第 3 章　习　题 答 案

题目及要求 2（4）, 5, 7（1）、7（2）、7（4）、 11（2）、12（1）

2（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？

参考答案：（略）

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( )函数以某个确定的 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：

可以看出 x*M 和“int t = x; x << = 4; x-=t;”三句对应，这些语句实现了 x 乘 15 的功能（左移 4 位相当于乘以 16，然后再减 1），因此，M 等于 15;

y/N 与“if (y < 0) y += 3; y>>2;”两句对应，第二句“y 右移 2 位”实现了 y 除以 4 的功能，因此 N 是 4。而第一句“if (y < 0) y += 3;”主要用于对 y= –1 时进行调整，若不调整，则–1>>2= –1 而 –1/4=0，两者不等；调整后 –1+3=2，2>>2=0，两者相等。

思考：能否把　　if (y < 0)　y += 3; 改成 if (y < 0)　y += 2; ？

不能！因为 y = - 4 时不正确。

7．已知 x = 10，y = – 6，采用 6 位机器数表示。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）用不恢复余数法计算$[x/y]_{原}$的商和余数。

参考答案：

$[10]_{补}$ = 001010　　$[–6]_{补}$ = 111010　　$[6]_{补}$ = 000110　　$[10]_{原}$ = 001010　　$[–6]_{原}$ = 100110

（1）　$[10+(– 6)]_{补}$= $[10]_{补}$+$[– 6]_{补}$= 001010+111010 = 000100 (+4)

$[10–(–6)]_{补}$= $[10]_{补}$+$[– (–6)]_{补}$ = 001010+000110 = 010000 (+16)

（2） 先采用无符号数乘法计算 001010× 000110 的乘积，原码一位乘法过程（前面两个 0 省略）如下：

C(进位)	P(乘积寄存器)	Y(乘数寄存器)	说明 (被乘数 001010, 前面两个 0 省略)
0	0 0 0 0	0 1 1 0	$P_0=0$
	+ 0 0 0 0	$y_1\ y_2\ y_3\ y_4$	$y_4=0$，+0
0	0 0 0 0		C, P 和 Y 同时右移一位
0	0 0 0 0	0 0 1 1	得 P_1
	+ 1 0 1 0		$y_3=1$，+X
0	1 0 1 0		C, P 和 Y 同时右移一位
0	0 1 0 1	0 0 0 1	得 P_2
	+ 1 0 1 0		$y_2=1$，+X
0	1 1 1 1	0 0 0 0	C, P 和 Y 同时右移一位
0	0 1 1 1	1 0 0 0	得 P_3
	+ 0 0 0 0		$y_1=0$，+0
0	0 1 1 1		C, P 和 Y 同时右移一位
0	0 0 1 1	1 1 0 0	得 P_4

若两个 6 位数相乘的话，则还要右移两次，得 000000 111100

符号位为：$0\oplus1=1$，因此，$[X\times Y]_{原}$ = 1000 0011 1100

即 $X\times Y=-11\ 1100B=-60$ [溢出,无法用 6 位原码表示]

（4） 因为除法计算是 2n 位数除 n 位数，所以$[6]_{原}$=0110，$[10]_{原}$=0000 1010，$[-6]_{补}$=1010，商的符号位：$0\oplus1=1$，运算过程（前面两个 0 省略）如下：

余数寄存器 R	余数/商寄存器 Q	说 明
0 0 0 0	1 0 1 0 □	开始 $R_0=X$
+ 1 0 1 0		$R_1=X-Y$
1 0 1 0	1 0 1 0 0	$R_1<0$，则 $q_4=0$，没有溢出
0 1 0 1	0 1 0 0 □	$2R_1$（R 和 Q 同时左移，空出一位商）
+ 0 1 1 0		$R_2=2R_1+Y$
1 0 1 1	0 1 0 0 0	$R_2<0$，则 $q_3=0$
0 1 1 0	1 0 0 0 □	$2R_2$ （R 和 Q 同时左移，空出一位商）
+ 0 1 1 0		$R_3=2R_2+Y$
1 1 0 0	1 0 0 0 0	$R_3<0$，则 $q_2=0$
1 0 0 1	0 0 0 0 □	$2R_3$ （R 和 Q 同时左移，空出一位商）
+ 0 1 1 0		$R_3=2R_2+Y$
1 1 1 1	0 0 0 0 0	$R_4<0$，则 $q_1=0$
1 1 1 0	0 0 0 0 □	$2R_4$ （R 和 Q 同时左移，空出一位商）
+ 0 1 1 0		$R_5=2R_4+Y$
0 1 0 0	0 0 0 0 1	$R_5>0$，则 $q_0=1$

商的数值部分为：00001。所以，$[X/Y]_{原}$=00001 (最高位为符号位)，余数为 0100。

11.2. 假设浮点数格式为:阶码是 4 位移码,偏置常数为 8, 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.)

答案请见下一页

(1) $(15/16)\times2^7+(2/16)\times2^5$　　(2) $(15/16)\times2^7-(2/16)\times2^5$

(3) $(15/16)\times2^5+(2/16)\times2^7$　　(4) $(15/16)\times2^5-(2/16)\times2^7$

【分析解答】

3+8

将上述各式中的数据用相应的变量 A、B、C、D 代替。

$A=(15/16)\times2^7=0.1111\text{B}\times2^7$，$[A]_{浮}=00.1111，1111$。

$B=(2/16)\times2^5=0.0010\text{B}\times2^5=0.1000\text{B}\times2^3$，$[B]_{浮}=00.1000，\underline{1011}$。

$C=(15/16)\times2^5=0.1111\text{B}\times2^5$，$[C]_{浮}=00.1111，1101$。

$D=(2/16)\times2^7=0.0010\text{B}\times2^7=0.1000\text{B}\times2^5$，$[D]_{浮}=00.1000，1101$。

不采用任何附加位时的计算结果如下。

(1) 计算 $A+B$：$[\Delta E]_{补}=[E_A]_{移}+[-[E_B]_{移}]_{补}=1111+0101=0100(\text{mod } 2^4)$，因此 $\Delta E=4$，故需对 B 进行对阶，因为采用就近舍入到偶数方式，所以，B 的尾数右移 4 位后直接舍去 1000，又因为 1000 是中间值，因此尾数取偶数 00.0000，故对阶后结果为 $[B]_{浮}=00.0000，1111$。由于 B 的尾数为 0，因此，$[A+B]_{浮}=[A]_{浮}=00.1111，1111$。故 $A+B=A=(15/16)\times2^7$。

(2) 计算 $A-B$：对阶结果与(1)相同，故 $[A-B]_{浮}=[A]_{浮}=00.1111，1111$。故 $A-B=A=(15/16)\times2^7$。

采用两位附加位时的计算结果如下。

(1) 计算 $A+B$：$[\Delta E]_{补}=[E_A]_{移}+[-[E_B]_{移}]_{补}=1111+0101=0100(\text{mod } 2^4)$，因此 $\Delta E=4$，故需对 B 进行对阶，对阶后结果为 $[B]_{浮}=00.0000\ 10，1111$。尾数相加结果为：$[M_A]_{补}+[M_B]_{补}=00.1111\ 00+00.0000\ 10=00.1111\ 10$，因此，$[A+B]_{浮}=00.1111\ 10，1111$。最后对尾数附加位 10 进行舍入，因为舍入的是中间值，所以尾数结果强迫为偶数，即尾数末位加 1，得尾数为 01.0000，因此，尾数需右规为 00.1000，同时，阶码 1111 加 1，产生阶码上溢，因而导致结果溢出。因此，$A+B$ 的结果溢出。

(2) 计算 $A-B$：对阶结果与(1)相同，尾数相减结果为：$[M_A]_{补}+[-M_B]_{补}=00.1111\ 00+11.1111\ 10=00.1110\ 10$，因此，$[A-B]_{浮}=00.1110\ 10，1111$。最后对尾数附加位 10 进行舍入，因为舍入的是中间值，所以尾数结果强迫为偶数，得尾数为 00.1110，因此，$[A-B]_{浮}=00.1110，1111$。故 $A-B=(14/16)\times2^7$。

12．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）

参考答案：

$x = 0.75 = 0.110...0\text{B} = (1.10...0)_2 \times 2^{-1}$

$y = – 65.25 = – 1000001.01000...0\text{B} = (–1.00000101...0)_2 \times 2^6$

用 IEEE 754 标准单精度格式表示为：

$[x]_{浮}$ = 0　01111110　10...0　　$[y]_{浮}$ = 1　10000101　000001010...0

所以，E_x = 01111110，M_x = 0 (1). 1...0 ，E_y = 10000101，M_y = 1(1).000001010...0

尾数 M_x 和 M_y 中小数点前面有两位，第一位为数符，第二位加了括号，是隐藏位“1”。

以下是计算机中进行浮点数加减运算的过程（假定保留 2 位附加位：保护位和舍入位）

（1）0.75+ (– 65.25）

① 对阶：　$[\Delta E]_{补} = [E_x]_{移} + [–[E_y]_{移}]_{补}$　(mod 2^n) = 0111 1110 + 0111 1011 = 1111 1001

$\Delta E = -7$，根据对阶规则可知需要对 x 进行对阶，结果为：$E_x = E_y = 10000101$，$M_x = 00.000000110...000$

x 的尾数 M_x 右移 7 位，符号不变，数值高位补 0，隐藏位右移到小数点后面，最后移出的 2 位保留

② 尾数相加：$M_b = M_x + M_y = 00.\ 000000110...000 + 11.\ 000001010\ ...000$（注意小数点在隐藏位后）

根据原码加/减法运算规则，得：$00.000000110...000 + 11.000001010...000 = 11.000000100…000$

上式尾数中最左边第一位是符号位，其余都是数值部分，尾数后面两位是附加位。

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 舍入：把结果的尾数 M_b 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：$M_b = 11.000000100…0$

⑤ 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。因此，阶码 $E_b = 10000101$。

最后结果为 $E_b = 10000101$，$M_b = 1(1).00000010…0$，即：-64.5。

（2） $0.75-(-65.25)$

① 对阶： $[\Delta E]_{补} = [E_x]_{移} + [-[E_y]_{移}]_{补} \pmod{2^n} = 0111\ 1110 + 0111\ 1011 = 1111\ 1001$

$\Delta E = -7$，根据对阶规则可知需要对 x 进行对阶，结果为：$E_x = E_y = 10000110$，$M_x = 00.000000110...000$

x 的尾数 M_x 右移一位，符号不变，数值高位补 0，隐藏位右移到小数点后面，最后移出的位保留

② 尾数相加：$M_b = M_x - M_y = 00.\ 000000110...000 - 11.\ 000001010...000$（注意小数点在隐藏位后）

根据原码加/减法运算规则，得：$00.000000110...000 - 11.000001010...000=01.00001000…000$

上式尾数中最左边第一位是符号位，其余都是数值部分，尾数后面两位是附加位（加粗）。

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 舍入：把结果的尾数 M_b 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：$M_b = 01.00001000…0$

⑤ 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。因此，阶码 $E_b = 10000101$。

最后结果为 $E_b = 10000101$，$M_b = 0(1).00001000…0$，即：$+66$。