COBOL语言

COMPUTERCOMPUTERCOMPUTER

譚浩强 编著

下

清华大学出版社

COBOL 语言

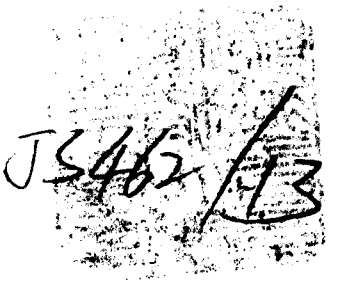
（下）

谭浩强　编著

清华大学出版社

内 容 提 要

本书是一本通俗而详细的COBOL语言教材。根据ANSI COBOL1974标准的规定，介绍了COBOL语言及其程序设计。作者针对初学者所遇到的问题，对各部分内容作了较好的安排。本书的叙述通俗易懂、循序渐进、例题丰富，使初学者容易理解。是一本学习COBOL的入门书。为适应不同读者的需要，本书对表处理的位标法、SEARCH语句、以及磁带文件和磁盘文件也作了较详细的介绍。它是作者"BASIC语言"和"FORTRAN语言"的姐妹篇。可作为大专院校师生、企事业单位管理人员、计算机程序工作人员学习COBOL的教材。也可供具有高中以上文化程度的读者自学。



COBOL 语言（下）

谭浩强 编著

清华大学出版社 出版

北京 清华园

轻工业出版社印刷厂排版

国防工业出版社印刷厂印刷

新华书店北京发行所发行·各地新华书店经售

开本：787×1092 1/16 印张：14 1/8 字数：350千字

1984年3月第一版 1984年3月第一次印刷

印数：1—60,000

统一书号：15235·95定价：1.75元

目　录

上　册

下　册

第七章　数据部之二

——数据部的较高技巧

§1．　数据在计算机内的表示形式

至今为止，我们介绍的是数据在计算机内部存放的最简单的情况，即：一个字符在计算机内存中占一个字节，无论是数字（0～9）或字母（A～Z）或其它字符（如：＋，$，*…）都各占一个字节。在本章我们将介绍其它的形式。

（一）计算机内存的组织形式

在计算机中是以二进制形式来表示数据的。内存单元是由一系列的二进制位(它的值是0或1）组成的，因此它的最小单位是二进制的“位”(bit)。

若干位组成一个字节(byte),一个字节为8个二进位。在计算机语言中一般是以字节来作处理单位的，即存取数据以字节为单位。譬如说，取数据时一次至少要取出一个字节八个二进位的内容，如00100100等。

字节又可组成半字、字、双字。各种不同计算机对一个字包含几个字节有不同的规定。一般计算机以4个字节（32位）作为一个“字”(word),以2个字节（16位）作为“半字”，以8个字节（64位）作为一个“双字”。

数据按指定形式放入。如一个数字（或其它字符）可以用ASCII代码或EBCDIC代码表示（见附录Ⅶ），把它放到计算机内存中，需要占一个字节（八位二进制）。从内存中取数据时，则是取出这个字节内容。

以上是需要了解的最简单的知识。

（二）字符数据在内存中的存放形式

字符型、字母型和数值编辑型、字符编辑型数据项中的数据，每一个字符都在内存中占一个字节。这种形式称为标准数据形式。

由于内存中数据都是以二进制数来表示的，因此要规定每一个字符用什么样的一组（八位)二进制数来表示。每类计算机系统分别选择其所用的代码形式。例如,在ASCII代码中，以八位二进制数01000001代表“A”，以00110001代表“1”。而在EBCDIC代码中，以11000001代表“A”,以11110001代表“1”。总之,一个字符以八位二进制数代表，占一个字节。字符转换成二进制代码不是由用户自己转换的,而是由计算机系统自动转换的。例如在键盘上按下A，B，C三个字符，就把相应的二进制代码输入到计算机内存中去。反之，从计算机中输出一个数据，先将这些字节中存放的二进制数（如EBCDIC码的01000001,01000010,01000011）取出，再把它们转换成字符（如A，B，C)打印出来。

如果采用字符型数据形式，不论是字母或数字，都按一个字节存放一个字符。如有两个数据项A和B，它们的描述为：

```
77  A  PIC  X(3)  VALUE 'ABC'.
```

77 B PIC X(3) VALUE '123'.

	A		
实际的二进码	11000001	11000010	11000011
代表字符	(A)	(B)	(C)

	B		
实际的二进码	11110001	11110010	11110011
代表字符	(1)	(2)	(3)

特别需要注意的是，如果采用字符型形式，则非数值常量中的数字亦按一般字符处理；如上述"1 2 3"，仅意味三个字符1、2、3，而不代表一百二十三。

（二）数值型数据的存放形式

数值型数据在内存中存放的形式有以下几种：

（1）**外部十进制**（或称扩张十进制）形式

按数值在机器外部的表现形式，一个数字在内存中占一个字节。如：

77 C PIC 9(3) VALUE 486.

在内存中占三个字节。每一个数字与二进制代码的关系同上述。**在内存中实际内容如下（假设机器内用EBCDIC码）。**

	C		
	11110100	11111000	11110110
	(4)	(8)	(6)

〔注〕

如果数据项D的描述如下（D的值为负）：

77 D PIC S9(3) VALUE -486.

此时，负号不占一个字节，而在最后一个字节中放入某个信息，一般是将此字节的前四位1111改为1101，后四位仍为0110。即

	D		
	11110100	11111000	11010110

计算机检查最后一个字节的前四位，如为1101，则按负数处理，如为1100，按正数处理。或者说，用十六进制[注]中"C"代表"正"，"D"代表"负"，"F"代表无符号，即

〔注〕：为表示方便，有时用十六进制数来表示一个数。一个十六进制数包括四个二进位数，用十六进制的"0"代表二进制数0000，"1"代表0001，"2"代表0010，"3"代表0011，"4"代表0100，"5"代表0101，"6"代表0110，"7"代表0111，"8"代表1000，"9"代表1001，"A"代表1010（十进制的10），"B"代表1011（十进制的11），"C"代表1100，"D"代表1101，"E"代表1110，"F"代表1111（F就是十进制中的15）。

因此本页中C在内存中的情况可以表示为：

F4	F3	F6

绝对值（也有些计算机系统不用“C”，“D”，而用其它数代表“正”，“负”）。

（2）**外部浮点数**形式

某些数据，它的值很大或很小（如$+1.23876\times10^{59}$或-1.38457×10^{-69}），用以前讲的外部十进制形式存贮是有困难的。COBOL允许用指数形式来表示一个数，例如上面两个数，用指数形式可以写为：

+1.23876E+59

−1.38457E−69

其中“E”表示“以10为底的指数”。“E”前面的部分称为“数值部分”或“尾数部分”，“E”后面的是“指数部分”或“阶码部分”。数值部分和指数部分各有一个符号以表示正或负。其一般形式为：

数符 数值部分 E 阶码符 阶码

为了表示这种指数形式的数据（外部浮点形式），在PIC子句中可以这样写：

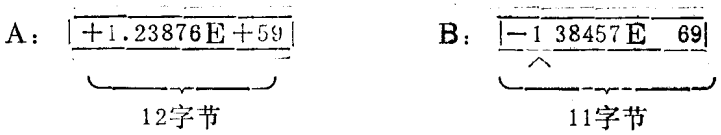
```
77  A  PIC  +9.99999E +99.
```

或

```
77  B  PIC  +9V99999E −99.
```

它表示在内存中按以上形式存放数据。其中每一个“9”表示此位置可放入一个0～9间的一个数字，一个“9”占一字节。E前面放数值部分，E后面放指数部分。小数点“.”表示此位置有一小数点，“V”表示此位置有一隐含的小数点，它不占内存字节。正号“+”的意思是，如果数值为正，此处为正号，数值为负时，此处为负号。负号“−”的意思是：数值为正时此处空白，数值为负时，此处为负号（它们的用法和数值编辑项中的“+”、“−”编辑符相同）。“E”也占一个字节。因此，A在内存占12个字节，B占11个字节。

A和B在内存中情况为



COBOL中浮点数可以表示数的范围是5.4×10^{-79}到0.72×10^{76}。用外部浮点形式，PIC字符串中数值部分最多可以出现16个“9”，指数部分除“E”外，应有一个“+”或“−”，以及两个“9”（不应有三个或更多个“9”，因为不可能出现10^{100}以上的数值）。

对外部浮点项不能用VALUE子句赋初值，如下面写法是不对的：

```
02 N PIC  +99.99E+99  VALUE  +12.45E+45.
```

只能从外部文件上读入以指数形式表示的数据（例如可以从卡片上相应的位置开始，按列依次穿孔“+12.45E+45”），也可以用MOVE语句给N传送数值。

下面是一些例子：

PIC字符串	外部数据格式	表示的值
+99V9E−99	␣183E−13	$+18.3\times10^{-13}$
−9V99E+99	␣687E+65	$+6.87\times10^{65}$
+9(3).99E+99	+875.46E−12	$+875.46\times10^{-12}$
−V9(6)E+99	␣678123E+37	$+0.678123\times10^{37}$

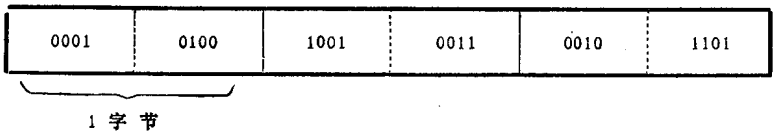
+.99E−99　　+.87E␣62　　$+0.87\times10^{62}$

（3）**内部十进制**（又称缩合十进制）形式

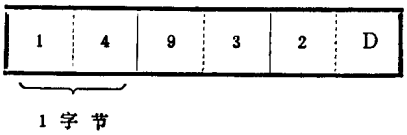
外部十进制形式在内存中一个字节放一个字符。数值型数据只用到 0 ～ 9 十个数字，如 129，23868 等。我们从下表中可以看到，0 ～ 9 的代码前四位是相同的。

十 进 制 数 字	EBCDIC码	ASCII码
0	1111 0000	0011 0000
1	1111 0001	0011 0001
2	1111 0010	0011 0010
3	1111 0011	0011 0011
4	1111 0100	0011 0100
5	1111 0101	0011 0101
6	1111 0110	0011 0110
7	1111 0111	0011 0111
8	1111 1000	0011 1000
9	1111 1001	0011 1001

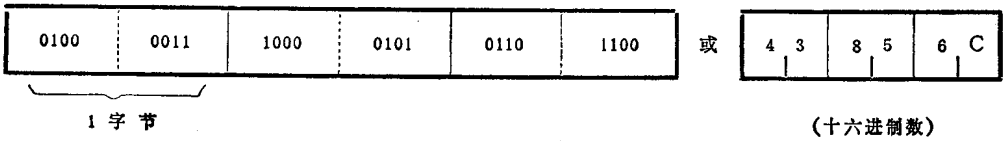
因此在表示 0 ～ 9 数字时，前四位并不起辨别作用，只是根据后四位的不同来判别是0～9中的哪个数字。因此，为节省内存，可以只用四位二进制数字来代表一个十进制数。在一个字节中放两个十进制数字。每个数字占四位，即半个字节。符号也占半个字节。如，-14932在内存中为：



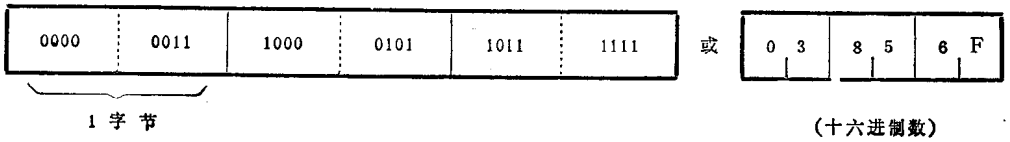
或用十六进制表示：



这种存放方式称内部十进制，或缩合十进制，+43856在内存中为：



无符号的数 3856，在内存中为：



F表示数值无符号，即取绝对值。由于存取的最小单位是字节，因此3856虽然只需两个半字节，但也占三个字节，最前面半个字节补零。

（3）**定点二进制**形式

不是以一个数字对应一个字节或半个字节，而是先把十进制数化成二进制数，然后存放在内存中。如：10先化成 $(1010)_2$，括弧外下角 2 表示是二进制的数。然后把1010放到内存单元。

如果在数据部中以 PIC 9(4) 描述一个数据项，即数据长度为 4 位数字，则需要在内存中开辟 2 个字节。COBOL规定在内存中根据数据项的长度分别用二字节、四字节或八字节来存放一个以定点二进制形式存放的数。

在PIC子句中描述字符9的个数	占内存字节
1～4	2
5～9	4
10～18	8

例如，十进制数10，在内存中以定点二进制形式存放时，占两个字节，为：

00000000	00001010

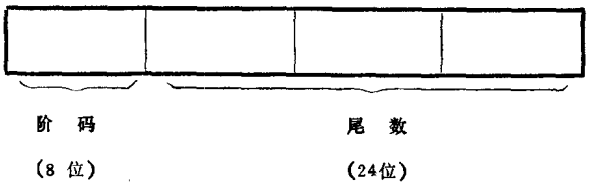
十进制数 $(83212)_{10}$用二进制形式表示为10100010100001100，在内存中占四个字节，存放形式为：

00000000	00000001	01000101	00001100

因为二个字节放不下十进制中五位整数，所以系统规定五位以上整数用四个字节存放。同样，四个字节放不下十位整数，故规定十位以上的整数一律以八个字节存放。

（4）**内部浮点**形式

它以内部的指数形式（二进制的指数形式）来表示一个数，以固定长度的内存单元来存放一个数。如以 4 个字节（32位）表示一个数，其中八位表示阶码（指数）部分，二十四位表示尾数部分。这称为短浮点形式。



也可以用 8 个字节（64位）表示一个数，阶码部分仍为 8 位，尾数部分为56位，称长浮点形式。长浮点形式比短浮点形式有更高的精确度（内部浮点形式的数值在内存中所占字节长度只有四个字节和八个字节两种）。

无论短浮点形式或长浮点形式，所表示的数值范围均为：

$$5.4\times10^{-79}\sim0.72\times10^{76}$$

有关内部浮点形式不准备作过多介绍。读者只需有个粗略的了解即可。

（三）小结

（1）字母型、字符型、编辑型、外部十进制数据和以外部浮点形式表示的数据只能用标准数据形式来表示，即一个字符占一个字节。

如：77 A PIC X(6).
77 B PIC A(6).
77 C PIC 999.99.

均占6个字节。

（2）数值型数据可以由程序员任意选定存放形式（外部十进制、外部浮点形式、内部十进制、定点二进制、内部浮点形式）。用外部十进制形式最简单、最好理解，但一般说，它占内存多。如135879326九位数，用外部十进制需9个字节，用内部十进制需5个字节，用定点二进制需4个字节，短浮点形式需4个字节，长浮点形式需8个字节。

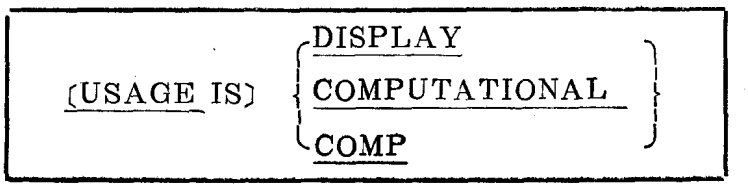
（3）数据在计算机内进行运算，都是化成二进制数以后再进行的，因此，外部十进制形式是不能直接进行运算的，需要由计算机把它们先化为定点二进制或内部浮点形式，然后再进行运算。假定A和B已定义为外部十进制形式，其值分别为11和12，它在内存中存贮形式为 |F 1|F 1| 和 |F 1|F 2|（假设以EBCDIC代码存放），A和B各占两个字节。如果执行 ADD A TO B，即 A+B ⇒B，并不是将A和B中各自第一个字节“F 1”相加，第二字节“F 1”和“F 2”相加。而是先将A和B化成 $(1011)_2$ 和 $(1100)_2$，然后进行二进位的相加，相加后得 $(10111)_2$，然后再把它转换成23的EBCDIC码，即“F 2”和“F 3”，再存到B中。计算前后各要转换一次。显然，化时间多，速度慢。

从运算速度上看，定点形式最快，内部浮点形式次之，外部浮点形式和外部十进制、内部十进制最慢。因此，用户应根据需要选择用哪一种数据形式。这就要用下一节介绍的USAGE子句。

§2. 用法子句（USAGE子句）

使用用法（USAGE）子句可以使程序设计者自由选择数据在内存中的存放形式。譬如，数据项A和B是需要多次进行运算，如果用外部十进制形式则来回转换会大大降低运算速度，这时，可以选择A和B为定点二进制形式或内部浮点形式。如果数据项C和D参加运算次数少，而且需要多次打印出C和D的结果，这时用外部十进制比较适合，因为它最适合打印的要求，不必再进行转换。

USAGE子句的一般格式为：



其中 USAGE IS DISPLAY 的意思是“显示型的用法”，即：此数据项适于显示、打印，它采用标准数据形式（一个字节放一个字符）。

COMPUTATIONAL 和 COMP 是同一意思（COMP 是COMPUTATIONAL的缩写）。USAGE IS COMP 的意思是“计算型的用法”（它只能用于数值型数据项）。表

示此数据项适于计算，它采用适于计算用的定点二进制形式或内部浮点形式等。

标准 COBOL 只列出 DISPLAY 和 COMPUTATIONAL（COMP）两种用法的格式。各种计算机还规定了它可采用的有哪几种数据存放形式以及用什么来代表它们。例如，在M系列、IBM 系列和许多计算机系统中提供的功能包含：

DISPLAY	（标准数据形式。一个字节放一字符）
COMPUTATIONAL COMP	（定点二进制形式）
COMPUTATIONAI-1 COMP-1	（内部短浮点形式）
COMPUTATIONAL-2 COMP-2	（内部长浮点形式）
COMPUTATIONAL-3 COMP-3	（内部十进制形式）

在使用前，应查阅所用计算机的COBOL说明书。

说明：（1）USAGE 子句是用来指定数据项在内存中的存贮形式的。如：

```
02  A  PIC  9(4)  USAGE  IS  COMP.
```

表示数据项A用定点二进制形式（假设该计算机系统所用的 COBOL 按我们介绍的上述规定）。

USAGE IS 这两个英文字可以省写。上面的数据项A描述体可简写成：

```
02 A PIC 9 (4)  COMP.
```

（2）如不写USAGE子句，则隐含表示为用 DISPLAY 形式。

如
```
02  B  PIC  9 (6) USAGE IS DISPLAY.
02  B  PIC  9 (6) DISPLAY.
02  B  PIC  9 (6) .
```

以上三种写法等价。字符型、字母型、编辑型、外部十进制、外部浮点形式的数据项必须用 USAGE DISPLAY（可以采用显式或隐含形式表示）。

（3）如果对组合项描述为某一种存放形式，则表示这个组合项的下属各初等项都是这种形式。如：

```
01  T.
    02  T1  USAGE  COMP.
        03  X  PIC  S9(3).
        03  Y  PIC  S9(3).
```

在T1的描述中用了COMP，表示定点二进制，它下属的X和Y也都是定点二进制形式。它相当于：

```
01  T.
    02  T1.
        03  X  PIC  S9(3)  COMP.
        03  Y  PIC  S9(3)  COMP.
```

组合项的描述中如果用了用法子句，其下属的初等项可以再用用法子句，但二者的说明不应矛盾。例如将上面的02层T1描述体改写为：

02 T1 USAGE COMP.

是可以的，因它与03层中用的一致。但如果改成：

02 T1 DISPLAY.

则02层与03层描述发生矛盾。不允许。

（4）USAGE 子句指定的数据存放形式不应与 PIC 子句指定的数据类型矛盾。如：

04 A PIC A(4) USAGE COMP.

或 04 B PIC $99.99 COMP-1.

都是错误的。前者A是字母型数据，不能采用数值型的存放形式。后者B是编辑数据型数据，当然也不能采用短浮点形式。

（5）长、短浮点形式已确定了内存的长度，不应再用PIC子句。如：

04 A1 COMP-2.

没用 PIC 子句，但已确定为长浮点形式，如果某一具体计算机规定长浮点形式占8个字节，则A1不论是什么值，在内存中都占8个字节。

（6）在传送或运算时不同“用法”的数值型数据间可互相转换。

如：77 A PIC 9(3) COMP.

77 B PIC 999V99 DISPLAY.

二者的存放形式不同。如果有语句：

MOVE A TO B

则先将A转换成B的形式再存入。如：

ADD A TO B

也由编译系统进行自动转换，使A和B具同一形式，相加后按B的形式存入B。

（7）在用 DISPLAY 语句显示数据项的内容时，如果数据项的“用法”不是USAGE DISPLAY的，则在显示前，计算机自动将内存中数据形式转换成 EBCDIC 码（或ASCII码，视计算机系统而定），然后显示。例如：

```
01 LP-REC.
   02 FILLER PIC X.
   02 A PIC S999 COMP.
   02 B PIC 999 COMP-3.
   02 C PIC S999.
   02 D COMP-1.
   02 E COMP-2.
   02 F PIC +99.99E+99.
```

如果执行 MOVE −555 TO A, B, C, D, E, F.

然后执行 DISPLAY A, B, C, D, E, F.

显示出的结果为：

−555555-555-.55500000E␣03-.5550000000000000E␣03-55.50E+01

A B C D E F

即先转换，后显示。COMP-1（内部短浮点形式）显示时给出8位有效数字，COMP-2

（内部长浮点形式）显示时给出17位有效数字（但不同计算机系统规定不同）。

可以看出，数值型的数据，不论指定的是什么“用法”，都可以进行计算，也都可以用来显示，只是在不同的场合下不同“用法”的效率不同，程序员根据需要选择较佳的“用法”。

（8）如果用 WRITE 语句，直接输出，则不进行转换，例如在上面例子中在MOVE语句之后，执行 WRITE LP-REC AFTER 1. 则打印结果为：

```
␣␣␣␣55N␣␣␣␣␣␣␣␣␣␣␣␣−55.50E+0
 A  B  C   D      E       F
```

由于 A，B，D，E 的内容不是 DISPLAY 用法的，它们无法打印出相应的字符，故这些位置留空白，只有C和F是 DISPLAY 用法的，按内存中存贮的实际情况输出相应的字符。C 在内存中的存放情况为：[F5][F5][D5]，按 EBCDIC 码“F 5”相应于字符“5”，“D 5”相应于字符“N”，故打印出“55N”。

但如果将A传送给一个数据项 G，G 是用 PIC X（4）描述的：

02　G　PIC　X（4）.

然后再用 WRITE 语句将G所在的记录输出，则在 G 的区域上打印出：555␣，因为在用 MOVE 语句向G传送时，将555传送给G（数值的负号按规定不能传送入字符型数据项），左对齐，右补空格。故打印出“555␣”。其它数据项（如 B，C，D，E）都与此相同。

关于传送时的规则细节，读者可以查阅说明书。在此，只要求知道，用WRITE 语句输出时，是直接按内存中存放数据形式输出（不加转换的）。这是和用 DISPLAY 语句时不同的。

§3. 符号子句 (SIGN子句)

ANSI COBOL 1974 提供了 SIGN子句。用它来指定数值型数据描述体中运算符号的状态和位置。

（一）前已介绍在没有 SIGN 子句时的情况：

02　A　PIC　9（3）USAGE DISPLAY.

如果A的值为“012”，当机器内码为EBCDIC码时，则在内存中A的值是以下面的形式存放的：

F 0	F 1	F 2

最后一个字节用十六进制码表示是“F2”，即二进码11100010。

如果在 PIC 子句中用“S”描述符，如：

02　A　PIC S9（3）USAGE DISPLAY.

则以最后一个字节的前半个字节作为符号的标志。如果A的值为“+012”，则在内存中A的值是这样存放的：

F 0	F 1	C 2

（二）用 SIGN 子句可以指定符号在数值的前部还是后部。

如果在A的描述体中包含以下的 SIGN 子句：

02 A PIC S9(3) USAGE DISPLAY SIGN IS LEADING.

则表示符号不在最后一个字节中表示，而用第一个字节中前四位表示。如果A的值仍为“+012”，则在内存中是这样的：

C 0	F 1	F 2

如A的值为“−012”，则为：

D 0	F 1	F 2

如果 SIGN 子句形式改写为：

02 A PIC S9(3) USAGE DISPLAY SIGN IS TRAILING.

则表示符号在最后字节中表示，和不写这个 SIGN 子句时作用相同。也就是说，当省略 SIGN 子句时，隐含“SIGN IS TRAILING”。

（三）还可以指定符号单独占一个字节。这时要用“ SEPARATE ”可选项，看下例：

02 A PIC S9(3) USAGE IS DISPLAY SIGN IS TRAILING SEPARATE.

表示符号在数值的后部，符号单独占一个字节。A值为“+012”时，内存区中情况为：

F 0	F 1	F 2	4 E

如果用二制数表示，即：

1111 0000	1111 0001	1111 0010	0100 1110
(0)	(1)	(2)	表示“+”

此时多占一个字节专用来放符号标志，在EBCDIC码中“+”是十六进制的“4E”，即二进制的“0100 1110”，“−”是十六制的“60”，即二进制的“0110 0000”。如果写成：

02 A PIC S9 (3) SIGN IS LEADING SEPARATE.

则符号单独占一个字节，它的位置在第一个字节。当A为“−012”时，内存中情况为：

6 0	F 0	F 1	F 2

即：

0110 0000	1111 0000	1111 0001	1111 0010
表示“−”	(0)	(1)	(2)

SIGN 子句的一般格式为：

〔SIGN IS〕{LEADING / TRAILING} 〔SEPARATE CHARACTER〕

在使用 SIGN 子句时注意：

（1）它只能用于 PIC 字符串中含有“S”的数值型数据描述体中。

（2）使用 SIGN 子句的数据项的用法（USAGE）应当是 USAGE DISPLAY（显式的或隐含的）。不能用于计算型用法的数据项。例如：

```
77  H  PIC S999 COMP SIGN LEADING SEPARATE.
```

是不合法的。

（3）用 SEPARATE 可选项时，内存中增加了一个字节，用来放符号标志，如上例中数据项A用S9（3）描述，实际上占四个字节。

但数值型数值在向字符型数据项传送时，符号不予传送，相当于没有加这个符号一样。如有：

```
77  W  PIC  X (4).
77  A  PIC  S9 (3) VALUE  −125  SIGN  LEADING
                SEPARATE.
```

在数据项A中存放四个字符“−125”的代码。

−	1	2	5

在执行 MOVE A TO W 后，由于符号不传送，W中从左到右存放“125␣”。即

1	2	5	␣

（4）如果一个数据项的描述体中包含 SIGN 子句，则数据项的值应包括正或负的符号，否则会出错。

§4. 重定义子句 (REDEFINES子句)

不同的数据项可以共用内存中的同一段空间。例如已给数据项A分配了一段内存空间，在经过某一段的过程后，A已不再使用了，但它仍占着内存中这部分空间，为了节约内存，可以将另一数据项B也分配在A所占的这段内存空间。如

```
02  A  PIC  X(5).
02  B  REDEFINES  A  PIC  9(5).
```

表示B对A进行重新定义。不仅与其共用空间，而且可以重新定义类型，A原为字符型，用 PIC X(5)描述，而B用 PIC 9(5) 描述，定义为数值型。

重定义子句也可以用于组合项，如：

```
02  A.
  03  A1  PIC  9 (4).   ┐
  03  A2  PIC  X (6).   ├ 共占14字节
  03  A3  PIC  X (4).   ┘
02  B  REDEFINES A.
  03  B1  PIC  X(5).    ┐
  03  B2  PIC  9 (6).   ├ 共占14字节
  03  B3  PIC  99V9.    ┘
```