# 汇编语言课程作业

**————报告3**

2251334 倪朗恩

## 基本要求

求1+2+…+100，并将结果“5050”打印到屏幕

1. 尝试结果放在寄存器中、数据段中、栈中等不同位置的操作
2. 用户输入1-100内的任何一个数，完成十进制结果输出（查找21号中断的功能表，找到输入数据的功能调用）
3. 用C语言编写后查看返汇编代码并加注释

## 汇编编写求和

### 框架

使用COUNT作为求和总数，将代码大致分为三个段：

1. SUM

    MOV CX,[COUNT]

    XOR AX,AX

SUM:

    ADD AX,CX

    LOOP SUM

求和计算

1. TRANSF/INSTK

获取每位的数字/字符，并进行存储，根据结果位置不同而有差别

1. OUTPUT

以字符形式输出结果，根据结果位置不同而有差别

1. 循环后操作

    MOV AX,4C00H

    INT 21H

函数返回

### 具体细节

1. 使用寄存器
2. TRANSF

    MOV CX,AX

    MOV BX,1

TRANSF:

    XCHG AX,BX

    MOV DX,10

    MUL DX

    XCHG AX,BX

    DIV BX

    CMP AX,0

    MOV AX,CX

    JNE TRANSF

循环求出位数（10的幂次），用于OUTPUT除法

1. OUTPUT

OUTPUT:

    XCHG AX,BX

    XCHG BX,CX

    MOV BX,10

    XOR DX,DX

    DIV BX

    XCHG BX,CX

    XCHG AX,BX

    XOR DX,DX

    DIV BX

    XCHG AX,DX

    ADD DL,30H

    XCHG AX,CX

    MOV AH,2

    INT 21H

    XCHG AX,CX

    CMP BX,1

    JNE OUTPUT

对余数进行除法，依次降低除数并对每位进行输出

1. 使用数据段
2. TRANSF

DATASEG SEGMENT

    STRING DB 32 DUP(0)

    COUNT DW 100

    ANS DW 0

DATASEG ENDS

TRANSF:

    MOV BL,10

    XOR DX,DX

    DIV BX

    MOV BX,CX

    ADD DL,30H

    MOV [STRING+BX],DL

    INC CX

    CMP AX,0

    JNE TRANSF

循环除10获得每位数字/字符并计数

1. OUTPUT

    XOR DX,DX

    MOV AH,2

    MOV BX,CX

OUTPUT:

    DEC BX

    MOV DL,[STRING+BX]

    INT 21H

    LOOP OUTPUT

循环输出（从高位到低位）

1. 使用栈段
2. INSTK

    XOR CX,CX

    XOR BX,BX

    MOV BL,10

INSTK:

    XOR DX,DX

    DIV BX

    ADD DL,30H

    PUSH DX

    INC CX

    CMP AX,0

    JNE INSTK

循环除10获得每位数字/字符并入栈

1. OUTPUT

    MOV AH,2

OUTPUT:

    POP DX

    INT 21H

    LOOP OUTPUT

退栈并输出

## 汇编编写输入输出

DATASEG SEGMENT

    STRING DB 32

DATASEG ENDS

    XOR AX,AX

    XOR BX,BX

    XOR CX,CX

    XOR DX,DX

    MOV AH,1

INPUT:

    INT 21H

    CMP AL,0DH

    JE OUTPUT

    MOV DS:[BX],AL

    INC BX

    ADD CX,2

    LOOP INPUT

是对输入输出的综合应用，本代码使用了数据段进行输入的存储，其中，CX保存了输入数目，之后使用循环输出（数据段能保存32位输入）

OUTPUT:

    XOR BX,BX

    XOR DX,DX

    MOV AH,2

L:

    MOV DL,DS:[BX]

    INC BX

    INT 21H

    LOOP L

## C语言编写

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

int main()

{

    int ans=0;

    for(int i=1;i<=100;++i)

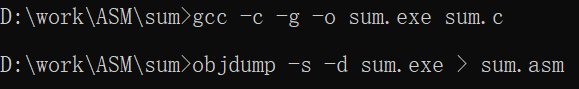
        ans+=i;

    printf("%d\n",ans);

    return 0;

}

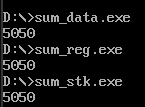
使用简单的循环和输出函数实现



在cmd窗口输入以上命令得到汇编代码（使用了Mingw工具），并加以注释，详见sum.asm

## 实验结果

1. 汇编编写求和



1. 汇编编写输入输出



1. C语言编写

Disassembly of section .text:

0000000000000000 <main>:

   0: 55                      push   %rbp                                     ;将%rbp寄存器的值入栈,以便之后恢复

   1: 48 89 e5                mov    %rsp,%rbp                                ;将%rsp(栈指针寄存器)的值复制到rbp中，从而更新rbp为新的栈帧基址

   4: 48 83 ec 30             sub    $0x30,%rsp                               ;从%rsp中减去30H/48个字节(在栈上分配48B的空间，用于存放局部变量或其他临时数据)

   8: e8 00 00 00 00          callq  d <main+0xd>                             ;调用函数main

   d: c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)                          ;将立即数（$0x0）存储到相对于%rbp指向位置偏移-0x4处，ans

  14: c7 45 f8 01 00 00 00    movl   $0x1,-0x8(%rbp)                          ;将立即数（$0x1）存储到相对于%rbp指向位置偏移-0x8处，i

  1b: eb 0a                   jmp    27 <main+0x27>                           ;无条件跳转,进行for循环的条件判断

  1d: 8b 45 f8                mov    -0x8(%rbp),%eax                          ;%eax=i

  20: 01 45 fc                add    %eax,-0x4(%rbp)                          ;ans+=%eax

  23: 83 45 f8 01             addl   $0x1,-0x8(%rbp)                          ;++i

  27: 83 7d f8 64             cmpl   $0x64,-0x8(%rbp)                         ;比较100和i

  2b: 7e f0                   jle    1d <main+0x1d>                           ;i<=100,继续循环,跳转到1d

  2d: 8b 45 fc                mov    -0x4(%rbp),%eax                          ;ans=%eax

  30: 89 c2                   mov    %eax,%edx                                ;%edx=%eax

  32: 48 8d 0d 00 00 00 00    lea    0x0(%rip),%rcx        # 39 <main+0x39>   ;计算RIP（指令指针）加上偏移量0后的有效地址，并将结果存入寄存器%rcx

  39: e8 00 00 00 00          callq  3e <main+0x3e>                           ;调用函数printf

  3e: b8 00 00 00 00          mov    $0x0,%eax                                ;%eax=0

  43: 48 83 c4 30             add    $0x30,%rsp                               ;%rsp+=30H,回收字节

  47: 5d                      pop    %rbp                                     ;从栈顶弹出值，恢复%rbp

  48: c3                      retq                                            ;从当前函数返回到调用者，并从栈中弹出返回地址，恢复%rip（指令指针寄存器）

  49: 90                      nop                                             ;空操作/占位符

  4a: 90                      nop

  4b: 90                      nop

  4c: 90                      nop

  4d: 90                      nop

  4e: 90                      nop

  4f: 90                      nop

## 问题与心得

1. 除法报错

DIV使用的是扩展的DX：AX数进行除法，如果被除数过大（如DX没有清0），则会进入异常（debug看见执行？？？指令），并且死循环

心得：不需要扩展时，使用DIV前要保存并清零DX，且尽量使用不冲突的寄存器进行除法

1. 出栈与入栈要保持类型相同

能用AX入栈，然后要求DX出栈，但是实际操作过程中要注意顺序，尤其注意函数调用本身有出入栈

1. 注意输入输出顺序以及计数
2. Demo先行

当进行多次类似的编程时，可以先编写demo文件作为框架，确定思想（本次作业编写了demo文件sum\_base.asm）

1. 选择合适的存储方式

可以依据输入输出、程序需求，针对性的选择栈、数据段、寄存器进行编写，合适的存储能提高效率以及精简编程