简单回答下列问题。

（1）为什么计算机内部采用二进制表示信息？既然计算机内部所有信息都用二进制表示，

为什么还要用到十六进制或八进制数？

（2）常用的定点数编码方式有哪几种？ 通常它们各自用来表示什么？

（3）为什么计算机中大多用补码表示带符号整数？

（4）在浮点数的基和位数一定的情况下，浮点数的表数范围和表数精度分别由什么决定？

两者如何相互制约？

（5）为什么要对浮点数进行规格化？有哪两种规格化操作？

（6）为什么有些计算机中除了用二进制外还用BCD 码来表示数值数据？

（7）为什么计算机处理汉字时会涉及到不同的编码（如，输入码、内码、字模码）？说明

这些编码中哪些是用二进制编码，哪些不是用二进制编码，为什么？

假定一台32 位字长的机器中带符号整数用补码表示，寄存器R1 和R2 的内容分别为：R1：

0000108BH，R2：8080108BH。不同指令对寄存器进行不同的操作，因而，不同指令执行

时寄存器内容的真值不同。

请问：执行下列指令时，寄存器R1 和R2 的内容所对应的真值分别为多少？

① 无符号数加法指令

② 带符号整数乘法指令

③ 单精度浮点减法指令（IEEE754）

参考答案：

**R1**：**0000108BH R2**：**8080108BH**

① **+108BH +8080108BH**

② **+108B -7F7FEF75H**

③ **0000 0000 0000 0000 0001 0000 1000 1011 1000 0000 1000 0000 0001 0000 1000 1011**

**+0.002116H** x**2-126 -1.002116H** x**2-126**

以下是一个C 语言程序，用来计算一个数组a 中每个元素的和。当参数len 为0 时，返回值

sum 应该是0，但是在机器上执行时，却发生了存储器访问异常。请问是什么原因造成的，

并说明程序应该如何修改。

1 floae sum\_elements(float a[], unsigned len)

2 {

3 int i;

4 float result = 0;

5

6 for (i = 0; i <= len-1; i++)

7 result += a[i];

8 return result;

9 }

参考答案：

参数 **len** 的类型是**unsigned**，所以，当**len=0** 时，执行**len-1** 的结果为**11…1**，是最大可表示

的无符号数，因而，任何无符号数都比它小，使得循环体被不断执行，引起数组元素的访问

越界，发生存储器访问异常。

只要将 **len** 声明为**int** 型，或循环的测试条件改为**i<len.**

14．假定一个变量的值为-2147483647，分别用32 位2-补码和IEEE754 单精度浮点格式表示该变量（结果用十六进制表示），并说明哪种表示的值完全精确，哪种表示的是近似值。

参考答案：

**-2147483647=-111 1111 1111 1111 1111 1111 1111 1111B**

**=-1.11 1111 1111 1111 1111 1111 1111 1111** x **230**

**32** 位**2-**补码形式为：**1000 0000 0000 0000 0000 0000 0000 0001** （**80000001H**）

**IEEE754** 单精度格式为：**1 10011101 1111 1111 1111 1111 1111 111** （**CEFFFFFFH**）

**32** 位**2-**补码形式能表示精确的值，而浮点数表示的是近似值，低位被截断

18．假定在一个程序中定义了变量x、y 和i，其中，x 和y 是float 型变量（用IEEE754 单精度

浮点数表示），i 是16 位short 型变量（用补码表示）。程序执行到某一时刻，x=-0.125、y=7.5、

i=100，它们都被写到了主存（按字节编址），其地址分别是100，108 和112。请分别画出在

大端机和小端机上变量x、y 和i 在内存的存放情况。

参考答案：

**-0.125=-0.001B = -1.0** x **2-3**

**x** 在机器内部的机器数为：**1 01111100 00…0 (BE000000H)**

**7.5=+111.1B= +1.111** x **22**

**y** 在机器内部的机器数为：**0 10000001 11100…0 (40F00000H)**

**100=64+32+4=1100100B**

**i** 在机器内部表示的机器数为：**0000 0000 0110 0100**（**0064H**）

大端机 小端机

地址 内容 内容

**100 BEH 00H**

**101 00H 00H**

**102 00H 00H**

**103 00H BEH**

**108 40H 00H**

**109 F0H 00H**

**110 00H F0H**

**111 00H 40H**

**112 00H 64H**

**113 64H 00H**

19．写出16 位数据字的SEC 码。假定数据字为0101 0001 0100 0110，说明SEC 码如何正确检

测数据位5 的错误。

参考答案：

码字排列如下：

**M16 M15 M14 M13 M12 P5 M11 M10 M9 M8 M7 M6 M5 P4 M4 M3 M2 P3 M1 P2 P1**

简单回答下列问题。（参考答案略）

（1）为何在高级语言和机器语言中都要提供“按位运算”？为何高级语言需要提供逻辑运算？按位运算和逻

辑运算的差别是什么？

（2）如何进行逻辑移位和算术移位？它们各用于哪种类型的数据？

（3）移位运算和乘除运算具有什么关系？

（4）高级语言中的运算和机器语言（即指令）中的运算是什么关系？假定某一个高级语言源程序P 中有乘、

除运算，但机器M中不提供乘、除运算指令，则程序P 能否在机器M上运行？为什么？

（5）为什么用一个ALU 和移位器就能实现定点数和浮点数的所有加、减、乘、除运算?

（6）影响加/减运算速度的关键问题是什么？可采取什么改进措施？

5．以下是两段C 语言代码，函数arith( )是直接用C 语言写的，而optarith( )是对arith( )函数以某M和N 编译生成的机器代码反编译生成的。根据optarith( )，可以推断函数arith( ) 中M和N 的值各是多少？

#define M

#define N

int arith (int x, int y)

{

int result = 0 ;

result = x\*M + y/N;

return result;

}

int optarith ( int x, int y)

{

int t = x;

x << = 4;

x - = t;

if ( y < 0 ) y += 3;

y>>2;

return x+y;

}

参考答案：

对反编译结果进行分析，可知：对于**x**，指令机器代码中有一条“**x** 左移**4** 位”指令，即：**x=16x**，然后有一条

“减法”指令，即**x=16x-x=15**，所以，根据源程序，知**M=15**；对于**y** ，有一条“**y** 右移**2** 位”指令，即**y=y/4**，

根据源程序，知**N=4**。**(**当**y<0** 时**, (y+3)/4=y/4**，若不调整，则“**-1>>2=-1** 而本来**-1/4=0”**，故使**-1+3=2**，**2/4=0)**

7．利用SN74181 和SN74182 器件设计一个16 位并行进位补码加/减运算器，画出运算器的逻辑框图，并给出零

标志、进位标志、溢出标志、符号标志的生成电路。

参考答案：（略）

• 62 •

9．已知二进制数x = 0.1010，y = -0.1101。请按如下要求计算，并把结果还原成真值。

（1） 求[x+y]补，[x-y]补。

（2） 用原码一位乘法计算[x∗y]原。

（3） 用布斯乘法计算[x∗y]补。

（4） 用不恢复余数法计算[xy]原的商和余数。

（5） 用不恢复余数法计算[xy]补的商和余数。

参考答案：（略）

13．假设浮点数格式为：阶码是4 位移码，尾数是6 位补码（采用双符号位），用浮点运算规则分别计算在不采

用任何附加位和采用2 位附加位（保护位、舍入位）两种情况下的值。（假定对阶和右规时采用就近舍入到

偶数方式）

（1）(15/16)\*27 +(2/16)\*25 （2）(15/16)\*27-(2/16)\*25

（3）(15/16)\*25 +(2/16)\*27 （4）(15/16)\*25-(2/16)\*27

参考答案： 采用 **2** 位附加位的情况：

**x= (15/16)\*27** 的浮点数表示为：**1111, 00.1111; y=(2/16)\*25** 的浮点数表示为：**1101, 00.0010**

对阶：**1111+0011=0010 (+2)**；对**y** 进行：**1111,00.000010**

尾数相加：**00.111100+00.000010=00.111110**

舍入：**1111,01.0000**

右规：右规前阶码已经为**1111**，所以结果“溢出”。

（其余略）

14．采用IEEE754 单精度浮点数格式计算：0.75+（-65.25）

参考答案：（略）

15．采用十进制数（NBCD 码）加法运算的方法，计算下列各式。并讨论在十进制BCD 码加法运算中如何判断

溢出。

（1） 234+567 （2） 548+729

参考答案：（略）

先确定位数，最高位有进位，则“溢出”