第三次作业

1）

**寄存器方式：**

code segment

assume cs:code

start:

mov ah,02h ;输出方式

mov dx,1 ;当前加数

mov bx,0 ;和

mov cx,100 ;计数器

loop1: ;循环开始

add bx,dx ;加

inc dx ;自增

loop loop1 ;循环结束

;以下代码将bx中的值分解成千、百、十、个位，以字符形式输出

mov cx,1000d;千位

mov ax,bx ;拷贝一份

mov dx,0 ;清空

div cx ;ax/cx

mov bx,dx ;余数给dx

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,100d ;百位

mov ax,bx ;拷贝一份

mov dx,0 ;清空

div cx ;ax/cx

mov bx,dx ;余数给dx

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,10d ;十位

mov ax,bx ;拷贝一份

mov dx,0 ;清空

div cx ;ax/cx

mov bx,dx ;余数给dx

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,1d ;个位

mov ax,bx ;拷贝一份

mov dx,0 ;清空

div cx ;ax/cx

mov bx,dx ;余数给dx

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

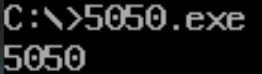
mov ah,4ch

int 21h

code ends

end start

**运行结果：**

****

**数据段方式：**

data segment

dw 0h

data ends

code segment

assume cs:code, ds:data

start:

mov bx,data ;段地址送入bx

mov ds,bx ;存放段地址

mov ah,02h ;输出方式

mov dx,1 ;当前加数

mov bx,0 ;和

mov ds:[0],bx;把和放入数据段

mov cx,100 ;计数器

loop1: ;循环开始

add bx,dx ;加

mov ds:[0],bx;把和放入数据段

inc dx ;自增

loop loop1 ;循环结束

;以下代码将bx中的值分解成千、百、十、个位，以字符形式输出

mov cx,1000d;千位

mov ax,ds:[0];内存直接寻址放入ax

mov dx,0 ;清空

div cx ;ax/cx

mov ds:[0],dx;余数给dx，放入数据段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,100d ;百位

mov ax,ds:[0];内存直接寻址放入ax

mov dx,0 ;清空

div cx ;ax/cx

mov ds:[0],dx;余数给dx，放入数据段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,10d ;十位

mov ax,ds:[0];内存直接寻址放入ax

mov dx,0 ;清空

div cx ;ax/cx

mov ds:[0],dx;余数给dx，放入数据段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,1d ;个位

mov ax,ds:[0];内存直接寻址放入ax

mov dx,0 ;清空

div cx ;ax/cx

mov ds:[0],dx;余数给dx，放入数据段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov ah,4ch

int 21h

code ends

end start

**运行结果：**

****

**栈方式：**

stk segment

dw 0h

stk ends

code segment

assume cs:code, ss:stk

start:

mov bx,stk ;段地址送入bx

mov ss,bx ;存放段地址

mov sp,0 ;栈指针

mov ah,02h ;输出方式

mov dx,1 ;当前加数

mov bx,0 ;和

push bx ;把和推入栈段

mov cx,100 ;计数器

loop1: ;循环开始

pop bx ;弹出当前和

add bx,dx ;加

push bx ;把和推入栈段

inc dx ;自增

loop loop1 ;循环结束

;以下代码将栈顶的值分解成千、百、十、个位，以字符形式输出

mov cx,1000d;千位

pop ax ;和放入ax

mov dx,0 ;清空

div cx ;ax/cx

push dx ;余数给dx，推入栈段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,100d ;百位

pop ax ;和放入ax

mov dx,0 ;清空

div cx ;ax/cx

push dx ;余数给dx，推入栈段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,10d ;十位

pop ax ;和放入ax

mov dx,0 ;清空

div cx ;ax/cx

push dx ;余数给dx，推入栈段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

mov cx,1d ;个位

pop ax ;和放入ax

mov dx,0 ;清空

div cx ;ax/cx

push dx ;余数给dx，推入栈段

mov dl,al ;一位数字

add dl,30h ;变成ASCII

mov ah,02 ;输出形式

int 21h ;打印

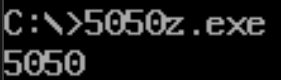
mov ah,4ch

int 21h

code ends

end start

**运行结果：**

****

2）.model small

.stack 100h

.data

prompt db 'Please enter a positive integer: $'

result\_msg db 'The sum is: $'

newline db 0Dh, 0Ah, '$'

num db 5 dup(0) ; 存储用户输入

sum dw 0 ; 累加和

digit db ? ; 存储每个数字字符

.code

main:

; 初始化段寄存器

mov ax, @data

mov ds, ax

; 提示用户输入

mov ah, 09h

lea dx, prompt

int 21h

; 读取输入 (最多允许5个数字)

mov bx, 0 ; BX 用作索引

input\_loop:

mov ah, 01h ; 等待并读取单个字符

int 21h

cmp al, 0Dh ; 检查是否是回车

je end\_input ; 如果是回车，结束输入

; 存储非回车字符

sub al, '0' ; 将字符转为数字

mov num[bx], al ; 存入缓冲区

inc bx ; 增加索引

cmp bx, 5 ; 检查是否超过输入长度

je end\_input

jmp input\_loop

end\_input:

; 转换字符串为整数

mov cx, 0 ; 清空cx (cx将存储n)

mov bx, 0 ; 重新设置索引为0

next\_digit:

cmp bx, 5 ; 检查是否处理完所有字符

je calculate\_sum

mov al, num[bx] ; 取出下一个数字

inc bx

mov ah, 0

mov si, 10

mul si ; 乘以10

add cx, ax ; 累加到CX

jmp next\_digit

calculate\_sum:

; 计算1到n的累加和

mov bx, cx ; 保存n

mov ax, 0 ; 初始化累加和

mov si, 1 ; 从1开始

sum\_loop:

add ax, si ; 累加当前的si

inc si ; 增加si

cmp si, bx ; 检查是否超过n

jg display\_sum ; 如果si > n，跳转到显示结果

jmp sum\_loop ; 否则继续循环

display\_sum:

; 输出结果

mov ah, 09h

lea dx, result\_msg

int 21h

; 输出累加和

call print\_number

; 换行

mov ah, 09h

lea dx, newline

int 21h

; 返回到 DOS

mov ah, 4Ch

int 21h

print\_number: ; 输出AX中的值 (最多5位)

mov bx, 10 ; 用于除法

mov cx, 0 ; 清空CX用于存储数字的位数

print\_loop:

xor dx, dx ; 清空DX

div bx ; AX / 10

push dx ; 保存余数

inc cx ; 统计位数

test ax, ax

jnz print\_loop

output\_digits:

pop dx ; 取出数字

add dl, '0' ; 转换为字符

mov ah, 02h ; 输出单个字符

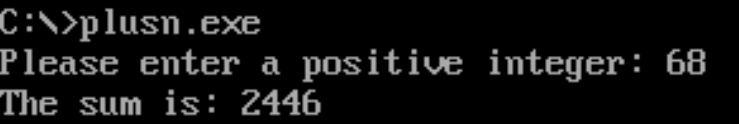
int 21h

loop output\_digits

ret

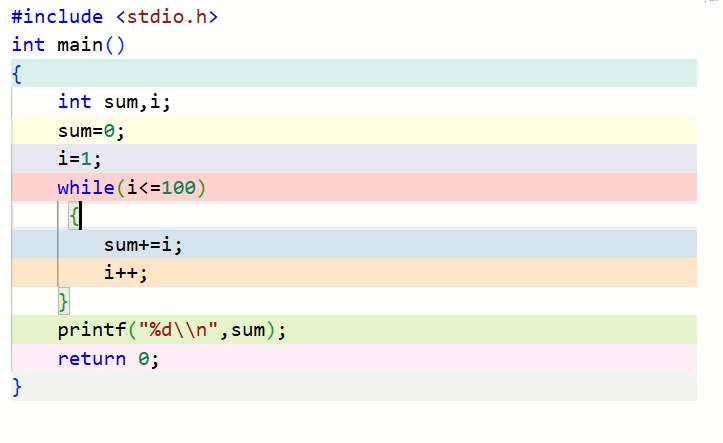
end main

**运行结果：**

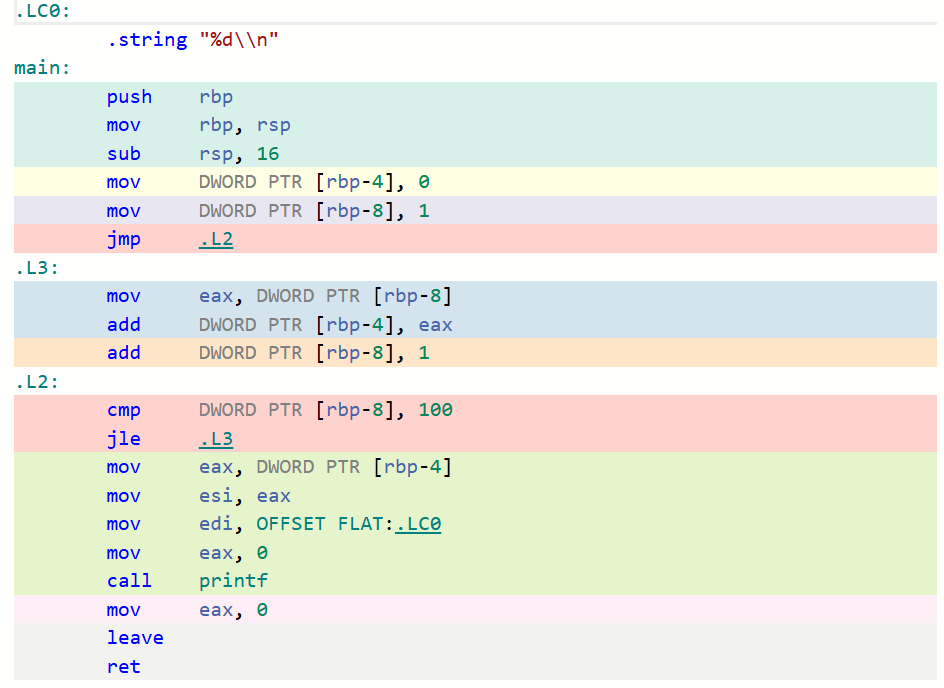
****

3）

**C语言实现：**



**反汇编：**

****

**注释：**

.LC0:

.string "%d\\n" # 字符串常量，用于printf函数格式化输出整数后换行

main: # 主函数入口点

push rbp # 保存旧的基指针

mov rbp, rsp # 设置新的基指针

sub rsp, 16 # 为局部变量分配栈空间

mov DWORD PTR [rbp-4], 0 # 初始化局部变量sum，用于存储累加结果

mov DWORD PTR [rbp-8], 1 # 初始化局部变量i，用于循环计数，从1开始

.L2: # 循环标签

cmp DWORD PTR [rbp-8], 100 # 比较i和100

jle .L3 # 如果i小于等于100，跳转到.L3继续循环

jmp .L4 # 如果i大于100，跳转到.L4结束循环

.L3: # 循环体

mov eax, DWORD PTR [rbp-8] # 将i的值移动到eax寄存器

add DWORD PTR [rbp-4], eax # 将eax的值加到sum上

add DWORD PTR [rbp-8], 1 # i加1

jmp .L2 # 跳转回.L2继续循环

.L4: # 循环结束后的代码

mov eax, DWORD PTR [rbp-4] # 将sum的值移动到eax寄存器

mov esi, eax # 将sum的值复制到esi寄存器，为printf准备

mov edi, OFFSET FLAT:.LC0 # 加载格式化字符串的地址到edi寄存器

mov eax, 0 # 将eax寄存器清零，为printf准备

call printf # 调用printf函数打印sum的值

mov eax, 0 # 设置函数返回值为0

leave # 恢复rbp和rsp寄存器

ret # 返回调用者