**汇编作业一**

**2151644 周一韬**

**作业一：按照要求打印ASCII表**

基本要求：输出ASCII表中的小写字母部分，要求每行13个字符

1. 用loop指令实现(test1-1)；
2. 用条件跳转指令实现(test1-2)；

1. 用loop指令实现(test1-1)

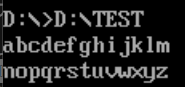
思路：使用loop循环来打印26个小写字母，以及在中间插入一个换行符，最后正常退出程序

实现代码如下：

|  |
| --- |
| Assembly language ; loop指令实现打印26个小写字母，并换行输出 CODESEG SEGMENT  ASSUME CS:CODESEG MAIN PROC FAR   MOV CX, 13  MOV AH, 2  MOV DL, 'a' ; 初始化  L:   INT 21H  INC DL  LOOP L   MOV AH, 2  MOV DL, 10 ; 打印换行符  INT 21H   MOV CX, 13 ; 重置计数器  MOV DL, 'n'   L2:  INT 21H  INC DL  LOOP L2   MOV AH, 4Ch  INT 21H  MAIN ENDP CODESEG ENDS  END MAIN |

其实现思路也很简单，就是我们从“a”开始打印，在打印之前设置cx为13，然后打印13个字符之后，打印换行符。这样我们就完成了a到m的打印，然后我们重新设置cx为13，再进行13次循环，打印后13个字符，即可完成题目要求。

实现结果如下：



1. 用条件跳转指令实现（test1-2）

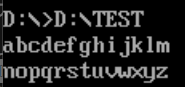
思路：类似于loop的实现，但是需要把loop循环改用跳转的指令实现

实现代码如下：

|  |
| --- |
| Assembly language ; 使用条件跳转实现打印26个小写字母，并换行输出 CODESEG SEGMENT  ASSUME CS:CODESEG MAIN PROC FAR   MOV CX, 13   MOV AH, 2  MOV DL, 'a'   L1:  INT 21H  INC DL  DEC CX   JNZ L1 ;如果CX不等于0，跳转到L1   ; 打印换行  MOV AH, 2  MOV DL, 10   INT 21H   MOV CX, 13 ;重置计数器  MOV DL, 'n'   L2:  INT 21H  INC DL  DEC CX   JNZ L2 ;如果CX不等于0，跳转到L2   MOV AH, 4Ch  INT 21H   MAIN ENDP CODESEG ENDS  END MAIN |

在这里我仅仅修改使用了JNZ指令来代替loop循环，JNZ的作用是判断cx是否等于0，如果不等于则跳转JNZ后面指向的位置，如果等于0则继续向下执行

实现结果如下：



**作业二：求和**

基本要求：求1+2+......+100,并将结果“5050”打印到屏幕

1. 结果放在寄存器中(test2-1)
2. 结果放在数据段中(test2-2)
3. 结果放在栈中(test2-3)
4. 用户输入1~100内的任何一个数，完成十进制结果输出(test2-4)

该作业的思路也比较清晰，主要的难点在于如何存放求和之后的值，以及如何把结果转换为10进制输出，本题的前三个要求实际上影响的都是存放数据问题，无论是存放在寄存器还是数据段还是栈中，最后其实对程序的影响不大。

1. 结果放在寄存器中（test2-1）

这是我们最容易想到的一种方法，直接把求和出来的结果存放在ax寄存器中，利用loop循环实现累加的过程

整个代码的实现分为两部分，一个是求和，一个是以十进制输出结果。

对于对于求和的实现，我们采用的是loop循环，cx从100开始，每次把cx的值加到寄存器ax中，这样最后进行100次循环之后，由于这一部分难度不大，不再过多赘述，实现部分代码如下所示：

|  |
| --- |
| Assembly language START:  MOV AX, DATA ; 将数据段地址加载到AX寄存器  MOV DS, AX ; 将数据段地址赋给DS寄存器  MOV AX, 0   MOV CX, 100 ; 循环100次  LOOP1:  ADD AX, CX ; 累加  LOOP LOOP1 |

最后我们寄存器ax中存放的就是我们的答案，但是我们不能直接输出。经过查阅相关资料后可知，如果想要输出“5050”，则意味着我们还要将其转换为字符串，以便将其显示出来，然后将其转换为10进制输出结果。DOS功能号9期望一个以终止字符（通常是"$"）结束的字符串作为输入，然后将其显示在屏幕上。如果直接尝试使用 INT 21H 来显示寄存器中的数值，它可能会将其解释为字符的ASCII码而不是数字的值，并以字符形式显示出来。

所以我们现在的思路是先把ax寄存器中的结果转换为十进制字符串

部分代码实现如下所示

|  |
| --- |
| Assembly language  ; 转换结果为十进制字符串  MOV CX, 10 ; 除数为10  MOV DI, 5 ; DI用于指向字符串的位置  MOV BYTE PTR [RESULT\_STRING + DI], '$' ; 设置字符串终止字符"$"  CONVERT\_LOOP:  XOR DX, DX ; 清空DX，存余数  DIV CX ; AX = AX / 10 (商), DX = AX % 10 (余数)  ADD DL, '0' ; 将余数转换为字符  DEC DI ; 移动到下一个字符串位置  MOV BYTE PTR [RESULT\_STRING + DI], DL ; 存储字符  TEST AX, AX ; 检查商是否为0  JNZ CONVERT\_LOOP ; 如果不为0，继续循环   ; 显示结果字符串  MOV AH, 9   MOV DX, OFFSET RESULT\_STRING ; DX指向要显示的字符串  INT 21H |

在这里我们定义最终存放的字符串是RESULT\_STRING，然后每次取最低位，放到字符串中，通过一个条件跳转实现转换，然后利用DOS的功能号9输出结果即可。

完整的代码如下所示

|  |
| --- |
| Assembly language ; 使用寄存器存放结果 DATA SEGMENT  RESULT\_STRING DB 6 DUP(?) ; 存放结果的字符串 DATA ENDS  STACK SEGMENT  DB 200 DUP(0) ; 定义栈段 STACK ENDS  CODE SEGMENT  ASSUME CS:CODE, DS:DATA, SS:STACK   START:  MOV AX, DATA ; 将数据段地址加载到AX寄存器  MOV DS, AX ; 将数据段地址赋给DS寄存器  MOV AX, 0   MOV CX, 100 ; 循环100次  LOOP1:  ADD AX, CX ; 累加  LOOP LOOP1    ; 转换结果为十进制字符串  MOV CX, 10 ; 除数为10  MOV DI, 5 ; DI用于指向字符串的位置  MOV BYTE PTR [RESULT\_STRING + DI], '$' ; 设置字符串终止字符"$"  CONVERT\_LOOP:  XOR DX, DX ; 清空DX  DIV CX ; AX = AX / 10 (商), DX = AX % 10 (余数)  ADD DL, '0' ; 将余数转换为字符  DEC DI ; 移动到下一个字符串位置  MOV BYTE PTR [RESULT\_STRING + DI], DL ; 存储字符  TEST AX, AX ; 检查商是否为0  JNZ CONVERT\_LOOP ; 如果不为0，继续循环   ; 显示结果字符串  MOV AH, 9   MOV DX, OFFSET RESULT\_STRING ; DX指向要显示的字符串  INT 21H   MOV AH, 4CH   INT 21H  CODE ENDS END START |

其结果如下所示



需要说明的是结果前面多了一个空格，这是因为我们使用输出结果的字符串是通过DOS功能号9来显示的，而DOS功能号9默认情况下在字符串前面添加一个空格。

1. 结果放在数据段中(test2-2)

这个要求的实现类似于存放于寄存器中，只是我们需要在开始之前额外设置一个变量SUM，来记录下最后的总和。

由于其具体实现类似于上一个，这里不再过多赘述

具体的代码如下所示

|  |
| --- |
| Assembly language ; 使用数据段存放 DATA SEGMENT  SUM DW ? ; 存放最终的累加和  RESULT\_STRING DB 6 DUP(?) ; 存放结果的字符串（新增部分） DATA ENDS  STACK SEGMENT  DB 200 DUP(0) ; 定义栈段 STACK ENDS  CODE SEGMENT  ASSUME CS:CODE, DS:DATA, SS:STACK   START:  MOV AX, DATA ; 将数据段地址加载到AX寄存器  MOV DS, AX ; 将数据段地址赋给DS寄存器  MOV AX, 0   MOV CX, 100 ; 循环100次  LOOP1:  ADD AX, CX ; 累加  LOOP LOOP1    MOV SUM, AX ; 存储最终累加和到SUM（新增部分）   ; 转换结果为十进制字符串  MOV AX, SUM ; 将累加和加载到AX  MOV CX, 10 ; 除数为10  MOV DI, 5 ; DI用于指向字符串的位置  MOV BYTE PTR [RESULT\_STRING + DI], '$' ; 设置字符串终止字符$  CONVERT\_LOOP:  XOR DX, DX ; 清空DX  DIV CX ; AX = AX / 10 (商), DX = AX % 10 (余数)  ADD DL, '0' ; 将余数转换为字符  DEC DI ; 移动到下一个字符串位置  MOV BYTE PTR [RESULT\_STRING + DI], DL ; 存储字符  TEST AX, AX ; 检查商是否为0  JNZ CONVERT\_LOOP ; 如果不为0，继续循环   ; 显示结果字符串  MOV AH, 9   MOV DX, OFFSET RESULT\_STRING ; DX指向要显示的字符串  INT 21H   MOV AH, 4CH   INT 21H  CODE ENDS END START |

1. 结果放在栈中(test2-3)

这个要求的实现类似于存放于寄存器中，只是我们需要在这段代码中，PUSH AX 用于将累加结果存储在栈中，然后在需要的时候使用 POP AX 从栈中弹出结果，并进行转换和显示。

由于其具体实现类似于上一个，这里不再过多赘述

具体的代码如下所示

|  |
| --- |
| Assembly language ; 使用栈存放 DATA SEGMENT  RESULT\_STRING DB 6 DUP(?) ; 存放结果的字符串 DATA ENDS  STACK SEGMENT  DW 200 DUP(0) ; 定义栈段，这里假设栈的大小为200字节 STACK ENDS  CODE SEGMENT  ASSUME CS:CODE, DS:DATA, SS:STACK   START:  MOV AX, DATA ; 将数据段地址加载到AX寄存器  MOV DS, AX ; 将数据段地址赋给DS寄存器   ; 使用栈来存放结果  MOV AX, 0  MOV CX, 100  LOOP1:  ADD AX, CX  LOOP LOOP1   PUSH AX ; 将累加和推入栈中   ; 转换结果为十进制字符串  MOV CX, 10  MOV DI, 5  MOV BYTE PTR [RESULT\_STRING + DI], '$'  CONVERT\_LOOP:  XOR DX, DX  DIV CX  ADD DL, '0'  DEC DI  MOV BYTE PTR [RESULT\_STRING + DI], DL  TEST AX, AX  JNZ CONVERT\_LOOP   ; 从栈中弹出累加和  POP AX   ; 显示结果字符串  MOV AH, 9  MOV DX, OFFSET RESULT\_STRING  INT 21H   MOV AH, 4CH  INT 21H CODE ENDS END START |

1. 用户输入1~100内的任何一个数，完成十进制结果输出(test2-4)

这个问题的与前三个问题的解决略有不同，但是本质上仍然是字符串转十进制输出，类似的我们只需要先从缓冲区中读取用户输入的数值，再转换十进制即可

对于输入，我们采取的方法是利用DOS 10 功能号读取字符串，然后存放到CX中，作为我们需要循环的次数。这里我们存放其的形式是字符串，将用户输入的字符串转化为整数存放。在这里，我们采取的方法是一个字节一个字节读取，然后再通过将字符减去 '0' 来将字符转化为数字的方式转换成整数存放到BL中，接着 将 AL 寄存器的内容和 BL 相乘，结果存入 AX 寄存器。这是在累积输入的数字，以便将它们转化为整数。这样便完成了一位的读取，接着我们对进行DI+2操作，对下一位字符进行转换。最后读取到回车后，表明读取结束，最终的结果即在AX中，便完成了读取。

其代码实现如下所示

|  |
| --- |
| Assembly language  ; 打印提示信息  MOV AH, 9  MOV DX, OFFSET INPUT   INT 21H    ; 中断输入字符  MOV AH, 10  MOV DX, OFFSET INPUT\_STRING   INT 21H   MOV AX, 0 ; 清零AX寄存器  MOV SI, 0 ; 清零SI寄存器  ; 将用户输入的字符串转化为整数存放 CONVERT\_LOOP:  MOV BX, 0  MOV BL, [SI+2] ; 从INPUT\_STRING中读取字符  SUB BL, '0' ; 将字符转化为数字  MOV [SI+2], BL ; 将转化后的数字存回INPUT\_STRING  MOV BX, 10   MUL BL ; 将AL和BL相乘，结果存入AX  ADD AL, [SI+2] ; 将AL和INPUT\_STRING中的字符相加  INC SI   MOV BL, [SI+2] ; 读取下一个字符  CMP BL, 0DH ; 检查是否为回车符  JNE CONVERT\_LOOP ; 如果不是回车符，继续转换   MOV CX, AX ; 将转化后的结果放入CX  MOV AX, 0 ; 清零AX |

然后我们需要进行求和操作，在这里我遇到了一个奇怪的问题，使用前面三个程序的转换方法不能正确得到结果，我推测是在进行前面读取操作的时候调用了部分寄存器或者栈，而后面可能没有清空初始化导致无法完成。最后，在多次尝试无果后，查阅了相关资料，采用了另一种方式来实现转换为10进制输出。

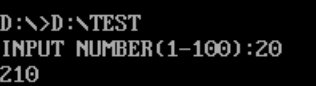
我这里的思路是使用栈的方式存放，每次取AX的最低位，存放到栈中，然后在输出的时候再依次出栈输出，这样就实现了以十进制输出结果的操作

具体的代码如下所示

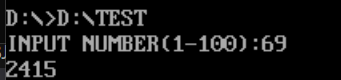
|  |
| --- |
| Assembly language ; 转化为十进制输出 CONVERT\_LOOP2:  MOV DX, 0   DIV CX ; 用AX除以CX，商在AX，余数在DX  PUSH DX ; 将余数推入栈中  CMP AX, 0 ; 检查商是否为0 ，不为0则继续转换  JNE CONVERT\_LOOP2   MOV AH, 2 ; 设置AH为2，表示打印字符  MOV DL, 0AH   INT 21H    MOV AH, 2    ; 打印累加和得到的数据 CONVERT\_LOOP3:  POP DX ; 弹出栈中的余数  ADD DL, '0' ; 将余数转化为字符  INT 21H   CMP SP, 10 ; 检查栈是否为空  JNE CONVERT\_LOOP3 ; 如果不为空，继续打印 |

实验结果如下所示

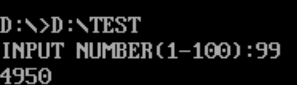
输入20



输入69



输入99



**心得体会**

本次作业对于我来说还是有一定难度的，由于前期缺少对语法的学习，在面对这两个题目的时候有点无从下手。为了完成本次作业中的要求，我主要去查阅了有关汇编中loop循环、J指令跳转、栈的一些语法的使用。第一个作业是需要我们完成a到z的打印，这个题目事实上在上节课已经完成了大部分的代码编写，我们需要做的额外任务是使用计数器来换行打印，通过这个作业让我基本掌握了loop循环和跳转指令的使用。而第二个作业相比较而言的难度则高于前者，求和的难度不高，而我主要卡在如何输出答案这个问题上，在参考了网上的相关资料之后，有了一定的初步思路，通过将其转换为字符串，以便将其显示出来，然后将其转换为10进制输出结果。

总体上来看，通过本次作业也让我系统地学习了汇编的基本语法结构，帮助我进一步了解认识这门语言。