南京理工大学计算机科学与工程学院

程序设计基础II实验报告

班级： 9191062302

姓名： 吴启来

学号： 9191160D0139

指导教师： 娄震

1. 设计思想

利用迭代法计算斐波那契数列: F[i+2]=F[i+1]+F[i]

同时将所求求得的每一项的16进制保存在F2中

N>0

NO

YES

FIB函数，计算数列并保存

是否有溢出

YES

NO

Printhout函数，打印数列

exit,程序结束

1. 有注释的程序源代码

#########################################################

# 程序: fibonaqie.asm 程序员：吴启来 课程:程序设计基础II

# 最后修改时间:2021年4月14日

##########################################################程序功能描述：

#程序调用FIB和printout函数计算斐波那契数列前N项，并打印

##########################################################Main函数的寄存器使用:

# $a0:将N传递到函数中 $a1:传递数组地址保存数列

# $a1:将F数组地址传递到函数

# $t7:将F2数组地址传递到函数

#数据段

#########################################################

.data

N: .word 0 #输入的N

F: .space 5000 #存数低4个字节

F2: .space 5000 #存数高4个字节

str\_hex: .space 5000 #存放16进制数

str\_space: .asciiz "\t\t" #制表符分隔

str\_nextline: .asciiz "\n" #换行符

str\_input: .asciiz "please input N:" #输入提示

str\_output: .asciiz "Fibonacci is below: \n" #输出提示

str\_error: .asciiz "the data is overflow\n" #溢出报错

str\_warn: .asciiz "N must > 0 !\n" #输入报错

##########################################################文本段

#########################################################

.text

main: la $a0,str\_input #打印输入提示

li $v0,4

syscall

li $v0,5

syscall #输入N

blt $v0,$0,warn #输入的N小于0，则跳转报错

la $t0,N #加载N的地址

la $t6,F #加载F的地址

la $t7,F2 #加载F2的地址

la $s1,str\_hex #加载str\_hex地址

lw $t5,($t0)

move $a0,$v0 #t5存放N的值

jal FIB #调用FIB函数

jal printout #调用输出函数

exit: li $v0,10 #程序结束

syscall

warn: la $a0,str\_warn #打印输入错误字符串

li $v0,4

syscall

b exit #跳转到结束

error: la $a0,str\_error #打印溢出字符串

li $v0,4

syscall

b exit #跳转到结束

##########################################################函数名：FIB($N,$F,$F2)

##########################################################功能描述：

# 使用FIB函数采用迭代法计算斐波那契数列，并保存其16进制数

# 其中F2保存数据高4个字节, F保存数据低四个字节

# 支持64位运算

#########################################################

#调用示例:

#########################################################

#函数中使用的寄存器：

# $t1:存储当前两项的前一项数的低4个字节

# $t2:存储当前两项的后一项数的低4个字节

# $a1:存储当前两项的前一项数的高4个字节

# $a2:存储当前两项的后一项数的高4个字节

# $t4:存储循环变量

# $s1:保存数列16位16进制数

#########################################################

FIB: move $t5,$a0

li $t1,1 #数列第一项低位

li $t2,1 #数列第二项低位

li $a1,0 #数列第一项高位

li $a2,0 #数列第二项高位

li $t4,0 #循环计数器

li $s7,45 #存放32十进制数溢出判断序号

addi $t8,$0,-1 #t8存放MAX\_INT

loop1: sw $t1,($t6) #保存低4个字节

sw $a1,($t7) #保存高4个字节

b hexout #转换成16进制

this: addu $t3,$t1,$t2 #t3=t1+t2

addi $t8,$0,-1 #t8存放MAX\_INT

subu $t8,$t8,$t1 #t8=t8-t1

sltu $t9,$t8,$t2 #若有溢出则进位t9=1；否则t9=0

move $t1,$t2 #t1=t2 低位迭代

move $t2,$t3 #t2=t3

add $a3,$a1,$a2 #a3=a1+a2

addu $a3,$a3,$t9 #加上进位

beqz $a1,a #判断溢出

beqz $a2,a

addi $t8,$a2,1

blt $a3,$t8,error

a: move $a1,$a2 #a1=a2 高位迭代

move $a2,$a3 #a2=a3

addi $t4,$t4,1 #循环计数器累加一

addi $t6,$t6,4 #低位地址指向下一个字

addi $t7,$t7,4 #高位地址指向下一个字

blt $t4,$t5,loop1 #判断是否结束循环

jr $ra

hexout: move $s0,$t1 #传递参数

li $s2,8 #设置循环变量

addi $s3,$s1,17 #准备存放16位16进制数

loop2: andi $s4,$s0,0x0f #取低一个字节

srl $s0,$s0,4 #逻辑右移4位

bge $s4,10,char1 #>=10转换成字符

addi $s4,$s4,0x30 #s4<10 转换成数字

b put1

char1: addi $s4,$s4,0x37 #转换成字符

put1: sb $s4,($s3) #保存数据

addi $s3,$s3,-1 #高一位bit

addi $s2,$s2,-1 #循环变量-1

bgtz $s2,loop2 #判断循环是否结束

move $s0,$a1 #传递高位4个字节

li $s2,8 #设置循环变量

loop3: andi $s4,$s0,0x0f #取低一个字节

srl $s0,$s0,4 #逻辑右移4位

bge $s4,10,char2 #>=10转换成字符

addi $s4,$s4,0x30 #s4<10 转换成数字

b put2

char2: addi $s4,$s4,0x37 #转换成字符

put2: sb $s4,($s3) #保存数据

addi $s3,$s3,-1 #高一位bit

addi $s2,$s2,-1 #循环变量-1

bgtz $s2,loop3 #判断循环是否结束

li $s5,0x78 #转换结束

sb $s5,1($s1) #设置16进制前的x

li $s5,0x30

sb $s5,($s1) #设置16进制前的0

sb $0,18($s1) #16进制数末尾置“\0”

addi $s1,$s1,19 #地址指向保存下一个16进制数的地方

b this #跳回

##########################################################函数名：printout()

##########################################################功能描述:

#打印已保存的N项斐波那契数列的10进制以及16进制

#$s1保存16进制数地址,$t6保存32位下的数列10进制

########################################################

#函数中使用的寄存器：

# $t4:循环变量

#########################################################

printout: la $a0,str\_output #打印输出提示

li $v0,4

syscall

la $t6,F #加载F地址

la $t7,F2 #加载F2地址

li $t4,0 #循环变量

la $s1,str\_hex #加载存放16进制数的地址

loop4: addi $s6,$t4,1 #s6存放下标

move $a0,$s6 #打印下标

li $v0,1

syscall

la $a0,str\_space #打印空格，分隔

li $v0,4

syscall

blt $s7,$t4,out #判断32位是否溢出

Nout: lw $a0,($t6) #打印10进制的数列

li $v0,1

syscall

la $a0,str\_space #打印空格分隔

li $v0,4

syscall

out: move $a0,$s1 #打印16进制数

li $v0,4

syscall

la $a0,str\_nextline #下一行

li $v0,4

syscall

addi $t4,$t4,1 #循环变量+1

addi $t6,$t6,4 #下一字节

addi $t7,$t7,4 #下一字节

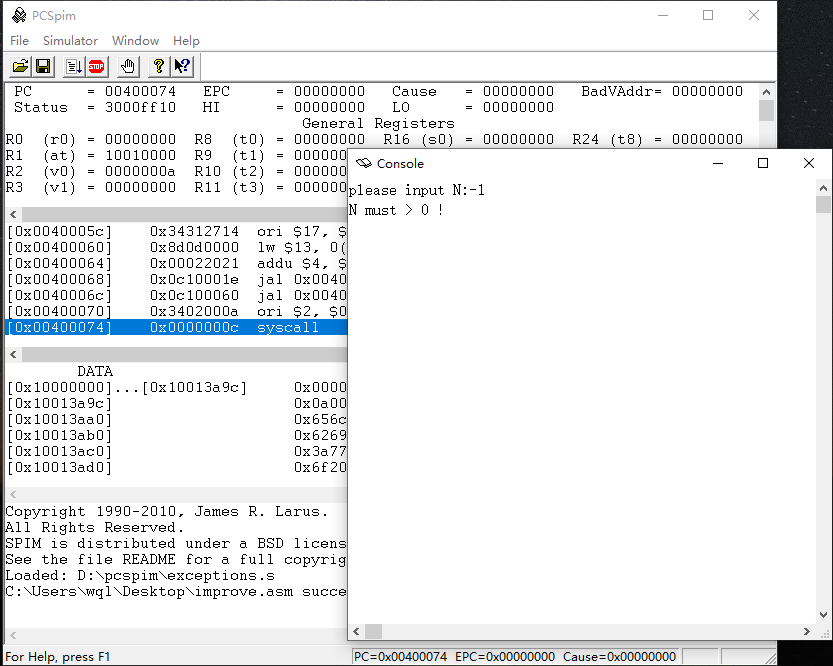
addi $s1,$s1,19 #保存下一个16进制的地址

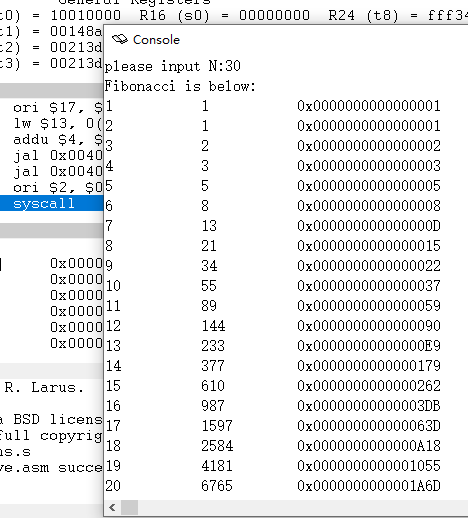
blt $t4,$t5,loop4 #判断循环是否结束

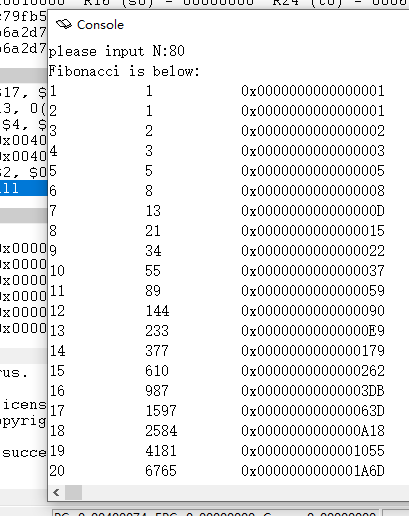
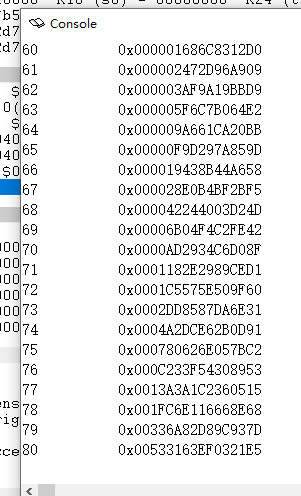
jr $ra

#########################################################

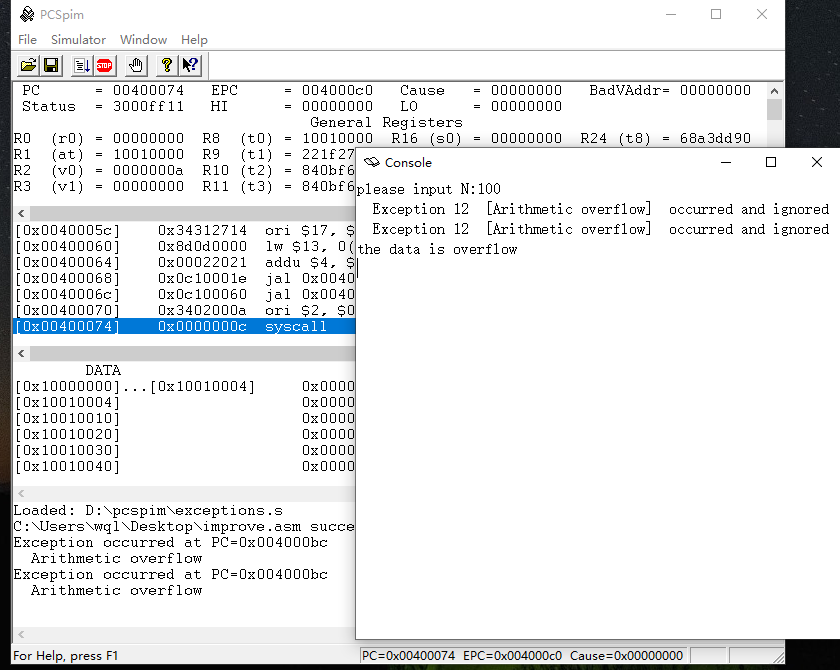
三：程序测试过程和结果

测试N<0 提示出错

测试N=30正常输出数列

测试N=80实现64位运算

测试数据溢出N=100



四：心得体会

汇编代码和以前学的c++等高级语言有很大不同，一时间有点不适应，且汇编·课还是兼听的，可以说很有困难。好在看书后对汇编代码有了大致的理解，再加上熟悉了各种指令后，也便勉强能够写简单程序了。斐波那契数列算是比较常见的数列，因此对程序的设计没有什么困难。然而，在实际编写中，会有很大困难，经常会出错：程序莫名进入死循环，输出不了想要的值，以及处理64位数据等。例如我定义了一个数组，应当是.sapce 200 我却定义成.word 200 ,调试过程中发现bug，我却没注意这个地方，耗费了大量时间。而且汇编代码不是结构化的，用到了很多跳转的思想，这就让修改代码有很大困难。本实验代码，存在着几个不足:对数据溢出判断不够精确,寄存器大量使用,未用16进制输出函数输出16进制，而是将16进制数保存在数组中，逐项输出以及代码不够高效和简捷。这些问题为以后写汇编代码提供了参考和经验，也让我下决心今后一定好好写c++!