第三章作业

1. 假定某数采用 IEEE 754 单精度浮点数格式表示为 C820 0000H，则该数的值是（ C ）。

   A. $(-1.01)_{10}\times 2^{17}$　　B. $(-1.01)_{10}\times 2^{144}$

   C. $(-1.25)_{10}\times 2^{17}$　　D. $(-1.25)_{10}\times 2^{144}$

2. 假定变量 i、f 的数据类型分别是 int、float。已知 i=12345，f=1.2345e3，则在一个 32 位机器中执行下列表达式时，结果为“假”的是（ C ）。

   A. i= =(int)(float)i　　B. i= =(int)(double)i

   C. f= =(float)(int)f　　D. f= =(float)(double)f

3. 某计算机字长为 8 位，其 CPU 中有一个 8 位加法器。已知带符号整数 x=–69，y=–38，现要在该加法器中完成 x+y 的运算，则该加法器的两个输入端信息和输入的低位进位信息分别为(A)。

   A. 1011 1011、1101 1010、0　　B. 1011 1011、1101 1010、1

   C. 1011 1011、0010 0101、0　　D. 1011 1011、0010 0101、1

4. 某 8 位计算机中，假定 x 和 y 是两个带符号整数变量，用补码表示，x=63，y= –31，则 x+y 的机器数及其相应的溢出标志 OF 分别是(B)。

   A. 1FH、0　　B. 20H、0　　C. 1FH、1　　D. 20H、1

5. 若两个 float 型变量（用 IEEE 754 单精度浮点格式表示）x 和 y 的机器数分别表示为 x=40E8 0000H，y=C204 0000H，则在计算 x+y 时，第一步对阶操作的结果$[\Delta E]_{补}$为(D)。

   A. 0000 0111　　B. 0000 0011　　C. 1111 1011　　D. 1111 1101

6. 某字长为 8 位的计算机中，x 和 y 为无符号整数，已知 x=68，y=80，x 和 y 分别存放在寄存器 A 和 B 中。请回答下列问题（要求最终用十六进制表示二进制序列）。

   （1）寄存器 A 和 B 中的内容分别是什么？

   （2）若 x 和 y 相加后的结果存放在寄存器 C 中，则寄存器 C 中的内容是什么？运算结果是否正确？加法器最高位的进位 Cout 是什么？零标志 ZF 和进位标志 CF 各是什么？

   （3）若 x 和 y 相减后的结果存放在寄存器 D 中，则寄存器 D 中的内容是什么？运算结果是否正确？加法器最高位的进位 Cout 是什么？零标志 ZF 和借位标志 CF 各是什么？

（4）无符号整数加/减运算时，加法器最高位进位 Cout 的含义是什么？它与进/借位标志 CF 的关系是什么？

（5）无符号整数一般用来表示什么信息？为什么通常不对无符号整数的运算结果判断溢出？

答：（1）x = 68 = 0100 0100 B = 44H；y = 80 = 0101 0000 B = 50H。所以，寄存器 A 和 B 中的内容分别是 44H 和 50H。

（2）x + y = 0100 0100 + 0101 0000 = (0) 1001 0100 = 94H，所以，寄存器 C 中的内容为 94H，对应的真值为 148，运算结果正确。加法器最高位的进位 Cout 为 0。因为结果不为 0，所以 ZF=0；进位标志 CF=Cout=0。

（3）$x–y = x + [–y]_{补}$ = 0100 0100 + 1011 0000 = (0) 1111 0100 = F4H，所以，寄存器 D 中的内容为 F4H，对应的真值为 244，运算结果不正确，这是因为相减结果为负数造成的。加法器最高位的进位 Cout 为 0。因为结果不为 0，所以 ZF=0；借位标志为 CF=Cout⊕1=1。

（4）在加法器中进行无符号整数加法运算时，若加法器最高位进位 Cout=1，则表示实际结果大于最大可表示数 255；在加法器中进行无符号整数减法运算时，若加法器最高位进位 Cout=1，则表示被减数大于减数，反之被减数小于减数。因此，在无符号数相加时，CF 就等于 Cout，表示进位；在无符号数相减时，通常将最高进位 Cout 取反来作为借位标志 CF，也即，无符号整数相减时，CF=$\overline{\text{Cout}}$，CF=1 表示有借位。

（5）无符号整数一般用来表示地址（指针）信息，当两个地址相加结果大于最大地址而取低位地址时，相当于取模，也即采用地址循环运算。因此通常不需要判断其运算结果是否溢出，即不考虑溢出标志 OF。

7. 考虑以下 C 语言程序代码：

```
int     func1 (unsigned word)
{
          return    (int) (( word <<24) >> 24);
}
int     func2 (unsigned word)
{
          return    ( (int) word <<24 ) >> 24;
}
```

假设在一个 32 位机器上执行这些函数，sizeof (int)=4，。说明函数 func1 和

func2 的功能，并填写下表。

W		func1(w)		func2(w)	
机器数	值	机器数	值	机器数	值
	127				
	128				
	255				
	256				

答：函数 func1 的功能是把无符号数高 24 位清零（左移 24 位再逻辑右移 24 位），结果一定是正的带符号整数；而函数 func2 的功能是把无符号数的高 24 位都变成和第 25 位一样，因为左移 24 位后左边第一位变为原来的第 25 位，然后进行算术右移，高位补符号，即高 24 位都变成和原来第 25 位相同。

W		func1(w)		func2(w)	
机器数	值	机器数	值	机器数	值
0000007FH	127	0000007FH	+127	0000007FH	+127
00000080H	128	00000080H	+128	**FFFFFF80H**	**–128**
000000FFH	255	000000FFH	+255	**FFFFFFFFH**	**–1**
00000100H	256	**00000000H**	**0**	**00000000H**	**0**

8. 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 IEEE 754 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= –130、y=7.25、i=130，它们都被写到了主存（按字节编址），其地址分别是&x，&y 和&i。请分别给出在大端机器和小端机器上变量 x、y 和 i 在内存的存放位置。

答：x = –130 = –100 00010B = –1.00 0001B×2^7，阶码 e=127+7=128+6=1000 0110，所以，用 IEEE 754 单精度浮点数表示为：1 100 0011 0 000 0010 0000 0000 0000 0000 = C302 0000H。

y=7.25= 111.01B = +1.1101B×2^2，阶码 e=127+2=128+1=1000 0001，所以，用 IEEE 754 单精度浮点数表示为：0 100 0000 1 110 1000 0000 0000 0000 0000 = 40E8 0000H。

i = 130 = 1000 0010B，用 16 位补码表示为 0082H。

上述三个数据在大端机器和小端机器上的存放位置如下所示。

地址	大端机器	小端机器
&x	C3H	00H
&x + 1	02H	00H
&x + 2	00H	02H
&x + 3	00H	C3H
&y	40H	00H
&y + 1	E8H	00H
&y + 2	00H	E8H
&y + 3	00H	40H
&i	00H	82H
&i + 1	82H	00H

9. 以下是函数 fpower2 的 C 语言源程序，它用于计算 2^x 的浮点数表示，其中调用了函数 u2f，u2f 用于将一个无符号整数表示的 0/1 序列作为 float 类型返回。请填写 fpower2 函数中的空白部分，以使其能正确计算结果。

```
1   float fpower2(int x)
2   {
3       unsigned exp, frac, u;
4
5       if (x<_______) {    /* 值太小，返回 0.0 */
6           exp = _________;
7           frac = _________;
8       } else if (x<_______) {    /* 返回非规格化结果 */
9           exp = _________;
10          frac = _________;
11      } else if (x<_______) {    /* 返回规格化结果 */
12          exp = _________;
13          frac = _________;
14      } else {        /* 值太大，返回+∞ */
15          exp = _________;
16          frac = _________;
17      }
18      u = exp << 23 | frac;
19      return u2f(u);
20  }
```

答：

```
1   float fpower2(int x)
2   {
3       unsigned exp, frac, u;
4
5       if (x<  -149 ) {    /* 值太小，返回 0.0 */
6           exp =  0  ;
7           frac =  0  ;
8       } else if (x<  -126  ) {        /* 返回非规格化结果 */
9           exp =  0  ;
10          frac =  0x400000>>(-x-127)  ;
11      } else if (x<  128  ) {        /* 返回规格化结果 */
12          exp =  x+127  ;
13          frac =  0  ;
14      } else {        /* 值太大，返回+∞ */
15          exp =  255  ;
16          frac =  0  ;
17      }
18      u = exp << 23 | frac;
19      return u2f(u);
20  }
```

10. 采用 IEEE 754 单精度浮点数格式计算下列表达式的值。

（1）0.75+(– 65.25）　　（2）0.75–(– 65.25）

答：

x=0.75=0.11B=$(1.10...0)_2\times2^{-1}$，$[x]_{浮}$=0 01111110 10...0

y= –65.25= –1000001.01B = $(–1.00000101...0)_2\times2^6$，$[y]_{浮}$=1 10000101 000001010...0

E_x=0111 1110，M_x=1.100 0000 0000 0000 0000 0000，

E_y=1000 0101，M_y=1.000 0010 1000 0000 0000 0000，

（1）0.75+ (– 65.25）

① 对阶。

$[\Delta E]_{补}=E_x+[–E_y]_{补}$=0111 1110 + 0111 1011=1111 1001 (mod 2^8)，ΔE=–7，故需对 x 进行对阶，结果为 $E_x=E_y$=1000 0101，M_x=0.000 0001 1000 0000 0000 0000,

② 尾数相加。

因为 x 与 y 符号相反，故做减法，即 Mx+$[-My]_{补}$

$[-My]_{补}$=0.111 1101 1000 0000 0000 0000

0.000 0001 1000 0000 0000 0000

\+ 0.111 1101 1000 0000 0000 0000

0.111 1111 0000 0000 0000 0000

最高数值位没有产生进位，表明结果为负，得到的是数值的补码形式，因此，需要对结果求补，即结果为 1.000 0001 0000 0000 0000 0000，结果符号与 x 的符号相反，即负号。

③ 规格化。

根据所得尾数的形式，数值部分最高位为 1，所以不需要进行规格化。

④ 溢出判断。

在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。

最终结果为 E_b=1000 0101，M_b=1.000 0001 0000 0000 0000 0000，符号为负，即：$(-1.0000001)_2\times2^6=-64.5$。

（2） 0.75–(– 65.25）

① 对阶。

同上述（1）中对阶过程一样。

② 尾数相减。

因为 x 与 y 符号相反，故做加法，即 Mx+My，符号与 x 的符号相同，为正号。

0.000 0001 1000 0000 0000 0000

\+ 1.000 0010 1000 0000 0000 0000

1.000 0100 0000 0000 0000 0000

结果没有溢出，符号为正号。

③ 规格化。

根据所得尾数的形式，数值部分最高位为 1，不需要进行规格化。

④ 溢出判断。

在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。

最后结果为 E_b=1000 0101，M_b = 1.00001000…0，符号为正，即：$(+1.00001)_2\times2^6=+66$。