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：2023 年 10 月 2 日

实验地点：

指导教师：

专业班级：

学　　号：

姓　　名

报告日期：2023 年 10 月 2 日

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

(1) 掌握汇编语言子程序设计方法；
(2) 熟悉汇编语言主程序和子程序不同的参数传递方法;
(3) 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。

2. 实验内容

设计汇编程序来实现如下数列计算，输入项数 n，输出对应第 n 项的 16 进制数值。数列以如下被以递推的方法定义：

F(0)=0，F(1)=1, F(2)=2， F(n)=F(n - 1)+F(n - 2)（n≥ 3，n∈ N*）

任务 1：子程序计算递推数列(用寄存器传递参数)

请编写 vscode 环境下完整的 MASM 可编译通过的汇编程序，其中需要**使用寄存器 ax 传递入口和出口参数**的形式递归子程序 ditui1 来计算数列。输入通过 dos 系统调用获取，输出需要通过 dos 系统调用**输出显示为 10 进制数**。

任务 2：子程序计算递推数列(用栈传递参数)

请编写 vscode 环境下完整的 MASM 可编译通过的汇编程序，其中需要**使用堆栈传递入口参数和出口参数**的形式递归子程序 ditui1 来计算数列。输入通过 dos 系统调用获取，输出需要通过 dos 系统调用**输出显示为 10 进制数**。

任务 3：子程序计算递推数列(用公共变量传递参数)

请编写 vscode 环境下完整的 MASM 可编译通过的汇编程序，其中**需要使用 N 作为入口参数和 result 作为出口参数**的形式子程序 ditui1 来计算数列。输入通过 dos 系统调用获取，输出需要通过 dos 系统调用**输出显示为 10 进制数**。

任务 4：计算数列(混合编程)

请用汇编语言编写计算上述数列的子程序 ditui1，用 C 语言函数来获取输入、调用 ditui1 汇编函数，并且显示子程序返回的结果。

3. 实验要求

(1) 前两次上机实验只需要贴线上平台通过截图或者代码截图即可。

(2) 第三次实验任务 1、2、3 需要首先在线上平台上提交通过，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，**不只是上机提交的子程序代码片段**。

(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。请大家选择自己熟悉的 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 语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。

4. 实验过程

一、 实验一截图：

图 1.1 实验一平台通过截图

提交时间	ID	状态	语言	用户名	姓名	班级
2023-09-11 20:04:09	48292	作答正确	MASM	U202211915	黄雅婷	网安202201

图 1.2 题目 1.1 平台通过具体信息截图

提交时间	ID	状态	语言	用户名	姓名	班级
2023-09-11 20:04:45	48307	作答正确	MASM	U202211915	黄雅婷	网安202201

图 1.3 题目 1.2 平台通过具体信息截图

二、 实验二截图：

图 2.1 实验二平台通过截图

提交时间	ID	状态	语言	用户名	姓名	班级
2023-09-18 20:59:37	57208	作答正确	MASM	U202211915	黄雅婷	网安202201

图 2.2 题目 2.1 平台通过具体信息截图

提交时间	ID	状态	语言	用户名	姓名	班级
2023-09-18 21:00:22	57225	作答正确	MASM	U202211915	黄雅婷	网安202201

图 2.3 题目 2.2 平台通过具体信息截图

三、 实验三：

任务 1：子程序计算递推数列(用寄存器传递参数)

（1）提纲

1. 程序结构
   (1) 数据段定义
   (2) 代码段定义
   (3) 主程序入口
   (4) 递归子程序实现
2. 数据段
   定义存储输入项数和递归计算结果的变量。
3. 代码段
   (1) 主程序入口：
      1) 使用 DOS 调用获取用户输入的项数。
      2) 将字符转换为对应的数字值。
      3) 调用递归子程序 ditui1 计算数列的特定项。
      4) 调用输出函数将结果以十进制数的形式显示。
   (2) 实现递归子程序 ditui1 来计算数列：
      1) 实现计算数列项的递归算法。
      2) 使用 ax 寄存器传递参数，计算特定项的数值。
      3) 考虑递归结束条件和递推公式。
      4) 保护和恢复寄存器状态，以免破坏外部调用的数据。
      5) 考虑栈溢出问题，因为递归可能导致栈溢出，尤其是在较大的项数上。

(3) 使用 DOS 系统调用来读取输入项数和输出计算结果：

1) 编写用于输出结果的子程序，将计算得到的数值以十进制数的形式显示。

2) 使用 DOS 调用退出程序。

（2）设计思想

在此程序中， 调用递归子程序 ditui1 用于计算递推数列。使用 push 和 pop 保存并恢复寄存器状态，以避免破坏外部调用的数据。使用 ax 寄存器作为参数传递，以计算数列中的特定项。递归结束条件为 n ≤ 2，此时直接返回 n 的值；否则，根据递推公式 F(n) = F(n-1) + F(n-2)，递归调用计算 F(n-1) 和 F(n-2)，然后返回它们的和。

（3）流程图

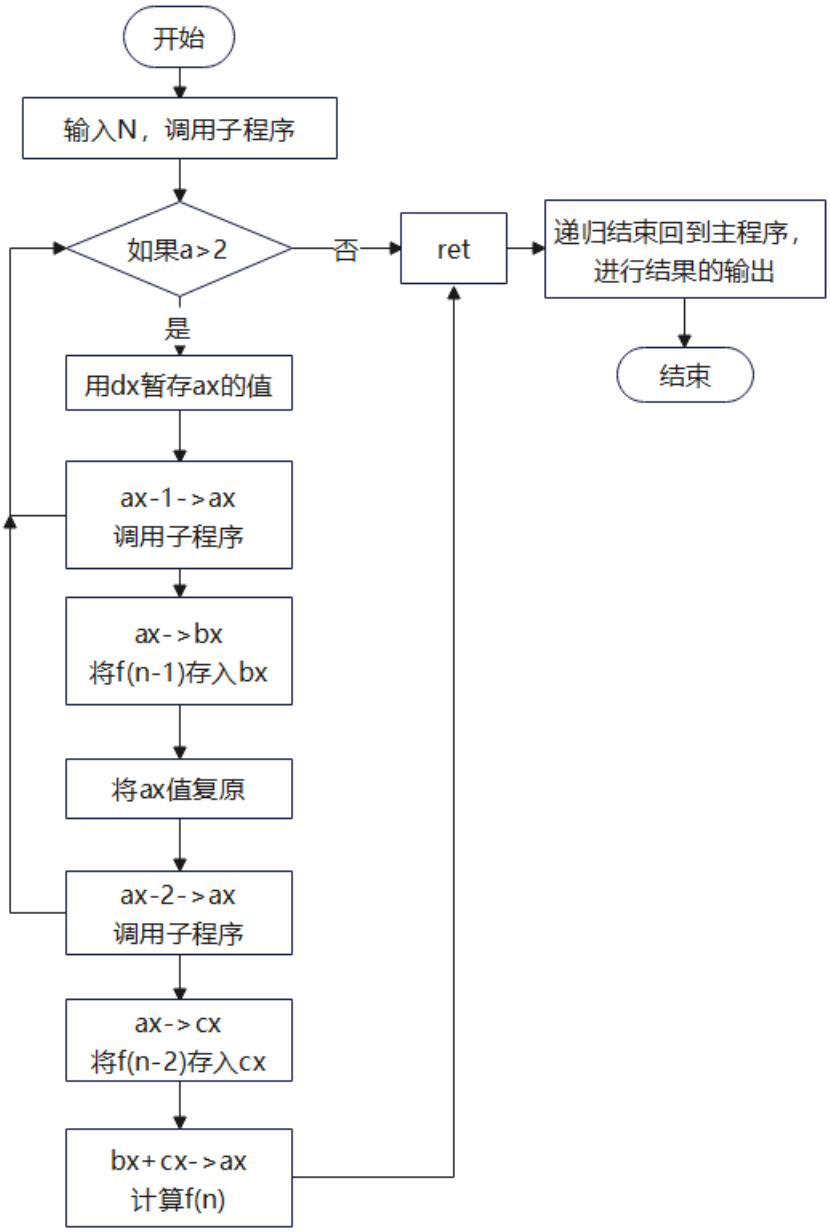

图 3.1.1 寄存器传参计算递推数列流程图

（4）源程序

```
;PREPEND BEGIN
    .model small
    .stack
    .data
    .code
start:
    mov ah, 08h
    int 21h
    sub al, '0'
    xor ah,ah
```

```
    call ditui1
    call print_ax
;******exit******
    mov ax, 4c00h
    int 21h
;******exit******
;PREPEND END
;TEMPLATE BEGIN
;ax=n 寄存器 ax 传递参数
ditui1 proc
;ditui1
;***begin****
    push bx
    push cx
    push dx

    cmp  ax,2   ;如果 n 等于 0/1/2，则 f(n)的值等于 n，可直接结束
    jbe   done

    ;计算公式为 f(n)=f(n-1)+f(n-2)，则分为-1 和-2 两种去进行递归计算
    ;同时将 ax 传递参数

    mov dx, ax  ;用 dx 把 ax 的值保留下来
    dec ax      ;计算 f(n-1)
    call ditui1
    mov bx, ax  ;将计算结果存入 bx

    mov ax, dx  ;将 ax 值复原
    sub ax, 2   ;计算 f(n-2)
    call ditui1
    mov cx, ax  ;将计算结果存入 cx

    mov ax, bx  ;计算 f(n-1)+f(n-2)
    add ax, cx

done:
    pop dx
    pop cx
    pop bx

    ret
;****end******
ditui1 endp
;TEMPLATE END
```

```
;APPEND BEGIN
print_ax proc
    push bx
    push ax
    push cx
    push dx
    xor dx, dx
    xor cx, cx  ;初始化 cx,用作计数器
    mov bx, 10  ;用 bx 让 ax 每次除以 10，余数为每一位的数字
pri_1:
    inc cx
    div bx      ;将 ax 除以 10，得到的余数即为最低位，最先压进栈
    add dx, 30h ;将余数转化为 ascii 码
    push dx     ;将余数的 ascii 码压入栈中
    xor dx, dx
    cmp ax, 0   ;如果 ax 除以 10 后得到 0，退出循环
    jnz pri_1
    mov ah, 02h ;准备输出字符
pri_2:
    pop dx      ;取出输入的值
    int 21h
    loop pri_2

    pop dx
    pop cx
    pop ax
    pop bx
    ret
print_ax endp
end start
;//APPEND END
```

（5）实验步骤

a.子程序整体框架与细节思考架

1. 目的思考：

   (1) 递推数列计算：编写子程序以递归方式计算数列的特定项。

   (2) 寄存器选择：使用 ax 寄存器传递参数。

2. 子程序框架：

   (1) 递归子程序：ditui1

(2) 寄存器选择：使用 ax 传递参数，其他寄存器用于中间计算。

3. 过程与细节：

(1) 判定条件：当 n <= 2 时直接返回 n 的值；否则按照递推公式 F(n) = F(n-1) + F(n-2) 递归调用 ditui1 计算并返回结果。

(2) 递归出口：确保递归结束条件为 n <= 2，避免栈溢出风险。

b. 编写代码

1. 配置环境：

(1) 打开 IDE，配置汇编环境，确保相关工具和插件已安装。

2. 汇编代码编写：

(1) 理解程序所给的部分代码包括数据段定义、代码段等，理解输入输出相关功能的实现，然后编写并实现子程序。

(2) 使用 ax 寄存器传递参数，符合任务要求。

(3) 对递归深度进行有效控制，避免栈溢出问题。

c. 调试与运行

1. 调试程序：

(1) 检查潜在的逻辑和语法错误。

(2) 使用单步调试功能，确保程序的逻辑正确性。

2. 运行程序：

(1) 在 DOSBox 中运行程序。

(2) 确保程序能够正确获取输入，计算数列并输出结果。

d. 在 VMCourse 平台对代码进行调整使其通过

1. 代码调整：

(1) 根据 VMCourse 平台的需求，对程序进行必要的调整。

(2) 调整代码以适应平台的特定要求。

2. 提交与测试：

(1) 在 VMCourse 平台提交程序。

(2) 测试程序在平台上执行，确保能够通过所有测试用例。

（5）实验结果

提交时间	ID	状态	语言	用户名	姓名	班级
2023-09-25 20:26:25	65314	作答正确	MASM	U202211915	黄雅婷	网安202201

图 3.1.2 题目 3.1 平台通过具体信息截图

为了区分输入和输出，在主程序调用子程序 print_ax 之前补充一段输出回车换行，代码如下（其下任务同）：

```
    mov bx,ax  ;暂存 ax 的值
    mov dl,0dh ;输出回车
    mov ah,02h
    int 21h
    mov dl,0ah ;输出换行
    mov ah,02h
    int 21h
    mov ax,bx  ;复原 ax 的值
```

以及为了将输入显示出来，将：

```
    mov ah, 08h
    int 21h
```

改为：

```
    mov ah, 01h
    int 21h
```

计算第 5 项递推数列的结果为 8。

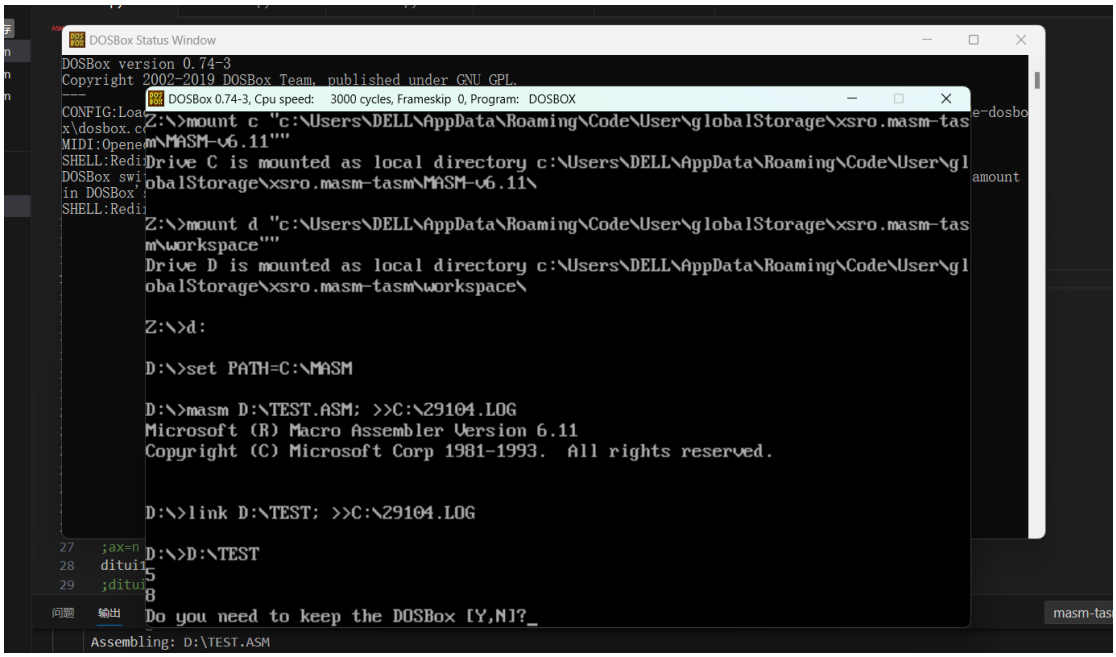

图 3.1.3 寄存器传参计算递推数列输入输出截图

任务 2：子程序计算递推数列(用栈传递参数)

（1）提纲

1. 程序结构

(1)段声明：

1) 使用 small 模型。

2) 定义堆栈、数据段和代码段。

(2)主程序入口 start：

1) 使用 DOS 调用获取用户输入的项数。

2) 将字符转换为对应的数字值。

3) 调用递归子程序 ditui1 计算数列的特定项。

(3)输出函数 print_ax：将计算结果转换为十进制，并通过 DOS 调用输出结果。

2. 递归子程序 ditui1

(1)寄存器选择：使用 bp 寄存器作为帧指针， ax, bx, cx, dx 用于临时数据存储。

(2)递归计算：

1) 根据 N 的值，如果 N <= 2，直接返回 N 的值；否则，按照递推公式递归调用 ditui1 计算 F(N-1) 和 F(N-2)。

2) 使用堆栈传递参数和返回值。

3) 确保堆栈操作的正确性，以避免出现错误。

3. 输入参数传递

(1)堆栈操作：使用 push 和 pop 在堆栈上传递参数。

(2)在堆栈上传递参数：将输入项数 N 推入堆栈中，作为递归子程序的参数。

4. 输出参数返回

(1)堆栈操作：通过堆栈返回计算结果。

(2)在堆栈返回参数：在递归子程序中将计算结果放回堆栈，以便主程序能够获取计算结果。

（2）设计思想

在此程序中，我们在主程序里将入口参数 N 保存到堆栈中，然后在递归子程序里取出 N 进行计算，并将计算结果放回堆栈，以便主程序能够获取计算结果。由于第 N 项的结果取决于第 N-1 项,第 N-2 项的和，故我们先将第 N-1 项的值赋给 bx，再将第 N-2 的值赋给 cx，直到为 0/1/2 时递归返回，每次递归计算完后 bx 得到第 N-1 项的值，cx 得到第 N-2 项的值，然后将 bx 与 cx 值的和存入 ax，然后再通过堆栈返回计算结果。

（3）流程图

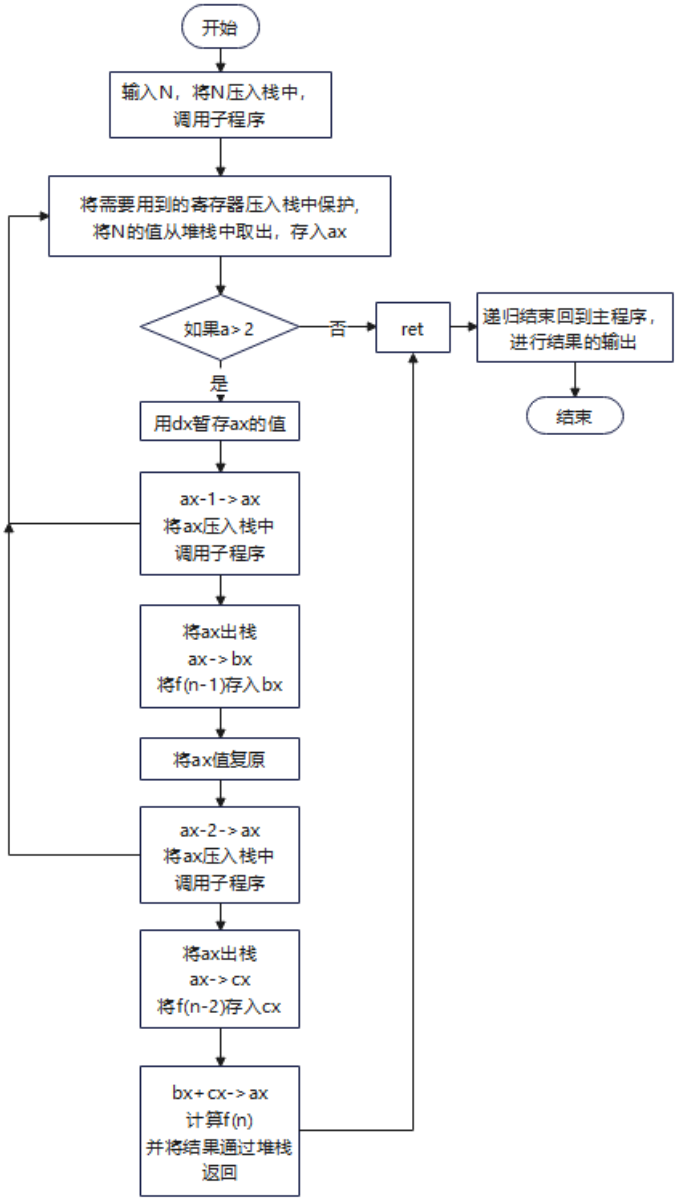

图 3.2.1 栈传参计算递推数列流程图

（4）源程序

```
;PREPEND BEGIN
    .model small
    .stack
    .data
    .code
start:
    mov ah, 08h
    int 21h
    sub al, '0'
    xor ah,ah
```

```
    push ax
    call ditui1
    pop ax
    call print_ax
;******exit******
    mov ax, 4c00h
    int 21h
;******exit******
;PREPEND END
;TEMPLATE BEGIN
ditui1 proc
;***begin****
    push ax
    push bx
    push cx
    push dx
    push bp

    mov bp,sp
    mov ax,[bp+12] ;将 N 的值从堆栈中取出，存入 ax
    cmp  ax,2   ;如果 n 等于 0/1/2，则 f(n)的值等于 n，可直接结束
    jbe   done

    ;计算公式为 f(n)=f(n-1)+f(n-2)，则分为-1 和-2 两种去进行递归计算
    ;用栈传递入口和出口参数的形式

    mov dx, ax  ;用 dx 把 ax 的值保留下来
    dec ax      ;计算 f(n-1)
    push ax     ;将 ax 压入栈中，用堆栈进行传递
    call ditui1
    pop ax      ;将值从堆栈中取出
    mov bx, ax  ;将计算结果存入 bx

    mov ax, dx  ;将 ax 值复原
    sub ax, 2   ;计算 f(n-2)
    push ax     ;作用同上
    call ditui1
    pop ax
    mov cx, ax  ;将计算结果存入 cx

    mov ax, bx  ;计算 f(n-1)+f(n-2)
    add ax, cx

done:
```

```
    mov [bp+12],ax ;将计算结果通过堆栈返回
    pop bp
    pop dx
    pop cx
    pop bx
    pop ax

    ret
;****end******
;TEMPLATE END
ditui1 endp
;APPEND BEGIN
print_ax proc
    push bx
    push ax
    push cx
    push dx
    xor dx, dx
    xor cx, cx  ;初始化 cx,用作计数器
    mov bx, 10  ;用 bx 让 ax 每次除以 10，余数为每一位的数字
pri_1:
    inc cx
    div bx      ;将 ax 除以 10，得到的余数即为最低位，最先压进栈
    add dx, 30h ;将余数转化为 ascii 码
    push dx     ;将余数的 ascii 码压入栈中
    xor dx, dx
    cmp ax, 0   ;如果 ax 除以 10 后得到 0，退出循环
    jnz pri_1
    mov ah, 02h ;准备输出字符
pri_2:
    pop dx      ;取出输入的值
    int 21h
    loop pri_2

    pop dx
    pop cx
    pop ax
    pop bx
    ret
print_ax endp
end start
;//APPEND END
```

（6）实验步骤

a.查阅书本资料，了解堆栈传递参数的基本格式和堆栈的数据存储情况的逻辑。

b. 子程序整体框架与细节思考

1. 目的思考：

   (1) 递推数列计算：编写子程序以递归方式计算数列的特定项。

   (2) 使用堆栈传参：确保在子程序间传递参数的正确性。

2. 子程序框架：

   (1) 递归子程序：ditui1

   (2) 堆栈传参：使用堆栈传递参数和返回值。

3. 过程与细节：

   (1) 判定条件：当 n <= 2 时直接返回 n 的值；否则按照递推公式 F(n) = F(n-1) + F(n-2) 递归调用 ditui1 计算并返回结果。

   (2) 堆栈传递参数：确保在调用子程序时使用堆栈传递参数和返回值。

c. 编写代码

1. 画出堆栈存储示意图：

   (1) 明确每一行执行的目的和对堆栈数据存储情况的影响。

2. 汇编代码编写：

   (1) 理解程序所给的部分代码包括数据段定义、代码段等，理解输入输出相关功能的实现，然后编写并实现子程序。

   (2) 使用堆栈传递参数，确保参数传递的正确性。

d.在 vscode 上调试并运行。

e. 在 VMCourse 平台对代码进行调整使其通过。

（7）实验结果

提交时间	ID	状态	语言	用户名	姓名	班级
2023-10-28 14:05:02	99696	作答正确	MASM	U202211915	黄雅婷	网安202201
2023-09-25 23:09:11	67191	作答正确	MASM	U202211915	黄雅婷	网安202201

图 3.2.2 题目 3.2 平台通过具体信息截图

计算第 4 项递推数列的结果为 5。

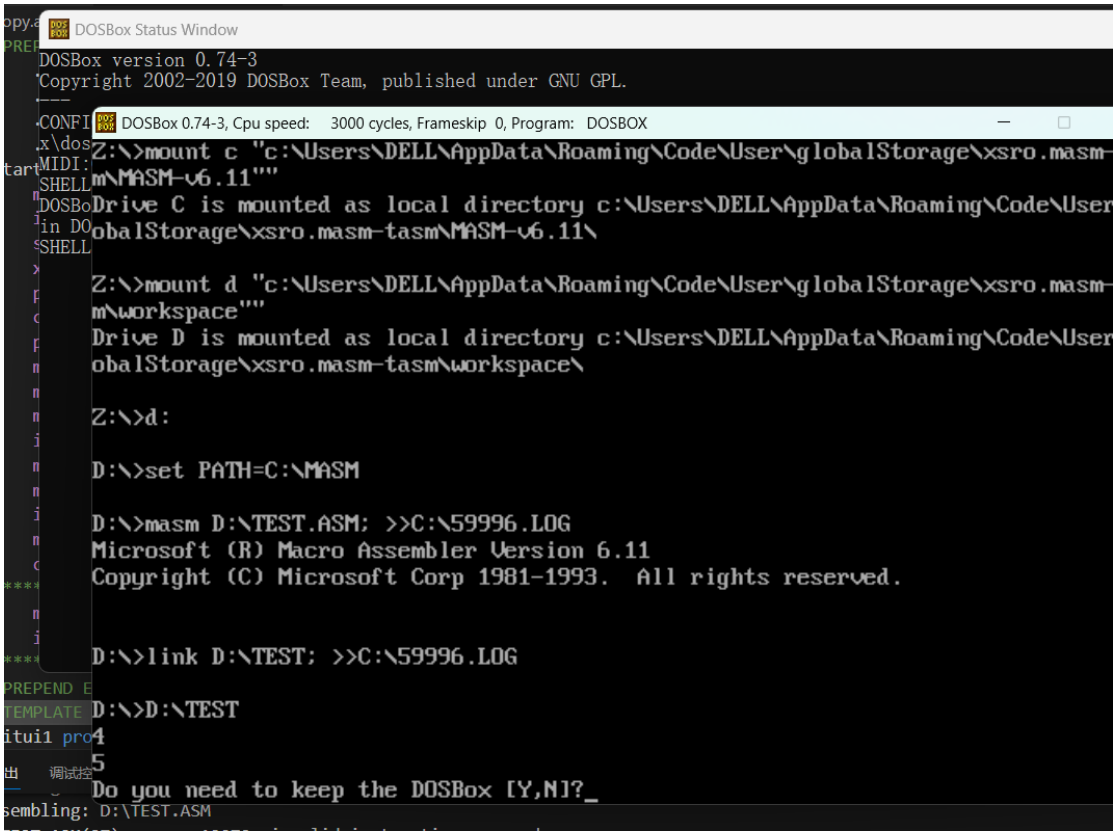

图 3.2.3 栈传参计算递推数列输入结果截图

任务 3：子程序计算递推数列(用公共变量传递参数)

（1）提纲

1. 程序结构

(1)段声明：使用 small 模型，定义堆栈、数据段和代码段。

(2)主程序入口 start：

1) 使用 DOS 调用来获取用户输入的项数 N。

2) 通过字符转换将输入项数转为对应的数字值。

3) 调用递归子程序 ditui1 计算数列的特定项。

(3)输出函数 print_ax：负责将计算结果转换为十进制，并通过 DOS 调用输出结果。

2. 递归子程序 ditui1

(1)寄存器选择：使用 ax, bx, cx, dx 作为临时存储寄存器。

(2)递归计算：

1) 通过递推公式 F(n) = F(n-1) + F(n-2) 计算数列的特定项。

2) 针对输入参数 N 的大小，如果 N <= 2，直接返回对应值；否则进行递归计算。

(3)参数传递和返回：通过公共变量 N 作为入口参数，result 作为出口参数，而且不使用堆栈传递参数和返回值。

3. 输入参数传递

(1)使用公共变量：N 作为入口参数。

(2)公共变量 result 的设置：result 作为出口参数，用于存储计算的数列特定项的结果。

4. 输出参数返回

(1)使用公共变量：result 作为出口参数，存储计算结果。

（2） 设计思想

在此程序中，利用递推公式 F(n) = F(n-1) + F(n-2) 计算数列的特定项，基于 N 的值，当 N <= 2 时，直接返回对应值；否则进行递归计算。使用公共变量 N 作为递归子程序 ditui1 的入口参数，用于控制递归的深度，而 result 作为出口参数，存储计算结果。在递归过程中，根据 N 的值，分别计算 F(0), F(1), F(2) 或使用递推公式计算更高项。为了避免使用堆栈则不能多次调用子程序用以递归，否则会使寄存器的值被改变，影响结果的正确性。

（3） 流程图

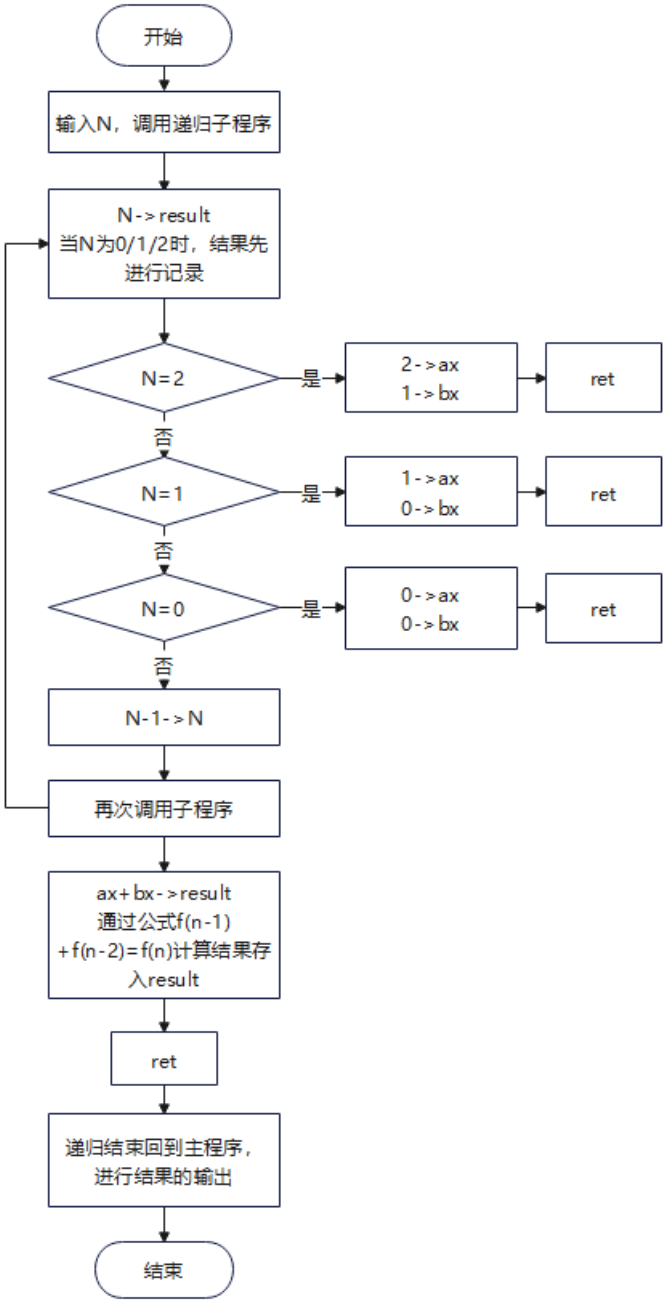

图 3.3.1 公共变量传参计算递推数列流程图

（4） 源程序

```
;PREPEND BEGIN
    .model small
    .stack
    .data
    N dw 0 ;
    result dw 0 ;
    .code
start:
```

```
    mov ah, 08h
    int 21h
    sub al, '0'
    xor ah,ah
    mov N, ax
    xor ax,ax
    call ditui1
    mov ax,result
    call print_ax
;******exit******
    mov ax, 4c00h
    int 21h
;******exit******
;PREPEND END
;TEMPLATE BEGIN
ditui1 proc
;ditui1
;***begin****
    mov CX, N       ;注意存储器间不能传值，mov result,N (×)
    mov result, CX  ;当 N 为 0/1/2 时，结果先进行记录
    cmp N, 2
    je two
    cmp N, 1
    je one
    cmp N, 0
    je zero
    dec N
    call ditui1
    mov dx,ax       ;暂存 f(n-1)
    add ax,bx       ;bx 中存放 f(n-2)，进行运算得 f(n)
    mov bx,dx       ;将 f(n-1)存在 bx，再进入下一次递归
    mov result, ax  ;将每次的结果存入 result
    ret
two:
    mov ax, 2       ;ax 存此时 n-1
    mov bx, 1       ;bx 存此时 n-2
    ret
one:
    mov ax, 1
    mov bx, 0
    ret
zero:
    mov ax, 0
    mov bx, 0
```

```
    ret
;****end******
ditui1 endp
;TEMPLATE END
;function: print_eax value in decimal;
;input:eax, use buffer (10 bytes)
print_ax proc
    push bx
    push ax
    push cx
    push dx
    xor dx, dx
    xor cx, cx  ;初始化 cx,用作计数器
    mov bx, 10  ;用 bx 让 ax 每次除以 10，余数为每一位的数字
pri_1:
    inc cx
    div bx      ;将 ax 除以 10，得到的余数即为最低位，最先压进栈
    add dx, 30h ;将余数转化为 ascii 码
    push dx     ;将余数的 ascii 码压入栈中
    xor dx, dx
    cmp ax, 0   ;如果 ax 除以 10 后得到 0，退出循环
    jnz pri_1
    mov ah, 02h ;准备输出字符
pri_2:
    pop dx      ;取出输入的值
    int 21h
    loop pri_2

    pop dx
    pop cx
    pop ax
    pop bx
    ret
print_ax endp
;function end
end start
;//APPEND END
```

（5） 实验步骤

a.子程序整体框架与细节思考

1. 目的思考：

(1) 递推数列计算：编写子程序以递归方式计算数列的特定项。

(2) 使用公共变量传参：确保在子程序间传递参数的正确性。

(3) 不使用堆栈：在没有堆栈保护下实现对数据的保留。

2. 子程序框架：

(1) 递归子程序：ditui1

(2) 公共变量传参：使用公共变量传递参数和返回值。

(3) 多次利用 jcc 进行跳转：进行多种情况的分类讨论。

3. 过程与细节：

(1) 判定条件：当 n <= 2 时直接返回 n 的值；否则按照递推公式 F(n) = F(n-1) + F(n-2) 递归调用 ditui1 计算并返回结果。

(2) 公共变量传递参数：确保在调用子程序时使用堆栈传递参数和返回值。

(3) 不使用堆栈：子程序只可以使用一次递归。

(4) 特殊值处理：对 N 等于 0/1/2 时，结果进行特殊处理。

b 编写代码

1. 注意汇编语言规范：

(1) 避免语法问题：存储器间不能传值

(2) 减少细节错误：寄存器使用前进行清零。

2. 汇编代码编写：

(1) 理解程序所给的部分代码包括数据段定义、代码段等，理解输入输出相关功能的实现，然后编写并实现子程序。

c.在 vscode 上调试并运行。

d. 在 VMCourse 平台对代码进行调整使其通过。

（6）实验结果

提交时间	ID	状态	语言	用户名	姓名	班级
2023-10-28 17:08:14	99950	作答正确	MASM	U202211915	黄雅婷	网安202201

图 3.3.2 题目 3.3 平台通过具体信息截图

计算第 7 项递推数列的结果为 21。

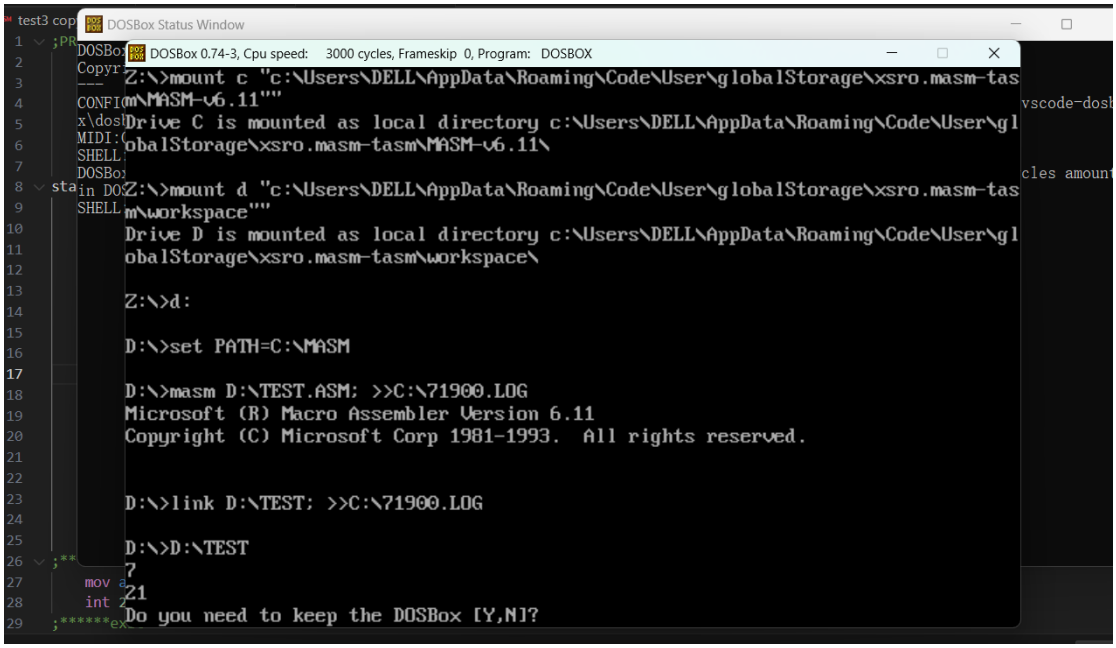

图 3.3.3 公共变量传参计算递推数列输入截图

任务 4：计算数列(混合编程)

（1）提纲

1. 理解任务需求：

需要通过 C 语言函数调用汇编函数来计算数列。

2. 代码设计

(1) C 语言主程序设计：

1) 编写主程序 main：通过 C 语言获取用户输入的项数 n，然后调用汇编函数 ditui1 计算数列的特定项，然后显示计算结果。

2) 汇编子程序设计：编写汇编子程序 ditui1，使用混合编程，以实现递归计算数列，并使用汇编语言的寄存器和 C 语言的变量交互。

3. 调试和测试

(1) C 语言调试：

1) 确保 C 语言的主程序没有语法错误。

2) 使用输出语句进行调试，验证输入和输出结果的准确性。

(2) 汇编语言调试：

1) 检查汇编函数 ditui1，确保逻辑正确性。

2)　逐行调试汇编代码，验证计算结果的正确性。

4. 整合与测试

(1) 整合代码：将汇编函数和 C 语言主程序进行整合。

(2) 在 vs2022 测试：确保能够正确接收用户输入、计算数列，并显示计算结果。

（2）设计思想

运用嵌入式汇编的思想，用汇编语言写一个子程序 ditui1，将其放入到 C 语言的函数 ditui1 中，再通过 C 语言的主函数调用子函数间接调用汇编程序 ditui1。在汇编子程序中，我们通过寄存器传递入口参数 n,计算结果通过寄存器传递到 result,再通过主函数显示输出。

（3）流程图

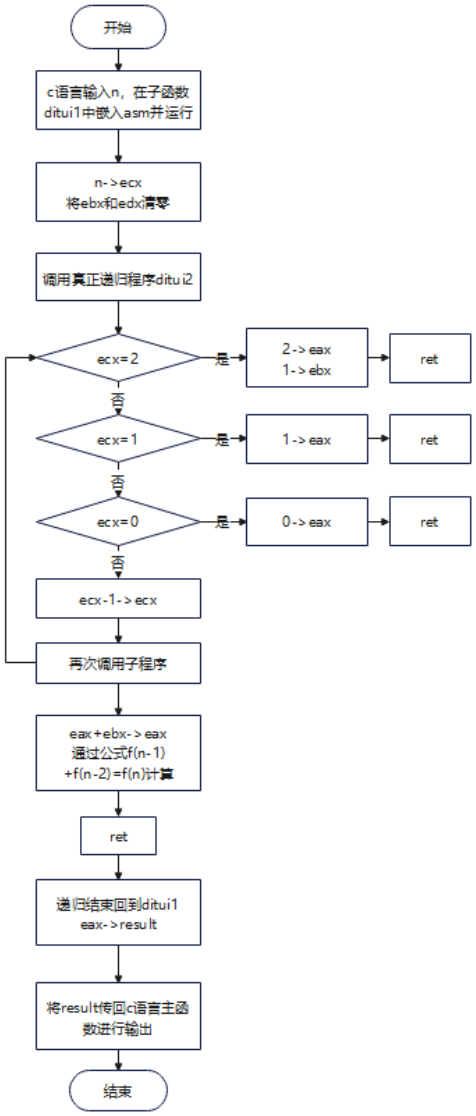

图 3.4.1 混合编程计算递推数列流程图

（4）源程序

```
#include<stdio.h>
int ditui1(int n);
int main(void) {
	int n = 0;
	printf("请输入n：");
	scanf_s("%d", &n);
	int result = ditui1(n);
	printf("第f(%d)项数列的值为：%d" ,n, result);
	return 0;
}
int ditui1(int n) {
	int result;
	__asm {
		mov ecx, n
		mov ebx, 0
		mov edx, 0
		call ditui2
		jmp next1
		ditui2:
		cmp ecx, 2
		je two
		cmp ecx, 1
		je one
		cmp ecx, 0
		je zero
		dec ecx
		call ditui2
		mov edx, eax
```

```
            add eax, ebx
            mov ebx, edx
            ret
            one :
            mov eax, 1
                ret
                two :
            mov eax, 2
                mov ebx, 1
                ret
                zero :
            mov eax, 0
                ret
                next1:
            mov result, eax
        }
        return result;
}
```

（5）实验环境

在 Visual Studio 2022 中进行 x86 汇编和 C 语言的混合编程需要特定的配置。

1. 创建 C++ 项目：

- 打开 Visual Studio 2022。
- 创建一个新的 C++ 项目。

2. 设置 x86 架构：

- 在项目属性中，选择 x86 作为目标平台（32 位）。
- 这可在项目属性的“生成”选项卡下的“平台”和“配置”中进行设置。

3. 添加汇编文件：

- 在解决方案资源管理器中，右键单击“源文件”。
- 生成自定义文件里面选择“masm”

- 编写所需的汇编代码。

4. 与 C++ 文件交互：

- 可以在 C++ 文件中使用 __asm 块，调用汇编函数。

（6）实验步骤

a.创建新项目

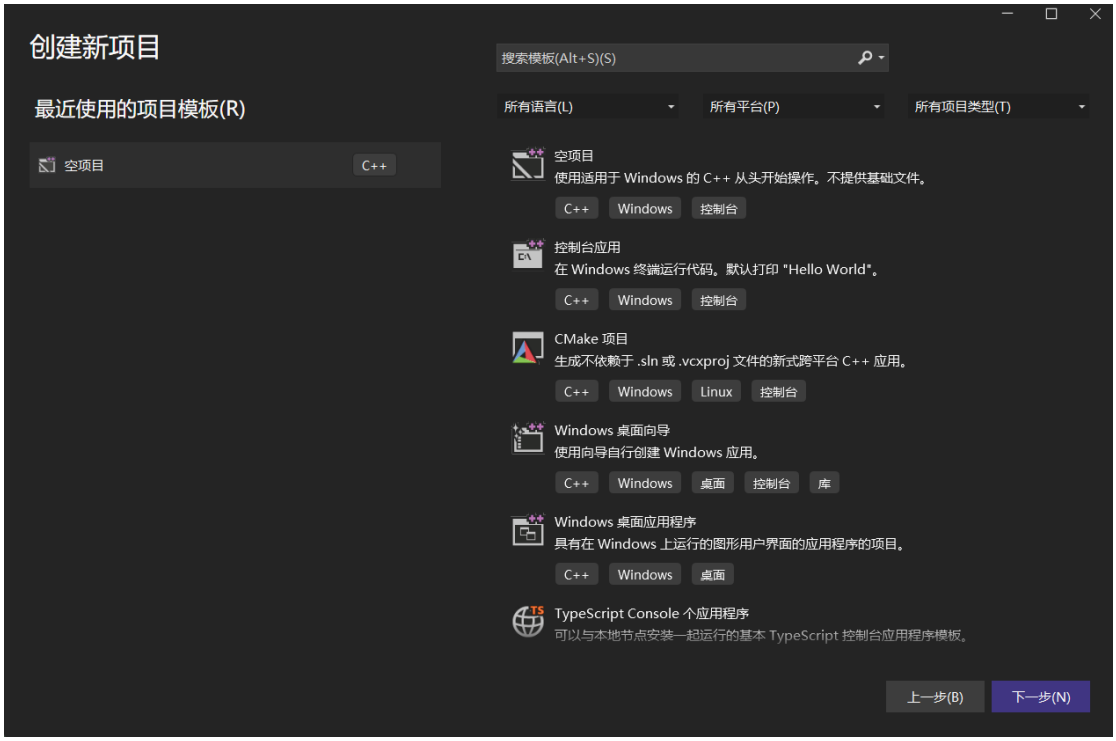

图 3.4.2 混合编程实验步骤 1

b．在生成自定义文件里面选择“masm”

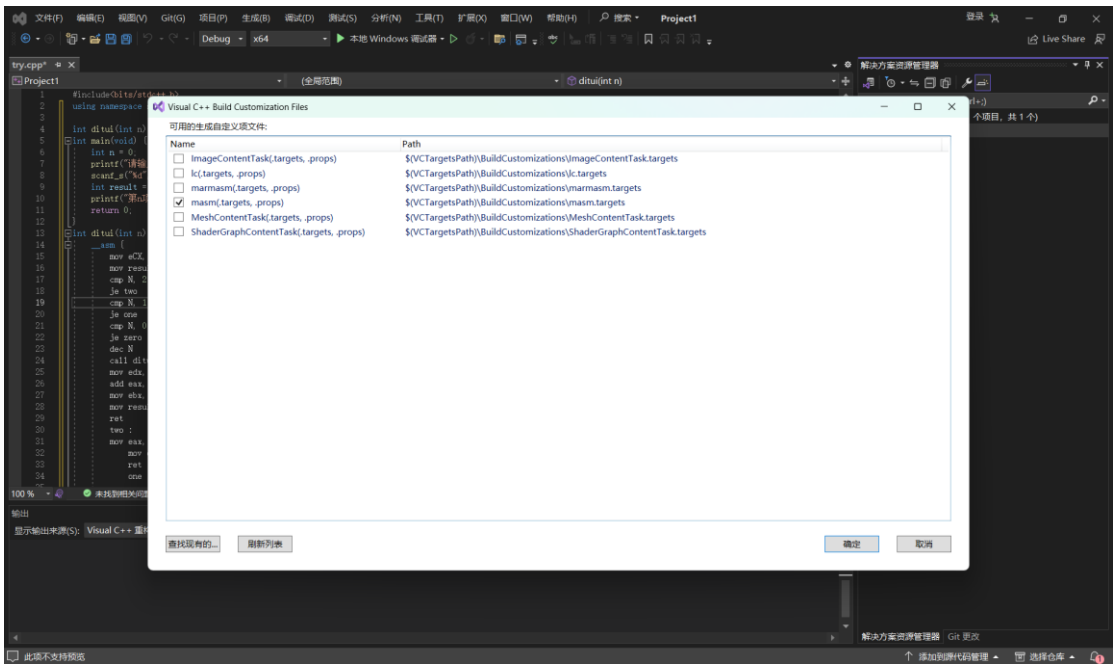

图 3.4.3 混合编程实验步骤 2

c.将 x64 管理器改为 x86

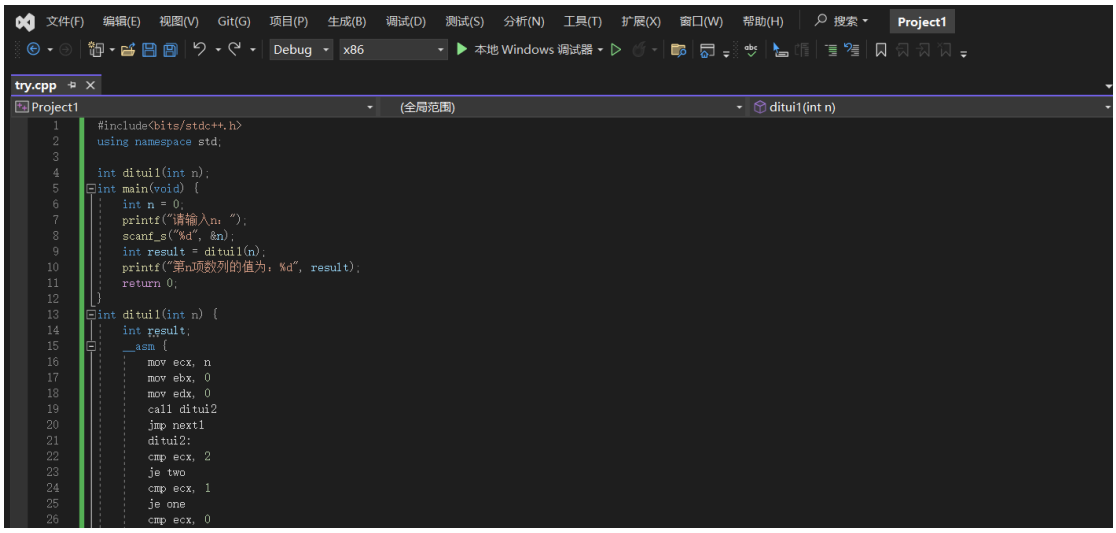

图 3.4.4 混合编程实验步骤 4

d.编辑源程序，调试运行

（7）实验结果

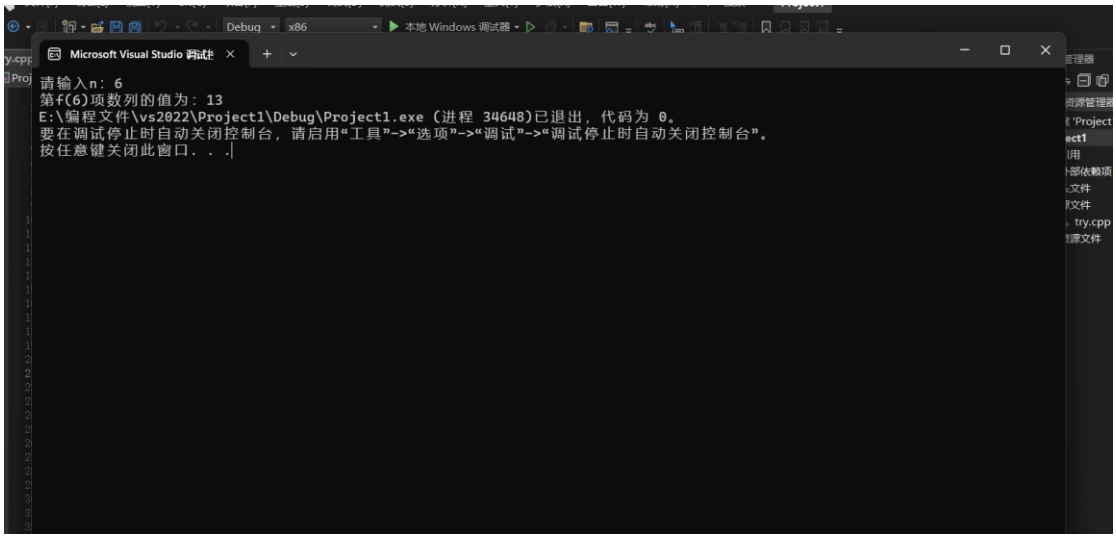

图 3.4.5 混合编程实验结果截图

5.总结与体会

（1）在编写的过程中对汇编语言有个更加准确的理解，在写的过程中有时候会犯一些小的语法错误，致使在运行的时候需要反复调试才可以正确执行，这更说明了理论课对实验课的奠基作用，并且实验课也加深了对理论学习知识的理解和运用，其相辅相成，利于增强我们对于汇编这门语言基础性的理解。

（2）在任务 1、2 和 3 中，我通过汇编语言编写了递归计算数列的程序，针对不同的任务要求，使用了不同的参数传递方式，包括使用寄存器传递参数、堆栈传递参数和使用公共变量传递参数。而其中堆栈的运用，我感觉通过画堆栈图对我理解堆栈传参有很大的作用，并可以用来检查代码的正确性。

任务 1 要求使用寄存器传递参数。通过这个任务，我深入了解了汇编中寄存器的使用方法，如何在不同的寄存器之间传递数据，并实现递归计算递推数列。使用寄存器传递参数，进行递归计算，简洁明了，但局限在参数传递量小且不适合大规模数据处理。保护和恢复寄存器状态，以免破坏外部调用的数据。考虑栈溢出问题，因为递归可能导致栈溢出，尤其是在较大的项数上。

任务 2 要求使用堆栈传递参数。这使我更深入地理解了堆栈的概念和在汇编语言中如何使用堆栈传递参数。递归计算的核心是堆栈操作，通过递归将参数和返回值存储在堆栈中，更适用于大规模数据处理。其中对于堆栈平衡的理解很重要，不仅是最开始调用前的压栈和最后的出栈，而且在多次调用递归的过程中，确保 bp 的数值指向的堆栈中的数值正确，ax 的再次压栈不可或缺，回到上一级函数也要确保其出栈。

任务 3 要求使用公共变量传递参数。这次任务的目的是通过公共变量实现参数的传递和返回值的存储，并不使用堆栈。通过公共变量实现参数传递，虽然简化了参数传递的复杂度，但不如堆栈灵活。而且因为没有堆栈对寄存器数值的保护，所以这一关我改变先前代码的思路使用单次递归的形式，避免寄存器的数值在多次递归的过程中混乱。

（3）任务 4 混合编程的体验让我对嵌入式汇编有更深刻的理解，在 Visual Studio 2022 中进行 x86 汇编和 C 语言的混合编程需要适当的环境配置。特别是对于汇编文件的引入和配置 x86 架构，这对于程序的正常运行至关重要。在汇编代码中，要确保语法正确，尤其是过程的声明、过程的结束等方面。对于汇编函数的编写，需要充分理解 x86 架构的语法和指令。结合 C 和汇编语言的混合编程，能够在性能关键的部分使用汇编语言来进行优化，同时也保持了 C 语言开发的便利性和灵活性。这次实验让我深入了解了两种语言之间的交互方式，同时也对汇编语言在性能优化方面的作用有了更直观的认识。这种混合编

程的实践，为更好地理解和应用底层编程提供了很好的机会。同时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。