计算机组成原理课程作业

——第二次

黄勖 22920212204392

厦門大學
XIAMEN UNIVERSITY

(1) [2015]由3个“1"和5个“0”组成的8位二进制补码，能表示的最小整数是（ B ）

A. -126　　B. -125　　C. -32　　D. -3

首先1应当放在负数符号位，其次根据补码定义1应当从低位放起，所以这个补码为10000011，转化为十进制即-125.

2.2

插入一段文本插入

(2) [2019]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 ）

A. -1　B. -32767　C. -32768　D. -65535

在这段代码中，我们首先声明了一个无符号短整型变量 usi，并将其初始化为最大值 65535，即二进制中所有位都为1。然后，我们声明了一个有符号短整型变量 si，并将其赋值为无符号短整型变量 usi。

由于 si 是有符号变量，因此在将无符号变量 usi 赋值给 si 时，会进行类型转换。根据C语言的类型转换规则，将一个无符号整型数值赋值给一个有符号整型变量时，需要进行符号扩展。也就是说，如果无符号整型数值的最高位为1，表示这是一个正数，而在有符号整型变量中则表示这是一个负数。因此，在进行符号扩展时，会在有符号变量的高位填充1。

在本题中，无符号短整型变量 usi 的值为 65535，即二进制的所有位都是1，因此在将其赋值给有符号短整型变量 si 时，需要进行符号扩展。符号扩展后的结果是一个负数，其值等于 65535 - 65536 = -1。

因此，执行完这段代码后，变量 si 的值为 -1

2.2

(3) [2012]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（ B ）

A. 0000 7FFAH

B. 0000 FFFAH

C. FFFF 7FFAH

D. FFFF FFFAH

在这段代码中，我们首先声明了一个无符号短整型变量 x，并将其初始化为 65530。由于短整型的长度为16位，因此变量 x 的二进制表示形式为：

1111 1111 1110 1010

然后，我们声明了一个无符号整型变量 y，并将其赋值为无符号短整型变量 x。根据C语言的类型转换规则，将一个短整型数值赋值给一个整型变量时，会进行整数扩展。在进行整数扩展时，会在整型变量的高位填充0。整数扩展后的结果为：

0000 0000 0000 0000 1111 1111 1110 1010

因此，变量 y 的机器数为 0000 FFFAh。

2.2

插入一段文本插入

(4) [2016]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（ D ）

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

在这段代码中，我们首先声明了一个有符号短整型变量 si，并将其初始化为 -32767。然后，我们声明了一个无符号短整型变量 usi，并将其赋值为有符号短整型变量 si。

根据C语言的类型转换规则，将一个有符号整型数值赋值给一个无符号整型变量时，如果有符号整型数值为负数，那么在进行类型转换时，会将其转换为无符号整型数值的补码表示形式。补码表示形式的计算方法是将原数的绝对值取反后加1。

在本题中，有符号短整型变量 si 的值为 -32767，即二进制的最高位为1，其他位都是0。将其转换为无符号短整型变量时，需要先将其转换为补码表示形式。因此，我们可以先将其取绝对值，然后取反加1，即：

|-32767| = 32767

32767 的二进制表示形式为：0111 1111 1111 1111

取反得到：1000 0000 0000 0000

加1得到：1000 0000 0000 0001

因此，无符号短整型变量 usi 的值为 32769。

2.2

插入一段文本插入

2.2

(5) [2011]float型数据通常用IEEE754单精度浮点数格式表示。若编译器将float 型变量x分配在一个32位浮点寄存器FR1中，且x=-8.25. 则FR1的内容是（ A ）

A. C104 0000H
B. C242 0000H
C. C184 0000H
D. C1C2 0000H

根据IEEE 754标准，32位单精度浮点数的表示形式为:

符号位（1位）	指数位（8位）	尾数位（23位）
s	e	m

其中，符号位s表示正负，指数位e采用移码表示，尾数位m用来表示数值的精度和大小。
对于一个32位浮点数x，现在我们来计算题目中给定的浮点数 x = -8.25 的IEEE 754单精度浮点数表示形式:

1. 确定符号位s：x为负数时，s为1。
2. 确定指数位的值：由于x的绝对值为8.25，可以将其转换为二进制数，即 1000.01，移动小数点后得到 1.00001 x 2^3，因此指数位的值为 127+3=130，需要将其转换为8位二进制数，即 1000 0010B。
3. 确定尾数位的值：由于指数位为130，所以尾数位的值需要将 1.00001 中整数部分的1去掉，保留小数部分并拓展到23位，即000 0100 0000 0000 0000 0000。
4. 拼接符号位、指数位和尾数位，得到32位二进制数即1100 0001 0000 0100 0000 0000 0000 0000。
5. 将32位二进制数转换为16进制数，得到C104 0000H。

(6) [2013]某数采用IEEE754单精度浮点数格式表示为C640 0000H,则该数的值是（ A ）

A. -1.5×2^{13}　B. -1.5×2^{12}　C. -0.5×2^{13}　D. -0.5×2^{12}

1. 先转化为二进制：1100 0110 0100 0000 0000 0000 0000 0000
2. 符号位s为1，表示负数。
3. 尾数位m为 100 0000 0000 0000 0000 0000，表示 1 + 0.5 = 1.5。
4. 指数位e为 1000 1100B（140），表示 e+127 = 140，因此指数e为 13。
5. 将符号位s、指数位e和二进制小数m按照公式计算得到该数的值为-1.5×2^{13}

2.2

插入一段文本插入

(7) [2012] float 型(即IEEE754单精度浮点数格式)能表示的最大正整数是（ D ）

A.2^{126}-2^{103} B. 2^{127}-2^{104} C.2^{127}-2^{103} D. 2^{128}-2^{104}

最大值：符号位为0表正，阶码1111 1110(1111 1111在IEEE754表示无穷大或NaN)：254（e值最大为127），尾数：111 1111 1111 1111 1111 1111（真值为1-2^(-23) +1），即 **0 11111110 11111111111111111111111**，经过计算最大值即2^128-2^104

2.2

插入一段文本插入

厦門大學
XIAMEN UNIVERSITY

(8) [2018] IEEE754单精度浮点格式表示的数中，最小规格化正数是（ A ）

A. 1.0×2^{-126}　　B. 1.0×2^{-127}　　C. 1.0×2^{-128}　　D. 1.0×2^{-149}

同上题，符号位为0表正，阶码1（e值为-126），尾数：0，经过计算答案为A

- IEEE754浮点数表示范围（单精度浮点数）：
  - 单精度规格化浮点数绝对值最小数：E=1，M=0，$f=2^{1-127} \times 1.0=2^{-126}$
  - 单精度规格化浮点数绝对值最大数：E=254，M=11...11（尾数=1.11...11），$f=2^{254-127} \times (2-2^{-23})=2^{128}-2^{-104} \approx +3.4 \times 10^{38}$
  - 单精度非规格化浮点数绝对值最小数：E=0，M=00...01（尾数=0.00...01），$f=2^{-126} \times 2^{-23}=2^{-149}$
  - 单精度非规格化浮点数绝对值最大数：E=0，M=11...11（尾数=0.11...11），$f=2^{-126} \times (1-2^{-23})=2^{-126}-2^{-149}$

2.2

插入一段文本插入

2.2

(9) [2014]float型数据通常用IEEE754单精度浮点格式表示。假定两个float 型变量x和y分别存放在32位寄存器fl和f2中，若(f1)=CC90 0000H (f2)=B0C0 0000H,则x和y之间的关系为（ A ）

A. x<y且符号相同

B. x<y且符号不同

C. x>y且符号相同

D. x>y且符号不同

根据IEEE754单精度浮点数的表示方法，首先需要确定这两个数的符号位、指数位和尾数位，然后才能进行比较。具体步骤如下：

(f1)= 1100 1100 1001 0000 0000 0000 0000 0000

(f2)= 1011 0000 1100 0000 0000 0000 0000 0000

1. 确定符号位：从左往右数第1位为符号位，0表示正数，1表示负数。
- 对于f1，符号位为1，表示负数。
- 对于f2，符号位为1，表示负数。
2. 确定指数位：从左往右数第2位到第9位为指数位，需要减去偏移量127，得到指数的真实值。
- 对于f1=CC90 0000H，指数位为10011001(153)，减去偏移量127得到指数值为26。
- 对于f2=B0C0 0000H，指数位为01100001(97)，减去偏移量127得到指数值为-30。
3. 确定尾数位：从左往右数第10位到第32位为尾数位，尾数位的值需要根据指数位进行规格化。
- 对于f1=CC90 0000H，尾数位为001……，进行规格化，得到-1.001乘以2的26次方。
- 对于f2=B0C0 0000H，尾数位为1……，进行规格化，得到-1.1乘以2的-30次方。
4. 对于f1和f2，都是负数，且f1的指数值大得多，因此f1的数值比f2小，即x<y且符号相同。

2.2

(10) [2010]假定变量i、f. d的数据类型分别为int. float. double (int用补码表示，float 和double用IEEE754标准中的单精度和双精度浮点数据格式表示) ,已知i=785, f=1.5678e3，d=1.5e100. 若在32位计算机中执行下列关系表达式，则结果为真的是（ B ）

I. i==(int)(float)i

II. f==(float)(int)f

II. f==(float)(double)f

IV. (d+f)-d==-f

A.仅I、II

B.仅I、III

C.仅II、III

D.仅III、IV

Ⅰ．i==(int)(float)i i值先扩充成float，精度增大，然后再减小，这个表达式结果为真。

Ⅱ．f==(float)(int)f 将f转换为int类型时，需要将其截断为32位整数，这里发生了精度损失，这个表达式结果为假。

Ⅲ．f==(float)(double)f 同理 Ⅰ

Ⅳ．(d+f)-d==f 由于d和f都是浮点数，并且d的指数部分远大于f的指数部分，所以在进行加减运算时会发生舍入误差。具体来说，

(d+f) - d ≈ d + 舍入误差 - d ≈ 舍入误差 -f ≈ -f + 舍入误差

由于舍入误差不一定相等或相反，并且可能很小或很大（取决于d和f的具体值），所以不能保证(d+f)-d和-f相等。所以这个表达式结果也是假。

（实际情况下，浮点运算对阶后f的尾数很可能直接有效位被舍去而变为0）

(11) [2013]用海明码对长度为8位的数据进行检错和纠错时，若能纠正一位错，则校验位数至少为（ C ）

A.2　　　B.3　　　C.4　　　D.5

通常情况下，海明码的校验位数是由以下公式确定的：$m + r + 1 \leqslant 2^r$，其中m是数据位数，r是校验位数。对于长度为8位的数据，如果能够纠正一位错，说明至少需要添加一个校验位，使得校验位数r满足以上公式中的条件。通过代入m=8和r=4，可以发现这个条件得到满足，因此至少需要添加4个校验位。

2.2

插入一段文本插入

厦門大學 XIAMEN UNIVERSITY

写出下列各数的原码、反码和补码。

0， -0， 0.10101 ， -0.10101， 0.11111 ， -0.11111， -0.10000， 0.10000

	真值	原码	反码	补码
1	0	0.000…0	0.000…0	0.000…0
2	-0	1.000…0	1.111…1	0.000…0
3	0.10101	0.10101	0.10101	0.10101
4	-0.10101	1.10101	1.01010	1.01011
5	0.11111	0.11111	0.11111	0.11111
6	-0.11111	1.11111	1.00000	1.00001
7	-0.10000	1.10000	1.01111	1.10000
8	0.10000	0.10000	0.10000	0.10000

2.4

插入一段文本插入

2.5已知数的补码表示形式，求数的真值。

补码	真值	补码	真值	补码	真值
0.10010	0.10010	1.10010	-0.01110	1.11111	-0.00001
1.00000	-1.00000	0.10001	0.10001	1.00001	-0.11111

2.5

2.6

2.6 C语言中允许无符号数和有符号整数之间的转换，下面是一段C语言代码。

```
int x = -1;
unsigned u=2147483648;
printf ("x=%u=%d\n", x, x);
printf ("u=%u=%d\n", u, u);
```

给出在32位计算机中上述程序段的输出结果并分析原因。

解：

在32位计算机中，上述程序段的输出结果如下：

x＝4294967295＝-1；

u＝2147483648＝-2147483648

- 对于第一个输出语句，变量 x 的值为 -1，因为它被声明为有符号整数。在第一个 %u 中，我们尝试以无符号十进制形式打印 x 的值，因此 -1 被解释为无符号整数，其对应的二进制值是 1111 1111 1111 1111 1111 1111 1111 1111。这个二进制值被解释为无符号整数的十进制值是 4294967295，因此输出结果是 x=4294967295。在第二个 %d 中，我们以有符号十进制形式打印 x 的值，因此 -1 被打印为 -1，输出结果是 x=-1。
- 对于第二个输出语句，变量 u 的值为 2147483648，因为它被声明为无符号整数。在第一个 %u 中，我们尝试以无符号十进制形式打印 u 的值，因此 2147483648 被解释为无符号整数，其对应的二进制值是 1000 0000 0000 0000 0000 0000 0000 0000。这个二进制值被解释为无符号整数的十进制值是 2147483648，因此输出结果是 u=2147483648。在第二个 %d 中，我们以有符号十进制形式打印 u 的值，但是因为 u 是无符号整数，2147483648 被解释为无符号整数，其对应的二进制值是 1000 0000 0000 0000 0000 0000 0000 0000，它被解释为有符号整数的十进制值是 -2147483648（在32位计算机中，有符号整数的范围是 -2^31 到 2^31-1）。因此，输出结果是 u=2147483648=-2147483648。

2.7

2.7分析下列几种情况下所能表示的数据范围分别是多少。
(1) 16位无符号数;
(2) 16 位原码定点小数;
(3) 16位补码定点小数;
(4) 16位补码定点整数。

定点格式，即约定机器中所有数据的小数点位置是固定不变的。在计算机中通常采用两种简单的约定：将小数点的位置固定在数据的最高位之前，或者是固定在最低位之后。一般常称前者为定点小数，后者为定点整数。

定点小数是纯小数，约定的小数点位置在符号位之后、有效数值部分最高位之前。若数据x的形式为x=x0.x1x2⋯xn(其中x0为符号位，x1～xn是数值的有效部分，也称为尾数，x1为最高有效位)

1) 16 位无符号数：0 ~ 1111 1111 1111 1111，即 0 ~ $2^{16}-1$＝65535
2) 16 位原码定点小数：1.111 1111 1111 1111 ~ 0.111 1111 1111 1111，即$-(1-2^{-15})$~ $1-2^{-15}$
3) 16 位补码定点小数：1.000 0000 0000 0000 ~ 0.111 1111 1111 1111，即-1 ~ $1-2^{-15}$
4) 16 位补码定点整数：1000 0000 0000 0000 ~ 0111 1111 1111 1111，即-2^{15} ~ $2^{15}-1$

2.9

2.9用IEEE754 32位单精度浮点数标准表示下列十进制数。
(1) -6(5/8) (2) 3.1415927 (3) 64000
解：
（1）首先分别将整数和分数部分转换成二进制数:
6(5/8) = 110.101
移动小数点,使其变成 1.M 的形式
1.10101 × 2^2
于是得到: s=1,e=2,E=10+01111111=100 0000 1, M=101 0100……
1100 0000 1101 0100 0000 0000 0000 0000＝（C0D40000）H
（2）首先分别将整数和分数部分转换成二进制数:
3.1415927 = 11.001001000011111110110101
移动小数点,使其变成 1.M 的形式
1.1001001000011111110110101 × 2^1
于是得到: s=0, e=1, E=1+01111111=100 0000 0, M=10010010000111111011010
拼接：0100 0000 0100 1001 0000 1111 1101 1011＝（40490FDB）H
（3）首先将 64000 转换成二进制数:
64000= 1111101000000000
移动小数点,使其变成 1.M 的形式
1.111101000000000 × 2^15
于是得到: S=0, e = 15 , E= 1111+01111111 = 100 0111 0, M = 111 1010 0……
0100 0111 0111 1010 0000 0000 0000 0000＝(477A0000) H

2.10求与单精度浮点数43940000H对应的十进制数。

0100 0011 1001 0100 0000 0000 0000 0000

S = 0，E=1000 0111=135，e=8，M=0010 1

1.00101 × 2^8 = 100101000，即296

2.10

插入一段文本插入

2.13设二进制浮点数的阶码为3位.尾数为7位。用模2补码写出它们所能表示的最大正数、最小正数、最大负数和最小负数. 并将它们转换成十进制数。

3位阶码范围为（-4~3）

	阶码（模2补码）	尾码（模2补码）	十进制数
最大正数	011	0.111111	$2^3x(1-2^{-6})$
最小正数	100	0.000001	$2^{-4}x2^{-6}$
最大负数	100	1.111111	$-2^{-4}x2^{-6}$
最小负数	011	1.000000	-2^3

2.13

插入一段文本插入

2.16 由6个字符的7位ASCII字符排列，再加上水平和垂直偶校验位构成表2.27所示的行列结构(最后一列HP为水平奇偶校验位，最后一行VP为垂直奇偶校验位)。

表 2.27 ASCII 交叉校验

字符	7 位 ASCII 字符							HP
3	0	X_1	X_2	0	0	1	1	0
Y_1	1	0	0	1	0	0	X_3	1
+	X_4	1	0	1	0	1	1	0
Y_2	0	1	X_5	X_6	1	1	1	1
D	1	0	0	X_7	1	0	X_8	0
=	0	X_9	1	1	1	X_{10}	1	1
VP	0	0	1	1	1	X_{11}	1	X_{12}

则X_1、X_2、X_3、X_4处的比特分别为1110；X_5、X_6、X_7、X_8;处的比特分别为1000；X_9、X_{10}、X_{11}、X_{12}处的比特分别为1011；Y_1和Y_2处的字符分别为 I 和 7（根据ASCII码表和偶校验定义计算）

2.16

插入一段文本插入

2.17

2.17设8 位有效信息为01101110，试写出它的海明校验码。给出过程，说明分组检测方式，并给出指错字及共逻辑表达式。如果接收方收到的有效信息变01101111,说明如何定位错误并纠正错误。

解：

发送端：

根据海明码的规则，对于 8 位有效信息，需要找到最小的 r，使得 8 + r <= 2^r - 1。可以得到 r = 4，即需要添加 4 个校验位。以下是构造海明码的过程：

1. 将 8 位有效信息进行编号，从右到左，从 1 到 8。将校验位编号为 1，2，4，8。将编号转化成二进制形式，准备插入。
2. 海明校验码放在索引号为2^n的位（n=0,1,2,...,k-1）上，本题中r=4，所以校验位的索引为第1，2，4，8位，于是在下表中把这几位空出来

索引号	1	2	3	4	5	6	7	8	9	10	11	12
	H1	H2	0	H4	1	1	0	H8	1	1	1	0

2.17

发送端:

3. 列出进制转换表:

索引号	8（2^3）	4（2^2）	2（2^1）	1（2^0）
3	0	0	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

索引号	1	2	3	4	5	6	7	8	9	10	11	12
	H1	H2	0	H4	1	1	0	H8	1	1	1	0

上表中，先说每一行的内容：从第二行开始，每一行的第一列代表索引号，这个索引号是除去了海明校验位之外的其他所有位。后面几列为该索引号对应的二进制表示，其位数取决于第（1）步计算得出的海明校验码的位数，比如第二行，索引号是3，十进制3对应的二进制就是0011，之所以用4位表示是因为这段信息码需要4个海明校验位。

再看列信息：第一行最右边数字1所对应的列里，出现1的，就表示可以用第H1位完成校验，出现数字0则表示不能用H1位进行校验，因此，由上表可知：

校验位H1负责校验：第3，5，7，9，11位（上表黄色高亮显示部分），对应位置上的值进行异或得：$0\oplus1\oplus0\oplus1\oplus1=1$，由于海明校验做的是偶校验，则H1=1；

校验位H2负责校验：第3，6，7，10，11位（上表蓝色高亮显示部分），对应位置上的值进行异或得：$0\oplus1\oplus0\oplus1\oplus1=1$；

校验位H4负责校验：第5，6，7，12位，对应位置上的值进行异或得：$1\oplus1\oplus0\oplus0=0$；

校验位H8负责校验：第9，10，11，12位，对应位置上的值进行异或得：$1\oplus1\oplus1\oplus0=1$。

2.17

得到最终要传输的数据串为

索引号	1	2	3	4	5	6	7	8	9	10	11	12
	1	1	0	0	1	1	0	1	1	1	1	0

接收端：

1. 进行校验：

索引号	1	2	3	4	5	6	7	8	9	10	11	12
	1	1	0	0	1	1	0	1	1	1	1	1

若此时接收方收到的数据相比源数据，在第12位发生了错误：
接收方先按照上一步中类似的方式进行计算检错字，区别在于要加上校验位自身这一位，即：
G1的值为第H1，3，5，7，9，11位上的值进行异或，计算结果为0
G2的值为第H2，3，6，7，10，11位上的值进行异或，计算结果为0
G4的值为第H4，5，6，7，12位上的值进行异或，计算结果为1
G8的值为第H8，9，10，11，12位上的值进行异或，计算结果为1

2. 错误位判定：

由于海明校验采取的是偶校验，所以判断出G1,G2监督式包含的数据位无错，错误位发生在G4,G8两个监督式包含的位上。
此时的做法是：①找到G4,G8这三个监督式共同包含的位；②找出共同包含的位之后，再剔除掉在G1,G2中出现过的位（因为已经验证过监督式中的位是正确的）；③剩下的位就是发生传输错误的位。
也可以根据指错码$G_4G_3G_2G_1$＝1100＝12,根据上述步骤可以得出，出错的位就是第12位，此时取反即可。

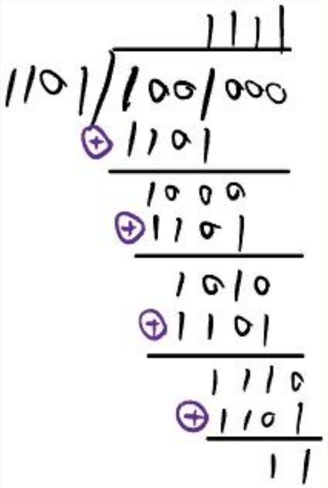
2.18

2.18设要采用CRC码传送数据信息x=1001，当生成多项式为G(x)=1101时，请写出它的循环冗余校验码。若接收方收到的数据信息为x'=1101,说明如何定位错误并纠正错误。

解：

求校验码：

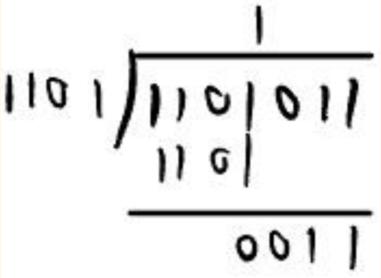
- 原始数据是：x=1001
- 生成待追加的校验码，需要使用一个生成多项式，G(x)(收发双方事先约定)，本题中 $G(x) = x^3 + x^2 + (0 * x^1) + x^0$（生成多项式的常数项必须是1）
- 构造被除数：原始数据 + 生成多项式最高次项个0，即：1001000
- 除数：除数实际上就是生成多项式的系数，G(x) 展开得到: $G(x) = 1 * x^3 + 1 * x^2 + 0 * x^1 + 1 * x^0$，即 1101
- 两数相除得余数，并进行补位(补到与生成多项式最高次项一致)，即得到校验码。但是，这里的除法跟常规除法并不相同
- 常规除法在上下两行数进行运算时，使用的是减法运算，而这里使用的是异或

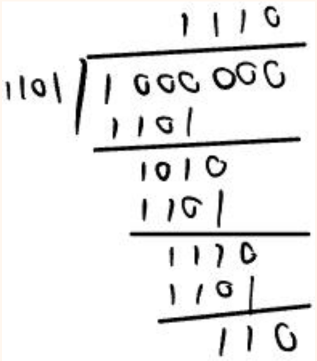


- 通过上述运算，得到校验码 011，添加到原始数据之后，得到的最终发送数据为 1001011

2.18

厦門大學 XIAMEN UNIVERSITY





接收方对收到的数据进行校验：

假设收到的数据为1101011

接收方对该数据做除法，除数仍然是之前使用的多项式的系数 1101

过程如下：

若余数为0，则表示被整除未出错，但这里出现了余数011，说明出现了错误。

注：对于CRC一位纠错性能的介绍，课本上语焉不详，只是说“可以利用CRC的编码特性设计组合逻辑电路来进行纠错”，并没有介绍如何纠错，在本题我研究了具体的做法，查找了具体做法的资料，很有意思，在此介绍。

现在开始纠错：

事实上，计算机不知道余数011对应哪一位出错了，计算机只知道第一位出错的余数是'?'，它的计算方法是： ?=2^{r+k-1}%G(x)现在知道余数为2^6%1001B=110

情况说明	校验码当前情况	运算	余数
发现出错，开始纠错	1101011	1101011 与 G(X)模二除	011
校验码循环左移，同时余数 011 补 0，成为 0110，与 G(X) 模二除	1010111	0110 与 G(X)模二除	110
发现当前余数位 110，已知 110 代表第一位出错，把第一位与 1 进行取反，纠错			

所以最后我们得到是第2位出错，将左侧起第 2 位的取反即可。

厦門大學
XIAMEN UNIVERSITY

ABOUT

感谢观看

THANKS FOR WATCHING

---汇 报 人： 黄勖