本次作业实现思路，使用数据段、栈、寄存器来记录累加的值，先循环 100 次完成累加，再将 5050 进行循环除 10，将得到的结果保存到栈中，再通过栈中结果加'0'完成字符的输出

源码一：使用数据段的方式来保留结果

```
MY SEGMENT
   A DW 1
   SUM DW 0 ;结果放到数据段中的做法
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
    MOV CX,100
    MOV BL,0
L:
    MOV AX,A
    ADD SUM,AX
    INC A
    LOOP L
    ; 先进行求和计算
    MOV CX,5

    MOV AX,SUM
    ; 使用栈存来输出 NN



CLEAR_STACK:
    POP DX          ; 弹出栈顶元素
    LOOP CLEAR_STACK ; 循环直到 CX 为 0
L2:
    XOR DX,DX
    MOV BX,10
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNZ L2
```

```
L3:
    POP DX
    ADD DL,'0'
    MOV AH,2
    INT 21H
    LOOP L3
```

```
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

使用寄存器保留数据的结果：每次完成加法操作后就将得到的值保存在 AX 寄存器当中

```
MY SEGMENT
   A DW 1
;    放到寄存器的做法
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
    MOV CX,100
    MOV BL,0
    MOV AX,0
L:
    ADD AX,A
    INC A
    LOOP L
    ; 先进行求和计算
    MOV CX,5

    ; 使用栈存来输出 NN


CLEAR_STACK:
    POP DX          ; 弹出栈顶元素
    LOOP CLEAR_STACK ; 循环直到 CX 为 0
L2:
    XOR DX,DX
    MOV BX,10
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNZ L2
```

```
L3:
    POP DX
```

```
    ADD DL,'0'
    MOV AH,2
    INT 21H
    LOOP L3
```

```
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

结果保留在栈中做法：先将初始值 0 放到栈中，再通过每次从栈中弹出，完成加法操作后再放回栈中。

```
MY SEGMENT
   A DW 1
;结果放到栈中的做法
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
    MOV CX,100
    MOV BL,0
    MOV AX,0
    PUSH AX
L:
    POP AX
    ADD AX,A
    INC A
    PUSH AX
    LOOP L
    ; 先进行求和计算
    MOV CX,5
    ; 使用栈存来输出



CLEAR_STACK:
    POP DX          ; 弹出栈顶元素
    LOOP CLEAR_STACK ; 循环直到 CX 为 0
L2:
    XOR DX,DX
    MOV BX,10
    DIV BX
```

```
    PUSH DX
    INC CX
    CMP AX, 0
    JNZ L2
```

```
L3:
    POP DX
    ADD DL,'0'
    MOV AH,2
    INT 21H
    LOOP L3
```

```
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

根据用户输入来计算累加结果：使用 ah 1 来实现用户输入，通过判断是否输入为回车符来结束输入，将每次输入的字符与后续字符合成数字后再进行累加操作。

```

MY SEGMENT
    A DW 1
    SUM DW 0 ;结果放到数据段中的做法
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
```

```
    MOV BL,0 ;当前输入位之前的结果
    MOV CL,10
INPUT:
    MOV AH,1
    INT 21H

    CMP AL, 0Dh          ; 判断是否为回车键
    JE OVER

    SUB AL,48
    MOV DL,AL
    MOV DH,0
    MOV AL,BL
```

```
    MUL CL
    ADD AX,DX
    MOV BX,AX
    JMP INPUT
```

```
OVER:
    MOV CX,BX
    MOV BL,0
L:
    MOV AX,A
    ADD SUM,AX
    INC A
    LOOP L
    ; 先进行求和计算
    MOV AX,SUM
    MOV CX,5

CLEAR_STACK:
    POP DX          ; 弹出栈顶元素
    LOOP CLEAR_STACK ; 循环直到 CX 为 0
L1:
    XOR DX,DX
    MOV BX,10
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNZ L1
```

```
L2:
    POP DX
    ADD DL,'0'
    MOV AH,2
    INT 21H
    LOOP L2
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

C 语言实现代码：

```
#include<stdio.h>
#define _CRT_SECURE_NO_WARNINGS
int main()
{

    int a=0;
    int sum = 0;
    scanf("%d", &a);
    for (int i = 1; i <= a; i++)
    {
        sum += i;
    }
    printf("%d\n", sum);
    return 0;
}
```

反汇编结果：

```
Disassembly of section .init:

0000000000001000 <_init>:
    1000:       f3 0f 1e fa             endbr64
    1004:       48 83 ec 08             sub    $0x8,%rsp
    1008:         48 8b 05 d9 2f 00 00      mov     0x2fd9(%rip),%rax          # 3fe8
<__gmon_start__@Base>
    100f:       48 85 c0                test   %rax,%rax
    1012:       74 02                   je     1016 <_init+0x16>
    1014:       ff d0                   call   *%rax
    1016:       48 83 c4 08             add    $0x8,%rsp
    101a:       c3                      ret

Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:          ff 35 8a 2f 00 00          push    0x2f8a(%rip)          # 3fb0
<_GLOBAL_OFFSET_TABLE_+0x8>
```

```
    1026:          ff 25 8c 2f 00 00          jmp      *0x2f8c(%rip)           # 3fb8
<_GLOBAL_OFFSET_TABLE_+0x10>
    102c:       0f 1f 40 00             nopl   0x0(%rax)
    1030:       f3 0f 1e fa             endbr64
    1034:       68 00 00 00 00          push   $0x0
    1039:       e9 e2 ff ff ff          jmp    1020 <_init+0x20>
    103e:       66 90                   xchg   %ax,%ax
    1040:       f3 0f 1e fa             endbr64
    1044:       68 01 00 00 00          push   $0x1
    1049:       e9 d2 ff ff ff          jmp    1020 <_init+0x20>
    104e:       66 90                   xchg   %ax,%ax
    1050:       f3 0f 1e fa             endbr64
    1054:       68 02 00 00 00          push   $0x2
    1059:       e9 c2 ff ff ff          jmp    1020 <_init+0x20>
    105e:       66 90                   xchg   %ax,%ax

Disassembly of section .plt.got:

0000000000001060 <__cxa_finalize@plt>:
    1060:       f3 0f 1e fa             endbr64
    1064:          ff 25 8e 2f 00 00          jmp      *0x2f8e(%rip)           # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    106a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

0000000000001070 <__stack_chk_fail@plt>:
    1070:       f3 0f 1e fa             endbr64
    1074:          ff 25 46 2f 00 00          jmp      *0x2f46(%rip)           # 3fc0
<__stack_chk_fail@GLIBC_2.4>
    107a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

0000000000001080 <printf@plt>:
    1080:       f3 0f 1e fa             endbr64
    1084:          ff 25 3e 2f 00 00          jmp      *0x2f3e(%rip)           # 3fc8
<printf@GLIBC_2.2.5>
    108a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

0000000000001090 <__isoc99_scanf@plt>:
    1090:       f3 0f 1e fa             endbr64
    1094:          ff 25 36 2f 00 00          jmp      *0x2f36(%rip)           # 3fd0
<__isoc99_scanf@GLIBC_2.7>
    109a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
Disassembly of section .text:

00000000000010a0 <_start>:
    10a0:       f3 0f 1e fa             endbr64
    10a4:       31 ed                   xor    %ebp,%ebp
    10a6:       49 89 d1                mov    %rdx,%r9
    10a9:       5e                      pop    %rsi
    10aa:       48 89 e2                mov    %rsp,%rdx
    10ad:       48 83 e4 f0             and    $0xfffffffffffffff0,%rsp
    10b1:       50                      push   %rax
    10b2:       54                      push   %rsp
    10b3:       45 31 c0                xor    %r8d,%r8d
    10b6:       31 c9                   xor    %ecx,%ecx
    10b8:       48 8d 3d ca 00 00 00    lea    0xca(%rip),%rdi        # 1189 <main>
    10bf:          ff  15  13  2f  00  00          call    *0x2f13(%rip)            # 3fd8
<__libc_start_main@GLIBC_2.34>
    10c5:       f4                      hlt
    10c6:       66 2e 0f 1f 84 00 00    cs nopw 0x0(%rax,%rax,1)
    10cd:       00 00 00

00000000000010d0 <deregister_tm_clones>:
    10d0:         48  8d  3d  39  2f  00  00     lea     0x2f39(%rip),%rdi          # 4010
<__TMC_END__>
    10d7:         48  8d  05  32  2f  00  00     lea     0x2f32(%rip),%rax          # 4010
<__TMC_END__>
    10de:       48 39 f8                cmp    %rdi,%rax
    10e1:       74 15                   je     10f8 <deregister_tm_clones+0x28>
    10e3:         48  8b  05  f6  2e  00  00     mov     0x2ef6(%rip),%rax          # 3fe0
<_ITM_deregisterTMCloneTable@Base>
    10ea:       48 85 c0                test   %rax,%rax
    10ed:       74 09                   je     10f8 <deregister_tm_clones+0x28>
    10ef:       ff e0                   jmp    *%rax
    10f1:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)
    10f8:       c3                      ret
    10f9:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001100 <register_tm_clones>:
    1100:         48  8d  3d  09  2f  00  00     lea     0x2f09(%rip),%rdi          # 4010
<__TMC_END__>
    1107:         48  8d  35  02  2f  00  00     lea     0x2f02(%rip),%rsi          # 4010
<__TMC_END__>
    110e:       48 29 fe                sub    %rdi,%rsi
    1111:       48 89 f0                mov    %rsi,%rax
    1114:       48 c1 ee 3f             shr    $0x3f,%rsi
```

```
    1118:       48 c1 f8 03             sar    $0x3,%rax
    111c:       48 01 c6                add    %rax,%rsi
    111f:       48 d1 fe                sar    $1,%rsi
    1122:       74 14                   je     1138 <register_tm_clones+0x38>
    1124:         48 8b 05 c5 2e 00 00      mov     0x2ec5(%rip),%rax          # 3ff0
<_ITM_registerTMCloneTable@Base>
    112b:       48 85 c0                test   %rax,%rax
    112e:       74 08                   je     1138 <register_tm_clones+0x38>
    1130:       ff e0                   jmp    *%rax
    1132:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
    1138:       c3                      ret
    1139:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001140 <__do_global_dtors_aux>:
    1140:       f3 0f 1e fa             endbr64
    1144:         80 3d c5 2e 00 00 00      cmpb    $0x0,0x2ec5(%rip)          # 4010
<__TMC_END__>
    114b:       75 2b                   jne    1178 <__do_global_dtors_aux+0x38>
    114d:       55                      push   %rbp
    114e:         48 83 3d a2 2e 00 00      cmpq    $0x0,0x2ea2(%rip)          # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    1155:       00
    1156:       48 89 e5                mov    %rsp,%rbp
    1159:       74 0c                   je     1167 <__do_global_dtors_aux+0x27>
    115b:         48 8b 3d a6 2e 00 00      mov     0x2ea6(%rip),%rdi          # 4008
<__dso_handle>
    1162:       e8 f9 fe ff ff          call   1060 <__cxa_finalize@plt>
    1167:       e8 64 ff ff ff          call   10d0 <deregister_tm_clones>
    116c:         c6 05 9d 2e 00 00 01      movb    $0x1,0x2e9d(%rip)          # 4010
<__TMC_END__>
    1173:       5d                      pop    %rbp
    1174:       c3                      ret
    1175:       0f 1f 00                nopl   (%rax)
    1178:       c3                      ret
    1179:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001180 <frame_dummy>:
    1180:       f3 0f 1e fa             endbr64
    1184:       e9 77 ff ff ff          jmp    1100 <register_tm_clones>

0000000000001189 <main>:
    1189:       f3 0f 1e fa             endbr64
    118d:       55                      push   %rbp
    118e:       48 89 e5                mov    %rsp,%rbp
```

```
    1191:       48 83 ec 20             sub    $0x20,%rsp
    1195:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    119c:       00 00
    119e:       48 89 45 f8             mov    %rax,-0x8(%rbp)
    11a2:       31 c0                   xor    %eax,%eax
    11a4:       c7 45 f0 00 00 00 00    movl   $0x0,-0x10(%rbp)
    11ab:       48 8d 45 ec             lea    -0x14(%rbp),%rax
    11af:       48 89 c6                mov    %rax,%rsi
    11b2:         48 8d 05 4b 0e 00 00      lea      0xe4b(%rip),%rax           # 2004
<_IO_stdin_used+0x4>
    11b9:       48 89 c7                mov    %rax,%rdi
    11bc:       b8 00 00 00 00          mov    $0x0,%eax
    11c1:       e8 ca fe ff ff          call   1090 <__isoc99_scanf@plt>
    11c6:       c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
    11cd:       eb 0a                   jmp    11d9 <main+0x50>
    11cf:       8b 45 f4                mov    -0xc(%rbp),%eax
    11d2:       01 45 f0                add    %eax,-0x10(%rbp)
    11d5:       83 45 f4 01             addl   $0x1,-0xc(%rbp)
    11d9:       8b 45 ec                mov    -0x14(%rbp),%eax
    11dc:       39 45 f4                cmp    %eax,-0xc(%rbp)
    11df:       7e ee                   jle    11cf <main+0x46>
    11e1:       8b 45 f0                mov    -0x10(%rbp),%eax
    11e4:       89 c6                   mov    %eax,%esi
    11e6:         48 8d 05 1a 0e 00 00      lea      0xe1a(%rip),%rax           # 2007
<_IO_stdin_used+0x7>
    11ed:       48 89 c7                mov    %rax,%rdi
    11f0:       b8 00 00 00 00          mov    $0x0,%eax
    11f5:       e8 86 fe ff ff          call   1080 <printf@plt>
    11fa:       b8 00 00 00 00          mov    $0x0,%eax
    11ff:       48 8b 55 f8             mov    -0x8(%rbp),%rdx
    1203:       64 48 2b 14 25 28 00    sub    %fs:0x28,%rdx
    120a:       00 00
    120c:       74 05                   je     1213 <main+0x8a>
    120e:       e8 5d fe ff ff          call   1070 <__stack_chk_fail@plt>
    1213:       c9                      leave
    1214:       c3                      ret
```

和上次反汇编结果类似的，使用 c 语言编写的程序在转化为汇编文件后会使用基指针方式来给变量分配空间，同时初始化变量。

除此之外，本次使用了输入输出函数，在反汇编后看到他需要将使用到的各个函数比如 stdin 和 printf 函数也进行相应的初始化处理，因此汇编代码显得十分冗长。