2023 春季学期-计算机组成原理-作业一

(要求：独立完成，3 月 31 日上课时上交)

1. (10 points)（课本 P88 页习题 3）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，该机器使用二进制补码表示带符号整数。无符号数采用逻辑移位，带符号整数采用算术移位。请填写下表，并说明函数 func1 和 func2 的功能。

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

func1的功能：先左移 24 位，再逻辑右移 24 位，相当于把高 24 位清零，将结果转换为有符号数，其一定为正数。

func2的功能：先左移 24 位，转换成有符号数，再算数右移 24 位，相当于高位补符号（原数从左开始第 25 位）。

2. (10 points)（课本 P89 页习题 6）设 $A_4 \sim A_1$ 和 $B_4 \sim B_1$ 分别是 4 位加法器的两组输入，C_0 为低位来的进位。当加法器分别采用串行进位和先行进位时，写出 4 个进位 C_4、C_3、C_2 和 C_1 的逻辑表达式。

串行进位加法：

$$\begin{aligned} C_1 &= A_1B_1 + (A_1 + B_1)C_0 \\ C_2 &= A_2B_2 + (A_2 + B_2)C_1 \\ C_3 &= A_3B_3 + (A_3 + B_3)C_2 \\ C_4 &= A_4B_4 + (A_4 + B_4)C_3 \end{aligned} \tag{1}$$

先行进位加法：

假设 $G_i = A_iB_i$，$P_i = A_i + B_i$

$$
\begin{aligned}
C_1 &= G_1 + P_1C_0 \\
C_2 &= G_2 + P_2C_1 = G_2 + P_2G_1 + P_2P_1C_0 \\
C_3 &= G_3 + P_3C_2 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_0 \\
C_4 &= G_4 + P_4C_3 = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_0
\end{aligned} \tag{2}
$$

3. (10 points)（课本 P89 页习题 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 $[x \times y]_{原}$。

（3）用基 4 布斯算法计算 $[x \times y]_{补}$。

（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。

$[x]_{原}$=$001010B$，$[x]_{补}$=$001010B$，$[-x]_{补}$=$110110B$。

$[y]_{原}$=$100110B$，$[y]_{补}$=$111010B$，$[-y]_{补}$=$000110B$。

（1）$[x+y]_{补}$= $[x]_{补}$+$[y]_{补}$= $001010 + 111010 = 000100B = 4$。$[x-y]_{补}$= $[x]_{补}$+$[-y]_{补}$= $001010 + 000110 = 010000B = 16$。

（2）$[x]_{原}$=$001010B$，$[y]_{原}$=$100110B$。将符号和数值部分分开处理。乘积的符号为 $0 \oplus 1 = 1$。数值部分采用无符号乘法计算 01010×00110 的乘积。原码一位乘法过程描述如下：初始部分积为 0，在乘积寄存器前增加一个进位位。每次循环首先根据乘数寄存器中最低位决定 $+X$ 还是 $+0$，然后将得到的新进位、新部分积和乘数寄存器中的部分乘数一起逻辑右移一位。共循环 5 次，最终得到一个 10 位无符号数表示的乘积 $00001\ 11100B$。所以，$[x \times y]_{原}$= $1\ 00001\ 11100B = -60$。若结果取 6 位原码，则因为高 5 位 00001 是一个非 0 数，所以结果溢出。

（3）$[x]_{补}$=$001010B$，$[-x]_{补}$=$110110B$，$[y]_{补}$=$111010B$。采用基 4 布斯算法时，符号和数值部分一起参加计算，最初在乘数后面添 0，初始部分积为 0，并在部分积前加一位符号位 0。每次循环先根据乘数寄存器中最低 3 位决定执行 $+X$、$+2X$、$-X$、$-2X$ 还是 $+0$ 操作，然后将新得到的部分积和乘数寄存器中的部分乘数一起算术右移两位。$-X$ 和 $-2X$ 分别采用 $+[-x]_{补}$ 和 $+2[-x]_{补}$ 的方式进行。共循环 3 次，最终得到一个 12 位补码表示的乘积 $1111\ 1100\ 0100B$。所以 $[x \times y]_{补}$= $1111\ 1100\ 0100B = -60$。

（4）$[x]_{原}$=$001010B$，$[y]_{原}$=$100110B$。将符号和数值部分分开处理。商的符号为 $0 \oplus 1 = 1$。数值部分采用无符号除法算法计算 01010×00110 的商和余数。无符号数不恢复余数除法过程描述如下：初始中间余数为 0 00000 01010 0，其中，最高位添加的为符号位，用于判断余数是否大于等于 0；最后一位添加的 0 为第一次上的商，该商位只是用于判断结果是否溢出，不包含在最终的商中，因为被除数和除数位数相同，结果肯定不溢出，所以该位的商可以直接上 0，并进行一次 $-Y$ 操作得到第一次中间余数，然后进入循环。每次循环首先将中间余数和商一起左移一位，然后根据上一次

上的商决定执行 $+Y$ 还是 $-Y$ 操作，以得到新的中间余数，最后根据中间余数的符号确定上商为 0 还是 1。$-Y$ 采用 $+[-y]_{补}$ 的方式进行。整个循环内的执行要点是“正、1、减；负、0、加”，共循环 5 次。最终得到一个 6 位无符号数表示的商 0 00001 和余数 00100，其中商的最高位 0 去掉后添上符号位得到最终的商的原码表示 1 00001 和余数的原码表示 00100。因此，$[x/y]_{原}$ 商为 -1，余数为 4。

（5）$[x]_{补}$=001010B，$[y]_{补}$=111010B，$[-y]_{补}$=000110B。符号和数值部分一起参加计算。初始对被除数进行符号扩展，并进行第一位商的计算。因为 X 与 Y 异号，因此做加法 $X+Y$，若中间余数与 Y 同号，则第一位商为 1，否则为 0。这里，第一位商也仅用于判断结果是否溢出。然后进入循环。每次循环先将中间余数和商一起左移一位，然后根据上一次上的商决定执行 $+Y$ 还是 $-Y$ 操作，以得到新的中间余数，最后根据中间余数的符号与除数符号是否一致，确定上商为 0（符号不一致）还是 1（符号一致）。$-Y$ 采用 $+[-y]_{补}$ 的方式进行。整个循环内的执行要点是“同、1、减；异、0、加”，共循环 6 次。最后进行商和余数的修正。对于商的修正，先将第一次的商移除，在最低位补上最后的商位。如果被除数与除数同号，则此结果为最终的商，否则末位加 1 修正。对于余数，若其符号与被除数相同，则无需修正，否则，当被除数与除数同号时，余数加除数进行修正，当被除数与除数异号时，余数减除数进行修正。最终得到一个 6 位补码表示的商 111111 和余数 000100。因此，$[x/y]_{补}$ 商为 -1，余数为 4。

4. (10 points)（课本 P89 页习题 8）若一次加法需要 $1ns$，一次移位需要 $0.5ns$。请分别给出用一位乘法、两位乘法、基于 CRA 的阵列乘法、基于 CSA 的阵列乘法 4 种方式计算两个 8 位无符号二进制数乘积时所需的时间。

（1）对于 8 位无符号数，一位乘法需要进行 8 次右移，8 次加法。所需时间总计 $12ns$。

（2）对于 8 位无符号数，两位乘法需要进行 4 次右移，4 次加法。所需时间总计 $6ns$。

（3）对于 CRA 阵列乘法，每一级的部分积不仅依赖于上一级部分积，还依赖于上一级最终的进位，而每一级进位又是串行的，所以最长的路径总共经过了 $8+2\times(8-1)=22$ 个单元节点。假设经过每个单元节点所花时间平均为 $0.5ns$，则共计约 $11ns$。

（4）对于 CSA 阵列乘法，本级进位与本级的和能同时传送到下一级，且同级部分积之间互不依赖，因此只需进行 $O(N)$ 次简单加法运算。假设简单加法时间为 8 位加法器时间的一半，则一共需要约 $4ns$。

5. (10 points)（课本 P90 页习题 12）采用 IEEE754 单精度浮点数格式计算下列表达式的值。

（1）$0.75+(-65.25)$

（2）$0.75-(-65.25)$

$x=0.75=(1.10\cdots0)_2\times2^{-1}$，转换成 IEEE 浮点数格式为 0 01111110 $10\cdots0$。其阶码 $E_x=$ 01111110，包含隐藏位的尾数 $M_x=1.10\cdots0$。

$y=-65.25=(-1.000001010\cdots0)_2\times2^6$，转换成 IEEE 浮点数格式为 1 10000101 $000001010\cdots0$。其阶码 $E_y=10000101$，包含隐藏位的尾数 $M_y=1.000001010\cdots0$。

$0.75+(-65.25)$ 运算过程如下：

1. 对阶

   $[\Delta E]_{补} = E_x + [-E_y]_{补} = 11111001 (mod 2^8)$，即 $\Delta E = -7$。因此，需要对 x 进行对阶，其尾数右移 7 位，得到 $M_x' = 0.000000110\cdots0$。

2. 尾数相加

   由于 y 为负数，因此实际进行的操作是尾数减法。$M_b = M_x' - M_y = 000.000000110\cdots0 - 001.000001010\cdots0 = 00.000000110\cdots0 + 110.111110110\cdots0 = 110.11111110\cdots0$。因此，结果为负数，而 IEEE 格式中，尾数使用原码表示，因此，需要将计算结果转换为原码形式，$001.00000010\cdots0$，且将结果的符号位置为 1。

3. 规格化

   上一步骤的计算结果已经是规格化的形式，因此，不需要进行规格化操作。

4. 舍入

   将多余的尾数去掉。由于多于 23 位的部分全部为 0，因此直接删除即可。

5. 溢出判断

   计算过程中，未进行规格化操作，因此结果的指数部分等于 y 的指数。因此，没有发生阶码上溢或者阶码下溢。

6. 最终计算结果：符号位为 1，阶码为 10000101，尾数为 $1.00000010\cdots0$。IEEE 浮点数格式为 1 10000101 $00000010\cdots0$。在十进制下验证计算结果，$-1.0000001 \times 2^6 = -64.5$。

$0.75 - (-65.25)$ 运算过程如下：

1. 对阶

   $[\Delta E]_{补} = E_x + [-E_y]_{补} = 11111001 (mod 2^8)$，即 $\Delta E = -7$。因此，需要对 x 进行对阶，其尾数右移 7 位，得到 $M_x' = 0.000000110\cdots0$。

2. 尾数相减

   由于 y 为负数，因此实际进行的操作是尾数加法。$M_b = M_x' + M_y = 000.000000110\cdots0 + 001.000001010\cdots0 = 001.000010\cdots0$。因此，结果为正数，其原码形式为，$001.000010\cdots0$，且将结果的符号位置为 0。

3. 规格化

   上一步骤的计算结果已经是规格化的形式，因此，不需要进行规格化操作。

4. 舍入

   将多余的尾数去掉。由于多于 23 位的部分全部为 0，因此直接删除即可。

5. 溢出判断

   计算过程中，未进行规格化操作，因此结果的指数部分等于 y 的指数。因此，没有发生阶码上溢或者阶码下溢。

6. 最终计算结果：符号位为 0，阶码为 10000101，尾数为 $1.000010\cdots0$。IEEE 浮点数格式为 0 10000101 $000010\cdots0$。在十进制下验证计算结果，$1.00001 \times 2^6 = 66$。