“计算机组织结构”作业 3 参考答案

1. 【2009 统考真题】一个 C 语言程序在一台 32 位机器上运行，程序中定义了三个变量 x、y、z，其中 x 和 z 为 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（）。

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

D

结合题干及选项可知，int 为 32 位，short 为 16 位；又因 C 语言的数据在内存中为补码形式，因此 x、y 的机器数写为 0000007F、FFF7H。执行 z=x+y 时，由于 x 为 int 型，y 为 short 型，因此需将 y 的类型强制转换为 int 型，在机器中通过符号位扩展实现，由于 Y 的符号位为 1，因此在 y 的前面添加 16 个 1，即可将 y 强制转换为 int 型，其十六进制形式为 FFFFFFF7H。然后执行加法，即 0000007FH+FFFFFFF7H=00000076H，其中最高位的进位 1 自然丢弃。

注意：数据转换时应注意的问题如下：

1）有符号数和无符号数之间的转换。例如，由 signed 型转换为等长 unsigned 型数据时，符号位成为数据的一部分，即负数转换为无符号数时，数值将发生变化。同理，由 unsigned 转换为 signed 时最高位作为符号位，也可能发生数值变化。

2）数据的截取与保留。当一个浮点数转换为整数时，浮点数的小数部分全部舍去，并按整数形式存储。但浮点数的整数部分不能超过整型数允许的最大范围，否则溢出。

3）数据转换中的精度丢失。四舍五入会丢失一些精度，截去小数也会丢失一些精度。此外，数据由 long 型转换为 float 型或 double 型时，有可能在存储时不能准确地表示该长整数的有效数字，精度也会受到影响。

4）数据转换结果的不确定性。当较长的整数转换为较短的整数时，要将高位截去。例如，long 型转换为 short 型，只将低 16 位送过去，这样就会产生很大的误差。浮点数降格时，如 double 型转换为 float 型，当数值超过 float 型的表示范围时，所得到的结果将是不确定的。对于此问题在“常见问题与知识点”的第 2 问中有另一角度的解析。

2. 【2010 统考真题】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 位寄存器中，则下列运算会发生溢出的是（）。

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

B

本题的真正意图是考查补码的表示范围，采用补码乘法规则计算出 4 个选项是费力不讨好的做法，且极易出错。8 位补码所能表示的整数范围为-128~+127。将 4 个数全部转换为十进制数，得 r1=-2，r2=-14，r3=-112，r4=-8，得 r2×r3=1568，远超出了表示范围，发生溢出。

3. 【2012 统考真题】假定编译器规定 int 和 short 类型长度分别为 32 位和 16 位，执行下列 C 语言语句：

unsigned short x=65530;
unsigned int y=x;
得到 y 的机器数为（）。

A. 0000 7FFAH
B. 0000 FFFAH

C. FFFF 7FFAH
D. FFFF FFFAH

B

将一个 16 位 unsigned short 转换成 32 位 unsigned int，因为都是无符号数，新表示形式的高位用 0 填充。16 位无符号整数所能表示的最大值为 65535，其十六进制表示为 FFFFH，因此 x　的十六进制表示为 FFFFH-5H=FFFAH，所以 y 的十六进制表示为 0000 FFFAH。

排除法：先直接排除 C、D，然后分析余下选项的特征。由于 A、B 的值相差几乎近 1 倍，可采用算出 0001 0000H（接近 B 且好算的数）的值。

4. 【2013 统考真题】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$[x]_{补}$ = 1 1110100，$[y]_{补}$ =1 0110000。若整型变量$z = 2x + y/2$，则z的机器数为（　　）。

A. 1 1000000
B. 0 0100100
C. 1 0101010
D. 溢出

A

x*2，将 x 算术左移一位为 1 1101000；y/2，将 y 算术右移一位为 1 1011000，均无溢出或丢失精度。补码相加为 1 1101000 + 1 1011000=1 1000000，亦无溢出。

5. 【2014 统考真题】若 x=103，y=-25，则下列表达式采用 8 位定点补码运算实现时，会发生溢出的是（）。

A x+y
B -x+y
C x -y
D -x-y

C

8 位定点补码表示的数据范围为-128~127，若运算结果超出这个范围则会溢出，A 选项 x + y = 103 – 25 = 78，符合范围，A 排除；B 选项-x + y = -103 - 25= -128，符合范围，B 除：D 选项-x – y = -103 + 25 = -78，符合范围，D 排除：C 选项 x – y = 103 + 25 = 128，超过 127。

6. 【2016 统考真题】有如下 C 语言程序段：

short si= -32767;
unsigned short usi = si;
执行上述两条语句后，usi 的值为（　　）。

A. -32767
B. 32767
C. 32768
D. 32769

D

因 c 语言中的数据在内存中为补码表示形式， si 对应的补码二进制表示为 1000 0000 0000 0001B，最前面的一位“1”为符号位，表示负数，即-32767。由 signed 型转化为等长的 unsigned 型数据时，符号位成为数据的一部分，即负数转化为无符号数（即正数）时，其数值将发生变化。usi 对应的补码二进制表示与 si 的表示相同，但表示正数，为 32769。

7. 【2018 统考真题】假定有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 的值及 x-y 的机器数分别是（）。

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

C

利用补码转换成原码的规则：负数符号位不变数值位取反加一；正数补码等于原码。两个机器数对应的原码是$[x]_{原}$ = 80000021H，对应的数值是-33，$[y]_{原}$ = $[y]_{补}$ = 00000041H = 65。

排除 A、D。x - y 直接利用补码减法准则，$[x]_{补}$ - $[y]_{补}$ = $[x]_{补}$ + $[-y]_{补}$，-y 的补码是连同符号位取反加一，最终减法变成加法，得出结果为 FFFFFF9EH。

8. 【2018 统考真题】整数×的机器数为 1101 1000，分别对×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（　　）。
   A. 1110 1100、 1110 1100
   B. 0110 1100、 1110 1100
   C. 1110 1100、 0110 1100
   D. 0110 1100、 0110 1100

B

逻辑移位：左移和右移空位都补 0，且所有数字参与移动；补码算术移位：符号位不参与移动，右移空位补符号位，左移空位补 0。

9. 【2018 统考真题】减法指令“sub R1, R2, R3”的功能为“(R1) – (R2) → R3”，该指令执行后将生成进位/借位标志 CF 和溢出标志 OF。若(R1) = FFFF FFFFH, (R2) = FFFF FFF0H，则该减法指令执行后，CF 与 OF 分别为（　　）。
   A. CF= 0, OF=0
   B. CF=1, OF=0
   C. CF=0, OF=1
   D. CF=1, OF=1

A

$[x]_{补}$ - $[y]_{补}$ = $[x]_{补}$ + $[-y]_{补}$，$[-R2]_{补}$ = 00000010H，很明显$[R1]_{补}$ + $[-R2]_{补}$的最高位进位和符号位进位都是 1（当最高位进位和符号位进位的值不相同时才产生溢出），可以判断溢出标志 OF 为 0。同时，减法操作只需判断借位标志，R1 大于 R2，所以借位标志为 0。

10. 【2019 统考真题】考虑以下 C 语言代码：
    unsigned short usi = 65535;
    short si= usi;
    执行上述程序段后，si 的值是（　　）。
    A -1
    B -32767
    C -32768
    D -65535

A

unsigned short 类型为无符号短整型，长度为 2 字节，因此 unsigned short usi 转换为二进制代码即 1111 1111 1111 1111。short 类型为短整型，长度为 2 字节，在采用补码的机器上，short si 的二进制代码为 1111 1111 1111 1111，因此 si 的值为-1。

11. 【2009 统考真题】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 和 7（均含 2 位符号位）。若有两个数 $X = 2^7 \times 29/32$ 和 $Y = 2^5 \times 5/8$，则用浮点加法计算$X + Y$的最终结果是（　　）。
    A. 00111 1100010
    B. 00111 0100010
    C. 01000 0010001
    D. 发生溢出

D

X的浮点数格式为 00, 111; 00, 11101（分号前为阶码，分号后为尾数），Y的浮点数格式为 00, 101; 00, 10100。然后根据浮点数的加法步骤进行运算。

① 对阶。X、Y阶码相减，即 00, 111 - 00, 101 = 00, 11 + 11, 0111 = 00, 010，可知X的阶码比Y的阶码大 2 (这一步可直接目测)。根据小阶向大阶看齐的原则，将Y的阶码加 2，尾数右移 2 位，将Y变为 00, 111; 00, 00101。

② 尾数相加。即 00, 11101 + 00, 00101 = 01, 00010，尾数相加结果符号位为 01，因此需

要右规。

③ 规格化。将尾数右移 1 位，阶码加 1，得 $X+Y$为 01, 000; 00, 10001。

④ 判断溢出。阶码符号位为 01，说明发生溢出。

本题容易误选 B、C，因为 B、C 本身并无计算错误，只是它们不是最终结果，选项 B 少了第 3 步和第 4 步，选项 C 少了第 4 步。

12. 【2015 统考真题】下列有关浮点数加减运算的叙述中，正确的是（ ）。

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

D

对阶是较小的阶码对齐至较大的阶码，I正确。右规和尾数舍入过程，阶码加 1 而可能上溢，II正确，同理III也正确。尾数溢出时可能仅产生误差，结果不一定溢出。

========== 分割线：以下内容不在小程序上提交 ==========

13. 【2011 统考真题】假定在一个 8 位字长的计算机中运行如下 C 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、z1、k1 和 k2。请回答下列问题（提示：有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 的内容分别是什么（用十六进制表示）？

2）执行上述程序段后，变量 m 和 k1 的值分别是多少（用十进制表示）？

3）上述程序段涉及有符号整数加减、无符号整数加减运算，这四种运算能否利用同一个加法器辅助电路实现？简述理由。

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

1)　134 = 128 + 6 = 1000 0110B，所以 x 的机器数为 1000 0110B，因此 R1 的内容为 86H。246 = 255 - 9 = 1111 0110B，所以 y 的机器数为 1111 0110B，x - y = 1000 0110 + 0000 1010 = (0)1001 0000，括号中为加法器的进位，因此 R5 的内容为 90H。x + y = 1000 0110 + 1111 0110 = (1)0111 1100，括号中为加法器的进位，因此 R6 的内容为 7CH。

2)　m 的机器数与 x 的机器数相同，皆为 86H = 1000 0110B，解释为有符号整数 m（用补码表示）时，其值为-111 1010B = -122。m - n 的机器数与 x - y 的机器数相同，皆为 90H = 1001 0000B，解释为有符号整数 k1（用补码表示）时，其值为-111 0000B = -112。

3)　能。n 位加法器实现的是模 2^n 无符号整数加法运算。对于无符号整数 a 和 b，a + b 可以直接用加法器实现，而 a - b 可用 a 加 b 的补数实现，即 $a - b = a + [-b]_{补} \pmod{2^n}$，所以 n 位无符号整数加减运算都可在 n 位加法器中实现。由于有符号整数用补码表示，补码加减运算公式为$[a + b]_{补} = [a]_{补} + [b]_{补} \pmod{2^n}$，$[a - b]_{补} = [a]_{补} + [-b]_{补} \pmod{2^n}$，所以 n 位有符号整数加减运算都可在 n 位加法器中实现。

4)　有符号整数加减运算的溢出判断规则为：若加法器的两个输入端（加法）的符号相同，且不同于输出端（和）的符号，则结果溢出，或加法器完成加法操作时，若次高位（最高数位）的进位和最高位（符号位）的进位不同，则结果溢出。最后一条语句执行时会发生溢出。因为 1000 0110 + 1111 0110 = (1)0111 1100，括号中为加法器的进位，根据上述溢出判断规则可知结果溢出。或者，因为 2 个带符号整数均为负数，它们相加之后，结果小于 8 位

进制所能表示的最小负数。

14. 【2020 统考真题】有实现 x × y 的两个 c 语言函数如下：

```
unsigned umul(unsigned x, unsigned y) { return x * y; }
int imul(int x, int y) { return x * y; }
```

假定某计算机 M 中的 ALU 只能进行加减运算和逻辑运算．请回答下列问题。

1）若 M 的指令系统中没有乘法指令，但有加法、减法和位移等指令，则在 M 上也能实现上述两个函数中的乘法运算，为什么？

2）若 M 的指令系统中有乘法指令，则基于 ALU、位移器、寄存器及相应控制逻辑实现乘法指令时，控制逻辑的作用是什么？

3）针对以下三种情况：a）没有乘法指令；b）有使用 ALU 和位移器实现的乘法指指令；c）有使用阵列乘法器实现的乘法指令，函数 umul()在哪种情况下执行的时间最长？在哪种情况下执行的时间最短？说明理由。

4）n 位整数乘法指令可保存 2n 位乘积，当仅取低 n 位作为乘积时，其结果可能会发生溢出。当 n＝32、x=2^{31}-1、y=2 时，带符号整数乘法指令和无符号整数乘法指令得到的 x×y 的 2n 位乘积分别是什么（用十六进制表示）？此时函数 umul()和 imul()的返回结果是否溢出？对于无符号整数乘法运算，当仅取乘积的低 n 位作为　　乘法结果时，如何用 2n 位乘积进行溢出判断？

1)　乘法运算可以通过加法和移位来实现。编译器可以将乘法运算转换为一个循环代码段，在循环代码段中通过比较、加法和移位等指令实现乘法运算。

2)　控制逻辑的作用是控制循环次数，控制加法和移位操作。

3)　a）最长，c）最短。对于 a），需要用循环代码段（即软件）实现乘法操作，因而需要反复执行很多条指令，而每条指令都需要取指令、译码、取数、执行并保存结果，所以执行时间很长；对于 b）和 c），都只需用一条乘法指令实现乘法操作，不过 b）中的乘法指令需要多个时钟周期才能完成，而 c）中的乘法指令可在一个时钟周期内完成，所以 c）的执行时间最短。

4)　当 n＝32、x＝2^{31} - 1、y＝2 时，带符号整数和无符号整数乘法指令得到的 64 位乘积都是 0000 0000 FFFF FFFEH。int 型的表示范围为[-2^{31}，2^{31} - 1]，故函数 imul()的结果溢出；unsigned int 型的表示范围为[0，2^{32} - 1]，故函数 umul()的结果不溢出。对于无符号整数乘法，若乘积高 n 位全为 0，即使低 n 位全为 1 也正好是 2^{32} - 1，不溢出，否则溢出。

15. 【2017 统考真题】已知 $f(n)=\sum_{i=0}^{n}2^i=2^{n+1}-1=\overbrace{11\cdots1}^{n+1\text{位}}\text{B}$，计算 f(n)的 c 语言函数 f1 如下：

```
int f1(unsigned n) {
    int sum=1, power=1;
    for(unsigned i=0; i<=n-1; i++){
        power *=2;
        sum += power;
    }
    return sum;
}
```

将 f1 中的 int 都改为 float，可得到计算 f(n)的另一个函数 f2。假设 unsigned 和 int 型数据都占 32 位，float 采用 IEEE754 单精度标准。请回答下列问题：

1)　当 n = 0 时，f1 会出现死循环，为什么？若将 f1 中的变量 i 和 n 都定义为 int 型，则 f1 是否还会出现死循环？为什么？

2)　f1(23) 和 f2(23) 的返回值是否相等？机器数各是什么（用十六进制表示）？

3)　f1(24) 和 f2(24) 的返回值分别为 33 554 431 和 33 554 432.0，为什么不相等？

4)　f(31)＝2^{32} - 1，而 f1(31)的返回值却为-1，为什么？若使 f1(n)的返回值与 f(n)相等，则最大的 n 是多少？

5)　f2(127) 的机器数为 7F80 0000H，对应的值是什么？若使 f2(n) 的结果不溢出，则最大的 n 是多少？若使 f2(n) 的结果精确（无舍入），则最大的 n 是多少？

1)　由于 i 和 n 是 unsigned 型，因此“i <= n - 1”是无符号数比较，n＝0 时，n - 1 的机器

数为全 1，值是 2^{32} - 1，为 unsigned 型可表示的最大数，条件“i <= n - 1”永真，因此出现死循环。若 i 和 n 改为 int 类型，则不会出现死循环。因为“i <= n - 1”是有符号整数比较，n = 0 时，n - 1 的值是-1，当 i = 0 时条件“i <= n - 1”不成立，此时退出 for 循环。

2) f1(23) 与 f2(23) 的返回值相等。f(23) = 2^{23+1} - 1 = 2^{24} - 1，其二进制形式是 24 个 1。int 占 32 位，没有溢出。float 有 1 个符号位，8 个指数位，23 个底数位，23 个底数位可以表示 24 位的底数，所以两者返回值相等。 f1(23) 的机器数是 00FF FFFFH，f2(23) 的机器数是 4B7FFFFFH。 显而易见，前者是 24 个 1，即 0000 0000 1111 1111 1111 1111 $1111\ 1111_{(2)}$，后者的符号位是 0，指数位为 $23 + 127_{(10)} = 1001\ 0110_{(2)}$，底数位是 111 1111 1111 1111 1111 $1111_{(2)}$。

3) 当 n = 24 时，f1(24) = 1 1111 1111 1111 1111 1111 1111B，而 float 型数只有 24 位有效位，舍入后数值增大，所以 f2(24) 比 f1(24) 大 1。

4) 显然 f(31)已超出了 int 型数据的表示范围，用 f1(31) 实现时得到的机器数为 32 个 1，作为 int 型数解释时其值为-1，即 f1(31)的返回值为-1。因为 int 型最大可表示的数是 0 后面加 31 个 1，因此使 f1(n) 的返回值与 f(n) 相等的最大 n 值是 30。

IEEE754 标准用“阶码全 1、尾数全 0”表示无穷大。f2 的返回值为 float 型，机器数 7F80 0000H 对应的值是 $+\infty$。当 n = 126 时，$f(126) = 2^{127} - 1 = 1.1\cdots1 \times 2^{126}$，对应阶码为 127 + 126 = 253，尾数部分舍入后阶码加 1，最终阶码为 254，是 IEEE754 单精度格式表示的最大阶码。因此使结果不溢出的最大 n 值为 126。当 n = 23 时，f(23)为 24 位 1，float 型数有 24 位有效位，所以不需要舍入，结果精确。因此使 f2 获得精确结果的最大 n 值为 23。