**汇编作业二**

**作业一：九九乘法表打印**

要求：使用双loop循环实现

目的：熟悉栈的使用

1 思路分析

在考虑如何用汇编实现前，可以先考虑如何用c语言实现，实际上在c语言中我们只需要使用两个for循环即可实现，而在汇编中我们首先可以想到的是用双loop循环，边计算边输出左右因子和答案即可。在这一过程中我们需要使用栈来暂时记录CX寄存器中循环次数的值，避免后续的操作导致CX内的值的改变，然后再输出AX寄存器中，两数相乘的结果即可，其实现在上一次的作业中已经完成。

2 具体实现

在这里我采用的是外循环+内循环嵌套使用的方法来实现，分别用CX和BX来记录当前的循环值。先确定左因子，然后右因子从最大值减小到1，依次和左因子相乘，然后得到乘积并打印出来，然后完成一次内循环之后再对左因子减一，继续执行。

然后我们将每次得到的乘积存放在寄存器AX中，然后利用栈的方法将其以10进制的形式打印出来，具体实现在上一个报告中已经详细展开，这边不再赘述。

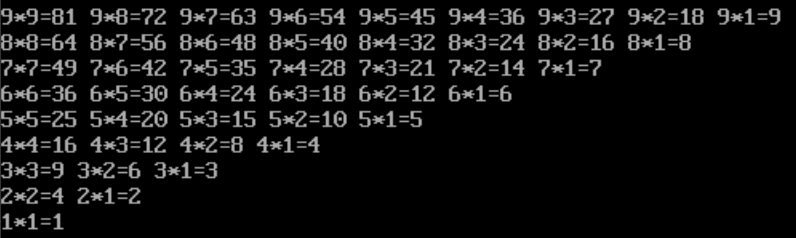
3 遇到的问题

本次作业的loop循环不难实现，难的在于需要使用栈来记录下来每次CX中的值，因为我们在循环的时候会用到CX来进行操作。其次是，在打印结果的时候，我们也需要使用栈来暂时记录CX中的值，因为这里后面会给CX赋值为10，进行除法操作。

部分代码实现如下所示（删去了打印字符的部分）

|  |
| --- |
| Assembly language ; 打印九九乘法表 STACK SEGMENT  DW 200 DUP(0) ; 定义栈段 STACK ENDS  CODESEG SEGMENT  ASSUME CS:CODESEG, SS:STACK MAIN PROC FAR   MOV AX, STACK ; 将栈段地址加载到AX寄存器  MOV SS, AX ; 将栈段地址传给SS寄存器  MOV SP, 100 ; 初始化栈顶指针为100   MOV CX, 9 ; 外层循环   MOV BX, 9 ; 内层循环  outer\_loop:  PUSH CX   MOV CX, BX   MOV DX, 0   inner\_loop:  MOV AX, BX   MUL CX  PUSH AX   POP AX  MOV DX, AX    ; 输出AX中的结果  PUSH CX    MOV CX, 10 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   ; 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, 96 ; 检查栈是否为空(因为在进行进制转换前两个CX在栈中，所以SP指针为96时余数为空)  JNE CONVERT\_LOOP3 ; 如果不为空，继续打印    POP CX ; 出栈，恢复计数器    LOOP inner\_loop    POP CX   DEC BX   LOOP outer\_loop    ; 程序结束  MOV AH, 4Ch  INT 21H  MAIN ENDP  CODESEG ENDS END MAIN |

4 结果如下所示



**作业二：九九乘法表纠错**

要求：检查九九乘法表内的数据是否正确，并将错误的位置打印出来

table db 7,2,3,4,5,6,7,8,9 ;9\*9表数据  
 db 2,4,7,8,10,12,14,16,18  
 db 3,6,9,12,15,18,21,24,27  
 db 4,8,12,16,7,24,28,32,36  
 db 5,10,15,20,25,30,35,40,45  
 db 6,12,18,24,30,7,42,48,54  
 db 7,14,21,28,35,42,49,56,63  
 db 8,16,24,32,40,48,56,7,72  
 db 9,18,27,36,45,54,63,72,81

1 思路分析

本题的思路其实和上一题类似，只是最后我们需要把打印结果的内容替换成通过查表比较是否相同。如果不同，则直接输出错误的数据的x和y值即可

2 具体实现

因为本题是在上一题的基础上实现的，我们只需要知道我们每次相乘的两数值，分别存放在BX和CX中，然后对应的就是表内（BX-1）\*9+CX-1 位置的数，再比较两者的数值，即可知道最后结果正确与否。

3 遇到的问题

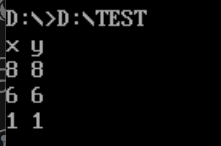
本题主要的问题在于读取table表内的元素，由于其访问的实现需要通过偏移量来实现。这里汇编中table的用法也类似于数组，一旦定义了TABLE，你可以使用偏移量（索引）来访问表中的元素。例如，如果有一个字节表TABLE，可以使用偏移量来获取或修改特定元素的值。如果想使用寄存器中的值作为TABLE中的偏移量，可以将寄存器的内容添加到TABLE的基址，并将结果存储在另一个寄存器中，然后使用该寄存器来访问或修改TABLE中的元素

4 实现部分代码如下所示

|  |
| --- |
| Assembly language  ; 下面进行纠错判断，先将我们用到的寄存器用栈保存   PUSH AX  PUSH CX  PUSH BX  PUSH DX    ; 计算AX=（BX-1）\*9+CX-1   MOV AX, BX  DEC AX  MOV BX, 9  MUL BX  ADD AX, CX  DEC AX   ; 使用si作为读取寄存器值作为偏移量  MOV SI, AX  MOV BX, OFFSET TABLE  ADD BX, SI  MOV AL, [BX] ; 去出纠错表内对应位置的元素  POP DX  POP BX  POP CX  CMP DL,AL  JNE error   POP AX |

以上这段代码实现的是对结果正确与否的判断，当发现结果有误时则会跳转到error，输出错误的行数和列数

5 结果如下



**心得体会**

本次作业主要使用了loop循环和栈的方法，由于在实际的完成中需要多次使用同一个寄存器，比如AX,BX,CX可能会在loop循环中被反复使用，这就需要我们使用栈来帮助我们暂时保存其内部的值。本次作业也大量采用了栈的方法，为了尽可能避免后续操作的影响，尤其是作业二中体现了这一点。

本次实验用c语言来实现没有难度，但是用汇编实现时仍遇到了不少问题，主要还是由于不熟悉具体的语法，以及由于进行MOV操作，破坏了寄存器中的值导致的。虽然总体的思路不难，后面还是花费了较多的时间进行debug。