第 3 篇 中央处理器

第 6 章 计算机的运算方法

计算机的应用领域极其广泛，但不论应用在什么地方，信息在计算机内部的形式都是一致的，即均为 0 和 1 组成的各种编码。

6.1 无符号数和有符号数
计算机中参与运算的数有两大类：无符号数和有符号数。
计算机中的数均存放在寄存器中，通常称寄存器的位数为机器字长。8 位，32 位等
所谓**无符号数**，即没有符号的数，在寄存器中的每一位均可用来存放数值。
当存放有符号数时，则需留出位置存放符号，符号的正、负机器是无法识别的，可以用 0 表示正，1 表示负，这样符号就被数字化了，并且规定将它放在有效数字的前面，即组成了**有符号数**。
通常把符号数字化的数叫做**机器数**，把带“+”或“-”符号的数叫做真值。

一旦符号数字化后，在运算过程中，符号位能否和数值部分一起参加运算？如果参加运算，符号位又需作哪些处理？这些问题都与符号位和数值位所构成的编码有关，这些编码就是原码、补码、反码和移码。

原码表示法
原码是是机器数中最简单的一种表示形式，符号位 0 表示正数，符号位 1 表示负数。
数值位即真值的绝对值。
为了书写方便以及区别整数和小数，约定整数的符号位与数值之间用逗号隔开，小数的符号位与数值位之间用小数点隔开。
$x = +0.1011$，$[x]_{原} = 0.1011$
$x = -0.1011$，$[x]_{原} = 1 - (-0.1011) = 1.1011$
$x = +1011$，$[x]_{原} = 0,1011$
$x = -1011$，$[x]_{原} = 2^4 - x = 10000 - (-1011) = 1,1011$

补码表示法
日常生活中，经常会遇到补数的概念。
时钟转一圈能指示 12 个小时，称 12 为模，写作 mod 12。对模 12 而言，-3 和+9 互为补数。-3 可以用+9 来代替。
补数的概念可以用到任意模上：
mod 10^2：-3 和+97 互为补数。
mod 2^4：-1011 和+0101 互为补数。
mod 2：-0.1001 和+1.0111 互为补数
补数的定义：
一个负数可以用它的正补数来代替，而这个正补数可以用模加上负数本身求得。
一个正数和一个负数互为补数时，它们的绝对值之和等于模数。
正数的补数即为该正数本身。
整数补码的定义：

$0 \le x < 2^n$, $[x]_{补} = 0,x$	$[+1100]_{补} = 0,1100$
$-2^n \le x < 0$, $[x]_{补} = 2^{n+1} + x$	$[-1101] = 2^5 + (-1101) = 100000 - 1101 = 1,0011$

小数补码的定义：

$0 \le x < 1$, $[x]_{补} = x$	$[+0.1100]_{补} = 0.1100$
$-1 \le x < 0 [x]_{补} = 2 + x$	$[-0.0110]_{补} = 2 - 0.0110 = 1.1010$

反码表示法：
反码通常用来作为原码求补码或补码求原码的中间过渡，真值为负时，反码是原码的每位取反。
[+1100]反 = 0,1100
[-1100]反 = 1,0011
[+0.0110]反 = 0.0110
[-0.0110]反 = 1.1001

总结：
三种机器数的最高位均为符号位。符号位和数值部分之间可用 . 或 ，隔开。
当真值为正时，原码、补码、反码的表现形式均相同，即符号位用 0 表示，数值部分和真值相同。
当真值为负时，三者符号位都用 1 表示，原码数值部分是**真值的绝对值**，补码是原码的**求反加 1**，反码是原码的**每位求反**。
补码不能直接反映真值的大小，移码和补码仅差一个符号位，把补码中的 1 换成 0，0 换成 1，就得到了真值的移码。

6.2 数的定点表示和浮点表示

计算机中，小数点不用专门的器件给出，而是按约定的方式标出。共有两种方法表示小数点的存在，即定点表示和浮点表示。定点表示的数称为定点数，浮点表示的数称为浮点数。

定点表示
定点数即小数点固定在某一位置的数。当小数点位于数符和第一数值位之间时，机器内的数为纯小数，当小数点位于数值位之后时，机器内的数为纯整数。
采用定点数的机器叫做定点机。
数值部分的位数 n 决定了定点机中数的表示范围，若机器数采用原码，小数定点机中数的表示范围是$-(1-2^{-n}) \sim (1-2^{-n})$，整数定点机中数的表示范围是 $-(2^n - 1) \sim (2^n -1)$。
在定点机中，由于小数点的位置固定不变，故当机器处理的数不是纯小数或纯整数时，必须乘上一个比例因子，否则会产生溢出。

浮点表示
小数点位置可以浮动的数称为浮点数。
浮点数在机器中的表现形式如下：
阶符(1 位)阶码的数值部分(m 位)数符(1 位)尾数的数值部分(n 位)
浮点数由阶码 j 和尾数 S 两部分组成。
阶码是整数，阶符和阶码的位数 m 合起来反映浮点数的表示范围及小数点的实际位置；
尾数是小数，其位数 n 反映了浮点数的精度；
尾数的符号 S_f 代表了浮点数的正负。
浮点数的表示范围：
以 $N = S \times r^j$ 为例，设浮点数阶码的数值位取 m 位，尾数的数值位取 n 位，当浮点数为非规格化时，设 m=4，n=5，它能表示的最小负数为-0.11111×2^{1111}，即$-(1-2^{-5})\times(2^{16-1}-1)$
它能表示的最大负数为-0.00001×2^{-1111}，即$-2^{-n}\times(2^{1-16}-1)$
它能表示的最大正数为 0.11111×2^{1111}，即$(1-2^{-n})\times(2^{16-1})$
它能表示的最小正数为 0.00001×2^{-1111}，即 $2^{-n}\times(2^{1-16}-1)$
1-16 表示 $1- 2^m$。
当浮点数阶码大于最大阶码时，称为上溢，机器停止运算，进行中断溢出处理。
当浮点数阶码小于最小阶码时，称为下溢，此时溢出的数绝对值很小，通常将尾数各位强制置零，按机器零处理，此时机器可以正常运行。

浮点数的规格化

尾数反映了浮点数的精度，尾数的位数是有限的，如果尾数前几位是 0，会占用了尾数的位数。
为了提高浮点数的精度，其尾数必须为规格化数。
基数为 2 时，尾数的最高位为 1 的数为规格化数。规格化时，尾数左移一位，变大 2 倍，阶码减 1，这种规格化称为向左规格化，简称左规；尾数右移一位，变小 2 倍，阶码加 1，这种规格化称为右规。
基数为 4 时，尾数的最高两位不全为零的数为规格化数。规格化时，尾数左移两位，阶码减 1；右移两位，阶码加 1。
基数为 8 时，尾数的最高三位不全为零的数为规格化数。规格化时，尾数左移三位，阶码减 1；右移三位，阶码加 1。

浮点机中的基数是隐含的，而且一旦基数确定后就不再变了。
不同基数的浮点数表示形式完全相同，但基数不同，对数的表示范围和精度都有影响。
一般来说，基数 r 越大，可表示的浮点数范围越大，所表示的数的个数越多，但表示的精度反而下降(r 为 16，规格化后前三位可能是 0)。

在浮点机中，判断补码规格化形式的原则是**尾数的符号位与第一数位不同**。

定点数和浮点数的比较

（1）当浮点机和定点机中的位数相同时，浮点数的表示范围比定点数的大得多。
（2）当浮点数为规格化数时，其相对精度远比定点数高。
（3）浮点数原算要分阶码部分和尾数部分，而且运算结果都要求规格化，故浮点运算步骤比定点运算步骤多，运算速度比定点运算低，运算线路比定点运算的复杂。
（4）在溢出判断方法上，浮点数是对规格化数的阶码进行判断，而定点数是对数值本身进行判断。例如，小数定点机中的数，其绝对值必须小于 1,否则溢出，此时要求机器停止运算，进行处理。为了防止溢出，上机前必须选择比例因子，这个工作比较麻烦，给编程带来不便。而浮点数的表示范围远比定点数大，仅当上溢时机器才停止运算，故一般不必考虑比例因子的选择。

总的来说，浮点数在数的表示范围、数的精度、溢出处理和程序编程方面(不取比例因子)均优于定点数。但在运算规则、运算速度和硬件成本方面又不如定点数。

IEEE 754 标准

和浮点数形式相比，IEEE 754 把数符放在第一位，把尾数数值部分的第一位 1 省略，称为隐藏位。
按 IEEE 标准，常用的浮点数有三种
短实数 符号位 1 阶码 8 尾数 23 总位数 32
短实数 符号位 1 阶码 11 尾数 52 总位数 64
短实数 符号位 1 阶码 15 尾数 64 总位数 80

阶码用移码表示，阶码的真值都被加一个常数(偏移量)

6.3 定点运算

定点运算包括移位、加、减、乘、除几种。

移位运算

1500 相当于 15 相对于小数点左移了两位，并在小数点前面添了两个 0。
某个十进制数左移 n 位，相当于该数乘以 10^n；右移 n 位，相当于该数除以 10^n。

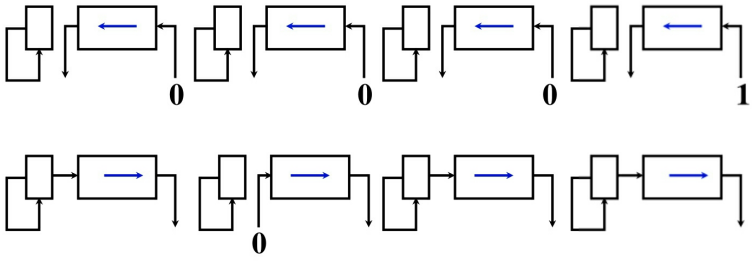
计算机中小数点的位置是事先约定的，因此，二进制表示的机器数在相对于小数点左移或右移时，实质就是该数乘以或除以 2^n（n = 1,2,3...）。
有符号数的移位称为算数移位；无符号数的移位称为逻辑移位。

算数移位规则

（1）机器数为正时，不论是左移还是右移，添补代码均为 0。
（2）由于负数的原码数值部分与真值相同，故在移位时只要符号位不变，其空位均添 0。
（3）由于负数的反码各位除符号位外与负数的原码正好相反，故移位后所添的代码应与原码相反，即全部添 1。
（4）补码中的最后一位与原码最后一位相同，前面的位数与原码相反(补码相同)。补码左移时，空位出现在低位，和原码一样，添 0；补码右移时，空位出现在高位，和反码相同，补 1。

例如-26
二进制原码为 10011010，左移一位，符号位不动，变成 10110100 (-52)；左移三位会丢掉 1，结果出错。
右移一位，变成 10001101 (-13)；左移两位，移丢一个 1，变成 10000110 (-6)，影响精度。
补码、反码同理。
反码和原码数值位相反，最低位丢 0，影响精度；最高位丢 0，结果出错。
补码中最低位和原码中一致，最低位丢 1，影响精度；最高位和反码一致，最高位丢 0，结果出错。

算数移位的硬件实现



	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

算数移位和逻辑移位的区别

有符号数的移位称为算数移位；无符号数的移位称为逻辑移位。
逻辑移位，左移，低位添 0；右移，高位添 0。
为了避免算数左移时，符号位移丢，可以把符号位移至 C_y

6.3.2 加法减法运算

减去一个数等于加上这个数的相反数。
现代计算机中都采用补码作加减法运算。

补码加法运算的基本公式
整数 $[A]_{补} + [B]_{补} = [A+B]_{补}$ （mod 2^{n+1}）
小数 $[A]_{补} + [B]_{补} = [A+B]_{补}$ （mod 2）
减法：
整数 $[A]_{补} + [-B]_{补} = [A-B]_{补}$ （mod 2^{n+1}）
小数 $[A]_{补} + [-B]_{补} = [A-B]_{补}$ （mod 2）

A = 0.1011，B = -0.0101，求$[A+B]_{补}$
$[A+B]_{补} = [A]_{补} + [B]_{补} = 0.1011 + 1.1011 = 10.0110 = 0.0110$

设机器数字长为 8 位，其中 1 位为符号位，令 A = -93，B = +45，求$[A – B]_{补}$。
[A]补 = [11011101]补 = 10100011
[-B]补 = [10101101]补 = 11010011
$[A – B]_{补}$ = [A]补 + [-B]补 = 1**01110110 = 01110110**

溢出判断：
参加操作的两个数符号相同，其结果的符号与原操作数的符号不同，即为溢出。
对于加法，只有正数加正数才会出现溢出，符号不同的两个数相加不会出现溢出。
对于减法，只有负数减正数或正数减负数才会出现溢出(符号位改变，上面的例子)，符号相同的两个数相减是不会出现溢出的。

变形补码(两位符号位的补码，模 4)的溢出判断
两位符号位不同时，表示溢出，否则，无溢出。不论是否发生溢出，高位符号位永远代表真正的符号。

6.3.3 乘法运算

笔算乘法

改进笔算乘法
（1）乘法运算可用移位和加法来实现，两个 4 位数相乘，总共要进行 4 次加法运算和 4 次移位运算。
（2）由乘数的末位值确定被乘数是否与原部分积相加，然后右移一位，形成新的部分积；同时乘数也右移一位，由次低位作新的末位，空出最高位放部分积的最低位。
（3）每次做加法时，被乘数仅仅与原部分积的高位相加，其低位被移至乘数所空出的高位位置。

第一章就介绍了运算器中的 ACC 寄存器、X 寄存器、MQ 寄存器。
ACC 寄存器用来存放乘积高位，MQ 寄存器用来存放乘数和乘积低位，X 用来存放被乘数。
ACC 和 MQ 需要有移位功能。
还需要一个全加器(n+1 位)。
计数器计算移位的次数。

原码乘法
原码和真值只差一个符号，可以用异或电路来得到结果的符号。
数值部分按绝对值相乘。
逻辑移位，符号位与数值位一起移位。
数值部分 n 位，n 次移位后乘法结束。

原码一位乘递推公式
符号位单独计算
$x^*y^* = x^*(0.y_1y_2y_3...y_n) = x^*(y_12^{-1} + y_22^{-2} + \ \dots + y_n2^{-n}) = 2^{-1}(y_1x^* + 2^{-1}(y_2x^* + \dots + 2^{-1}(y_nx^* + 0))$

$z_0 = 0$
$z_1 = 2^{-1}(y_nx^* + z_0)$
$z_2 = 2^{-1}(y_{n-1}x^* + z_1)$

…
$z_n = 2^{-1}(y_1x^* + z_{n-1})$

原码一位乘的硬件配置

除法计算
数符单独处理、

恢复余数法：
被除数大于除数(移位后的比较)，上商 1，否则，上商 0。
小数相除，被除数减去除数，余数为正，结果异常；余数为负，上商 0，加余数(恢复余数)；
左移，此时被除数减去除数，结果为正，上商 1。
再移位，减去除数，结果为正，上商 1，继续移位减除数。结果为负，上商 0，恢复余数后，移位减除数。

不恢复余数法(加减交替法，改进的恢复余数法)
当余数大于 0，商上 1，移位、减除数。
当余数小于 0，商上 0，移位、加除数。

浮点运算
浮点数由阶码和尾数组成，设基值为 2,两浮点数为：
$x = S_x \times 2^{jx}$　　$y = S_y \times 2^{jx}$
浮点运算的特点是：
阶码运算和尾数运算分开进行。

加减运算
浮点加减运算一律采用补码
（1）**对阶**
先求阶差，然后以小阶向大阶看齐原则，将阶码小的尾数右移一位，阶码加 1，直到两数阶码相等为止。
（2）**尾数求和**
将对阶后的尾数按定点加减运算规则运算。
（3）**规格化**
分左规和右规
（4）**溢出判断**

两浮点数 $x = 0.1101 \times 2^{10}$，$y = 0.1011 \times 2^{01}$，求 x+y
解：
对阶：
$y = 0.1011 \times 2^{01} = 0.0101 \times 2^{10}$
[x]补 = 00,10;00.1101　　[y]补 = 00,10;00.0101
[x+y]补 = 00,10;01.0010
右规：
00,11;00.1001
溢出判断
尾数出现 01.0010 不表示溢出，只有将此数右规后，根据阶码来判断浮点运算是否溢出。

浮点乘除法运算
两个浮点数相乘，乘积的阶码应为相乘两数的阶码之和，乘积的尾数应为相乘两数的尾数之积。

两个浮点数相除，商的阶码应为被除数的阶码减去除数的阶码，尾数为被除数的尾数除以除数的尾数所得的商。
规格化和加减运算相同。

浮点运算所需要的硬件配置
浮点运算分阶码和尾数两部分，因此浮点运算器的硬件配置比定点运算器的复杂。
对于阶码，只有加减运算，对于尾数则有加减乘除四种运算。
浮点运算器主要由两部分组成，一个是阶码运算部件，用乘法完成阶码加、减，以及控制对阶时小阶的尾数右移次数和规格化时对阶码的调整；另一个是对尾数的运算部件，用来完成尾数的运算(包括加、减、乘、除)，以及判断尾数是否已规格化。此外，还需有判断运算结果是否溢出的电路。

现代计算机可把浮点运算部件做成独立的选件，或称协处理器，用户可根据需要选择，不用选件的机器，也可以用编程的方法来完成浮点运算，不过这会影响机器的运算速度。

6.5 算术逻辑单元(ALU)

针对每一种算术运算，都必须有一个相对应的基本硬件配置，其核心部件是加法器和寄存器，当需要完成逻辑运算时，势必要配置相应的逻辑电路，而 ALU 电路是既能完成算术运算又能完成逻辑运算的部件。

快速进位链
随着操作数位数的增加，电路中进位的速度对运算时间的影响越来越大，为了提高运算速度，需要分析进位过程，设计快速进位链。
进位链是影响加法器速度的瓶颈
提高进位产生的速度

并行加法器

串行进位链

并行进位链
为了提高运算速度，除了采用高速芯片和改进算法(如用两位乘替代一位乘)，普遍采用先行进位的办法，即高位的进位不必等待低位的进位传递产生，而是与低位的进位同时产生。
如果把传递进位的电路称作进位链，那么实现先行进位的进位链通常采用**单重分组进位链**和**双重分组进位链**。

还有好多，看后面的题目

选择题

13. 当用一个 16 位的二进制数表示浮点数时，下列方案中最好的是**阶码取 5 位，尾数取 11 位**。

15. [x]补 = 1.000...0，它表示的真值是**-1**。

16. 设 x 为整数，$[x]_{补} = 1,x_1x_2x_3x_4x_5$，若要 x<-16，应满足的条件是 $\mathbf{x_1}$**必须为 0，**$\mathbf{x_2}$**~**$\mathbf{x_5}$**任意**。

18. 设[x]原 = $1.x_1x_2x_3x_4$，当**满足 $\mathbf{x_1}$必须为 0，$\mathbf{x_2}$~$\mathbf{x_4}$任意**，x>-1/2 成立。

21. 设[x]补 = $1.x_1x_2x_3x_4$，当**满足 $\mathbf{x_1}$必须为 1，$\mathbf{x_2}$~x4 至少有一个为 1** 时，x>-1/2 成立。

33. **8421 码**属于有权码。

37. 两个八进制数$(7)_{八}$和$(4)_{八}$相加后得$(13)_{八}$。

43. 下列说法有误差的是，**任何十进制小数都可用二进制表示**。(0.15 用二进制表示，无限循环)

50. 最少需用 **14** 位二进制数表示任一 4 位长的十进制整数。

52. 若 9BH 表示移码(含一位符号位)，其对应的十进制数是 10011011(10011011 原，**-27**)。

54. 设寄存器内容为 10000000，若它等于 0，则为**移码**。

58. 设寄存器内容为 11111111，若它等于-0，则为**反码**。

59. 设寄存器内容为 11111111，若它等于-127，则为**原码**。

60. 设寄存器内容为 11111111，若它等于-1，则为**补码**。

61. 设寄存器内容为 11111111，若它等于+127，则为**移码**。

65. 补码加减法是指**操作数用补码表示，连同符号位直接相加减，减某数用加负某数的补码代替，结果的符号在运算中形成**。

66. 在原码两位乘中，符号位单独处理，参加操作的数是**绝对值的补码**。

69. 两补码相加，采用 1 位符号位，则**当最高位进位和次高位进位异或结果为 1 时**，表示结果溢出。

73. 在浮点机中，判断原码规格化形式的原则是**尾数的第一数位为 1，数符任意**。

74. 在浮点机中，判断补码规格化形式的原则是**尾数的符号位与第一数位不同**。

75. 设机器字长 8 位(含 2 位符号位)，若机器数 DAH(1101 1010)为补码，，则算术左移一位(最高位不变)得(10110100，**B4H**)，算术右移一位(高位添 1,低位添 0)得(11101101，**EDH**) 。

79. 定点运算器用来进行**定点运算**。

80. 串行计算器结构简单，其运算规律是**由低位到高位逐位运算**。

81. 4 片 74181 和一片 74184 相配合，具有组内并行进位，组间并行进位传递功能。

82. 早期的硬件乘法器设计中，通常采用加和移位相结合的方法，具体算法是**并行加法和串行右移**，但需要有**计数器**控制。

83. 关于浮点运算器的描述中，正确的是浮点运算器可用两个松散连接的定点运算部件(阶码部件和尾数部件)来实现、阶码部分只实现加、减和比较运算。

84. 下面有关定点补码乘法器的描述，正确的句子是被乘数的符号和乘号的符号都参与运算、用计数器控制乘法次数。

90. 计算机中表示地址时，采用**无符号数**。

91. 浮点数的表示范围和精度取决于**阶码的位数和尾数的位数**。

95. 芯片 74181 可完成 **16 种算术运算和 16 种逻辑运算**。

96. ALU 电路属于**组合逻辑电路**。

97. 在补码定点加减法运算器中，无论采用单符号位还是双符号位，必须有溢出判断电路，它一般用**异或门**实现。

填空题

1. 计算机中广泛采用**二**进制数进行计算、存储和传递，其主要理由是**物理器件性能所致**。

2. 在整数定点机中，机器数为补码，字长 8 位(含两位符号位)，则所能表示的十进制数的范围为**-64 至+63**，前者的补码形式为 **11000000**，后者的补码形式为 **00111111**。

3. 机器数为补码，字长为 16 位(含一位符号位)，用 16 进制写出对应于整数定点机的最大正数补码是 **7FFF**，最小负数补码是 **8000**。

4. 机器数为补码，字长为 16 位(含一位符号位)，用 16 进制写出对应于小数定点机的最大正数补码是**0.FFFE**(0.11111111111)，最小负数补码是**1.0000**。(1.1111 1111 111)（小数最后可以补 0）
5. 某整数定点机，字长 8 位(含一位符号位)，当机器数分别采用原码、补码、反码以及无符号数时，其对应真值的表示范围分别是**-127~+127, -128~+127, -127~+127, 0~255**。
6. 某小数定点机，字长 8 位(含一位符号位)，当机器数分别采用原码、补码、反码时，其对应真值的表示范围分别是**-127/128~+127/128, -1~+127/128, -127/128~+127/128**。
7. 在整数定点机中，字长 8 位(含一位符号位)，若寄存器内容为 10000000，当它分别表示为原码、补码和反码及无符号数时，其对应的真值分别为**-0，-0，-127，128**。
8. 在小数定点机中，采用 1 位符号位，若寄存器内容为 10000000，当它分别表示为原码、补码和反码时，其对应的真值分别为**-0，-1 ，-127/128**。
9. 在整数定点机中，采用 1 位符号位，若寄存器内容为 11111111，当它分别表示为原码、补码和反码时，其对应的真值分别为**-127，-1 ，-0**。
10. 在小数定点机中，采用 1 位符号位，若寄存器内容为 11111111，当它分别表示为原码、补码和反码时，其对应的真值分别为**-127/128，-1/128 ，-0**。
11. 机器字长为 8 位(含一位符号位)，当 x=-128(十进制)时，其对应的二进制为**-1000000**，$[x]_{原}$= **不能表示** $[x]_{补}$= **11111111**， $[x]_{反}$= **不能表示**， $[x]_{移}$ = **01111111**。
12. 机器字长为 8 位(含一位符号位)，当 x= -127(十进制)时，其对应的二进制为**-1111111**，$[x]_{原}$= **11111111** ， $[x]_{补}$ = **10000001**， $[x]_{反}$ = **10000000**， $[x]_{移}$ = **00000001**。
13. 在整数定点机中，机器字长为 8 位(一位符号位)，当 x = - 1 时，其对应的二进制为**-0000001**，$[x]_{原}$ =**10000001** ，$[x]_{反}$ = **11111110**，$[x]_{补}$ = **11111111**，$[x]_{移}$ = **01111111**。
14. 在整数定点机中，机器字长为 8 位(一位符号位)，当 x = - 0 时，其对应的二进制为**-0000000**，$[x]_{原}$ =**10000000** ，$[x]_{反}$ = **11111111**，$[x]_{补}$ = **00000000**，$[x]_{移}$ = **10000000**。
15. 在整数定点机中，机器字长为 8 位(一位符号位)，当 x = + 100 (十进制)时，其对应的二进制为**1100100**，$[x]_{原}$ = **01100100** ，$[x]_{反}$ = **01100100**，$[x]_{补}$ = **01100100**，$[x]_{移}$ = **11100100**。
16. 在整数定点机中，机器字长为 8 位(一位符号位)，当 x = + 127 (十进制)时，其对应的二进制为**1111111**，$[x]_{原}$ = **01111111** ，$[x]_{反}$ = **01111111**，$[x]_{补}$ = **01111111**，$[x]_{移}$ = **11111111**。
17. 设机器数字长为 8 位(含一位符号位)，若机器数为 00H(十六进制)，当它分别代表原码、补码、反码和移码时，等价的十进制整数分别为 **0，±0，0，-128**。
18. 设机器数字长为 8 位(含一位符号位)，若机器数为 80H(十六进制)，当它分别代表原码、补码、反码和移码时，等价的十进制整数分别为**-0，-128，-127，±0**。
19. 设机器数字长为 8 位(含一位符号位)，若机器数为 81H(十六进制)，当它分别代表原码、补码、反码和移码时，等价的十进制整数分别为**-1，-127，-126，+1**。
20. 设机器数字长为 8 位(含一位符号位)，若机器数为 FEH(十六进制)，当它分别代表原码、补码、反码和移码时，等价的十进制整数分别为**-126，-2，-1，+126**。
21. 设机器数字长为 8 位(含一位符号位)，若机器数为 FFH(十六进制)，当它分别代表原码、补码、反码和移码时，等价的十进制整数分别为**-127，-1，-0，+127**。
22. 采用浮点表示时，若尾数为规格化形式，则浮点数的表示范围取决于**阶码**的位数，精度取决于**尾数**的位数，**数符**确定浮点数的正负。
23. 一个浮点数，当其尾数右移时，欲使其值不变，阶码必须**增加**。尾数右移一位，阶码**加一**。
24. 对于一个浮点数，**阶码的大小**确定的小数点的位置，当其位数左移时，欲使其值不变，必须使**阶码减小**。
25. 采用浮点数表示时，最大浮点数的阶符一定为**正**，尾数的符号一定为**正**。最小浮点数的阶符一定为**正**，尾数的符号一定为**负**。
26. 移码常用来表示浮点数的**阶码**部分，移码和补码除了符号位不同外，其他各位相同。
27. 采用浮点表示时，当阶码和尾数的符号位均为正，其他的数字全部为 **1** 时，表示的是最大的浮点数。当阶码的符号为**正**，尾数的符号为**负**，其他数字全部为 1 时，这是最小的浮点数。

28. 设浮点数字长为 24 位，欲表示-6×10^4~6×10^4之间的十进制数，在保证数的最大精度条件下，除阶符、数符各取 1 位外，阶码应取 **5** 位，尾数应取 **17** 位。按这样分配，这 24 位浮点数的溢出条件是**阶码大于+31**。

29. 已知 16 位长的浮点数，欲表示-3×10^4~3×10^4之间的十进制数，在保证数的最大精度条件下，除阶符、数符各取 1 位外，阶码应取 **4** 位，尾数应取 **10** 位。这种格式的浮点数补码形式)，当**阶码小于-16** 时，按机器零处理。

30. 当 0>x>-1 时，满足[x]原 = [x]补的 x 值是-1/2， 当 0>x>-128 时，满足[x]原 = [x]补的 x 值是-64。

31. 最少需用 **17** 位二进制数可表示任一五位长的十进制数。

32. 设 24 位长的浮点数，其中阶符 1 位，阶码 5 位，数符 1 位，尾数 17 位，阶码和尾数均用补码表示，且尾数采用规格化形式，则它能表示的最大正数真值是 $\mathbf{2^{31}\times(1-2^{-17})}$，非零最小正数真值是 $\mathbf{2^{-33}}$，绝对值最大的负数真值是$\mathbf{-2^{31}}$，绝对值最小的负数真值是 $\mathbf{2^{-32}\times(-2^{-1}-2^{-17})}$。

33. 设浮点数阶码为 8 位(含一位阶符)，尾数为 24 位(含一位阶符)，则在 32 位进制补码浮点规格化数对应的十进制真值范围内：最大正数为 $\mathbf{2^{127}\times(1-2^{-23})}$，最小正数为 $\mathbf{2^{-129}}$，最大负数为 $\mathbf{2^{-128}\times(-2^{-1}-2^{-23})}$，最小负数为$\mathbf{-2^{127}}$。

34. 设机器数字长为 8 位(含一位符号位)，对应十进制数 x = -0.6875 的[x]原为 1.1011000，[x]补为 **1.0101000**，[x]反为 **1.0100111**，[-x]原为 **0.1011000**，[-x]补为 **0.1011000**，[-x]反为 **0.1011000**。

35. 设机器数字长为 8 位(含一位符号位)，对应十进制数 x = -52 的[x]原为 **1,0110100**，[x]补为 **1,1001100**，[x]反为 1**,1001011**，[-x]原为 **0,0110100**，[-x]补为 **0,0110100**，[-x]反为 **0,0110100**。

36. 补码表示的二进制浮点数，尾数采用规格化形式，阶码 3 位(含一位阶符)，尾数 5 位(含一位符号位)，则所对应的最大正数真值是 **7.5**，最小正数真值是 **1/32**，最大负数真值是**-9/256**(1,00;1.0111)，最小负数真值是**-8(**0,11;1.0000)。

37. 某机器字长 16 位(含一位符号位)，它能表示的无符号整数的范围是 **0~65535**，用原码表示的定点小数范围是$\mathbf{-(1-2^{-15})}$~$\mathbf{(1-2^{-15})}$，用补码表示的定点小数范围是$\mathbf{-1}$~$\mathbf{(1-2^{-15})}$，用补码表示的定点整数范围是-32768~32767。

38. 已知十进制数 x = -2.75，分别写出对应 8 位字长的定点小数(含 1 位符号位)和浮点数(其中阶符 1 位、阶码 2 位、数符 1 位，尾数 4 位)的各种机器数，要求定点数比例因子选取2^{-4}，浮点数为规格化数，则定点表示法对应的[x]原为 **1.0010110**，[x]补为 **11101010**， [x]反为 **1.1101001**，浮点表示法对应的[x]原为 **0,10;11011**，[x]补为 **010;10101**， [x]反为 **0,10;10100**。

39. 已知十进制数 x = -5.5，分别写出对应 8 位字长的定点小数(含 1 位符号位)和浮点数(其中阶符 1 位、阶码 2 位、数符 1 位，尾数 4 位)的各种机器数，要求定点数比例因子选取2^{-4}，浮点数为规格化数，则定点表示法对应的[x]原为 **1.0101100**，[x]补为 **1.1010100**， [x]反为 **1.1010011**，浮点表示法对应的[x]原为 **0,11;1.1011**，[x]补为 **011;1.0101**， [x]反为 **0,11;1.0100**。

40. 设浮点数字长为 16 位(其中阶符 1 位，阶码 5 位，数符 1 位，尾数 9 位)，对应十进制数-87 的浮点规格化补码形式为 **0,00111;1.010100100**，若阶码采用移码，尾数采用补码，则机器数形式为 **1,00111;1.010100100**。

41. 设浮点数字长为 16 位(其中阶符 1 位，阶码 5 位，数符 1 位，尾数 9 位)，对应十进制数-95 的浮点规格化补码形式为 **0,00111;1.010000100**，若阶码采用移码，尾数采用补码，则机器数形式为 **1,00111;1.010000100**。

42. 在计算机中，一个二进制代码表示的数可理解成**指令**或**数据**或**字符**或**地址**或**逻辑值**。

44. 已知[x]补 = 1.0000，则$[1/2x]_{补}$ =**1.1000** ，x = **-1**，$[x]_{原}$ = **不能表示**，$[x]_{反}$ = **不能表示** 。

45. 设机器代码为 FCH，机器数为补码形式(采用一位符号位)，则对应的十进制真值为**-4**，其原码形式为 **84H**，反码形式为 **FBH**(均用十六进制表示)。

46. 设机器代码为 C5H，机器数为补码形式(采用一位符号位)，则对应的十进制真值为**-59**，其原码形式为 **BBH**，反码形式为 **C4H**(均用十六进制表示)。

47. 已知$[x]_{补}$ = 1.1010100，则 x = **-11/32**，$[1/2x]_{补}$ = **1.1101010**。

48. 若$[x]_{反}$ = 1.0101011，则$[-x]_{补}$ = **0.1010100**，设 x*为绝对值，则$[-x^*]_{补}$ = **1.0101100**。

49. 若$[x]_{反}$ = 0.01010，则$[-x]_{补}$ = **1.10110**，设 x*为绝对值，则[-x*]补 = **1.10110**。
50. 设 x*为绝对值，等式$[-x]_{补}$ = $[-x^*]_{补}$成立的条件是 **x 为正数或 0**。
51. 最少需用 **14** 位二进制数就能表示任一四位长的无符号整数。
52. 某浮点数基值为 2，阶码 4 位(含一位阶符)，尾数 8 位(含一位阶符)，阶码和尾数均用补码表示。它能表示的最大正数真值是 **127**，非零最小正数真值是 $2^{-8}\times2^{-7}$ = $\mathbf{2^{-15}}$，最大负数真值是 $2^{-8}\times(-2^{-7})$ = $\mathbf{-2^{-15}}$，最小负数真值是**-128**。如果尾数采用规格化表示，上述值分别是 **127**，$\mathbf{2^{-9}}$，$\mathbf{-(2^{-9}+2^{-15})}$，**-128**。如果阶码采用移码表示，上述值**不变**。
56. 假设阶码取 3 位，尾数取 8 位(均不包括符号位在内)，则对应十进制数-73.5 的原码是 **0,111;1.10010011** ，补码是 **0,111;1.01101101**，反码是 **0,111;01101100**。若阶码用移码表示，尾数用补码表示，则机器数为 **1,111;1.01101101**。
57. 在浮点表示时，若用全 0 表示机器零(尾数为 0, 阶码最小)，则阶码应采用**移码**机器数形式。在定点表示时，若要求数值 0 在计算机中唯一表示为全 0，则机器数为**补码**。
58. 正数原码算术移位时，**符号位**不变，空位补 **0**。负数原码算术移位时**符号位**不变，空位补 **0**。
59. 正数补码算术移位时，**符号位**不变，空位补 **0**。负数补码算术左移时**符号位**不变，低位补 **0**, 负数补码算术右移时，**符号位**不变，高位补 **1**。
60. 正数反码算术移位时，**符号位**不变，空位补 **0**。负数反码算术左移时**符号位**不变，低位补 **0**, 负数反码算术右移时，**符号位**不变，高位补 **1**。
61. 已知寄存器位数为 8 位，机器数取 1 位符号位，设其内容为 11110101 当它代表无符号数时，逻辑左移一位后得 **11101010**，逻辑右移一位后得 **01111010**。当它代表补码时，算术左移一位后得 **11101010**，算术右移一位后得 **10111010**。
62. 已知寄存器位数为 8 位，机器数取 1 位符号位，设其内容为 01101100 当它代表无符号数时，逻辑左移一位后得 **11011000**，逻辑右移一位后得 **00110110**。当它代表补码时，算术左移一位后得 **01011000**，算术右移一位后得 **00110110**。
63. 已知寄存器位数为 8 位，机器数为补码(含 2 位符号位)，设其内容为 00101101，算术左移一位后得 **01011010**，此时机器数符号为**正**；算术右移一位后得 **00010110**，此时机器数符号为**正**。
64. 已知寄存器位数为 8 位，机器数为补码(含 2 位符号位)，设其内容为 11001011，算术左移一位后得 **10010110**，此时机器数符号为**负**；算术右移一位后得 **11100101**，此时机器数符号为**负**。
65. 设机器数字长为 8 位(含两位符号位)，对应真值 x = -5/16 的$[x]_{补}$ = **11.101100** ，算术左移 1 位后得 **11.011000**，算术左移两位后得 **10.110000** ，算术右移一位后得 **11.110110**，算术右移两位后得 **11.111011**。移位后对应的真值分别为**-5/8**、**负溢**、**-5/32**、**-5/64**。
66. 设机器数字长为 8 位(含两位符号位)，对应真值 x = -26 的$[x]_{补}$ = **11,100110** ，算术左移 1 位后得 **11,001100**，算术左移两位后得 **10,011000**，算术右移一位后得 **11,110011**，算术右移两位后得 **11,111001**。移位后对应的真值分别为**-52**、**负溢**、**-13**、**-7**。
67. 正数原码左移时，符号位不变，高位丢 1，结果出错；右移时低位丢 1，结果引起误差。负数原码左移时，符号位不变，高位丢 1，结果出错，右移时低位丢 0，结果正确。
73. 两个 n+1 位(含 1 位符号位)的原码在机器中作一位乘运算，共需做 **n 次移位**操作，最多需做 **n 次加法**操作，才能得到最后乘积，乘积的符号位需**通过两数符号位异或运算获得**。
74. 设操作数字长 16 位(不包括符号位)，机器做原码两位乘运算，共需做 **8 次移位操作**和 **9 次加法操作**，才能得到最后的乘积，乘积的符号位**由两数符号位异或运算获得**。
75. 设操作数字长 15 位(不包括符号位)，机器做原码两位乘运算，共需做 **8 次移位操作**和 **8 次加法操作**，才能得到最后的乘积，乘积的符号位**由两数符号位异或运算获得**。
76. 定点原码除法和定点补码除法均可采用**加减交替法**，但补码除法中**符号位**参与运算。
77. 在补码一位乘中，设$[x]_{补}$为被乘数，$[y]_{补}$为乘数，若 y_ny_{n+1} = 00，应执行**右移一位**操作，若 y_ny_{n+1} = 01，应执行**+$[x]_{补}$，右移一位**操作。若 y_ny_{n+1} = 10，应执行**+$[-x]_{补}$，右移一位**操作。若 y_ny_{n+1} = 11，应执行**右移一位**操作。若机器字长为 16 位(不包括符号位)，则补码乘法最多需做 **16 次移位操作**和 **17 次加法操作**。

78. 在补码除法中，设$[x]_{补}$为被除数，$[y]_{补}$为除数。除法开始时，若$[x]_{补}$和$[y]_{补}$同号，需做**$[x]_{补}+[-y]_{补}$**操作，得余数$[R]_{补}$，若$[x]_{补}$和$[y]_{补}$异号，上商 **0**，再执行 **$2[R]_{补}+[y]_{补}$**操作。若机器数为 8 位(含一位符号位)，共需上商 **8** 次，且最后一次上商 **1**。

79. 在补码除法中，设$[x]_{补}$为被除数，$[y]_{补}$为除数。除法开始时，若$[x]_{补}$和$[y]_{补}$异号，需做**$[x]_{补}+[y]_{补}$**操作，得余数$[R]_{补}$，若$[x]_{补}$和$[y]_{补}$同号，上商 **1**，再执行 **$2[R]_{补}+[-y]_{补}$**操作。若机器数为 15 位(不包含符号位)，共需上商 **16** 次，且最后一次上商 **1**。

80. 在浮点补码二进制加减运算中，当尾数部分出现 **01.xxxx** 和 **10.xxxx** 形式时，需进行右规，此时**尾数右移一位，阶码加 1**；当尾数部分出现 **00.0xxx** 和 **11.1xxx** 形式时，需进行左规。

83. 已知浮点数尾数 24 位(不包括符号位)，当它分别代表原码、补码、反码时，左规的最多次数分别为 23、24、23，右规的最多次数分别为 1、1、1 次。

84. 在浮点加减运算中，对阶时需**小阶向大阶看齐**，即小阶的尾数向**右**移位，每移一位，阶码**加 1**，直到两数的阶码相等为止。

85. 假设机器数字长为 32 位(不包括符号位)，若一次加法需 1 微秒，一次移位需 1 微秒，则完成原码一位乘、原码两位乘、补码一位乘、补码加减交替法(不考虑上商时间)各需 **64 微秒**、**33 微秒**、**65 微秒**、**64 微秒**时间。

86. 运算器的技术指标一般用**机器字长**和**运算速度**表示。

87. 定点数和浮点数是按数的**小数点的位置**来区分的，定点运算器的结构**简单**，但表示数的范围**小**，常用于**小型机、微型机、单片机**类机器。

88. 运算器能进行**算术逻辑运算**。运算器中通常需有三个寄存器，称为**累加器**、**乘商寄存器**和**操作数寄存器**。

89. 一些大中型通用计算机的运算器既能进行**定点运算**，又能进行**浮点运算**，这主要取决于机器的**指令系统**。

90. 浮点运算器由**阶码运算器**和**尾数运算**器组成，它们都是**定点**计算器。前者只要求执行**加减运算**，后者要求能进行**加减乘除运算**。

91. 现代计算机中，通常将运算器和**控制器**制作在一个芯片内，称为 **CPU** 芯片。

92. 按信息的传送方式分、运算器可分为**串行**、**并行**、**串并行**三种结构，其中**串行运算器**最省器材，**并行运算器**运算速度最快。

93. 存放在两个寄存器中的 n 位长补码，欲实现串行加减运算，最基本的电路应有**一位全加器**和**一位触发器**，前者用来**实现加减运算**，后者用作**存放进位**。若 t_1 和 t_2 分别代表它们的延迟，则执行 n 位加法所需的时间为 **$n(t_1+t_2)$**，随着 n 的增加，**全加器和触发器的数目**不变。

94. 为了提高运算器的速度，通常可采用**高速器件**、**快速进位链**和**改进算法**三种方法。

95. 三态缓冲门可组成运算器的数据总线，其输入电平有**高电平**、**低电平**、**浮空**三种状态，它是靠**允许/禁止(控制)**输入端上的高低电平来控制的，当该输入端**为无效电平**时，输出阻抗呈现**高阻**。

96. 多路开关是一种用来从 n 个数据源中选择**一**个数据到其输出端的器件，假设 $n=2^p$，则源的选择由 **P 位的编码格式**所决定的。

97. 算术/逻辑运算单元 74181 ALU 可以对 **4** 位信息完成 **16 种算术运算**和 **16 种逻辑运算**。

98. 进位的逻辑表达式中有**本地进位**和**传送进位**两部分，影响速度的是**传送进位**。

99. ALU 属于**组合逻辑电路**，因此在运算过程中，其**输入数据**必须保持不变，欲获得运算结果，必须在 ALU 的输入端设置**暂存器**。

100. 进位链是**传送进位的逻辑电路**。

101. 先行进位是指**高位的进位不必等低位的进位产生后再形成，高位的进位与低位的进位同时产生**。

102. 单重分组跳跃进位链的工作原理是**将 n 位全加器分成若干小组，小组内进位同时产生，小组之间采用串行进位**。

103. 双重分组跳跃进位链的工作原理是**将 n 位全加器分成几个大组，每个大组里又包含若干个小组，大组内每个小组的最高进位是同时产生的，大组与大组之间采用串行进位；小组内的其他位进位也同时产生**。

105. 在定点运算器中，无论采用单符号位还是双符号位，必须有**判断溢出**电路，它一般用**异或门**来实现。
106. 运算器内通常设有反映**运算结果**状态的寄存器，利用该寄存器的内容可以提供**判断条件**，以实现程序的**控制转移**。
107. 74187 可进行**算术逻辑**运算，74182 称作**先行进位**部件、它可实现**小组与小组**之间的先行进位。一个具有二级先行进位的 32 位 ALU 电路需有 **8** 片 74181 和 **2** 片 74182。
108. 运算器由许多部件组成，除寄存器外，其核心部分是**算术逻辑运算单元**，记为 **ALU**。
109. 若移码的符号为 1，则该数为**正**数；若符号为 0，则为**负**数。
110. 在原码、补码、反码和移码中，**原码、反码**对 0 有两种形式，**补码、移码**对 0 的表示只有一种形式。
111. 设机器字长为 8 位，-1 的补码在整数定点机中表示为 1,1111111，在小数定点机中表示为 1.0000000。
112. 在浮点数中，尾数用原码表示时，其规格化特征是**符号位任意，第一数字位为 1**。尾数用补码表示时，其规格化特征是**符号位与第一数字位不同**。
113. 一个定点数由**数符**和**数值位**两部分组成。根据小数点的位置不同，定点数有**纯小数**和**纯整数**两种表示方法。
114. 16 位二进制补码(含 1 位符号位)所能表示的十进制整数的范围是-32768~+32767，前者的十六进制补码表示为 8000H，后者的十六进制补码表示为 7FFFH。
115. 在各种机器数中，0 为唯一形式的机器数是补码和移码；表示定点整数时，要求数值 0 在计算机中唯一表示为全 0，应采用补码；表示浮点数时，若要求机器零在计算机中表示为全 0，则阶码应采用移码。
116. 若寄存器内容为 FFH，若其表示 127，则为**移码**，若表示-127 则为**原码**；若其表示-1，则为**补码**；若其表示-0，则为**反码**。
117. 在浮点数的基值确定后，且尾数采用规格化形式，则浮点数的范围取决于**阶码的位数**，精度取决于**尾数的位数**，小数点的真正位置取决于**阶符和阶码值**。
118. 32 位长的浮点数，其中阶码 8 位(含一位符号位)，基值为 2，尾数 24 位(含一位数符)。当阶码和尾数均用原码表示时，且尾数为规格化形式，所对应的最小负数是-$\mathbf{2^{127}\times(1-2^{-23})}$，最小正数是 $\mathbf{2^{-128}}$；当阶码和尾数均用补码表示时，且尾数为规格化形式，则对应的最大负数是-$\mathbf{2^{-128}\times(2^{-1}+2^{-23})}$，最小正数是 $\mathbf{2^{-129}}$。
121. 在计算机中，有符号数共有**原码**、**反码**、**补码**、**移码**四种表示法。
122.若[x]补 = 1.0000000，则 x = **-1**；若[x]补 = 1,0000000，则 x = **-128**。
123. 在浮点数中，当数的绝对值太大，以至于大于阶码所能表示的数值时，称为浮点数的**上溢**，当绝对值太小，以至于小于阶码所能表示的数值时，称为浮点数的**下溢**。**上溢**时，机器需停止运算，做中断处理。
124. 当浮点数的尾数部分为 0，不论阶码为何值，机器都把该浮点数当作**机器零**处理。

问答题

1. 设浮点数字长 16 位，其中阶码 4 位(含 1 位符号位)，尾数 12 位(含一位数符)，将$(51/128)_{+}$转换成二进制规格化浮点数及机器数(其中阶码采用移码，基值为 2,尾数采用补码)，并回答次浮点格式的规格化数表示范围。

二进制表示：0.01100110000

答：

二进制规格化浮点数为 $2^{-1}\times0.11001100000$

0,0111;0.11001100000

表示范围

最大正数：$2^{7}\times(1-2^{-11})$　　1,111;0.11111111111

最小正数：$2^{-8}\times0.100.. = 2^{-9}$　　0,111;0.10000000001

最大负数：$2^{-8}\times1.011...1 = -2^{-8}\times(2^{-1}+2^{-11})$　　0,111;1.01111111111

最小负数：$2^{7}\times(-1) = -2^{7}$　　1111;1.00000000000

2. 设浮点数字长 16 位，其中阶码 4 位(含 1 位阶符)，尾数 12 位(含 1 位数符)，将$(-43/128)_+$转换成二进制规格化浮点数及机器数(其中阶码采用移码，基值为 2,尾数采用补码)，并回答此浮点格式的规格化数的表示范围。

答：

$(-43/128)_+ = -0.0101011$ 规格化后-0.1010110×2^{-1}

规格化浮点数：$2^{-1}\times1.01010100000$

机器数：1,111;1.01010100000

表示范围：最大正数：$2^7\times(1-2^{-11})$　　1,111;0.11111111111

最小正数：$2^{-8}\times0.100.. = 2^{-9}$　0,111;0.10000000001

最大负数：$2^{-8}\times1.011...1 = -2^{-8}\times(2^{-1}+2^{-11})$　　0,111;1.01111111111

最小负数：$2^7\times(-1) = -2^7$　　1111;1.00000000000

3. 设浮点数字长 16 位，其中阶码 5 位(含 1 位阶符)，尾数 11 位(含 1 位数符)，将(-13/64)十转换成二进制规格化浮点数及机器数(其中阶码采用移码，基值为 2，尾数采用补码)，并回答此浮点数的规格化数表示范围。

答：

$(-12/64)_+ = -0.001101$ 规格化后-0.1101×2^{-2}

规格化浮点数：$2^{-2}\times1.00110000000$

机器数：1,1110;1.00110000000

表示范围：最大正数：$2^{15}\times(1-2^{-10})$

最小正数：$2^{-16}\times0.100.. = 2^{-17}$

最大负数：$2^{-16}\times1.011...1 = -2^{-16}\times(2^{-1}+2^{-10})$

最小负数：$2^{15}\times(-1) = -2^{15}$

5. 设浮点数字长 16 位，其中阶码 8 位(含一位阶符)，尾数 8 位(含一位数符)，阶码采用移码表示，基值为 2,尾数采用补码表示，计算：

(1) 机器数为 81D0H 的十进制数值。

(2) 此浮点格式的规格化表示范围。

答：

（1）$1,0000001;1.1010000 = 2^1 \times -0.01100000 = -0.11 = (-0.75)_+$

（2）表示范围：

最大正数：$2^{127} \times (1-2^{-7})$

最小正数：$2^{-128} \times (2^{-1}) = 2^{-129}$

最大负数：$-2^{-128}\times(2^{-1}+2^{-7})$

最小负数：-2^{127}

9. 设浮点数字长 32 位，其中阶码 8 位(含一位阶符)，尾数 24 位(含 1 位数符)，当阶码的基值分别为 2 和 16 时：

(1) 说明 2 和 16 在浮点数中如何表示。

(2) 当阶码和尾数均用补码表示，且尾数采用规格化表示时，给出两种情况下所能表示的最大正数真值和非零最小正数真值。

答：

（1）基值 2 和 16 是隐含约定的，在浮点数中的表示形式完全相同，阶码和尾数均用二进制表示，运算规则也基本相同。但对阶码和规格化操作时，若基值为 2,则每当阶码增 1 或减 1 时，尾数相应移 1 位；若基值为 16，当阶码增 1 或减 1 时，尾数相应移 4 位。

（2）基值为 2，所能表示的最大正数为$2^{127}\times(1-2^{-23})$，所能表示的最小正数是$2^{-128}\times2^{-1} = 2^{-129}$

基值为 16，所能表示的最大正数为$16^{127}\times(1-2^{-23})$，所能表示的最小正数是$16^{-128}\times2^{-4} = 2^{-129}$

11. 给定下列十六进制数，若将此数分别视为无符号数、原码、反码和移码表示，写出对应的十进制整数值(有符号数的符号位占 1 位)。

00H，05H，7FH，80H，85H，FEH，FFH

答：

十六进制数	无符号数	原码	反码	移码
00H，0000 0000	0	+0	+0，-0	-128
05H，0000 0101	5	+5	+5	-123
7FH，0111 1111	127	+127	+127	-1
80H，1000 0000	128	-0	-128	+0，-0
85H，1000 0101	133	-5	-123	+5
FEH，1111 1110	254	-126	-2	+126
FFH，1111 1111	255	-127	-1	+127

13. 已知$[y]_{补} = y_0.y_1y_2...y_n$，求$[-y]_{补}$。

答：写不清楚，看笔记本。

14. 若$[x]_{补} > [y]_{补}$，是否有 x>y?

答：不一定成立。

x、y 为正时，若$[x]_{补} > [y]_{补}$，则 x>y 成立

x 为正 y 为负时，若$[x]_{补} > [y]_{补}$，不管怎样，一定有 x>y 成立

x 为负 y 为正时，若$[x]_{补} > [y]_{补}$，依然有 x<y，假设不成立

x、y 为负时，若$[x]_{补} > [y]_{补}$，则 x>y 成立

19. 设浮点数字长 16 位，其中阶码 5 位(含 1 位阶符)，尾数 11 位(含 1 位数符)，写出$(-29/1024)_+$对应的浮点规格化数的原码、补码、反码和阶码用移码，尾数用补码表示的形式。

$(-29/1024)_+ = -0.0000011101$

原码：$2^{-5}\times1.11101$，即 1.0101;1.1110100000

补码：$2^{-5}\times1.11101$，即 1,1011;1.0001100000

反码：$2^{-5}\times1.11101$，即 1,1010;1.0001011111

阶码用移码，尾数用补码：0,1011;1.0001100000

21. 如何判断一个七位二进制整数 $A = a_1a_2a_3a_4a_5a_6a_7$ 是否是 4 的倍数?

答：最后两位，即 a_6、a_7 为 0 时，A 是 4 的倍数。

22. 简述算术移位和逻辑移位的区别，举例说明。

答：算术移位时，符号位(最高位)不变，左移时最高数值位移丢，右移时最低数值位移丢，移位时出现的空位根据不同机器数的移位规则确定添 0 或 1。

逻辑移位时，没有符号位，左移时最高位移丢，低位补 0；右移时最低位移丢，高位补 0。

例如，10110011，逻辑左移一位得 01100110，算术左移一位得 11100110，结果不同。

23. 讨论三种机器数在算术左移或右移时，对结果的影响(指出何时正确，何时有误)。
答：当真值为正时，左移或右移，符号位不变，左移时最高位丢 1，结果出错，右移时最低位丢 1，出现误差。
当真值为负时，原码移位时，符号位不变，左移时最高位丢 1，结果出错，右移时最低位丢 1，出现误差。
补码移位时，符号位不变，左移时最高位丢 0，结果出错，右移时最低位丢 1，出现误差。
反码移位时，符号位不变，左移时最高位丢 0，结果出错，右移时最低位丢 0，出现误差。

24. 在定点机中采用单符号位，如何判断补码加减法是否溢出，有几种方案?
答：两种方案。
（1）参加运算的两个操作数(减法时减数需连同符号位在内每位取反，末位加 1)符号相同，结果的符号位与原操作数的符号位不同，则为溢出。
（2）求和时最高位进位与次高位进位异或结果为 1 时，则为溢出。

25. 在浮点机中如何判断溢出?
答：浮点机中根据阶码来判断溢出，当阶码大于最大正阶码时，视为溢出。阶码小于最小负阶码时，按机器零处理。

26.补码一位乘法中，部分积为什么采用双符号位?
答：补码一位乘是由重复加和移位操作实现的，移位时按补码右移规则进行。以小数乘法为例，由于乘法过程中相加结果可能大于 1，即小数点前面第一位为数值，占去了符号位的位置，若采用一位符号位，则原符号位被破坏，移位时会出错。所以采用双符号位，最高位代表真正的符号，就可避免移位时会出错的现象。

27. 补码两位乘法中，部分积需采用几位符号位，为什么?
答：需采用三位符号位。
以小数乘法为例，乘法过程中相加结果可能大于 2，占去小数点前面的两个位置，所以需采用三位符号位，最高位代表真正的符号，就可以根据该位的状态进行移位，结果不会出错。

28. 在原码两位乘法形成部分积的过程中，参加运算的数是否为原码，为什么?
答：不是原码。
因为由原码两位乘的运算规则得出，符号位的运算和数值部分的运算是分开进行的，而数值部分的运算是绝对值参加运算。但由于由两位乘的运算规则得出，运算过程中可能出现减 1 倍被乘数的绝对值操作(记为减被乘数*)，计算机中减法用加法代替，即需作加[-被乘数*]补的操作，故数值运算时，参加运算的数实际是绝对值的补码而不是原码。

29. 在原码除法形成余数的过程中，参加运算的数是否为原码，为什么?
答：参加运算的数不是原码。
原码除法过程中，商符和商值的运算是分开进行的。求商值可用加减交替法，即加 y*和减 y*，在计算机内则用加$[y^*]_{补}$和加$[-y^*]_{补}$来实现，故参加运算的数不是原码而是绝对值的补码。

30. 试比较原码和补码在加减交替法除法的过程中有何相同和不同之处?
答：原码和补码在加减交替除法过程中的相同之处是形成新余数的规则相同。
不同之处有四点：
(1) 原码除法的商符由两数符号位异或运算获得，补码除法的商符在求商的过程中自然形成。
(2) 原码除法参加运算的数是绝对值的补码，补码除法参加运算的数是补码。
(3) 两种除法上商的原则不同。原码除法中，余数为正上商 1，余数为负上商 0；补码除法中，余数与除数同号上商 1，余数与除数异号上商 0。
(4) 两种除法第一步的操作不同。原码除法第一步做被除数减除数的操作；补码除法第一步要根据被除数和除数的符号决定做加法还是减法(同号做减法，异号做加法)。

31. 在浮点补码加减运算中，当尾数运算结果的符号位为 01 或 10 时，即表示运算结果溢出，这种说法是否正确，为什么?
答：这种说法不对。因为浮点数的溢出不是以尾数溢出为判断依据的(以阶码所能表示的范围为依据)。若尾数溢出，可通过右规使尾数恢复正常。()

32. 写出浮点补码规格化形式，当尾数出现什么形式时需规格化？如何规格化?
答：设浮点数尾数采用双符号位，当尾数呈现 00.1××...×或 11.0××...×时，记为补码规格化形式。
当尾数出现 01.××...×或 10.××...×时，需右规，右规时尾数右移一位，阶码加 1。
当尾数出现 00.00××...×或 11.111××...×时，需右规，右规时尾数左移一位，阶码减 1。

33. 写出十进制数 x = -41，y = +101，设机器数字长 8 位(含一位符号位)，计算[x+y]补和[x-y]补，并给出相应的 Z(零标志)、V(溢出标志)和 C(进位标志)。
答：$[x]_{补}$ = 1,1010111　　$[y]_{补}$ = 0,1100101　　$[-y]_{补}$ = 1,0011011
$[x+y]_{补}$ = 11010111 + 01100101 = 100111100 = 00111100，Z = 0，V = 0，C = 1
$[x-y]_{补}$ = 11010111 + 10011011 = 101110010 = 01110010，Z = 0，V = 1，C = 1
参加加法操作的数均为负，结果为正，溢出

34. 写出十进制数 x = 25/32，y = -21/64，设机器数字长 8 位(含一位符号位)，计算[x+y]补和[x-y]补，并给出相应的 Z(零标志)、V(溢出标志)和 C(进位标志)。
答：$[x]_{补}$ = 0.1100100　　$[y]_{补}$ = 1.1010110　　$[-y]_{补}$ = 0.0101010
$[x+y]_{补}$ = 10.0111010 = 0.0111010 ，Z = 0，V = 0，C = 1
$[x-y]_{补}$ = 1.1001110，Z = 0，V = 1，C = 0

35. 已知二进制数 x = -0.1100，y = 0.1001，按一位乘法计算 x*y，要求列出详细过程，机器数形式自定。
答：

部分积 被乘数	乘数 乘积低位
0.0000 +0.1100	1001
0.1100 右移 0.0110 +0.0000	0100
0.0110 右移 0.0011 +0.0000	0010
0.0011 右移	

0.0001 +0.1100	1001
0.1101 右移 0.0110	1100

符号位异或运算得 -
结果 1.01101100

37. 已知二进制数 x = -0.1011，y = -0.1101，用补码一位乘计算$[x*y]_{补}$。
答：[x]补 = 1.0101　　[y]补 = 1.0011
补码比较法：

部分积 被乘数	乘数 乘积低位	附加位	说明
00.0000 +00.1011	1001<u>1</u>	<u>0</u>	初值$[z_0]_{补}$ = 0
00.1011 00.0101 00.0010 +11.0101	1100<u>1</u> 11100	<u>1</u> <u>1</u>	右移一位，得$[z_1]_{补}$
11.0111 11.1011 11.1101 +00.1011	11 11110 1110 1111	<u>0</u> <u>0</u>	
00.1000	1111		

$[x*y]_{补}$ = 0.10001111

45. 原码两位乘有何特点？归纳一下共有几种运算规则。
答：原码两位乘的特点是：乘积的符号位由两原码符号位异或运算获得，数值部分是两原码绝对值相乘。设部分积为 z，被乘数的绝对值为 x*，乘数的绝对值为 y*，乘数的判断位为 $y_{n-1}y_n$，标志位为 C_j，具体规则归纳如下表所示。

乘数判断位 $y_{n-1}y_n$	标志位 C_j	操作内容
00	0	$z\rightarrow 2$，$y^*\rightarrow 2$，C_j 保持 0
01	0	$z+x^*\rightarrow 2$，$y^*\rightarrow 2$，C_j 保持 0
10	0	$z+2x^*\rightarrow 2$，$y^*\rightarrow 2$，C_j 保持 0
11	0	$z-x^*\rightarrow 2$，$y^*\rightarrow 2$，置 1 C_j
00	1	$z+x^*\rightarrow 2$，$y^*\rightarrow 2$，置 0 C_j
01	1	$z+2x^*\rightarrow 2$，$y^*\rightarrow 2$，置 0 C_j
10	1	$z-x^*\rightarrow 2$，$y^*\rightarrow 2$，C_j 保持 1
11	1	$z\rightarrow 2$，$y^*\rightarrow 2$，C_j 保持 0

46. 两个浮点数规格化相乘，是否可能需要右规？为什么？
答：不可能右规。两个规格化浮点数的尾数均在 1/2 到 1 之间，其乘积不会大于 1，不可能右规。

47. 两个浮点规格化相乘，是否可能需要左规？为什么？
答：可能左规，且左规次数为 1 位。 两个规格化浮点数的尾数均在 1/2 到 1 之间，乘积在 1/4 到 1 之间，故出现左规时，左规的次数只能为 1 次。

48. 假设阶码取 3 位，尾数取 6 位(均不包含符号位)，机器数形式自定，计算$[2^5\times(11/16)]+[2^4*(-5/8)]$，并给出真值。
答：$[2^5\times(11/16)]+[2^4\times(-5/8)] = [2^5\times(11/16)]+[2^5\times(-5/16)] = 2^5\times(3/8)$
原码：
~~0,101;0.101100 + 0,100;1.101000 = 0,101;0.101100 + 0,101;1.0101000 = 0,101;0.011000~~
0,101;0.101100 + 0,100;1.101000 = 0,101;0.101100 + 0,101;1.1011000 = 0,101;(1)0.011000

53. 假设机器数字长为 16 位(包含 1 位符号位)，若一次移位需 100ns，一次加法需 100 ns，试问原码一位乘、原码两位乘、补码一位乘和补码加减交替法各最多需多少时间。
答：原码一位乘，加 15 次，移位 15 次，共 3 微秒。
原码两位乘，加 8 次，移位 8 次(最后一次移一位)，共 1.6 微秒。
补码一位乘，加 16 次，移位 15 次，共 3.1 微秒。
补码加减交替法(采用末位恒置 1 法)，加 15 次，移位 15 次，共 3 微秒。

54. 你知道几种方法判断补码定点加减运算的溢出。
答：有三种判断补码定点溢出的方法。
(1) 采用一位符号位，若两操作数符号相同(减法时减数需每位取反，末位加 1)，结果的符号位又与原操作数符号不同，则为溢出。
(2) 采用一位符号位，加法时最高位(符号位)的进位和次高位的进位异或结果为 1 时，即为溢出。
(3) 采用双符号位，当结果的两个符号位不同时，即为溢出。

55. 如何判断原码和补码小数除法运算溢出。
答：以小数除法为例，原码除法以第一次上商的商值来判断是否溢出，若上商 1，即为溢出。
补码除法以第一次上商的商值(即商符)与两操作数符号位异或结果不同即为溢出。例如两操作数符号位异或结果为 1，即为溢出。

57. 设机器内没有“取反码”指令，如何得到一个数的反码？
答：将此数与全 1 异或，即可得到该数的反码。

58. 如何判断定点和浮点补码除法的溢出？
答：以小数除法为例，补码除法第一次上商即为商符，若商符与两操作数符号位异或结果不同，即为溢出。如两操作数符号相同，第一次上商若为 1 即为溢出；两操作数符号不同，第一次上商若为 0 即为溢出。

76. 计算机中如何判断原码、补码和反码的规格化形式？
答：浮点机中，机器数采用原码时，不论尾数的符号是 0 或 1，只需第一数值位为 1，即为规格化形式。机器数采用反码或补码时，尾数的符号位与第一数值位不同即为规格化形式。

80. 试比较串行、串并行、全并行补码定点加减法运算器的硬件组成，哪种结构运算速度最快？
答：串行、串并行和全并行补码定点加减法运算器都需有相应的寄存器和全加器，但全加器的位数不同。
串行运算器只需 1 位全加器，完成 1 位加减运算，还需 1 位触发器，用来存放每位求和时产生的进位。
串并行运算器的全加器位数取决于并行处理信息的位数，例如能并行处理 4 位信息，相应就设置 4 位全加器。同时还需设置 1 位触发器，用来存放这 4 位并行处理信息的最高位进位，该进位作为下一组 4 位并行处理信息的外来进位。
全并行运算器的全加器位数与寄存器位数相同，而且全加器本身就包含了进位电路，无需再设置触发器存放进位。
可见全并行运算器结构最复杂，但速度最快。

82. 什么是进位链？什么是先行进位？你知道有几种先行进位？简要说明。
答：进位链就是传递进位的逻辑电路。高位进位和低位进位同时产生的进位叫先行进位。先行进位有两种，一种是单重分组跳跃进位，即将 n 位全加器分成若干小组，小组内进位同时产生，小组间采用串行进位，简称组内并行、组间串行。另一种是多重分组跳跃进位，即将 n 位全加器分成几个大组，每个大组又包含若干小组，大组内每个小组的最高位进位是同时产生的，小组内的其他各位进位也是同时产生的，而大组之间采用串行进位，简称组(小组)内并行，组(小组)间并行。