M 微处理器 icroprocessor 与 M 微计算机 icrocomputer

第五册

第五机械工业部第二〇七研究所

目　录

8080微型计算机实用子程序库

微计算机8080A、8085用PROM监控程序的正确使用方法

8080微型计算机实用子程序库

前　言

这里介绍的子程序库（ＳＳＬ—2000）是制造微型/小型计算机的（日本）电子测器株式会社为该社的8080系列微型计算机ＡＩＤＡＣ　Ｓ—2000系统所配的。征得该社的同意，从该社的手册转载于此。

该程序库收集了编制各种运算程序所必须的子程序：4字节浮点数的四则运算子程序、输入输出转换子程序、操作子程序、各种代码转换程序、平方根子程序、三角函数子程序、反三角函数子程序、对数函数子程序、指数函数子程序以及单字节整数乘除子程序和双字节整数四则运算子程序等。

1．子程序库概要

微型计算机ＡＩＤＡＣＳ——2000系统的开发子程序库ＳＳＬ——2000收录了为编制演算处理程序所必需的各种子程序。诸如4字节的浮点形式的四则运算子程序，输入输出变换子程序、操作子程序、各种数码变换子程序、平方根子程序、三角函数子程序、反三角函数子程序、对数函数子程序、指数函数子程序，以及1字节的整数乘除子程序、2字节的四则运算子程序等。

函数子程序库ＳＳＬ——2000的程序占有存贮器０Ａ66Ｈ（2662）个字节，只由数字排成的数如2662表示十进制数（后面有Ｈ的０Ａ66Ｈ表示是16进制数）。因此，由用户使用的一些必要的子程序，可直接从后面列出的程序表中，适当地选出某些子程序作为源程序加以汇编，即可使存贮容量最小。

但是，有的子程序套用了另外的子程序，用户必须注意这一点。

2．4字节浮点形式数据的表示

4字节浮点形式的内部数据如图1所示，由1字节的指数（ＥＸＰ）、3字节的尾数（上尾数、中尾数、下尾数）构成。每个数据需要4个存贮单元。

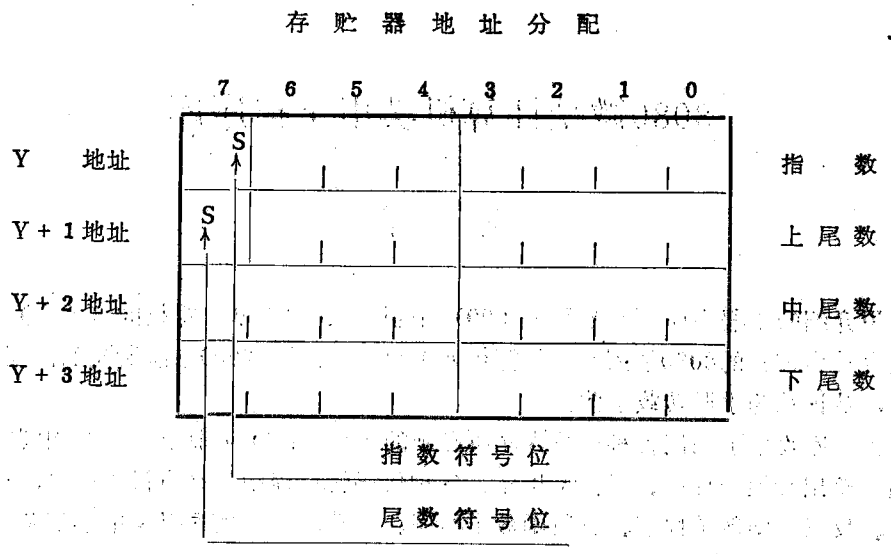


图 1　4 字节浮点形式数据的内部表示

2—1　尾数表示

4 字节浮点形式的尾数，由图 2 的 3 字节构成。

最高位(M S B)是符号位，“0”表示正数、“1”表示负数。

尾数符号位的右侧是小数点的位置，小数点右侧各位的权依次为2^{-1}，2^{-2}，2^{-3} …

尾数为正时（符号位为“0”），各位的值（1 或 0）与其权相乘后，结果的总和就是尾数的值。

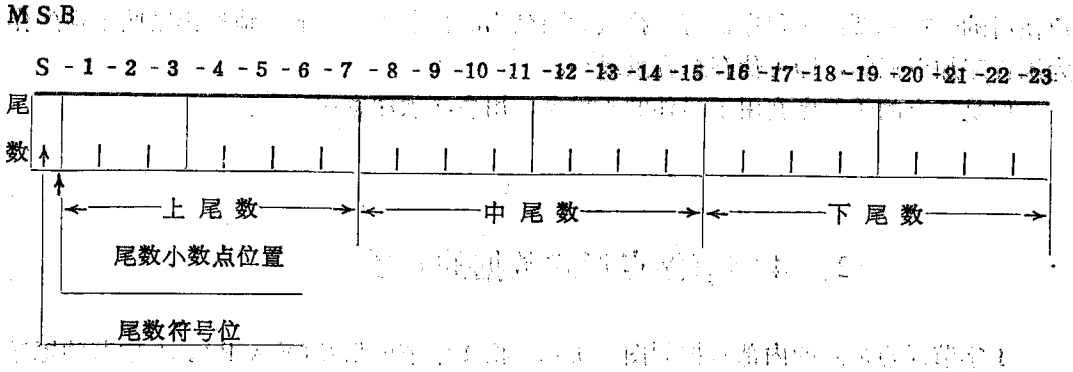


图 2　4 字节浮点形式数据的尾数表示

例1）

当二进制表示的尾数值如下图时，其十进制表示的数值由下式给出。

	S	-1 -2 -3	-4 -5 -6 -7	-8 -9 -10 -11	-12 -13 -14 -15	-16 -17 -18 -19	-20 -21 -22 -23
尾数	0	1 1 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

符 号 位

〔说明〕

尾数的值 $= 1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}+0\times2^{-5}+1\times2^{-6}+0\times2^{-7}+\cdots\cdots$

$= 0.5+0.25+0+0.0625+0+0.015625+0+\cdots\cdots$

$= 0.828125$

尾数为负时（符号位为“1”），把3字节长的尾数取补码后，按正数那样转换，再将结果加负号。

列2）

当二进制表示尾数的是下述数值时，其十进制表示的数值由下列转换给出。

	S	-1 -2 -3	-4 -5 -6 -7	-8 -9 -10 -11	-12 -13 -14 -15	-16 -17 -18 -19	-20 -21 -22 -23
尾数	1	0 1 1	0 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

符号位（“1”，表示负数）

〔说明〕

将各位的值取反码。

0	1 0 0	1 0 1 0	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

上列的数据＋1得下面的值

0	1 0 0	1 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

尾数的值 $= -\ (1\times2^{-1}+0\times2^{-2}+0\times2^{-3}+1\times2^{-4}+0\times2^{-5}$

$+1\times2^{-6}+1\times2^{-7}+0\times2^{-8}+\cdots\cdots)$

$= -\ (0.5+0+0+0.0625+0+0.015625+0.0078125)$

$= -\ 0.5859375$

尾数表示的数值取在1＞｜（尾数值）｜≥0.5的范围内。（但数值“0”除外）。使尾数符合这个数值范围的操作叫做规格化。

4字节浮点形式的数据，其尾数一定要规格化。

2—2 指数表示

4字节浮点数据的指数，由图3的1字节构成，最高位（MSB）是符号位，“0”表示正的指数值，“1”表示负的指数值。

指数最低位（LSB）的右侧是指数的小数点位置，小数点位置的左侧各位的权依次为2^0、2^1、2^2、2^3……。

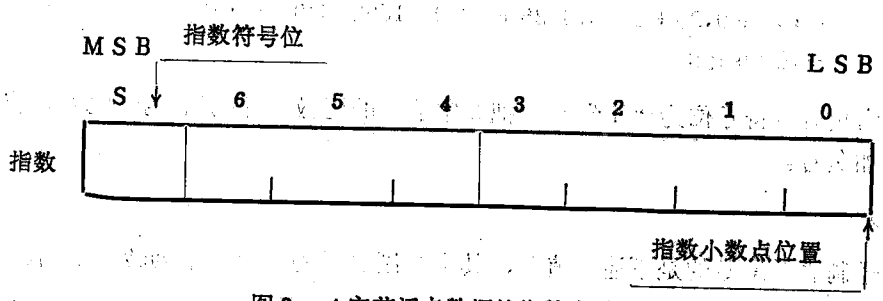
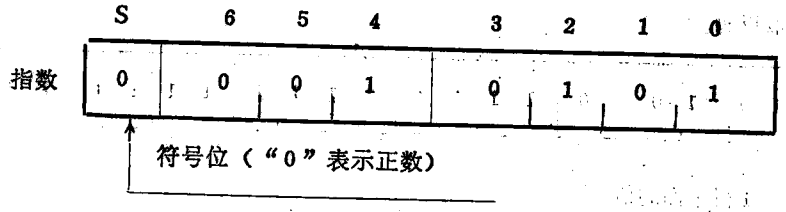


图3 4字节浮点数据的指数表示

指数为正时（符号位为“0”），各位的值与其权相乘后，结果的总和就是指数的值。

例3）

当二进制表示的指数如下时，其十进制指数值由下列转换给出。



〔说明〕

指数的值$=1\times2^0+0\times2^1+1\times2^2+0\times2^3+1\times2^4+0\times2^5+0\times2^6$

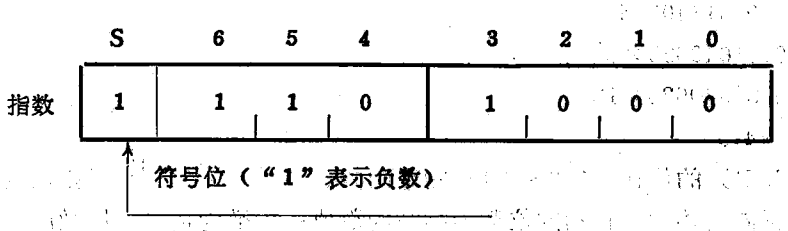
$=1+0+4+0+16+0+0$

$=21$

指数为负时（符号位为1”），取指数的补码后，把结果按正数那样变换，再取负号。

例4）

当二进制表示的指数如下时，其十进制指数值由后面转换给出。



〔说明〕

把各位的值取反码。

把上列的数值＋1。

$$\text{指数的值} = -\ (0\times2^0+0\times2^1+0\times2^2+1\times2^3+1\times2^4+0\times2^5+0\times2^6)$$
$$= -\ (0+0+0+8+16+0+0)$$
$$= -24$$

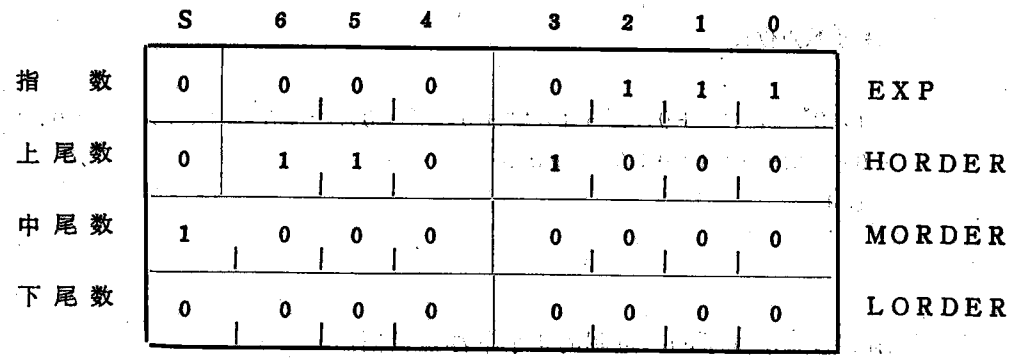
3．4字节浮点数据

用上面所示的指数和尾数，4字节浮点数据可以表示为如下的十进制实数。

$$\text{10进制实数值} = (\text{尾数的值}) * 2^{(\text{指数的值})}$$

例5）

二进制表示的4字节浮点数据的10进制实数值可用下列的变换求出。



〔说明〕

指数的值＝ 7

尾数的值＝0.81640625

实数值＝0.81640625×2^{7}

$=0.81640625\times128$

$=104.5$

指数（ＥＸＰ）的值在－128～＋128的范围内。表示$2^{(EXP)}$。指数为正时，尾数的小数点位置向右移动，移动的位数为指数的数值；指数为负时，尾数的小数点位置向左移动，移动的位数为指数原码的绝对值的数值（｜ＥＸＰ｜）移动后的位置是实际的小数点位置。

以小数点的位置为中心，在左侧各位的权依次是2^{0}、2^{1}、2^{2}、2^{3}……，在右侧各位的权依次是2^{-1}、2^{-2}、2^{-3}……。

各位的值与其权相乘后所得结果的总和即为实数的值。

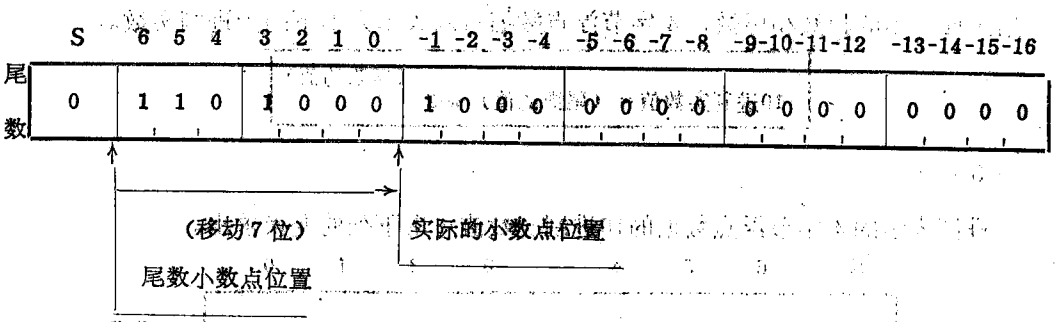
但是，尾数的符号为负时，要取3字节尾数的补码进行上列的变换并需对结果给出相应的符号。

例６），

用10进制实数表示前例的４字节浮点数据。

〔说明〕

指数的值是正的，用例３的方法将指数的值变换为７，尾数的小数点位置向右移动７位，尔后再按权相乘求和。



实数值＝$1\times2^{6}+1\times2^{5}+0\times2^{4}+1\times2^{3}+0\times2^{2}+0\times2^{1}+0\times2^{0}+1\times2^{-1}+0\times2^{-2}$

$=64+32+0+8+0+0+0+0.5+0+\cdots\cdots$

$=104.5$

例７）

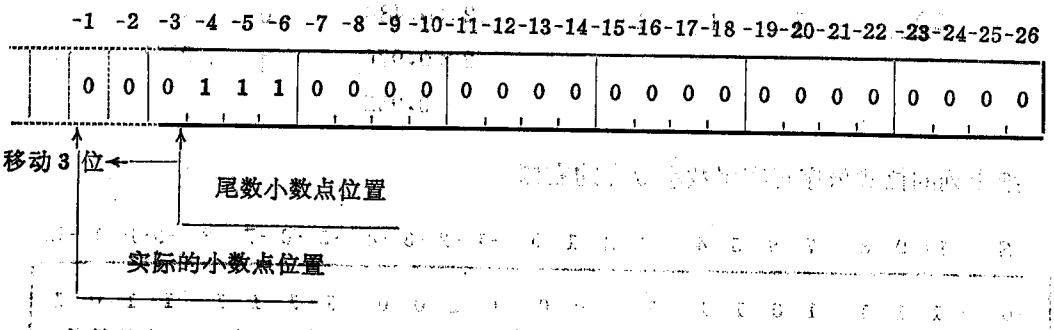
用10进制实数表示如下的二进制的４字节浮点数据。

	S	6	5	4	3	2	1	0
指　数	1	1	1	1	1	1	0	1
上尾数	1	0	0	1	0	0	0	0
中尾数	0	0	0	0	0	0	0	0
下尾数	0	0	0	0	0	0	0	0

〔说明〕

指数是负的，指数的值用例4的方法变换为－3。

尾数是负的，用例2的方法取尾数的补码，把尾数小数点位置向左移动3位，尔后按权相乘求和。



$$\text{实数值} = 0\times2^{-1}+0\times2^{-2}+0\times2^{-3}+1\times2^{-4}+1\times2^{-5}+1\times2^{-6}+0\times2^{-7}+0\times2^{-8}+\cdots\cdots$$
$$= 0+0+0+0.0625+0.03125+0.015625+0+0+\cdots\cdots$$
$$= 0.109375$$

反之，也可以把10进制实数变换为4字节的浮点数据。

根据上述说明，二进制数是以小数点的位置为中心，在左侧各位的权为2^0、2^1、2^2、2^3……，在右侧各位的权为2^{-1}、2^{-2}、2^{-3}……。

因此，为了把10进制数变换为二进制数，必须按2^n的位（bitpatten）进行变换。

例8.

把正的10进制实数1976.312的数值变换为4字节的浮点数据。

〔说明〕

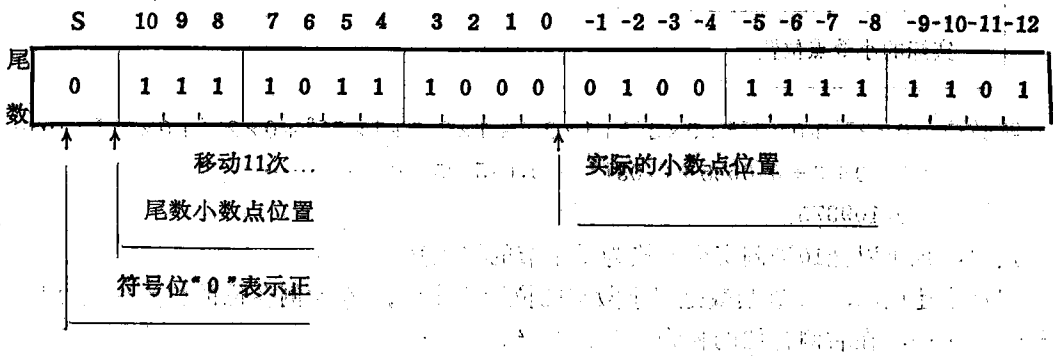
把整数和小数分开，按如下的步骤变换。

整数用2除，结果写在下面，余数写在右侧，反复上面操作，直到结果为“0”（构成2^n的位式）。

小数的处理是，用2乘以后，把结果的小数点以后的值写在下面，把结果的小数点以前的位写在右侧，构成2^{-n}的位式。

	位			位
2）1976	0	2^0	2×0.312	0 2^{-1}
2）988	0	2^1	2×0.624	1 2^{-2}
2）494	0	2^2	2×0.248	0 2^{-3}
2）247	1	2^3	2×0.496	0 2^{-4}
2）123	1	2^4	2×0.992	1 2^{-5}
2）61	1	2^5	2×0.984	1 2^{-6}
2）30	0	2^6	2×0.968	1 2^{-7}
2）15	1	2^7	2×0.936	1 2^{-8}
2）7	1	2^8	2×0.872	1 2^{-9}
2）3	1	2^9	2×0.744	1 2^{-10}
2）1	1	2^{10}	2×0.488	0 2^{-11}
			2×0.976	1 2^{-12}
			0.952	

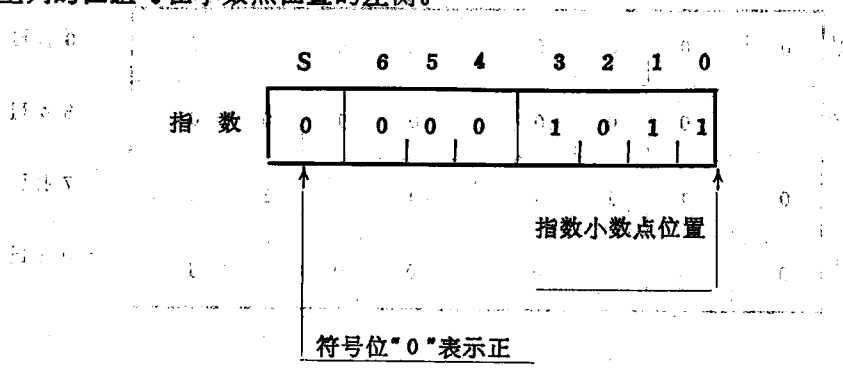
把上列的位式依序写在尾数小数点的右侧



由于实际小数点的位置向尾数小数点的位置移动了11位，故指数可像上述的整数一样地把11变换成指数的二进制位式。

		位
2）11	1	2^0
2）5	1	2^1
2）2	0	2^2
2）1	1	2^3

上列的位组写在小数点位置的左侧。



由上述可知，正的10进制实数1976.312，可表示为如下的4字节浮点形式。

													16进制数
指 数	0	0	0	0	1	0	1	1	= 0 B H
上尾数	0	1	1	1	1	0	1	1	= 7 B H
中尾数	1	0	0	0	0	1	0	0	= 8 4 H
下尾数	1	1	1	1	1	1	0	1	= F D H

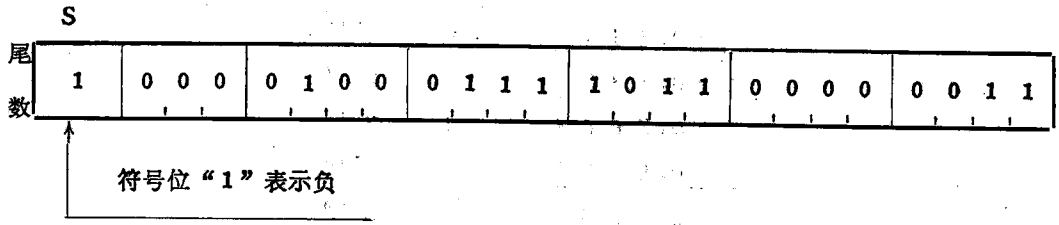
例9）

把负的10进制实数－1976.312的数值变换为4字节浮点数。

〔说明〕

取负的10进制实数的绝对值，与正的十进制实数相同把绝对值变换为如下列的3字节长的补码形式的尾数。

另外，指数与正的10进制实数相同。



因此，负的10进制实数－1976.312可表示为如下的4字节浮点形式。

	S								
指　数	0	0	0	0	1	0	1	1	= 0 B H
上尾数	1	0	0	0	0	1	0	0	= 8 4 H
中尾数	0	1	1	1	1	0	1	1	= 7 B H
下尾数	0	0	0	0	0	0	1	1	= 0 3 H

例10)

把正的10进制小数0.0887的数值变换为4字节浮点数据。

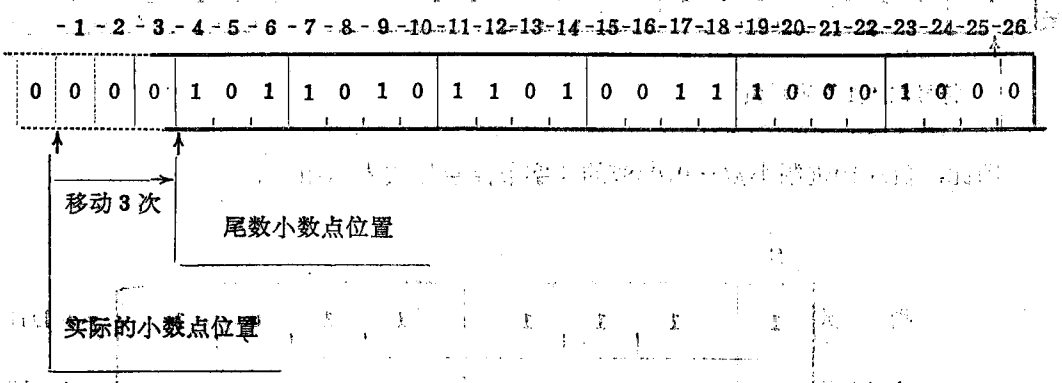
〔说明〕

因为仅仅是小数，所以变换与例8的小数部分相同。

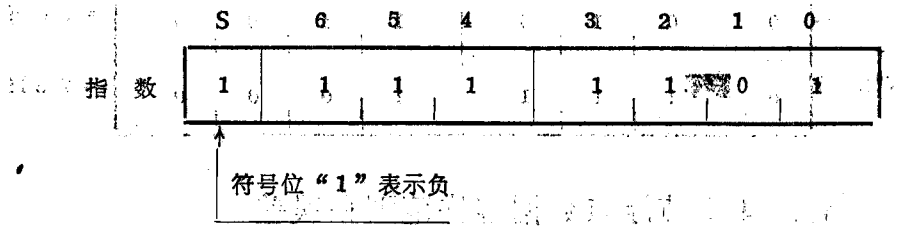
2×0.0887	0	2^{-1}
2×0.1774	0	2^{-2}
2×0.3548	0	2^{-3}
2×0.7096	1	2^{-4}
2×0.4192	0	2^{-5}
2×0.8384	1	2^{-6}
2×0.6768	1	2^{-7}
2×0.3536	0	2^{-8}
2×0.7072	1	2^{-9}
2×0.4144	0	2^{-10}
2×0.8288	1	2^{-11}
2×0.6576	1	2^{-12}
2×0.3052	0	2^{-13}
2×0.6104	1	2^{-14}
2×0.2208	0	2^{-15}
2×0.4416	0	2^{-16}
2×0.8832	1	2^{-17}
2×0.7664	1	2^{-18}
2×0.5328	1	2^{-19}
2×0.0656	0	2^{-20}
2×0.1312	0	2^{-21}

2×0.2624	0	2^{-22}
2×0.5248	1	2^{-23}
2×0.0496	0	2^{-24}
2×0.0992	0	2^{-25}
2×0.1984	0	2^{-26}
0.3968		

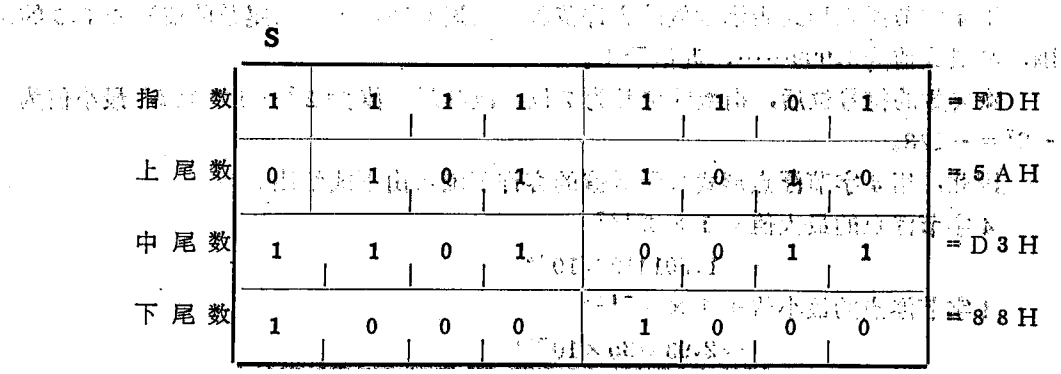
把上列的位组，写成如下的规格化尾数的形式（尾数符号位的右侧为“1”）。



因为小数点的位置向右移动3位，所以指数的值为－3，用例4的方法变换即可。



因此，正的10进制小数0.0887可用4字节浮点形式表示如下。



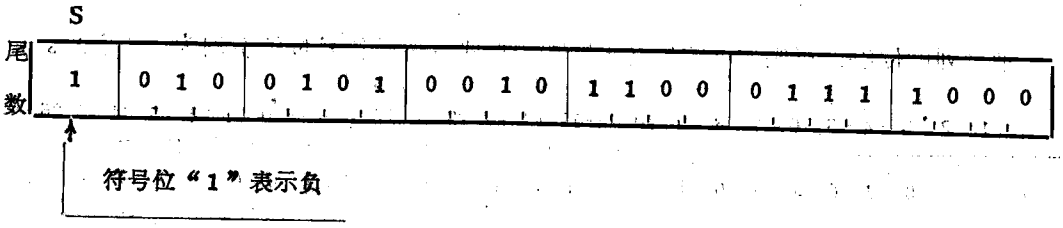
例11）

把负的10进制小数－0.0887变换为4字节浮点数。

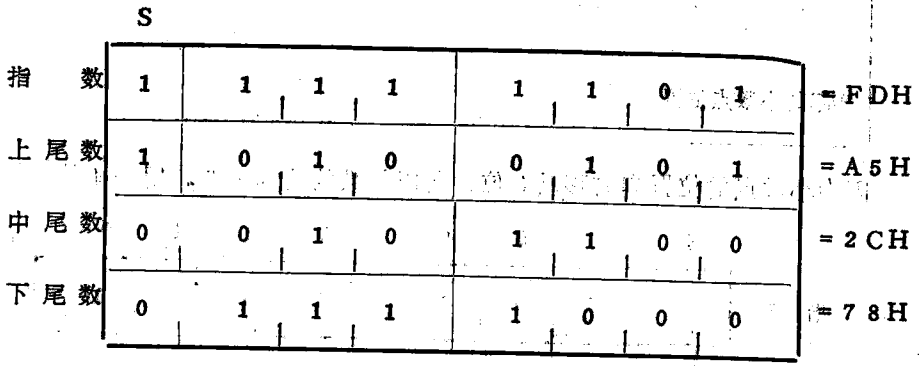
〔说明〕

取负的10进制小数的绝对值，像正的10进制小数同样变换，所得3字节长的补码形式尾数如下。

另外，指数变换与正的10进制小数相同。



因此，负的10进制小数－0.0887的4字节浮点形式表示如下。



4. 4字节浮点数据容许范围和误差

用4字节浮点形式表示数值的容许范围，其尾数在：1＞（尾数的值）≥0.5的范围，其最大值为0.999……，近似于1。

除去数的符号位后，指数的位数为7位，故其最大值为$2^7-1=127$，最小值为$-2^7=-128$。

因此，用4字节浮点形式表示数值的容许范围可由下式给出。

4字节浮点的最大值≒1×2^{127}

$=1.701412\times10^{38}$

4字节浮点的最小值≒1×2^{-128}

$=2.938735\times10^{-39}$

如上所述，用浮点形式表示数值的容许范围极大：即±（1.701412×10^{38}～2.938735×10^{-39}），这在通常用于科学计算中是足够了。

但是，4字节浮点形式的数据，其有效位数受到尾数的位数的限制。

由于除了符号位以外的尾数的位数为23。$2^{23}=8388608$，十进制的7位为有效值，其他部分即为误差。

例12)

考虑如下的数值运算，用线划出的数是有效数，其他的数位作为误差。

```
  1749568230          0.0003741562 8
+)  53214683 4      +) 0.0000329482 0
  2281714000          0.0004071044 0
```

〔说明〕

只要做4字节浮点运算，上述的运算误差就会产生，因此，在使用中间必须考虑运算的精度。

另外，为了提高运算精度（有效位数），就需要位数多的浮点子程序。例如5字节、6字节等的浮点算术运算有时也是需要的。但是，必须注意，随着字节数的增加，存贮器容量也大幅度增多。另外，还会导致处理速度的降低。

5. 4字节浮点子程序库

在4字节浮点子程序库ＳＳＬ—2000中收录了各种子程序，以下分别加以说明。

所有变换都要通过或者参照浮点累加器（＊FACC(1)）进行的，所以必须预先把需要的数据置于浮点累加器。但是，子程序＊FPSH，＊FPOP除外。

除浮点累加器之外，有时还必须指定源数据区域（或者目的数据区域）。这时，数据区域的地址已置位于寄存器对（H，L），给出并进行子程序调用。

尚且，寄存器的内容是不受特别保护的，所以需要保护的寄存器必须在子程序调用前推入堆栈。

在作各项说明之前，先叙述文中使用的记号的意义。

注：＊（1）函数子程序库内的4字节浮点数据区（参照源数据目录的101页）

＊（2）（H，L）也被破坏，必须注意。

符　号	意　义
（＊FACC）	浮点累加器的内容。
X	标注源数据或目的数据区（4字节）的起始地址的符号名。
（X）	X的内容（由4字节浮点形式表示的数据）。
Z	标志位。
S	标志位。
Ca	标志位。
H，L	寄存器对。

5—1　数据形式

源数据（或目的数据），以及置在浮点累加器数据，如前面详细叙述的那样，是用浮点形式表示的。

源（目的）数据区

```
X:      〔指数〕        ; Y地址
        〔上尾数〕      ; Y + 1
        〔中尾数〕      ; Y + 2
        〔下尾数〕      ; Y + 3
```

浮点累加器

```
＊FACC:  〔指数〕        ; Z地址
```