****

计算机原理与汇编语言程序设计实验报告

|  |  |
| --- | --- |
| 题 目： | 输出素数和斐波那契数列 |
| 学生姓名： |  |
| 指导教师： | 雷向东 |
| 学 院： |  |
| 专业班级： |  |
| 学生学号： |  |

本科生院制

2020年12月

目录

实验一 输出10万以内的素数3

1.1 目的与要求3

1.2 操作环境3

1.3 实验内容和思路3

1.4 实验结果5

实验二 输出前50位斐波那契数列9

2.1 目的与要求9

2.2 操作环境9

2.3 实验内容和思路9

2.4 实验结果12

实验总结13

附录14

参考文献20

实验一 输出10万以内的素数

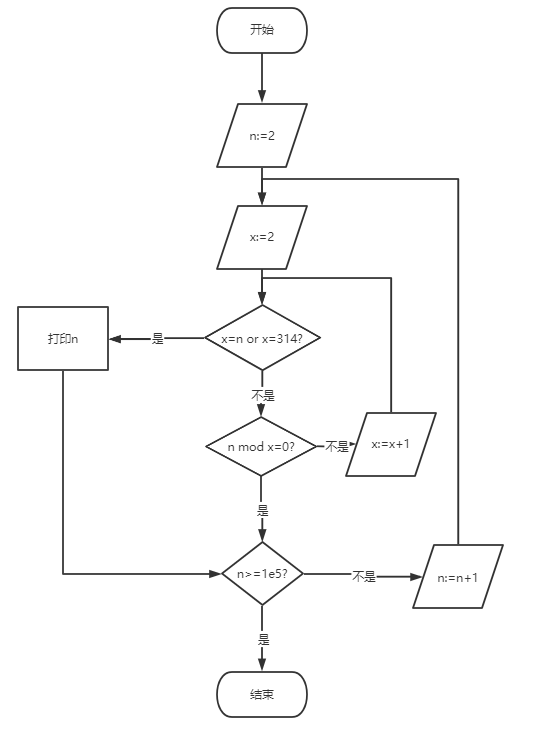
**1.1 目的与要求**

利用汇编语言输出10万以内的素数

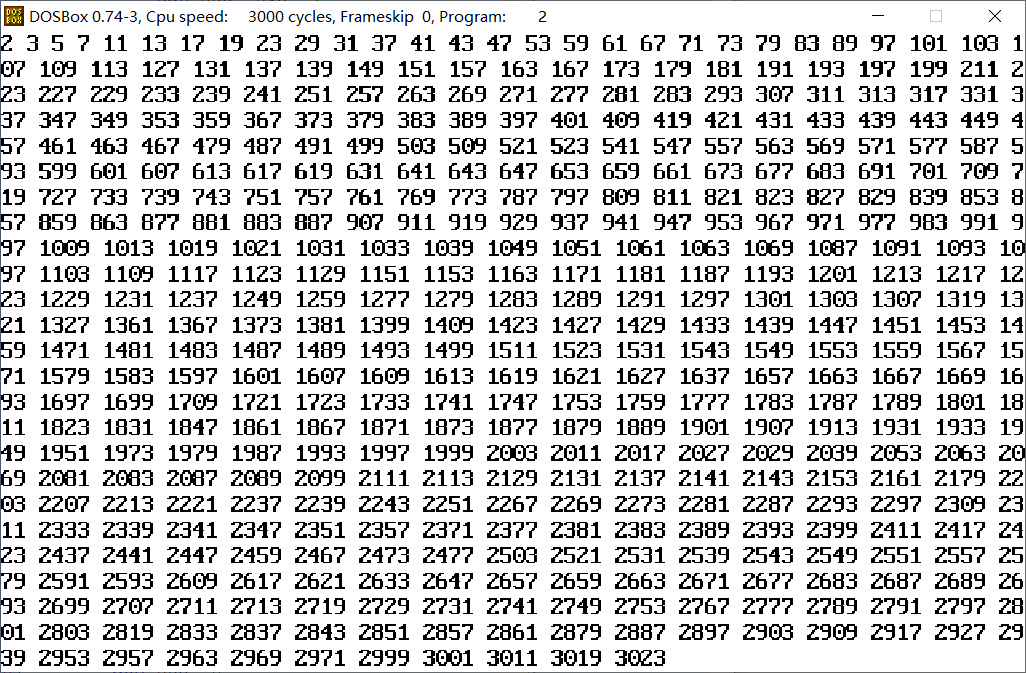
**1.2 操作环境**

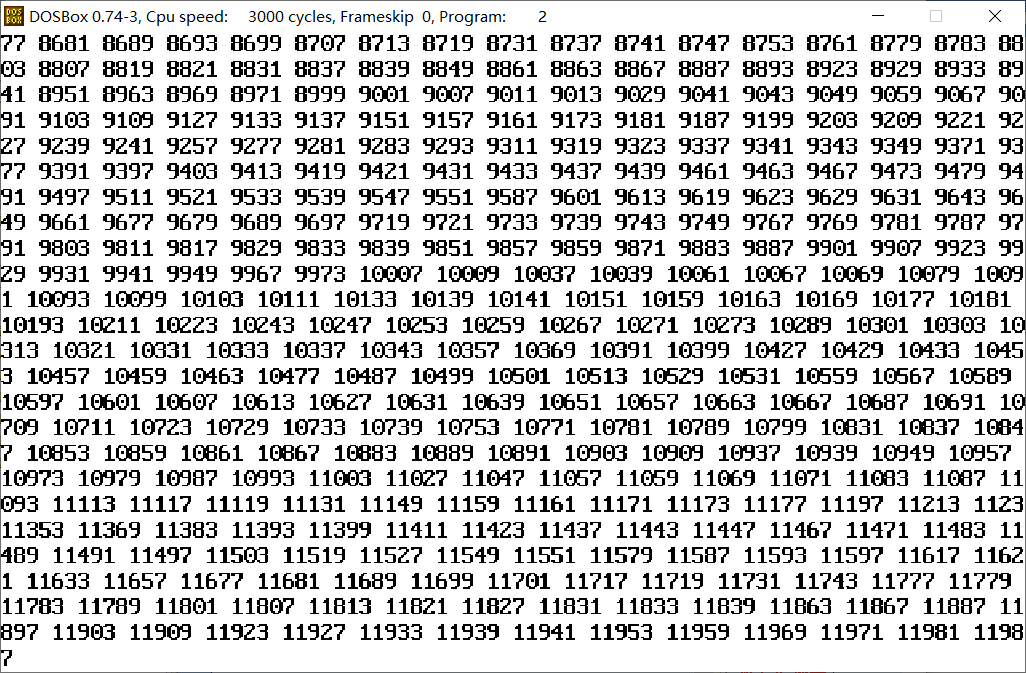
Masm for windows 集成开发环境

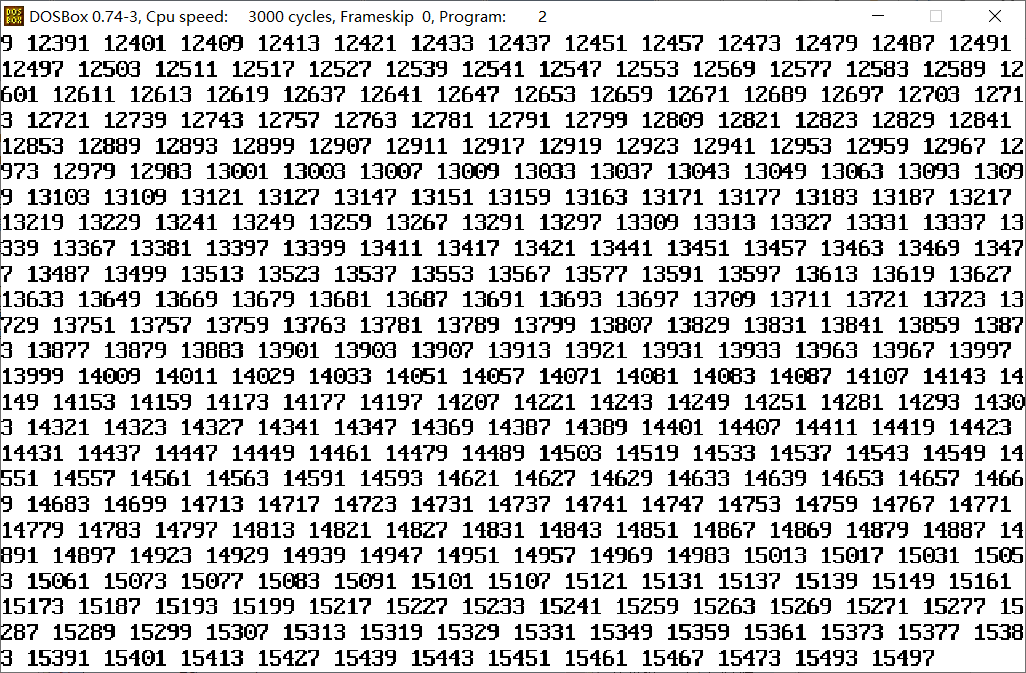
**1.3 实验内容与思路**

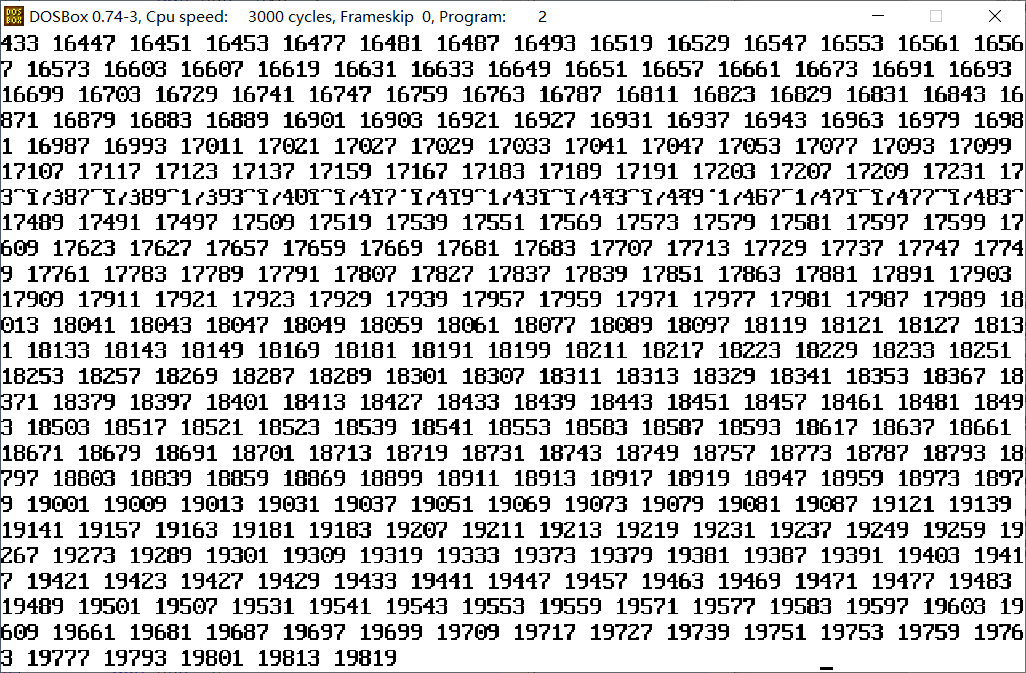
考虑效率需求，筛素数很容易联想到使用线性筛。但是同时考虑到内存不便于开大以及数据范围不是很大(1e5内)，故判断没有效率需求。  
本程序的实现思路是直接枚举每个因子暴力试除。如果存在一个大于1小于待验证的数能整除它，那么这个数就不是素数，跳过。否则一直筛到因子等于待验证数都没有筛掉它，那么这个数就是素数，进行输出即可。  
而实际运行时如果采用上述方法会非常慢，故考虑优化。  
一个优化效率的方法是，我们只需要筛到即可输出素数，以下附上简单证明：  
设待判断数为n，它能被一个大于且小于n的整数x整除。  
那么设整除结果为y，有:  
即：  
而：  
因为：  
所以：  
即：  
所以当我们在枚举因子试除n时，如果n是一个合数，直到时一定已经存在一个因子y能把n整除。那么反之如果直到都没有一个数能整除n，n就是一个质数。  
使用这个优化，可以把原本判断一个素数的时间复杂度降为。从而算法总时间复杂度变为。  
，313是满足小于317条件的最大的质数，故只需要筛到313即可。  
代码实现中通过将314与EBX比较作为上限，在标签圈定代码块***div\_try***中试除来实现素数筛。满足条件的素数会跳入***got\_prime***分支实行输出流程，不满足条件的会跳出循环块继续往后找数。

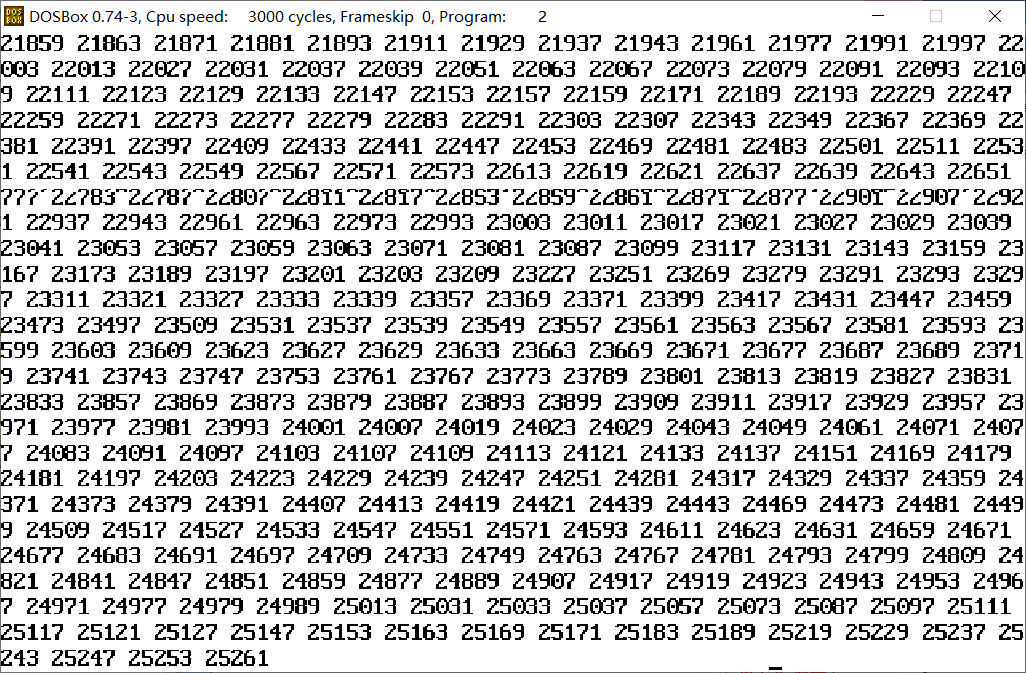
**1.4 实验结果**

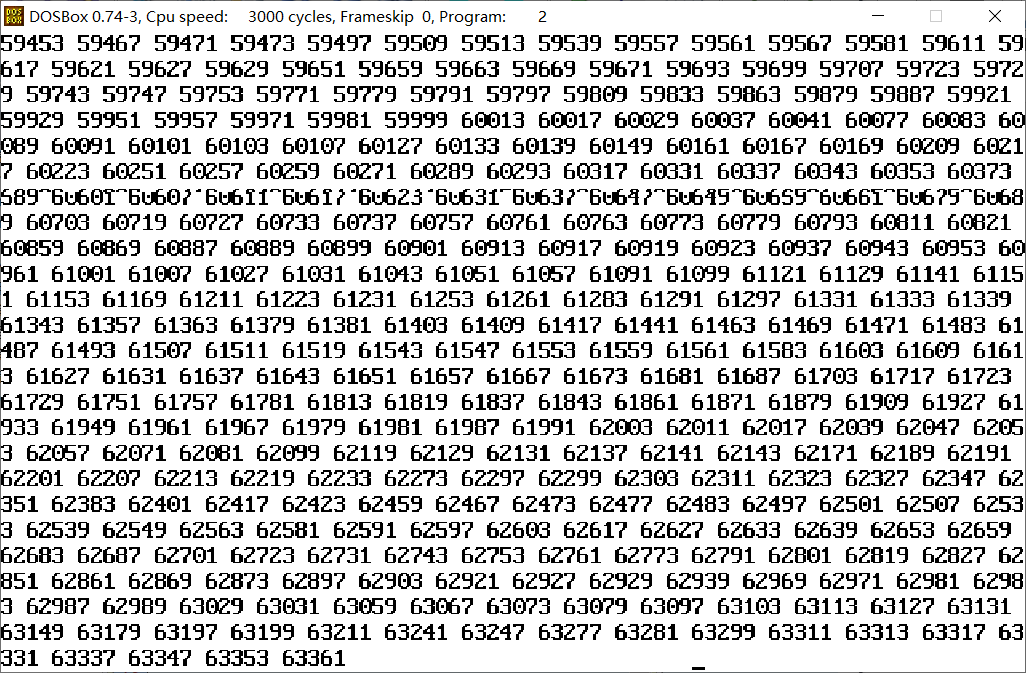


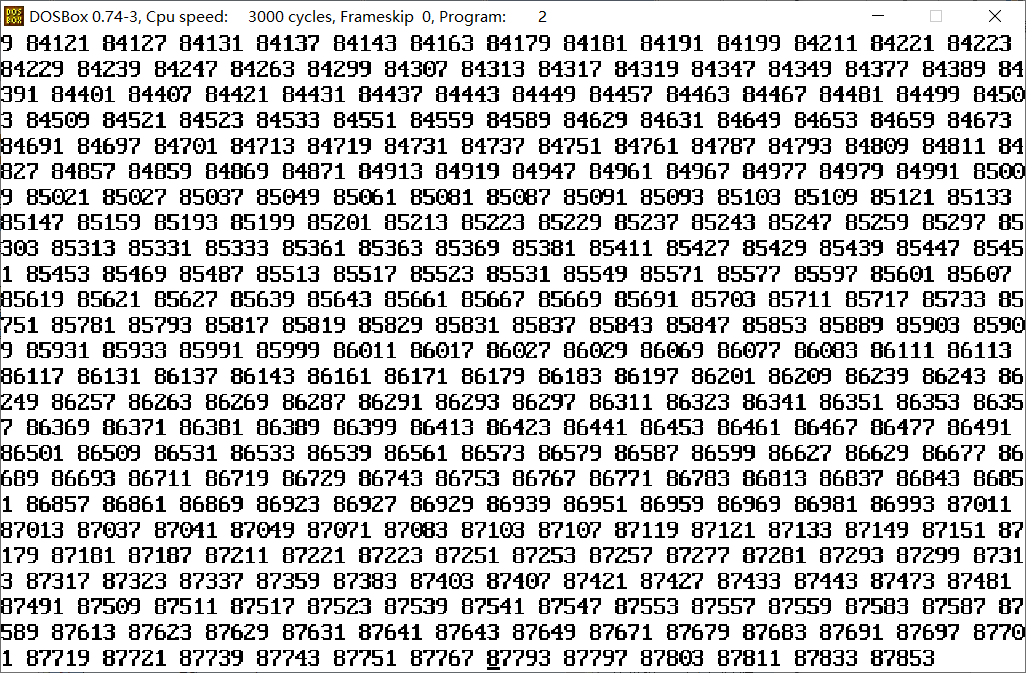


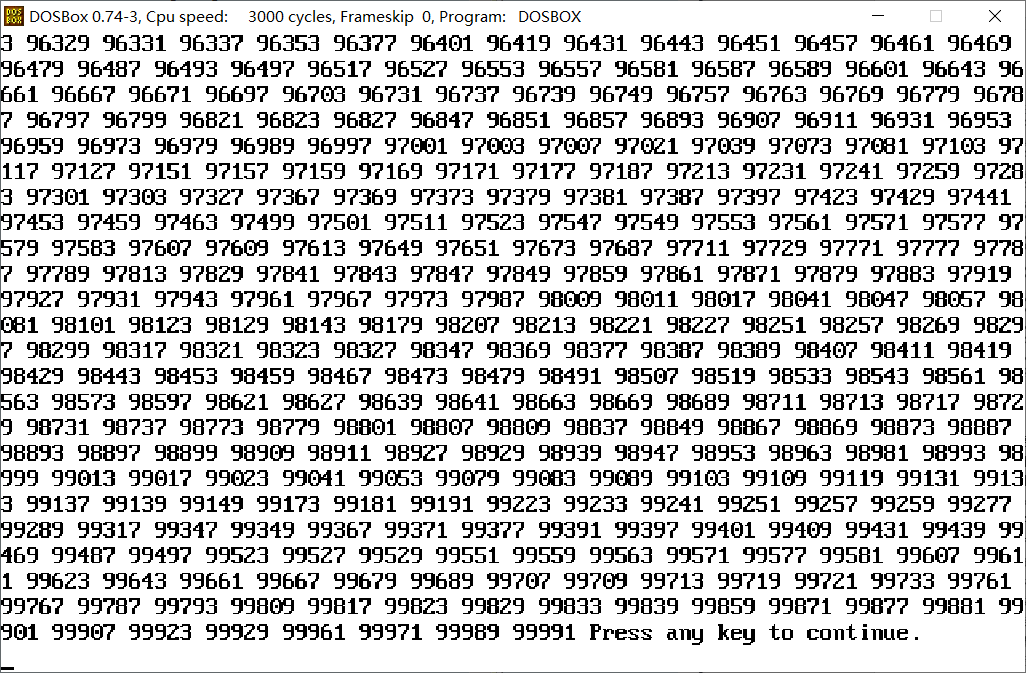












实验二 输出前50位斐波那契数列

**2.1 目的与要求**

利用汇编语言输出前50位斐波那契数列

**2.2 操作环境**

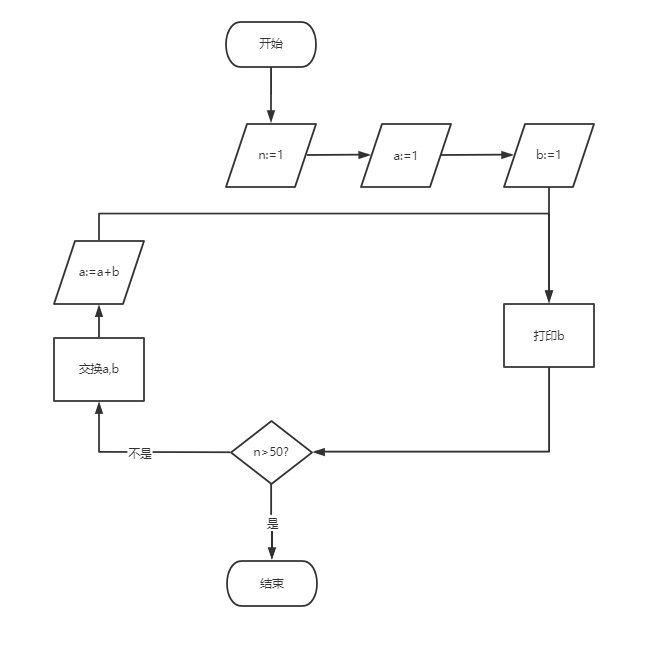
Masm for windows 集成开发环境

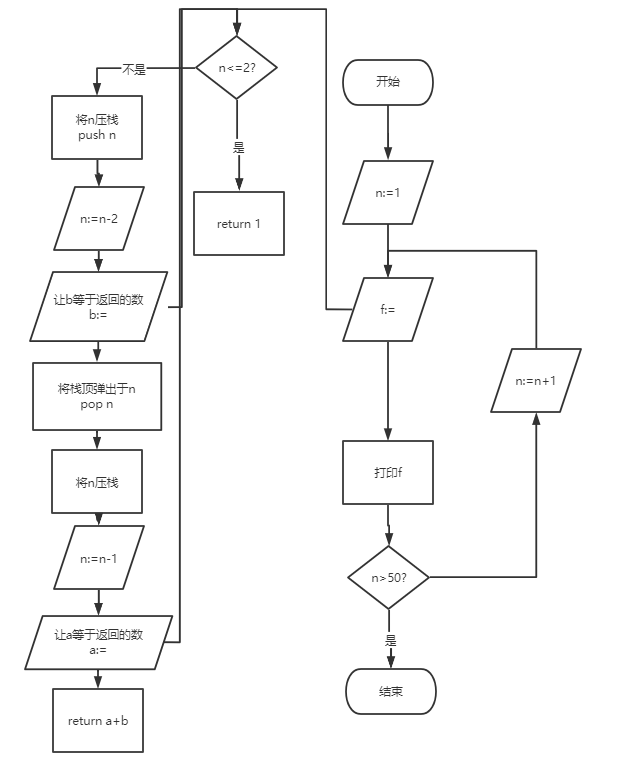
**2.3 实验内容与思路**

**非递归打印斐波那契数列前50项**  
非递归的思路很简单，使用两个寄存器存当前递推到的值，再将他们求和，把大的转移到其中一个寄存器中，再把和放入另外一个寄存器中，然后循环如上流程并输出结果即可。  
但实际操作时遇到了溢出问题，32位寄存器最多只能表示0xffffffff的整数，而斐波那契数列的第46项为0x11e8d0a40，已经超过了32位寄存器所能表示的范围，故之后的运算会造成溢出，使得输出结果不正确。故仅使用2个32位寄存器存中间结果并不能解决问题，需要额外的空间存取中间结果。  
本程序中使用新开的双字内存块来解决额外空间的问题。其中DS:[4]和DS:[12]用来装前两项斐波那契数列的高位，DS:[0]和DS:[8]用来装低位。由于只使用两位即可完成斐波那契数列前50项，故可直接先让高位相加，再让低位相加，再考虑进位，而无需逐个低位相加计算进位值再往高位加。考虑到十进制输出的便利性，高位是采用了1e9作为基而并非其他数字，这样的设计方便直接打印高位后连接低位，不用考虑进制转换。  
如上设计，最多可以打印到斐波那契数列的第91项而不发生溢出。

**递归打印斐波那契数列前50项**  
递归实现打印斐波那契数列的主要思想是，将当前待求问题转换成两个子问题：求当前数-1的斐波那契数列值和当前数-2的斐波那契数列值，再将他们相加，再打印结果。由于递归实现，不使用记忆化搜索，故效率不会很高。其核心基于以下公式：  
其递归边界是n=1和n=2，此时：  
具体实现程序中通过fib代码块进行递归。fib1是边界条件，通过ECX传参，判断ECX等于1的时候将两中间结果内存位置装入1，并向上返回，回溯相加最后得到结果。

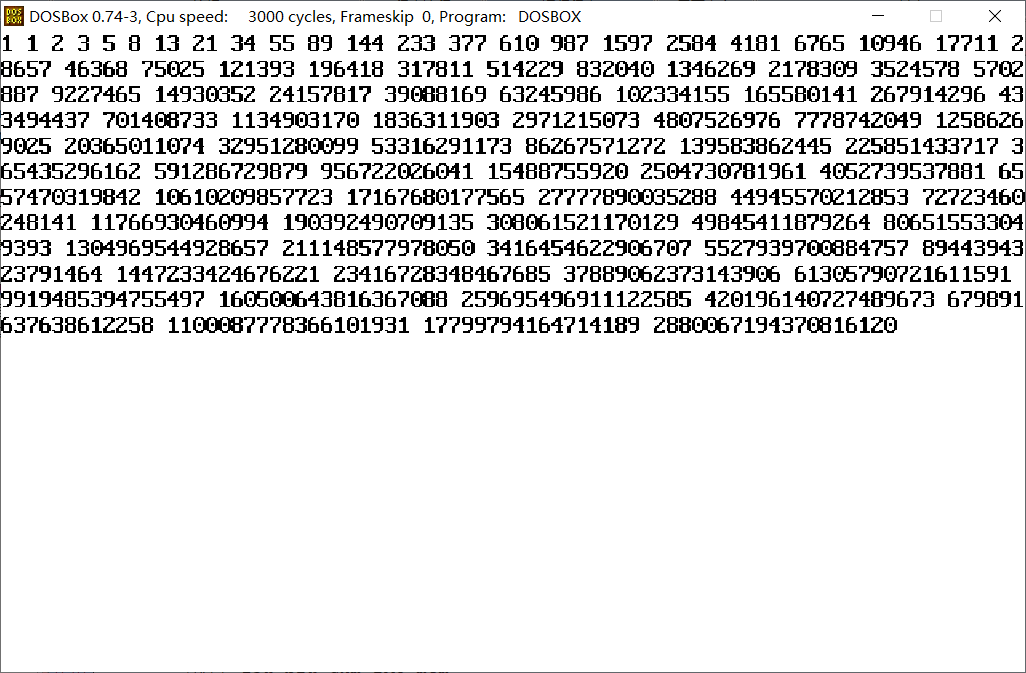
**非递归：**



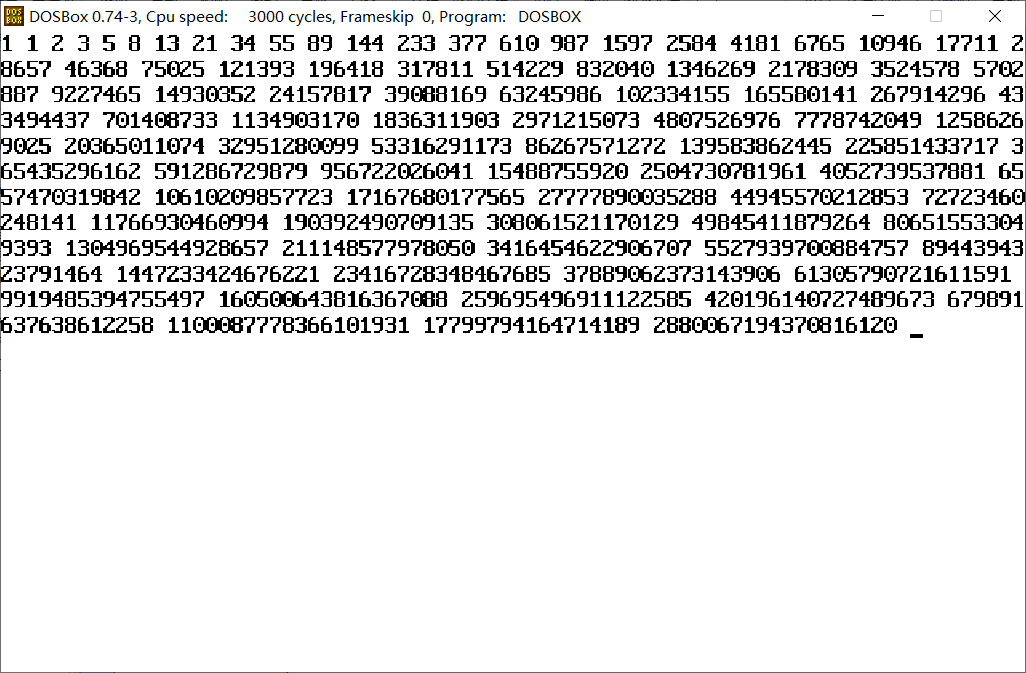
**递归：**

**2.4 实验结果**

递归



递推



实验总结

附录

输出10万以内的素数

.MODEL HUGE ;100000以内的素数

.STACK 16384

.CODE

MAIN PROC FAR

.386

MOV EAX,1

while\_true:

INC EAX

MOV EBX, 2

CMP EAX, 100000

JNC done

while\_true\_2:

CMP EBX, EAX ;<

JNC got\_prime

CMP EBX, 314

JNC got\_prime

div\_try:

PUSH EAX

MOV EDX,0

DIV EBX

POP EAX

CMP EDX,0

JZ while\_true

INC EBX

JMP while\_true\_2

got\_prime:

MOV ECX, EAX

PUSH EAX

CALL far ptr fun\_output

back1:

CALL far ptr print\_space

back2:

POP EAX

JMP while\_true

print\_space:

MOV EDX,20H

MOV EAX,200H

INT 21H ;输出空格

RET

fun\_output: ; ECX是待打印

CMP ECX, 10 ; 后者大，CF=1，相等则ZF=1

JC print

MOV EAX, ECX

MOV EBX, 10

MOV EDX, 0

DIV EBX

MOV ECX, EAX

PUSH EDX

CALL far ptr fun\_output

POP ECX

print:

MOV EDX,ECX

ADD EDX,30H

MOV EAX,200H

INT 21H

RET

done:

MOV EAX,4C00H

INT 21H

MAIN ENDP

END

递归实现前50位斐波那契数列

.model HUGE

.386

.STACK 16384

.DATA

dd 50 dup(0)

.CODE

MAIN PROC FAR

; MOV EAX, 233

MOV ECX, 1

CALL far ptr fun\_output

CALL far ptr print\_space

MOV ECX, 1

CALL far ptr fun\_output

CALL far ptr print\_space

MOV EAX, 1

MOV DS:[0], EAX

MOV DS:[8], EAX

MOV EAX, 0

MOV DS:[4], EAX

MOV DS:[12], EAX

MOV EAX, DS:[0]

MOV EDX, 1

MOV ECX, 3

while\_true:

CMP ECX, 91 ; 两位精度（第二位权值为1e9）最大只能打到90位

JNC done

PUSH ECX

CALL far ptr fib

POP ECX

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

; PUSHA

MOV ECX, DS:[4]

CMP ECX, 0 ;不打前导零

JZ lowwer

CALL far ptr fun\_output

lowwer:

MOV ECX, DS:[0]

CALL far ptr fun\_output

CALL far ptr print\_space ; 打空格

; POPA

POP EDX

POP ECX

POP EBX

POP EAX

INC ECX

JMP while\_true

; 这里往下是函数区，不建议直接走

fib: ; 生成fib(ECX)的值，存在DS:[0-4]里

CMP ECX, 3

JC fib1

PUSH ECX

SUB ECX, 2

CALL far ptr fib

MOV ECX, DS:[0]

MOV DS:[8], ECX

MOV ECX, DS:[4]

MOV DS:[12], ECX

POP ECX

SUB ECX, 1

PUSH ECX

CALL far ptr fib

POP ECX

CALL far ptr sum\_two\_mem

RET

fib1:

MOV ECX, 1

MOV DS:[0], ECX

MOV ECX, 0

MOV DS:[4], ECX

RET

sum\_two\_mem:

MOV EAX, DS:[4]; 对位相加，先加高位

ADD EAX, DS:[12]

XCHG EAX, DS:[4]

MOV DS:[12], EAX

MOV EAX, DS:[0]; 再加低位

MOV EBX, DS:[0]

ADD EAX, DS:[8]

MOV DS:[8], EBX

MOV EBX, 1000000000

MOV EDX, 0

DIV EBX

MOV DS:[0], EDX

ADD DS:[4], EAX ; 进位

RET

print\_space:

PUSH EDX

PUSH EAX

MOV EDX,20H

MOV EAX,200H

INT 21H ;输出空格

POP EAX

POP EDX

RET

fun\_output: ;打印ECX的值，用前最好push一下各个寄存器

CMP ECX, 10 ; 后者大，CF=1，相等则ZF=1

JC print

MOV EAX, ECX

MOV EBX, 10

MOV EDX, 0

DIV EBX

MOV ECX, EAX

PUSH EDX

CALL far ptr fun\_output

POP ECX

print:

MOV EDX,ECX

ADD EDX,30H

MOV EAX,200H

INT 21H

RET

done:

MOV EAX,4C00H

INT 21H

MAIN ENDP

END

非递归实现50位斐波那契数列

.model HUGE

.386

.STACK 16384

.DATA

dd 50 dup(0)

.CODE

MAIN PROC FAR

; MOV EAX, 233

MOV ECX, 1

CALL far ptr fun\_output

CALL far ptr print\_space

MOV EAX, 1

MOV DS:[0], EAX

MOV DS:[8], EAX

MOV EAX, 0

MOV DS:[4], EAX

MOV DS:[12], EAX

MOV EAX, DS:[0]

MOV EDX, 1

MOV ECX, 2

while\_true:

CMP ECX, 51 ; 两位精度（第二位权值为1e9）最大只能打到90位

JNC done

MOV EAX, DS:[4]; 对位相加，先加高位

ADD EAX, DS:[12]

XCHG EAX, DS:[4]

MOV DS:[12], EAX

MOV EAX, DS:[0]; 再加低位

MOV EBX, DS:[0]

ADD EAX, DS:[8]

MOV DS:[8], EBX

MOV EBX, 1000000000

MOV EDX, 0

DIV EBX

MOV DS:[0], EDX

ADD DS:[4], EAX ; 进位

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

; PUSHA

MOV ECX, DS:[12]

CMP ECX, 0 ;不打前导零

JZ lowwer

CALL far ptr fun\_output

lowwer:

MOV ECX, DS:[8]

CALL far ptr fun\_output

CALL far ptr print\_space ; 打空格

; POPA

POP EDX

POP ECX

POP EBX

POP EAX

INC ECX

JMP while\_true

print\_space:

PUSH EDX

PUSH EAX

MOV EDX,20H

MOV EAX,200H

INT 21H ;输出空格

POP EAX

POP EDX

RET

fun\_output: ;打印ECX的值，用前最好push一下各个寄存器

CMP ECX, 10 ; 后者大，CF=1，相等则ZF=1

JC print

MOV EAX, ECX

MOV EBX, 10

MOV EDX, 0

DIV EBX

MOV ECX, EAX

PUSH EDX

CALL far ptr fun\_output

POP ECX

print:

MOV EDX,ECX

ADD EDX,30H

MOV EAX,200H

INT 21H

RET

done:

MOV EAX,4C00H

INT 21H

MAIN ENDP

END

参考文献

[1] 雷向东,雷振阳,龙军. 汇编语言程序设计[M].长沙:中南出版社,2019(1):8.