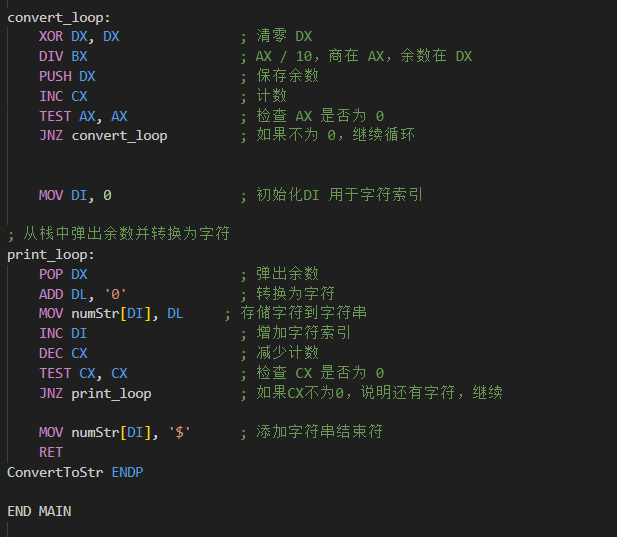
1. 求和
2. 结果存在寄存器中





这段代码首先初始化数据段寄存器，使用CX寄存器控制循环，从100递减至1，并将每次循环的值累加到AX寄存器中。累加完成后，通过调用ConvertToStr函数，将计算结果转换为字符串。该函数利用除法操作将数字按位分解，并通过栈操作逐位存入字符串。最后，程序调用DOS中断21H的功能号09H，将存储的字符串输出到屏幕。程序执行完毕后，使用DOS中断正常退出。

1. 结果存在内存中

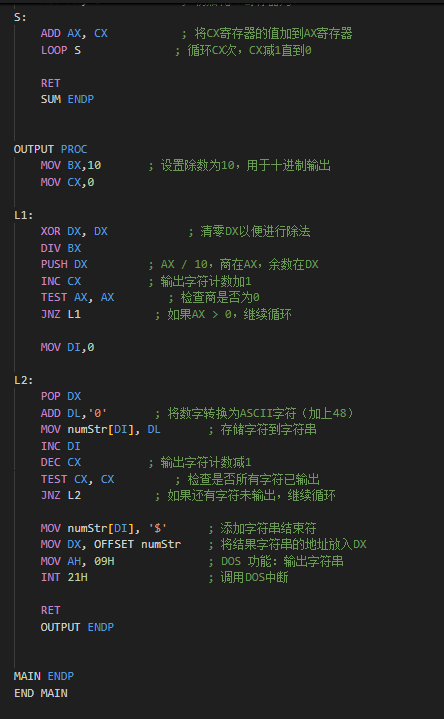
这种方法实现过程与第一种方法几乎相同，只是会把AX寄存器中累加的结果存在内存中。

1. 结果存在栈中

这种方法实现过程与第一种方法也几乎相同，只是会把AX寄存器中累加的结果压入栈中保存。

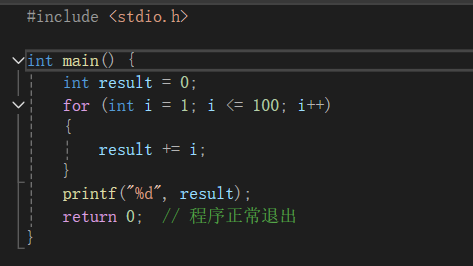
二．求和（输入数字）





程序首先初始化数据段寄存器，并通过调用 INPUT 过程来获取用户输入的数字。在 INPUT 过程中，程序逐个读取字符，直到检测到回车键（Enter），将输入的每个字符转换为相应的数字，并根据当前位置累加到结果中，最终将结果保存在BX寄存器中。然后回调用SUM过程进行求和，并把结果存在AX寄存器中。最后程序调用 OUTPUT 过程输出结果。在 OUTPUT 过程中，程序使用除法将数字逐位转换为字符形式，通过推入栈来保存每位数字，然后再逐位弹出并转换为字符串，最终通过DOS中断将字符逐个输出到控制台。程序最后通过调用DOS中断正常退出。

1. 求和（C语言实现）



反汇编结果：

00007FF77F481890 push rbp

;将rbp（基址指针）压入栈中，保存调用者的栈帧基址

00007FF77F481892 push rdi

;将rdi寄存器压入栈中，保存它的值（一般用于函数调用）

00007FF77F481893 sub rsp,128h

;将栈指针rsp减少128h（即304字节），为局部变量分配栈空间

00007FF77F48189A lea rbp,[rsp+20h]

;将rbp设置为rsp+20h，建立栈帧的基址

00007FF77F48189F lea rcx,[\_\_0FE406C9\_print\_sum@c (07FF77F492008h)]

00007FF77F4818A6 call \_\_CheckForDebuggerJustMyCode (07FF77F48136Bh)

00007FF77F4818AB nop

00007FF77F4818AC mov dword ptr [result],0

;将局部变量result初始化为0

00007FF77F4818B3 mov dword ptr [rbp+24h],1

; 将局部变量i初始化为1

00007FF77F4818BA jmp main+34h (07FF77F4818C4h)

; 跳转到循环条件判断的位置

00007FF77F4818BC mov eax,dword ptr [rbp+24h]

; 将变量i的值加载到eax寄存器中

00007FF77F4818BF inc eax

; 将eax的值加1

00007FF77F4818C1 mov dword ptr [rbp+24h],eax

; 更新变量i的值

00007FF77F4818C4 cmp dword ptr [rbp+24h],64h

; 比较i的值是否大于100

00007FF77F4818C8 jg main+49h (07FF77F4818D9h)

; 如果i > 100，则跳出循环

00007FF77F4818CA mov eax,dword ptr [rbp+24h]

; 将i的值加载到eax寄存器中

00007FF77F4818CD mov ecx,dword ptr [result]

; 将result的值加载到ecx寄存器中

00007FF77F4818D0 add ecx,eax

; 将i的值加到result中

00007FF77F4818D2 mov eax,ecx

; 更新eax为新的result值

00007FF77F4818D4 mov dword ptr [result],eax

; 将result的值保存回内存

00007FF77F4818D7 jmp main+2Ch (07FF77F4818BCh)

; 跳回循环起始处

00007FF77F4818D9 mov edx,dword ptr [result]

; 将result的值加载到edx寄存器中

00007FF77F4818DC lea rcx,[string "%d" (07FF77F48AC24h)]

; 加载格式化字符串到rcx中

00007FF77F4818E3 call printf (07FF77F481195h)

; 调用printf函数打印result的值

00007FF77F4818E8 nop

00007FF77F4818E9 xor eax,eax

; 将eax设置为0，表示返回值为0

00007FF77F4818EB lea rsp,[rbp+108h]

; 恢复栈指针rsp的值

00007FF77F4818F2 pop rdi

; 恢复rdi寄存器的值

00007FF77F4818F3 pop rbp

; 恢复之前的栈帧基址

00007FF77F4818F4 ret

; 返回，结束main函数

程序开始时，通过压入基址指针（rbp）和rdi寄存器的值来保存当前栈帧的状态，并为局部变量分配栈空间。

接着，局部变量result被初始化为0，而循环变量i则初始化为1。

然后程序进入一个循环，通过将i的值加1并与100进行比较来判断是否继续循环。在循环内部，i的值被累加到result中，直至i的值超过100，循环结束。此时，程序将累加结果加载到edx寄存器，并调用printf函数打印输出。

最后，程序通过设置返回值为0来标识正常退出，并恢复栈指针和寄存器的状态，完成主函数的返回。

1. 总结
2. 直接用汇编语言完成的程序，变量的管理大多是通过寄存器和内存地址直接进行，二反汇编则会涉及到更为复杂的数据管理，例如使用栈帧管理局部变量，所以反汇编生成的代码往往会有更多的入栈和出栈操作，同时也涉及函数调用时的参数传递和返回值处理。
3. 直接用汇编语言完成需要调用DOS中断来实现输出，所以涉及到多位数字的输出就很不方便，需要逐位转化为字符并压入栈中，再逐位弹出转化为字符串，最后输出字符串。
4. 通过这次实验，我学会了使用DOS中断（INT 21H）实现输入、输出字符、输出字符串以及结束程序等功能。这些功能在编写汇编程序时是非常重要的，为程序的用户交互提供了基础。