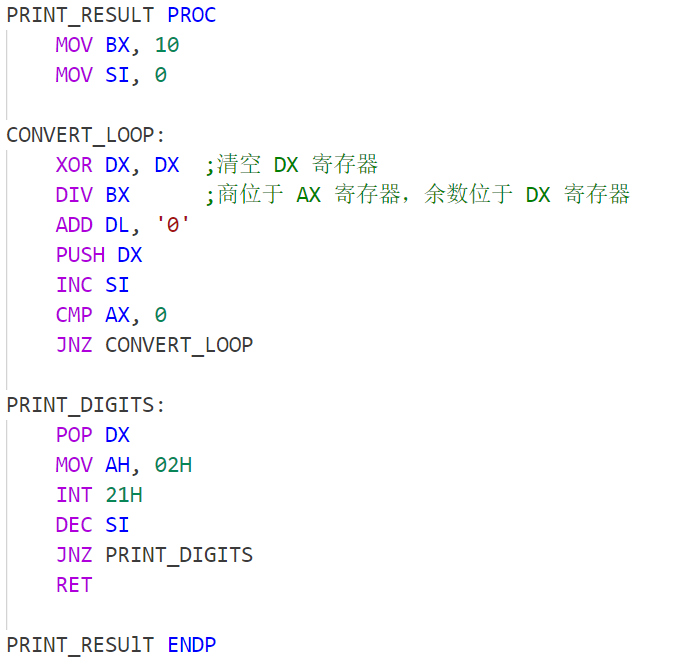
**求 1+2+......+100，并将结果“5050”打印到屏幕**

1. 该作业的基本要求是通过汇编程序计算 1 到 100 的累加和，并将最终的结果（即 5050）打印到屏幕。重点在于：

结果要存储在不同位置：寄存器、数据段和栈。

最终结果需要进行十进制转换后输出。

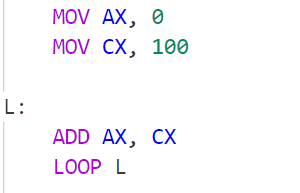
本作业使用不同的存储方式来保存中间结果。在每个版本中，PRINT\_RESULT 子程序负责将累加和转换为十进制字符串并打印。



PRINT\_RESULT 子程序将累加结果转换为十进制字符串并逐字符打印。通过 DIV BX 实现除 10 操作，将余数转换为 ASCII 码并存入栈，最后逐个从栈中弹出并打印。

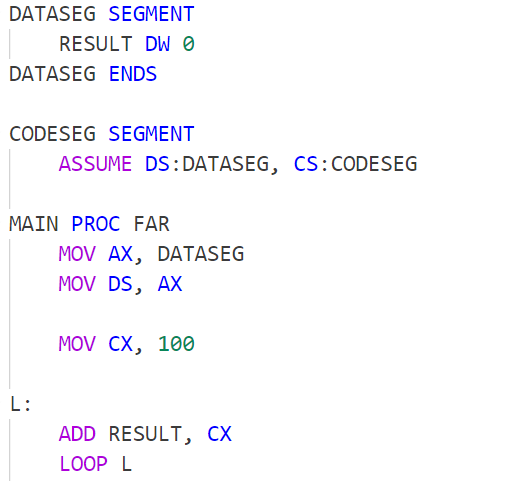
1. 结果存储在寄存器中

程序使用 AX 寄存器存储累加结果，CX 作为循环控制变量。每次将 CX 的值加到 AX，并在每次循环后递减 CX。累加和存储在 AX 寄存器中。



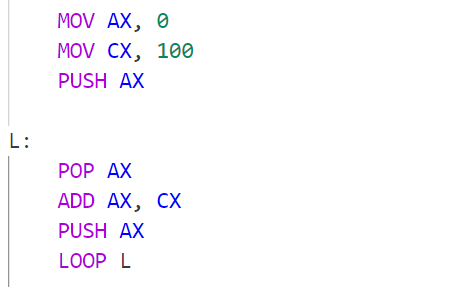
1. 结果存储在数据段中

累加和存储在 RESULT 变量中。



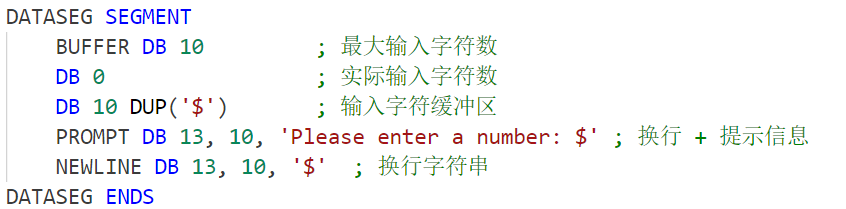
1. 结果存储在栈中

所有的累加结果都保存在栈中。每次累加前，程序先从栈中弹出当前累加和，将新的和计算后再次压入栈。



1. 用户输入 1~100 内的任何一个数，完成十进制结果输出

该程序使用 DOS 中断调用来读取用户输入，并完成十进制结果输出。用户输入一个1到100的数字，程序会读取用户输入并回显输入的内容。本程序使用 INT 21H 中断调用来处理输入和输出。



BUFFER：用户输入的缓冲区，前两个字节分别用于保存最大输入长度和实际输入长度。其后的字符区域用于存储输入的内容。

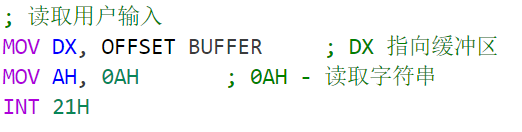
PROMPT：提示信息，提示用户输入一个数字。

NEWLINE：用于打印换行符。

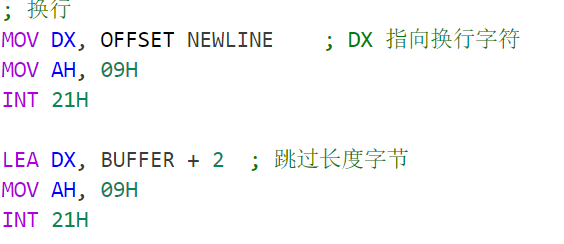
本程序通过 INT 21H 的 09H 功能调用显示提示信息，使用 LEA DX, PROMPT 将提示信息的地址加载到 DX，然后通过 INT 21H 调用打印。



之后，使用 INT 21H 的 0AH 功能读取用户输入的字符串

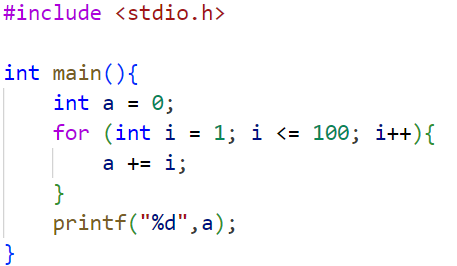


读取完输入后，程序会打印一个换行符，以便格式化输出，之后使用 INT 21H 的 09H 功能打印出用户输入的内容



1. 用C语言实现后察看反汇编代码并加注释

求和程序C代码：



反汇编

0x0040146e c7 44 24 1c 00 00 00 00   mov DWORD PTR [esp+0x1c],0x0    ; 将 [esp+0x1c] 的值设置为 0，初始化变量 sum = 0

0x00401476 c7 44 24 18 01 00 00 00   mov DWORD PTR [esp+0x18],0x1    ; 将 [esp+0x18] 的值设置为 1，初始化变量 i = 1

0x0040147e eb 0d                     jmp 0x40148d                    ; 跳转到 0x40148d，进入主循环检查部分

0x00401480 8b 44 24 18               mov eax,DWORD PTR [esp+0x18]    ; 将 i 的值（[esp+0x18]）加载到 eax 中

0x00401484 01 44 24 1c               add DWORD PTR [esp+0x1c],eax    ; 将 i 的值加到 sum（[esp+0x1c]）中

0x00401488 83 44 24 18 01            add DWORD PTR [esp+0x18],0x1    ; 将 i 的值自增（i++）

0x0040148d 83 7c 24 18 64            cmp DWORD PTR [esp+0x18],0x64   ; 比较 i 是否等于 100（十六进制的 0x64）

0x00401492 7e ec                     jle 0x401480                    ; 如果 i <= 100，跳回到 0x401480 继续累加

0x00401494 8b 44 24 1c               mov eax,DWORD PTR [esp+0x1c]    ; 将 sum 的值加载到 eax 中

0x00401498 89 44 24 04               mov DWORD PTR [esp+0x4],eax     ; 将 sum 存入栈中以便之后调用

0x0040149c c7 04 24 50 40 00 00      mov DWORD PTR [esp],0x405050    ; 将地址 0x405050 （可能是格式化字符串地址）存入 esp（传递给 printf）

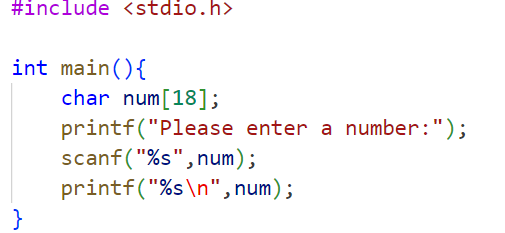
0x004014a3 e8 d8 25 00 00            call 0x403a80 <printf>          ; 调用 printf 函数，打印结果

0x004014a8 b8 00 00 00 00            mov eax,0x0                     ; 将 eax 设置为 0，返回值

0x004014ad c9                        leave                           ; 恢复栈指针

0x004014ae c3                        ret                             ; 返回主函数

打印程序C代码：



反汇编：

0x00401469 e8 72 05 00 00            call 0x4019e0 <\_\_main>          ; 调用 main 函数的准备阶段或主程序入口的预处理

0x0040146e c7 04 24 64 50 40 00      mov DWORD PTR [esp], 0x405064   ; 将字符串常量内存地址 0x405064 压入栈中

0x00401475 e8 16 26 00 00            call 0x403a90 <printf>          ; 调用 printf 函数输出格式化字符串

0x0040147a 8d 44 24 1e               lea eax, [esp+0x1e]             ; 计算栈中偏移地址 [esp+0x1e]，并将其加载到 eax

0x0040147e 89 44 24 04               mov DWORD PTR [esp+0x4], eax    ; 将 eax 的值（偏移地址）存入栈中 [esp+0x4]

0x00401482 c7 04 24 7b 50 40 00      mov DWORD PTR [esp], 0x40507b   ; 将字符串常量内存地址 0x40507b 压入栈中

0x00401489 e8 ea 25 00 00            call 0x403a78 <scanf>           ; 调用 scanf 函数读取用户输入

0x0040148e 8d 44 24 1e               lea eax, [esp+0x1e]             ; 计算栈中偏移地址 [esp+0x1e]，并将其加载到 eax

0x00401492 89 04 24                  mov DWORD PTR [esp], eax        ; 将 eax 的值存入栈顶，用于后续 puts 函数调用

0x00401495 e8 ee 25 00 00            call 0x403a88 <puts>            ; 调用 puts 函数输出字符串

0x0040149a b8 00 00 00 00            mov eax, 0x0                    ; 将 eax 设置为 0，返回值

0x0040149f c9                        leave                           ; 恢复栈指针

0x004014a0 c3                        ret                             ; 函数结束