3.2 选择题（考研真题）。

（1）[2009] 一个 C 语言程序在一台 32 位机器上运行，程序中定义了 3 个变量 x、y、z，其中 x 和 z 是

int 型，y 为 short 型。当 $x=127$，$y=-9$ 时，执行赋值语句 $z=x+y$ 后，x、y、z 的值分别是 ___**D**___。

A．x = 0000007FH，y = FFF9H，z = 00000076H

B．x = 0000007FH，y = FFF9H，z = FFFF0076H

C．x = 0000007FH，y = FFF7H，z = FFFF0076H

D．x = 0000007FH，y = FFF7H，z = 00000076H

分析：y 为 short 类型，x 为 int 类型。在进行运算的时候会先将 y 转为 int 类型再相加。所以 y 会再运算前转为-9 的补码形式，为 1111 1111 1111 0111，然后与 x 相加得到 0000 0076.

（2）[2010] 假定有 4 个整数用 8 位补码分别表示 r_1= FEH，r_2= F2H，r_3= 90H，r_4= F8H，若将运算结果存放在一个 8 位的寄存器中，则下列运算会发生溢出的是 ____**B**___。

A．$r_1 \times r_2$ B．$r_2 \times r_3$ C．$r_1 \times r_4$ D．$r_2 \times r_4$

分析：A：FEH * F2H = 28，没有溢出；B：F2H * 90H = -2016，超出 8 位补码表示范围，发生了溢出；C：FEH * F8H = 16，没有溢出；D：F2H * F8H = 112，没有溢出

（3）[2013] 某字长为 8 位的计算机中，已知整型变量 x、y 的机器数分别为 $[x]_{补}$=11110100，$[y]_{补}$= 10110000。若整型变量 $z=2\times x+y/2$，则 z 的机器数为 ___**A**____。

A．11000000 B．00100100 C．10101010 D．溢出

分析：相当于将 x 左移一位，得到 1110 1000，y 右移一位，得到 1101 1000 然后将结果相加，得到 1100 0000.

（4）[2018] 假定带符号整数采用补码表示，若 int 型变量 x 和 y 的机器数分别是 FFFF FFDFH 和 0000 0041H，则 x、y 的值以及 $x-y$ 的机器数分别是 ___**C**____。

A．$x=-65$，$y=41$，$x-y$ 的机器数溢出

B．$x=-33$，$y=65$，$x-y$ 的机器数为 FFFF FF9DH

C．$x=-33$，$y=65$，$x-y$ 的机器数为 FFFF FF9EH

D．$x=-65$，$y=41$，$x-y$ 的机器数为 FFFF FF96H

分析：将 x 与 y 的机器数转为十进制可知 x = -33，y = 65.将机器数相减，得到 FFFF FF9EH

（5）[2018] 整数 x 的机器数为 1101 1000，分别对 x 进行逻辑右移 1 位和算术右移 1 位操作，得到的

机器数各是 ____**B**___。

A．1110 1100、1110 1100 B．0110 1100、1110 1100

C．1110 1100、0110 1100 D．0110 1100、0110 1100

分析：逻辑右移最高位补 0，算术右移最高位补符号位，仅有 B 符合要求。

（6）[2019] 浮点数加减运算过程一般包括对阶、尾数运算、规格化、舍入和判断溢出等步骤。

设浮点数的阶码和尾数均采用补码表示，且位数分别为 5 位和 7 位（均含 2 位符号位）。若有两个数

$X=2^7\times29/32$，$Y=2^5\times5/8$，则用浮点加法计算 $X+Y$ 的最终结果是 ___**D**____。

A．001111100010 B．001110100010 C．010000010001 D．发生溢出

分析：x 可记为 00,111;00,11101(分号前为阶码，分号后为尾数)，同理，y 可记作 00,101;00,10100；再对 x 和 y 的阶码进行相减，即 00,111 － 00,101,可知 x 的阶码大于 y，因此需要将 y 的阶码+2，尾数右移两位，得到 y 为 00,111;00,00101;之后尾数相加，可得到符号位为 01，需要进行右规。规格化后将尾数右移 1 位，阶码+1 得到阶码符号位为 01，发生了溢出。

（7）[2015] 下列有关浮点数加减运算的叙述中，正确的是 ___**D**____。

Ⅰ．对阶操作不会引起阶码上溢或下溢

Ⅱ．右规和尾数舍入都可能引起阶码上溢

Ⅲ．左规时可能引起阶码下溢

Ⅳ．尾数溢出时结果不一定溢出

A．仅Ⅱ、Ⅲ B．仅Ⅰ、Ⅱ、Ⅳ

C．仅Ⅰ、Ⅲ、Ⅳ D．Ⅰ、Ⅱ、Ⅲ、Ⅳ

分析：对于 1，对阶操作是调整两个浮点数的指数部分，使指数相同，不会引起阶码上溢或下溢，1 正确；对于 2，右规是便于尾数相加运算，相加后可能引起阶码上溢，2 正确；对于 3，左规是将结果规范化，可能导致阶码下溢，3 正确；对于 4，尾数溢出结果不一定溢出，4 正确。

3.3 回答下列问题。

（3）如何判断浮点数运算结果是否为规格化数？如果不是规格化数，如何进行规格化？

1.看尾数部分的表示形式。若浮点数最高位为 1，则是规格化的，若尾数全为 0，这表示+-0，也是规格化的。

2.首先将尾数左移一位，使最高位变为 1，指数部分减 1；再确保指数部分没有下溢，根据需要进行舍入适应尾数的位数范围；最后检查溢出，看指数有没有上溢

（4）为什么阵列除法器中能用 CAS 的进位 / 借位控制端作为上商的控制信号？

CAS 的主要功能是将两个操作数的部分和计算出来，同时生成借位和进位。因为在除法器的计算过程中需要根据部分和的情况来确定商的值。

3.4 已知 x 和 y，用变形补码计算 $x+y$，并判断结果是否溢出。

（3）$x=-0.10111$，$y=-0.11000$。

$x = -0.10111$　　$y = -0.11000$

$[x]_{补} = 11.01001$　　$[y]_{补} = 11.01000$

相加：$[x]_{补} + [y]_{补} = 10.10001$

负溢出

3.5 已知 x 和 y，用变形补码计算 $x-y$，并判断结果是否溢出。

（3）$x = -0.11111$，$y = -0.11001$。

$x = -0.11111$　$[x]_{补} = 11.00001$

$y = -0.11001$　$[y]_{补} = 11.00111$

$[-y]_{补} = 00.11001$

$11.00001 + 00.11001 = 11.11010$

$\therefore [x]_{补} + [-y]_{补} = 11.11010$

未溢出

3.6 用原码一位乘法计算 $x \times y$。

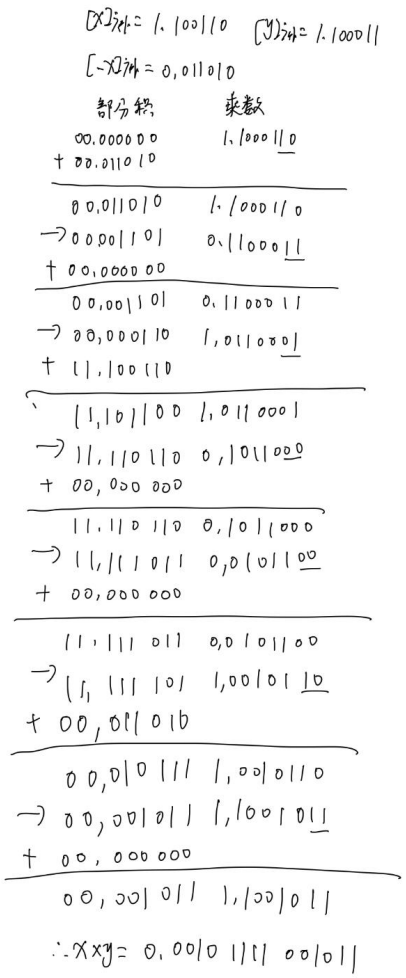
（2）$x = -0.11010$，$y = -0.01011$。

部分积	乘数
00.00000	11010
+ 00.00000	
00.00000	11010
→ 00.00000	01101
+ 00.11010	
00.11010	01101
→ 00.01101	00110
+ 00.00000	
00.01101	00110
→ 00.00110	10011
+ 00.11010	
00.00000	10011
→ 00.00000	11001
+ 00.11010	
00.11010	11001
→ 00.01101	00011

由于符号位为正，结果为 0.00110 00010

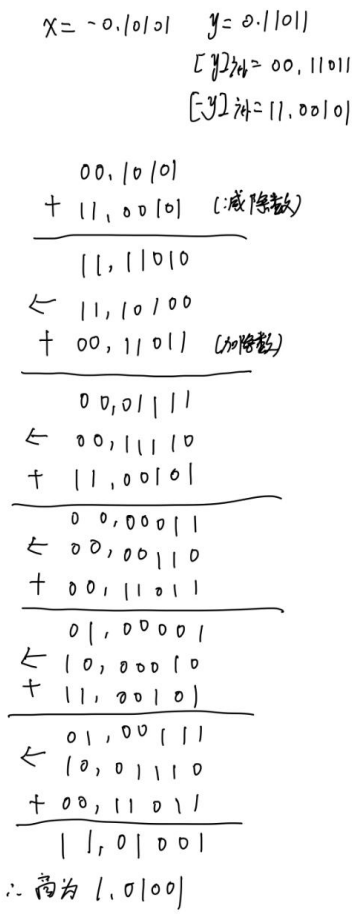
3.7 用补码一位乘法计算 $x \times y$。

（2）$x = -0.011010$，$y = -0.011101$。



3.8 用原码不恢复余数法计算 $x \div y$。

（2）$x = -0.10101$，$y = 0.11011$。



3.9 设数的阶码为 3 位，尾数为 6 位（均不包括符号位），按机器补码浮点运算规则完成下列 $[x+y]_{补}$ 运算。

（1）$x=2^{011}\times0.100100$，$y=2^{010}\times(-0.011010)$。

对于阶码，X 补 = 00011 00100100，Y 补 = 00010 1100110，阶差为 00001，，将 Y 补的尾数右移一位得到 00011 11110011

X 补+Y 补 = 00011 00010111，尾数相加为 00010111

再将结果规格化，由于尾数符号位跟最高有效位相同，需要左规，规格化结果如下

X 补+Y 补 = 00010 00101110，不需要舍入，无溢出，则 X 补+Y 补 = 000100010110

3.11 假定在一个 8 位字长的计算机中运行如下 C 语言程序段。

```
unsigned int x=134;
unsigned int y=246;
int m=x; int n=y;
unsigned int z1=x-y;
unsigned int z2=x+y;
int k1=m-n;
```

```
int k2=m+n;
```

若编译器编译时将 8 个 8 位寄存器 R1 ～ R8 分别分配给变量 *x*、*y*、*m*、*n*、*z*1、*z*2、*k*1 和 *k*2。请回答

下列问题（提示：带符号整数用补码表示）。

（1）执行上述程序段后，寄存器 R1、R5 和 R6 中的内容分别是什么？（用十六进制表示）

R1 = 134，转为 16 进制为 86H；R5 = x - y = -112，由于 z1 为无符号数，且机器是 8 位子长，所以-112 会转成 256-112 = 144，因此 R5 = 144，转为 16 进制为 90H；同样，R6 会发生溢出，z2 = 380 - 256 = 124，转为 16 进制为 7CH.

因此，他们的内容如下

R1 = 86H, R5 = 90H, R6 = 7CH

（2）执行上述程序段后，变量 *m* 和 *k*1 的值分别是多少？（用十进制表示）

对于 m，将 134 转为二进制为 1000 0110，由于 m 为有符号数，因此首位 1 代表符号，根据补码运算可以得到 m = -122；同理，k1 也是有符号数，经过计算 k1 为两个有符号 int 的差，结果为 134 - 246 = -112

因此，m 和 k1 的值如下

m = -122，k1 = -112

（3）上述程序段涉及带符号整数加减、无符号整数加减运算，这 4 种运算能否利用同一个加法器及辅助电路实现？简述理由。

能。加法器可以执行有符号数和无符号数的加减法，可以根据运算的类型来选择正确的操作数以及如何处理结果。无符号整数加减法可以直接用加法器进行运算，带符号整数加减法需要考虑符号位的影响，但仍然可以通过加法器实现

（4）计算机内部如何判断带符号整数加减运算的结果是否发生溢出？上述程序段中，哪些带符号整数运算语句的执行结果会发生溢出？

若符号位进位 Cf 和最高位数据为进位 Cd 不同，则结果溢出。最后一条语句执行时会发生溢出。