言源程序的程序设计以及主要伪指令的格式和应用。以 IBM PC/XT 为样机介绍微型计算机的 CPU 子系统、RAM 子系统、ROM 子系统以及微型机的接口设计和主要接口芯片 8255、8253 等。

本书可作为电子机械类各专业本科生教材，也可作为非计算机类各专业、成人教育和培训班教材。

微型计算机原理

季维发　王典金　编

电子科技大学出版社出版
（中国成都建设北路二段四号）邮编 610054
四川省自然资源研究所印刷厂印刷
四川省新华书店经销
*
开本 787×1092　1/16　印张 11.375　字数 285 千字
版次 1994 年 5 月第一版　印次 1994 年 12 月第二次印刷
印数 6001—12000 册
中国标准书号　ISBN 7-81016-755-3/TP・58
定价：8.50 元

出 版 说 明

根据国务院关于高等学校教材工作的规定，我部承担了全国高等学校和中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978～1990年，已编审、出版了三个轮次教材，及时供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻国家教委《高等教育“八五”期间教材建设规划纲要》的精神，“以全面提高教材质量水平为中心，保证重点教材，保持教材相对稳定，适当扩大教材品种，逐步完善教材配套”，作为“八五”期间工科电子类专业教材建设工作的指导思想，组织我部所属的九个高等学校教材编审委员会和四个中等专业学校专业教学指导委员会，在总结前三轮教材工作的基础上，根据教育形势的发展和教学改革的需要，制订了1991～1995年的“八五”（第四轮）教材编审出版规划。列入规划的，以主要专业主干课程教材及其辅助教材为主的教材约300多种。这批教材的评选推荐和编审工作，由各编委会或教学指导委员会组织进行。

这批教材的书稿，其一是从通过教学实践、师生反映较好的讲义中经院校推荐，由编审委员会（小组）评选择优产生出来的，其二是在认真遴选主编人的条件下进行约编的，其三是经过质量调查在前几轮组织编写出版的教材中修编的。广大编审者、各编审委员会（小组）、教学指导委员会和有关出版社，为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还可能有缺点和不足之处，希望使用教材的单位，广大教师和同学积极提出批评和建议，共同为不断提高工科电子类专业教材的质量而努力。

机械电子工业部电子类专业教材办公室

前　言

本教材系按机械电子工业部的工科电子类专业教材 1991～1995 年编审出版规划，由电子机械教材编审委员会评选审定，并推荐出版。责任编委张江陵。

本教材由西安电子科技大学季维发副教授、烟台大学王典金副教授主编，由华中理工大学胡盛斌副教授主审。编者根据教材编审委员会通过的《微型计算机原理》教材大纲，并结合编者多年的教学讲义编写而成。与本教材一起，还有《微型计算机接口技术》和《微型计算机控制基础》两本教材。这三本教材既有连贯性，也有独立性，共同作为电子机械类专业本科生的专业基础课教材，也可作为非计算机类各专业、成人教育和培训班教材。以上三本教材均由电子科技大学出版社出版。

本课程的参考教学时数为 50 学时，主要介绍三部分内容：第一部分介绍计算机的一般知识，计算机的基本组成和微型计算机的组成特点，计算机中信息的表示方法、数的运算和存储以及硬件和软件的关系等。第二部分介绍 16 位微处理器，并以 8086/8088 微处理器为例叙述其内部结构、基本工作原理、指令系统及应用，并以 ASM 86 汇编程序来介绍汇编语言源程序的设计方法、主要伪指令的格式和应用。第三部分从微型机系统角度来介绍微型计算机的系统设计、组成、存储器结构以及微型计算机的接口方法和几种常用接口芯片等。配合课堂教学还有 10 学时的实验内容。

本教材由王典金副教授编写第五、六、七章，其余各章均由季维发副教授编写。由于编者水平有限，对书中的缺点和错误，敬请读者批评指正。

编　者

1994.4

目　录

第一章 概 述

第一节 计算机的发展

现在，通常提到的计算机都是指电子数字计算机。由于计算机应用的广泛性，其高超的计算能力是家喻户晓人人皆知的。然而，从科学的角度来看，它仍然只是一种计算和信息加工的机器。由于它的存储容量大，运算速度快和精度高等优点，使它远远胜过其他计算工具。当然，作为计算和信息加工的工具，计算机与人类发明的其他计算工具相比，有了质的飞跃。它减轻并部分地代替了人的脑力劳动，在特定的时间限制内，能够完成大脑无法完成的工作，也因此获得电脑的美称。

自从1946年第一台电子数字计算机问世以来，随着电子器件的不断更新和发展，使得计算机的发展也是日新月异，通常把它们的发展划分为电子管、晶体管、小规模集成电路和大规模集成电路四代，现在正酝酿着第五代。

第一代(1946～1956)，计算机的基本电路采用电子管，故称电子管计算机时代。这代计算机体积大，耗电多，价格贵，速度慢，可靠性差。主要用于科研机构和院校。

第二代(1956～1962)，计算机的基本电路元件采用晶体管，故称晶体管计算机时代，这代计算机与第一代计算机比较，体积减小，耗电降低，价格下降，速度更快，可靠性提高，应用面已扩大到工矿企业和机关事务管理中。

第三代(1962～1970)，计算机采用集成电路，这时计算机的性能价格比都显著提高，而且应用面迅速扩大。

第四代(1970～1992)，计算机采用大规模、超大规模集成电路芯片，这时计算机的特点是体积小，重量轻，耗电省，速度快，价格低，其应用面已扩大到工矿企业、机关团体、家庭个人等社会各个领域。

人们预示不久的将来，会出现第五代计算机，它将是人工智能计算机，是综合计算机科学和控制论而发展起来的一门新技术，它能模拟人的智能，如识别图形、语言、物体等，将对社会的发展带来不可估量的影响。

前面谈的是一般计算机的发展，作为计算机大家庭中的后起之秀微型计算机(Microcomputer)，简称微机。它把计算机中核心部件运算器和控制器集成在一块芯片上，称为中央处理单元(Central Processing Unit，简称CPU)，又称微处理器(Microprocessor)。它本身的发展又可划分为五个阶段。

第一阶段(1971～1973)，典型的微处理器有4004和8008，其特点是采用PMOS工艺，字长4～8位，并行处理，多总线结构，平均指令周期为2μs，时钟2.5～5MHz，集成度为2000器件/片，16～18引脚。

第二阶段(1973～1975)，典型的微处理器有8080和M6800，其特点是采用NMOS工艺，字长8位，并行处理，单总线结构，平均指令周期为2μs，时钟2MHz，集成度为5000器件/片，40引脚。

第三阶段(1975～1977)，典型的微处理器有8085，M6801，M6803，Z80，其特点是采用E/D MOS工艺，注入逻辑工艺，字长8位，平均指令周期1μs，时钟2.5～5MHz，集成度为

1 万器件/片。随着集成电路技术发展，外围接口电路芯片 RAM，ROM 也获得很大发展。此时出现的单板计算机和单片计算机，给新的工业革命和传统的工业改造注入强有力的生命力。

第四阶段(1978～1980)，微处理器进入超大规模时代，典型微处理器有 8086，8088，6809，Z8000 等，其特点采用 HMOS 工艺，字长 16 位，指令周期 0.5μs，时钟 5～10MHz，集成度为 3 万器件/片。

第五阶段(1981 至现在)，典型微处理器有 IAPX43201，80386，80486，MC68010，MC68020 等，字长 32 位，时钟高达 16MHz，集成度为 10 万器件/片以上。

微型计算机的出现标志着计算机的发展进入了新的时代。由于其体积小，重量轻；价格便宜，性能价格比好，应用面广泛，故它的发展速度和深度都大大超过了它的前代。微机发展特点是新快多广。所谓新就是技术新，工艺新，产品新；快就是变化快，换代快(平均每两年换代一次)；多就是品种多；广就是应用面广，涉及面广。

微型机的出现和发展给机械工业注入新的生命力，它是传统机械工业的技术改造和新的机电一体化工程中不可缺少的技术手段。例如机床数控系统，工业机器人，传感技术，智能化仪表，工业对象的遥测，故障诊断及工业过程实时控制、数据采集等，无一不应用微机技术。现在工业中已经大量使用微机系统，可编程控制器(Programmable Controller)，STD 总线工业控制机，单板机，单片机，它大大地提高了劳动生产率和生产自动化程度，从而推动了工业技术革命进程。

第二节　计算机中数制和编码

在第一节中，我们已经看到计算机只是一种计算和信息加工的机器，数字和信息在计算机中只能用 0 和 1 来表示。下面将介绍数字、符号、汉字、图形等在计算机中的表示方法。

一、常用进位制

在日常生活中，人们最常用的是十进制。例如：

$$1992.5=1\times10^3+9\times10^2+9\times10^1+2\times10^0+5\times10^{-1}$$

任意一个十进制数都可以表示为

$$\begin{aligned}X &= x_m10^m+\cdots+x_010^0+x_{-1}10^{-1}+\cdots+x_{-n}10^{-n}\\ &=\sum_{i=-n}^{m}x_i10^i\end{aligned}\qquad(1\text{-}1)$$

其中，10 称为十进制的基数，而所在数位 i 的权为 10^i。十进制数的特点如下：

(1)有十个基本数字 0～9。

(2)逢 10 进 1，借 1 当 10。

在计算机内如果采用十进制数，那就要有 10 种状态代表 10 个数字，在电路上难以实现，所以，计算机内无论是计算、内部传送、还是存储，一般都采用二进制数，两种状态的器件在电路上是很容易实现的。

任意二进制数都可展开为

$$\begin{aligned}X &= x_m2^m+\cdots+x_02^0+x_{-1}2^{-1}+x_{-2}2^{-2}+\cdots+x_{-n}2^{-n}\\ &=\sum_{i=-n}^{m}x_i2^i\end{aligned}\qquad(1\text{-}2)$$

其中，2 称为二进制的基数，第 i 位 x_i 是 0 或 1，2^i 称为第 i 位的权。二进制数有如下特点：

(1)有两个基本数字 0 和 1。

(2)逢 2 进 1，借 1 当 2。

例如，二进制数 1011.010 可展开为

$$1011.0101 = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4}$$

二进制数在计算机内很容易表示，机内也用二进制数进行运算。但二进制数位数较多，书写很繁，容易写错。为了编程书写方便，在微机中还引用十六进制数。任意一个十六进制数都可表示为

$$X = x_m 16^m + \cdots + x_0 16^0 + x_{-1}16^{-1} + \cdots + x_{-n}16^{-n} = \sum_{i=-n}^{m} x_i 16^i \qquad (1\text{-}3)$$

其中，16 称为十六进制数的基数，第 i 位数 x_i 应是下述 16 种符号之一，对应的数值为 $x_i\times 16^i$，16^i 称为第 i 位的权。十六进制数的特点：

(1)有 16 个基本数字 0～9，A，B，C，D，E，F。这里的 A，B，C，D，E，F 分别表示十进制数 10，11，12，13，14，15。

(2)逢 16 进 1，借 1 当 16。

为了避免在编程时使用多种进制而引起混乱，在各进制数字最后加一个后缀，以示区分。例如十进制数后缀是 D（十进制数后缀可省略），二进制数后缀是 B，十六进制数后缀是 H。同样一个十进制数值 155，在各不同进制中分别表示为 155D，10011011B，9BH。

二、各种进制数的转换

尽管有不同的进制，但在计算机中的数仍然只能用二进制表示，十六进制是适应于编程时读写方便的需要，而十进制则是日常生活所必需的。因此，就要掌握各种进制的转换关系。

1. 十进制数和二进制数的相互转换

十进制数转换成二进制数。根据式 (1-2)，十进制整数转换成二进制数，一般采用除 2 取余法。十进制纯小数转换成二进制小数，采用乘 2 取整法。例如，把十进制数 215.625 转换成二进制数。整数转换过程：

	余数	
2|215	1	x_0
2|107	1	x_1
2|53	1	x_2
2|26	0	x_3
2|13	1	x_4
2|6	0	x_5
2|3	1	x_6
2|1	1	x_7
0		

小数转换过程：

	0.625		
×	2	整数	
	1.250	1	x_{-1}
	0.250		
×	2		
	0.500	0	x_{-2}
	0.500		
×	2		
	1.000	1	x_{-3}

上例转换结果可写成：

$$215.625D = 11010111.101B$$

二进制数转换成十进制数。对所给的二进制数，只要按式(1-2)展开，即得对应的十进制数。例如：

$$1011.101B = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-3}$$
$$= 11.625D$$

2. 十进制数和十六进制数的相互转换

十进制数转换成十六进制数同十进制数转换成二进制数道理一样，整数采用除 16 取余法，纯小数采用乘 16 取整法。例如，把十进制数 12472.65625 转换成十六进制数。

整数转换过程：

	余数	
16|12472	8	x_0
16|779	B	x_1
16|48	0	x_2
16|3	3	x_3
0		

小数转换过程：

	0.65625		
×	16		
	393750		
	65625	整数	
	10.50000	A	x_{-1}
	0.50000		
×	16		
	300000		
	50000		
	8.00000	8	x_{-2}

转换结果可写成：

$$12472.65625D = 30B8.A8H$$

十六进制数转换为十进制数。对给定的十六进制数，只要按式(1-3)展开，即得对应的十进制数。例如：

$$9AC6H = 9 \times 16^3 + 10 \times 16^2 + 12 \times 16^1 + 6 \times 16^0$$
$$= 39622D$$

3. 二进制数和十六进制数的相互转换

由于一位十六进制数需要四位二进制数表示，它们之间存在着直接而又唯一的对应关系，如表 1-1 所示。可见二进制和十六进制之间转换是很方便的。

二进制数转换为十六进制数，方法是从小数点开始向左右两边，每四位二进制数为一组转换成相应的一位十六进制数，最后不足四位用 0 补。例如：

0001 1011 0111. 0101 0100
↓ ↓ ↓ ↓ ↓
1 B 7 . 5 4

即 110110111.010101＝1B7.54H

十六进制数转换为二进制数。不论十六进制整数还是小数，只要把每一位十六进制数用相应的四位二进制数代替，就可以转换成二进制数。例如：

6 B . 5 4
↓ ↓ ↓ ↓
0110 1011 . 0101 0100

去掉无用 0 后，该十六进制数就转换成为

6B.54H＝1101011.010101B

上述转换必须注意到，有时一个十进制小数不一定能完全准确地转换成二进制小数，例如，0.1D＝0.000110011001100……B，这就是说十进制小数 0.1 转换成二进制后成为一个无限循环的小数，不能准确地表示出来。不能用有限位的二进制小数去表示任意一个有限位的十进制小数，这是二进制数的缺点。

表 1-1 各种进制数对照表

十进制	二进制	十六进制	十进制	二进制	十六进制
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

三、带符号数的表示方法

1. 原码表示

前面讲的二进制数，没有提到符号问题，实际是一种无符号数的表示。在计算机中，把数的符号也用数码表示，通常做法是将一个二进制数码的最高位作为符号位，规定 0 表示正号，1 表示负号，剩余位作为数值部分，这种表示法称原码表示法。例如：

X＝＋1110010B　　$[X]_{原}$＝01110010B

Y=－1110010B　　$[Y]_{原}$=11110010B

符号被数值化了的二进制数称为机器数，而带有正负号的数称为机器数的真值。上例字长 8 位，最高位为符号位，后面 7 位为数值部分，如果字长为 16 位，后面 15 位为数值部分。原码优点是简单，转换方便。缺点有两个，一是原码的 0 有两种（见表 1-2 所示）。另一个是当两个异号数相加时要做减法运算。为了把减法改用加法，而引出反码和补码表示。

表 1-2　数的表示法

机　器　数		真值（十进制）			
二进制数	十六进制表示	无符号数	原　码	反　码	补　码
00000000	00	0	+0	+0	+0
00000001	01	1	+1	+1	+1
00000010	02	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮	⋮
01111110	7E	126	+126	+126	+126
01111111	7F	127	+127	+127	+127
10000000	80	128	－0	－127	－128
10000001	81	129	－1	－126	－127
⋮	⋮	⋮	⋮	⋮	⋮
11111110	FE	254	－126	－1	－2
11111111	FF	255	－127	－0	－1

2. 反码表示

反码的定义：

$$[X]_{反}=\begin{cases}X & 2^{n-1}>X\geqslant 0\\ 2^{n}-1-|X| & 0>X>-2^{n-1}\end{cases}$$

对一个 n 位纯整数来说，正数的反码与原码相同。负数的反码，在原码基础上，符号位不变，数值位按位求反。例如：

X=＋0000100B　　$[X]_{原}=[X]_{反}$=00000100B

Y=－0000100B　　$[Y]_{原}$=10000100B

$[Y]_{反}$=11111011B

3. 补码表示

补码的定义：

$$[X]_{补}=\begin{cases}X & 2^{n-1}>X\geqslant 0\\ 2^{n}-|X| & 0>X>-2^{n-1}\end{cases}$$

对一个 n 位纯整数来说，正数的补码与原码相同，负数的补码，在反码基础上，最低位加 1，例如：

X=－0001010B　　$[X]_{原}$=10001010B

$[X]_{反}$=11110101B

$[X]_{补}$=11110110B

引入补码后，就可把两数相减，转换成补码相加。例如：

$X=24-10=14$　　$[X]_{补}=[24]_{补}+[-10]_{补}$

$[24]_{补}=00011000B$　　$[-10]_{补}=11110110B$

```
    00011000
  +11110110
丢失←[1] 00001110
```

$[X]_{补}=00001110B$ 的真值为 14，同直接相减结果一致。必须指出，一个有符号数，由于编码不同，可有几种机器数。反之，一个机器数，由于解释方法不同，又可代表无符号数，也可代表有符号数，还可代表符号、字符、图形等。它究竟代表什么，是由编程者确定的。

前面带符号数的表示，都是以八位数为例讨论的。如果要把一个八位数补码扩展成 16 位时这样进行。即

数值	8 位表示	16 位表示
+1	00000001B	0000000000000001B
−1	11111111B	1111111111111111B

是把原来符号位的值扩展到附加的高八位上去，这种操作称为符号扩展，只有这样，才能正确执行补码，8086 中就应用这种方法。

四、小数点问题

在计算机中，小数点的位置有两种表示法，定点表示法和浮点表示法。这两种表示法不但关系到小数点的位置，而且关系到数的表示范围、计算精确性和电路实现的复杂程度等。

1. 定点表示法

定点表示法，就是小数点在数中的位置是固定不变的，这样的数称为定点数，使用定点表示法的计算机称为定点计算机。

原则上讲，在定点表示法中，小数点的位置规定在哪一位都没有关系。但为了方便，总是把小数点规定在数的最前面或数的最后面，即把所有的数都化成纯小数或纯整数来进行运算。因此，一个带符号数定点表示法的一般形式为

数符 . 数　码	或	数符　数　码	.
小数点		小数点	

必须指出，这里的小数点不占机内一位，机器也无法表示出来，只是使用者在编程时需要明白小数点定在什么位置。定点表示法优点是电路实现简单，但编程时，应把原始数据用比例因子化成小数或整数来处理，最后计算结果再用比例因子化成实际值。为了保持计算精度，保证计算正确不溢出，在运算过程中还要多次调整比例因子。

2. 浮点表示法

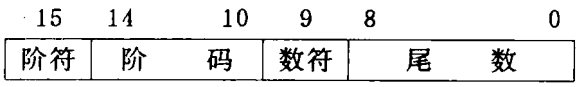
浮点表示法，就是小数点在数中的位置是浮动的，这样的数称为浮点数。使用浮点表示法的计算机称为浮点计算机。

在浮点表示时，计算机把任何一个二进制数分成阶码和尾数两部分组成。因为二进制数可写成如下一般形式：

$$N=\pm S\cdot 2^{\pm J}$$

其中 J 称阶码(整数)，2 称阶码的底，S 称尾数(一般规定为纯小数，即小数点在尾数的前面)。

阶码正负号(称阶符)用来指示小数点的实际位置，而尾数的正负用来指示整个数是正数还是负数。阶符和尾数符号各占一位，阶码和尾数本身各占多少位，由不同机器来定。例如有一台16位字长的计算机，若用前6位作为阶码，后10位作为尾数，符号各占一位，则该机的浮点数格式为



如果字长相同，浮点数比定点数所能表示的数值范围要大。利用浮点数进行一般科学计算，不需要用比例因子，这是浮点表示法的优点。但是，浮点表示法中，小数点的实际位置不固定，所以，进行加减运算时就存在对位(即对阶)问题，这给编程带来麻烦，这是浮点表示法的缺点。

定点计算机也可采用浮点表示和处理实数，但由于机内结构是定点式的，所以只能通过程序来转换和加工。这些程序一般都标准化的，称为浮点运算子程序。

五、溢出问题

由于计算机运算器的位数有限，它所能表示数的范围也就受到限制，运算过程中一旦超出这个限制，运算结果就会产生错误，这就是计算机的溢出问题。一个n位带符号的二进制补码，所能表示的最大正数是$2^{n-1}-1$，最大负数是-2^{n-1}。例如8位字长，用补码所表示的数值范围是$-128\sim+127$。16位字长，用补码所表示的数值范围是$-32768\sim+32767$。

微型机中常用双高位判别法来判别溢出问题。所谓双高位判别法，即规定符号位(用C_S代表)有进位时$C_S=1$，否则$C_S=0$。数值部分最高位(用C_P代表)有进位时$C_P=1$，否则$C_P=0$。当微机中“异或”电路判别出$C_S\oplus C_P=1$成立，则有溢出产生，否则无溢出。下面以8位字长补码运算，来说明双高位判别法。

例
$$\begin{array}{rll} & 01011010 & +90 \\ & +01100011 & +99 \\ & 10111101 \quad (\text{求补}) & -67 \end{array}$$

$\because C_S=0 \quad C_P=1 \qquad \therefore C_S\oplus C_P=1$

产生溢出，结果出错（超出正数范围）。

例
$$\begin{array}{rll} & 10010010 & -110 \\ & +10100100 & -92 \\ \text{丢失}\rightarrow \boxed{1} & 00110110 \quad (\text{求补}) & +54 \end{array}$$

$\because C_S=1,\ C_P=0 \qquad \therefore C_S\oplus C_P=1$

产生溢出，结果出错（超出负数范围）。

例
$$\begin{array}{rll} & 10010010 & -110 \\ & +00110110 & +54 \\ & 11001000 \quad (\text{求补}) & -56 \end{array}$$

$\because C_S=0,\ C_P=0 \qquad \therefore C_S\oplus C_P=0$

无溢出，结果正确（未超出数值范围）。

任何运算都不允许产生溢出。因此，编程时，都要避免溢出产生。如果运算中产生溢出，应使计算机停止运算并进行必要的处理。