# 数字逻辑课程作业1----陈鹏宇20204227

## 一.计算机是如何处理（编码）中文汉字的？

1.首先用户从键盘上键入汉字，创造汉字输入码（外码），例如数字编码（区位码），拼音编码（搜狗，微软拼音输入法等），字形编码（五笔等）

2.其次，汉字外码先转换为国标码（GB2312），再转换为汉字机内码。经查询 区位码先转换成十六进制数表示，加上2020H变成国标码，再加上8080H变为机内码（为了和ASCII中的英文区分）。

除了国标码之外还有其他汉字编码如GBK，UNICODE，utf。

3.通过机内码在字库里面查找对应输出码，输出码可以提供输出汉字时所需要的汉字字形。而在屏幕中输出的汉字其实是以可编程点阵储存的，每一个点用一位二进制表示。计算机通过汉字内码在汉字字库中找到唯一匹配的字形码，实现其转换

## 二.如何仅用加法实现除法？

理论实现;利用补码加法，用被除数减除数，减一次，就在那个地方加个，直到被除数小于除数，那么在那个地方放的数就是商，剩下的被除数就是余数。

代码实现（基于c++）

#include <bits/stdc++.h>

using namespace std;

bool jinwei = true;

int result1 = 0; //商

int result2 = 0; //余数

void add(deque<int> &d1 , deque<int> &d2)

{

jinwei = false;

int temp = d1.size();

for(int i = temp - 1 ; i > 0 ; i--)

{

d1[i] += d2[i];

if(d1[i] == 2)

{

d1[i] = 0;

d1[i-1] += 1;

}

else if(d1[i] == 3)

{

d1[i] = 1;

d1[i-1] += 1;

}

}

d1[0] += d2[0];

if(d1[0] == 2)

{

jinwei = true;

d1[0] = 0;

result1++;

}

else if(d1[0] == 3)

{

jinwei = true;

d1[0] = 1;

result1++;

}

}

void bu(deque<int> &d) //求补码

{

int n = d.size();

for(int i = 0 ; i < n ; i++) //按位取反

{

if(d[i] == 1)

d[i] = 0;

else

d[i] = 1;

}

d[n-1] += 1;

for(int i = n-1 ; i > 0 ; i--) //末尾加一

{

if(d[i] == 2)

{

d[i] -= 2;

d[i-1] += 1;

}

}

if(d[0] == 2)

d[0] = 0;

}

int main()

{

deque<int> bin1;//被除数

deque<int> bin2;//除数

string b1 , b2 ; cin >> b1 >> b2;

int n1 , n2;

n1 = b1.size() ; n2 = b2.size();

for(int i = 0 ; i < n1 ; i++)

{

bin1.push\_back(b1[i]-48);

}

for(int i = 0 ; i < n2 ; i++)

{

bin2.push\_back(b2[i]-48);

}

if(n1 > n2)

for(int i = 0 ; i < n1-n2 ; i++){

bin2.push\_front(0); //除数补齐

n2++;

}

bu(bin2);

while(jinwei)

add(bin1,bin2);

cout<<"商为"<<result1<<endl;

bu(bin2);

add(bin1,bin2);

for(int i = 0 , j = n1-1 ; i < n1 && j >= 0 ; i++ , j--)

{

result2 += bin1[j]\*pow(2,i);

}

cout<<"余为"<<result2<<endl;

}

文本

低可信度描述已自动生成

## 三.如何判断二进制补码加法运算中的进位和溢出？

我们令A表示符号位是否进位，B表示最高数值位是否进位。

若A⊕B为真，则发生了溢出，否则运算进位。

例

[64]补 0100 0000

[-64]补 + 1100 0000

[0]补 =10000 0000

A = 1 B = 1 A⊕B = 0 进位，运算正确

[64]补 0100 0000

[64]补 + 0100 0000

[128]补 = 1000 0000

A = 0 B = 1 A⊕B = 1 溢出，运算错误

负数加正数和负数加负数以此类推。简单的判断就是若相同符号的数相加后，得到的数的符号位不同，则发生溢出，反之进位。