华 中 科 技 大 学

课 程 实 验 报 告

课程名称：汇编语言程序设计实验

实验名称： 汇编语言子程序设计与C语言混合编程

实验时间：2022/10/4

实验地点：南五楼522

指导教师：鲁赵骏

专业班级： 密码2101

学 号： U202116003

姓 名：侯竣

报告日期：2022/10/25

**成绩评定**

|  |  |  |
| --- | --- | --- |
| 实验完成质量（70分） | 报告撰写质量（30分） | 总成绩 |
| 实验步骤清晰、详细、深入，实验记录真实完整等 | 报告规范、完整、通顺、详实 |
|  |  |  |

1. 实验目的
2. 掌握汇编语言子程序设计方法；
3. 熟悉汇编语言主程序和子程序不同的参数传递方法;
4. 了解C语言和汇编语言混合编程方法以及主、子程序之间参数传递的机制。
5. 实验内容

## 任务1.1：立即数寻址与间接寻址



## 任务1.2：实现简单的数学公式

## 任务2.1：显示"hello,everyone!"

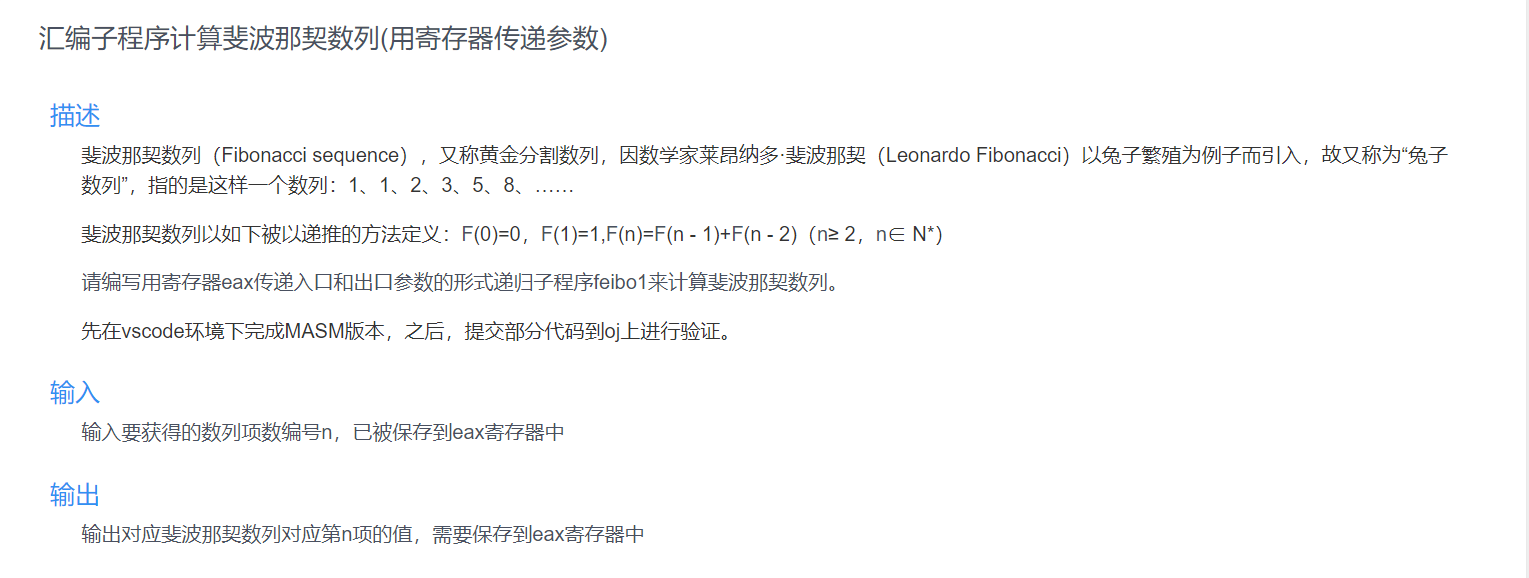
## 任务2.2：熟练应用变量定义和标号属性

## 任务3.1：去除字符串中的数字和空格

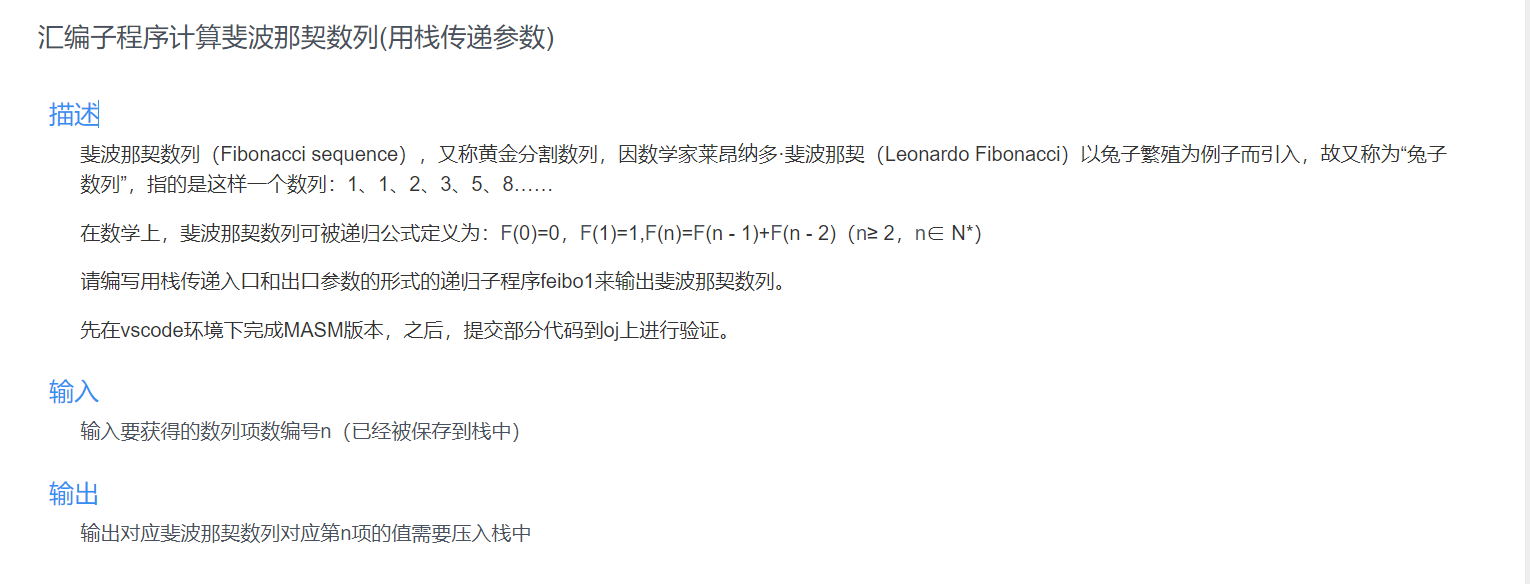
## 任务3.2：计算字符串的出现次数



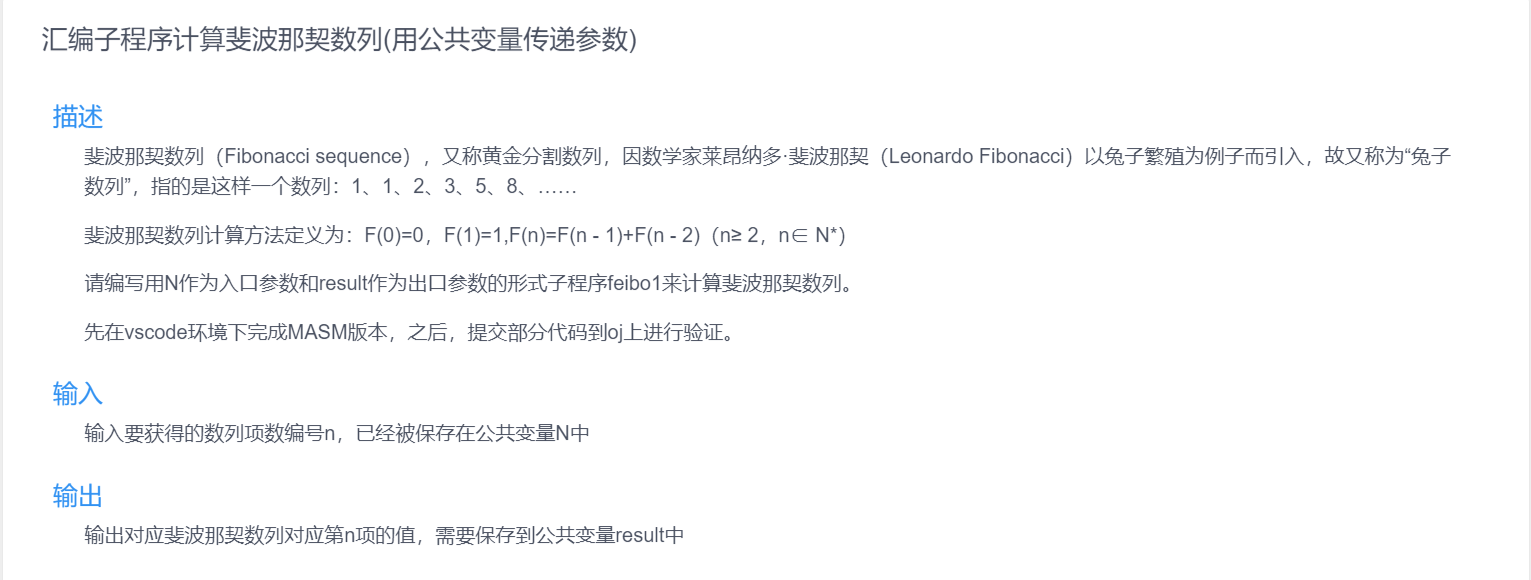
## 任务4.1：汇编子程序计算斐波那契数列(用寄存器传递参数)



## 任务4.2：汇编子程序计算斐波那契数列(用栈传递参数)



## 任务4.3：汇编子程序计算斐波那契数列(用公共变量传递参数)



1. 实验要求
2. 任务1、2、3需要首先在oj平台上提交通过，在实验报告中需要填写可masm6.x以上版本编译通过简化段定义的完整汇编语言程序包括输入输出，**不只是子程序代码片段**。
3. 理解思考任务1、2、3中不同参数传递方式对汇编子程序编写的区别，掌握dosbox下调试汇编程序的方法。
4. 任务4中在不同的C语言开发环境中实现与汇编语言程序的混合编程，其操作方法有可能是不同的。请大家选择白己熟悉的C语言开发环境并查找相关的资料完成本实验。在实验报告中，详细地描述采用的开发环境及其实现方法。
5. 观察C语言编译器中对各种符号的命名规则（指编译器内部可以识别的命名规则，比如，符号名前面是否加下划线“\_”等），主、子程序之间参数传递的机制，通过栈传递参数后堆栈空间回收的方法。
6. 对混合编程形成的执行程序，用调试工具观察由C话言形成的程序代码与由汇编语言形成的程序代码之间的相互关系，包括段、偏移的值，汇编指令访问C的变量时是如何翻译的等。
7. 通过本次实验，希望大家掌握汇编子程序的设计方法以及不同的编程语言是可以协同解决一个问题的，而且可以利用不同语言的特点来更好地解决问题；利用汇编语言的知识，能够更好地理解高级语言的内部处理原理与策略。
8. 实验过程

任务1.1



非常简单, 将61h移入ax, 再将ax移入buffer即可

.model small

.stack

.data

buffer dw ?

.code

.startup

mov ax,61h

mov [buffer],ax

mov cx, buffer ; 参数二：要显示的字符串

mov dx, 2 ; 参数三：字符串长度

mov bx, 1 ; 参数一：文件描述符(stdout)

mov ax, 4 ; 系统调用号(sys\_write)

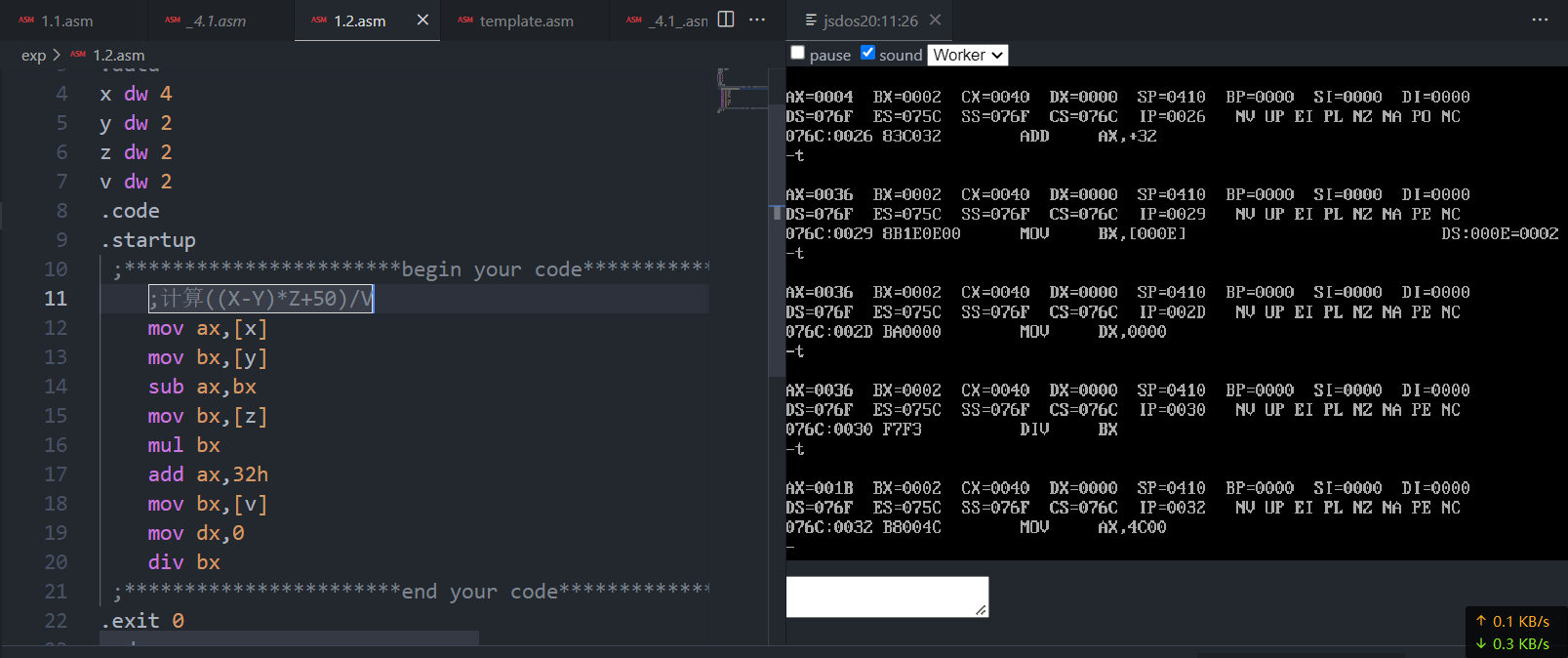
int 80h ; 调用内核功能

ret

.exit 0

end

任务1.2



如图可见, 结果(dx.ax)为1BH, 即27

.model small

.stack

.data

x dw 4

y dw 2

z dw 2

v dw 2

.code

.startup

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*begin your code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;计算((X-Y)\*Z+50)/V

mov ax,[x]

mov bx,[y]

sub ax,bx

mov bx,[z]

mul bx

add ax,32h

mov bx,[v]

mov dx,0

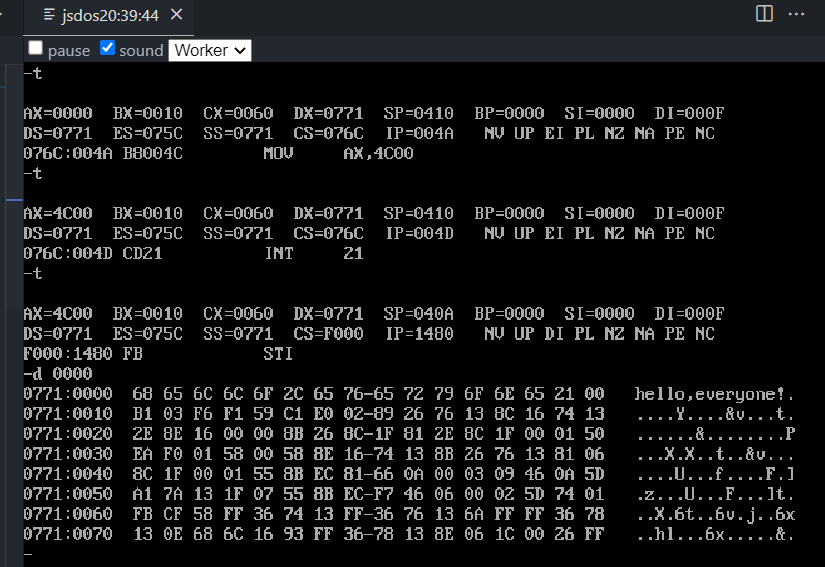
div bx

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end your code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.exit 0

end

任务2.1



用mov 修改数据即可

.model small

.stack

.data

msg db 'hello,world!',3 dup(?), '$'

.code

.startup

mov di, offset msg

add di, 6h

mov [di], 'e'

inc di

mov [di], 'v'

inc di

mov [di], 'e'

inc di

mov [di], 'r'

inc di

mov [di], 'y'

inc di

mov [di], 'o'

inc di

mov [di], 'n'

inc di

mov [di], 'e'

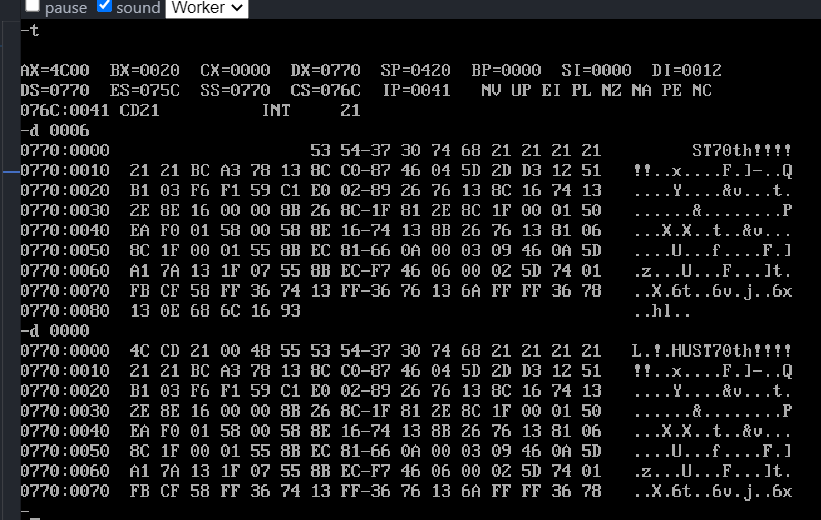
inc di

mov [di], '!'

.exit 0

end

任务2.2



mov赋值即可

.model small

.stack

.data

v\_word db 'hUST'

target db 10 dup(0)

.code

.startup

mov di, offset v\_word

sub [di], 20h

mov di, offset target

mov [di], '7'

inc di

mov [di], '0'

inc di

mov [di], 't'

inc di

mov [di], 'h'

inc di

mov cx, 6

mov al, '!'

lp:

mov [di], al

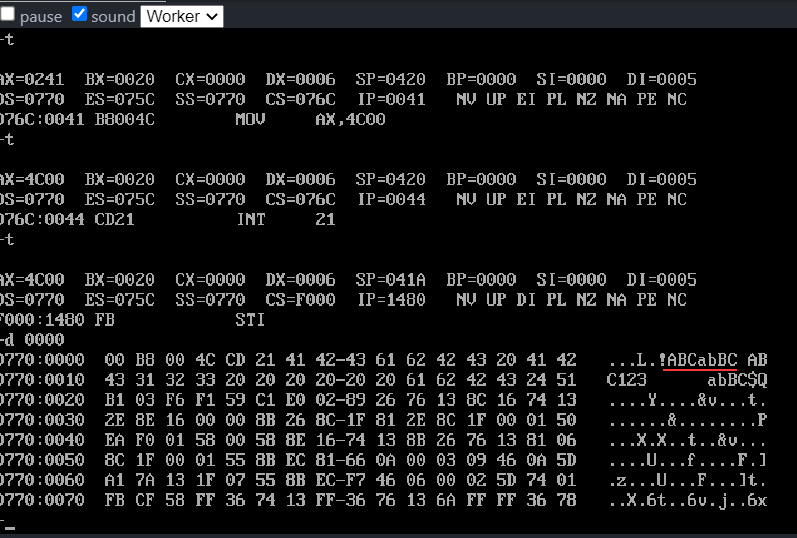
inc di

loop lp

.exit 0

end

任务3.1



思路:遍历string, 如果遇到大小写字母就入栈, 计数+1, 遇到’$’结束, 最后再根据计数逐个出栈对string赋值, 实现去空格和数字的功能

.model small

.stack

.data

string db " ABC123 abBC"

db '$'

slength equ $-string

.code

.startup

mov dx, 0

mov cx, slength

mov di, offset string

eliminate:

mov ax, 0

mov al, [di]

inc di

cmp al, 40h

js lp

inc dx

push ax

lp: loop eliminate

mov cx, dx

mov di, offset string

add di, dx

dec di

restore:

pop ax

mov [di], al

dec di

loop restore

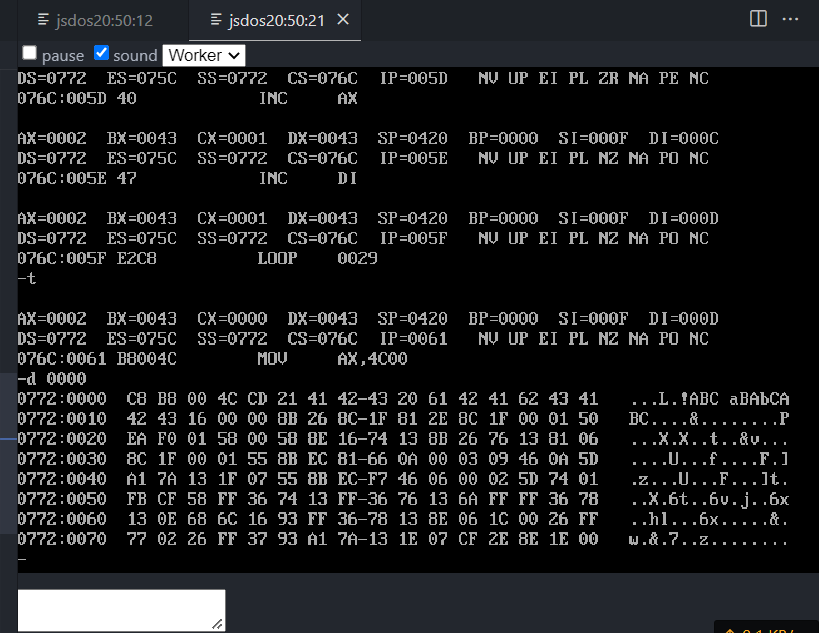
mov ah,02h

mov dx, offset string

.exit 0

end

任务3.2



如图:string="ABC aBAbC"，cmpstr="ABC"，cmpstr在string中出现2次, 即ax=2

思路:先比较第一个字符, 如果成功则跳转到比较第二个的代码, 第三个同理, 否则向前移动, 比较下一个字符

.model small

.stack

.data

string db "ABC aBAbC"

slength equ $-string

cmpstr db 'ABC'

cmpslength equ $-cmpstr

.code

.startup

mov cx, slength

sub cx, 2

mov si, offset cmpstr

mov di, offset string

xor ax,ax

xor bx,bx

xor dx,dx

com:

mov bl, [di]

cmp bl, 60h

js upper1

sub bl, 20h

upper1:

mov dl, ds:[si]

cmp bl, dl

jnz fal

mov bl, ds:[di+1]

cmp bl, 60h

js upper2

sub bl, 20h

upper2:

mov dl, ds:[si+1]

cmp bl, dl

jnz fal

mov bl, ds:[di+2]

cmp bl, 60h

js upper3

sub bl, 20h

upper3:

mov dl, ds:[si+2]

cmp bl, dl

jnz fal

inc ax

fal:

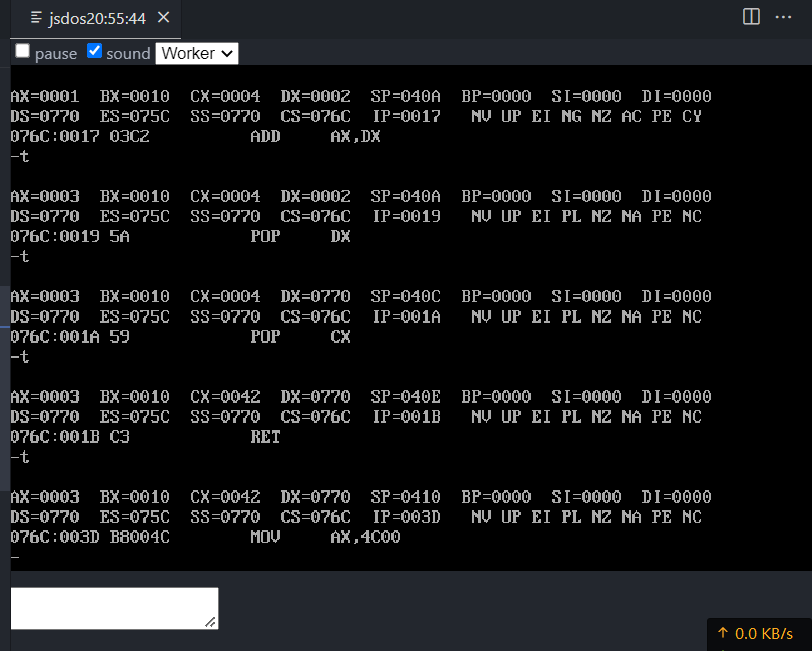
inc di

loop com

.exit 0

end

任务4.1



寄存器传参, 这是第4项的结果, ax=3

.model small

.stack

.data

.code

;寄存器传参

feibo1 proc

;比较特殊情况

cmp ax, 3

jl LessThanThree

;计算

push cx

push dx

mov cx, ax

dec ax

call feibo1 ;前一项

mov dx, ax

mov ax, cx

sub ax, 2

call feibo1 ;前第二项

add ax, dx

pop dx

pop cx

ret

;小于3直接输出1

LessThanThree:

mov ax, 1

ret

feibo1 endp

.startup

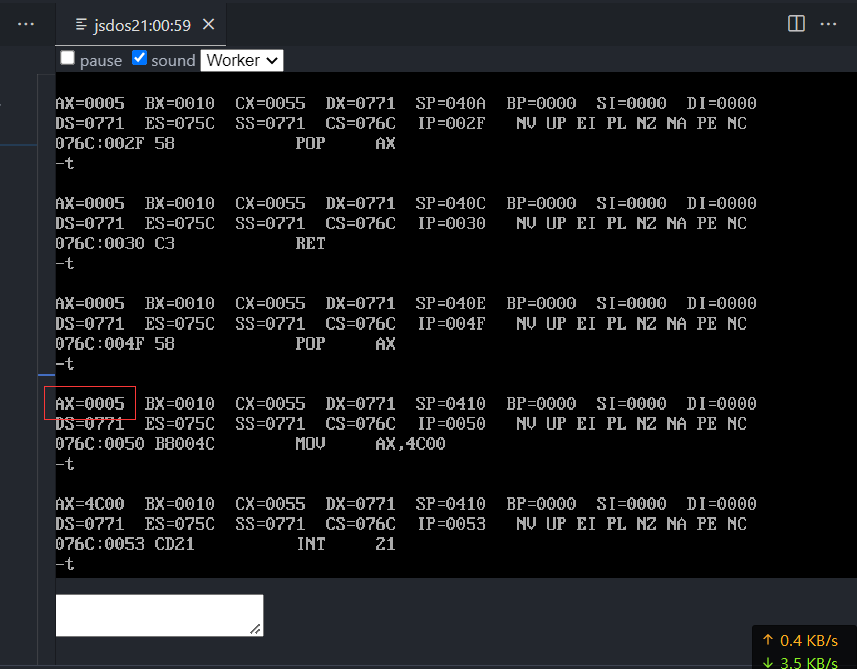
mov ax, 4

call feibo1

.exit 0

end

任务4.2



堆栈传参, 关键需要找对参数相对sp的偏移位置即可, 记得+IP, 此处是计算第5项的结果, ax从栈中pop得到5

.model small

.stack

.data

.code

;栈传参

feibo1 proc

;比较特殊情况

push ax

push bp

push cx

push dx

mov bp, sp

mov ax, [bp+10]

cmp ax, 3

jl LessThanThree

;计算

mov cx, ax

dec ax

push ax

call feibo1 ;前一项

pop ax

mov dx, ax

mov ax, cx

sub ax, 2

push ax

call feibo1 ;前第二项

pop ax

add ax, dx

jmp done

;小于3直接输出1

LessThanThree:

mov ax, 1

done:

mov [bp+10], ax

pop dx

pop cx

pop bp

pop ax

ret

feibo1 endp

.startup

mov ax, 5

push ax

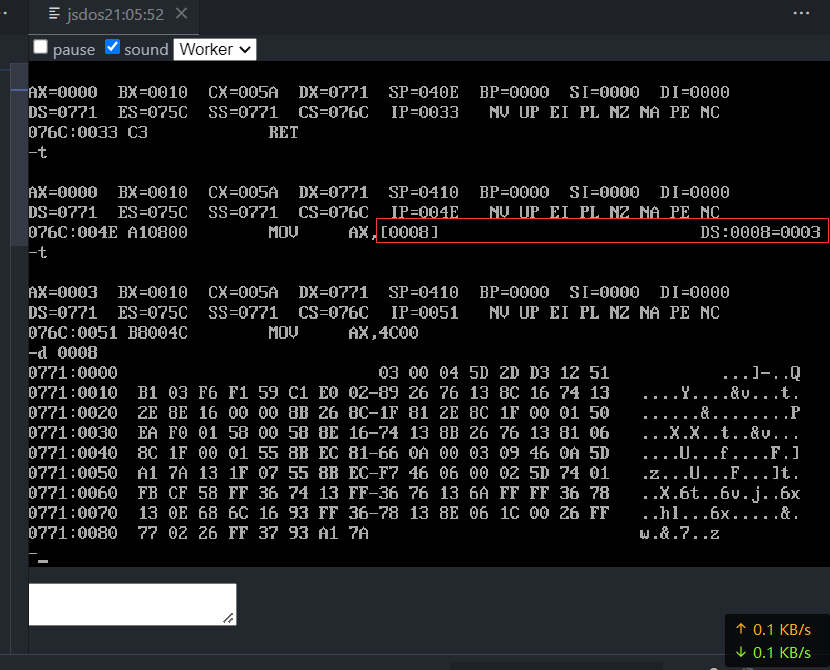
call feibo1

pop ax

.exit 0

end

任务4.3



公共变量传参, 如图为n=4的结果, result=3

.model small

.stack

.data

N dw 4

result dw ?

.code

;公共变量

feibo1 proc

push ax

push cx

push dx

mov ax, N

;比较特殊情况

cmp ax, 3

jl LessThanThree

;计算

mov cx, ax

dec ax

mov N, ax

call feibo1 ;前一项的值

mov dx, result

mov ax, cx

sub ax, 2

mov N, ax

call feibo1 ;前第二项的值

mov ax, result

add ax, dx

jmp done

;小于3直接输出1

LessThanThree:

mov ax, 1

done:

mov result, ax

pop dx

pop cx

pop ax

ret

feibo1 endp

.startup

call feibo1

mov ax, result

.exit 0

end

1. 总结与体会

在oj上无法调试, 做汇编实验十分考验对知识的掌握运用能力, 以及对内存的熟悉程度, 同时oj环境是32位, 也考验了我们对所学知识的迁移类比能力, 最后三题斐波那契递归程序比较有难度, 需要综合所学才能做出来.