汇编语言程序设计课程实验报告

实验 3

姓名		院系	计算机工程与科学学院	学号	
实验目的：					
1. 通过循环控制编程方式用表格形式显示 ASCII 字符表					
实验任务：					
按 15 行×16 列的表格形式显示 ASCII 码为 10H—100H 的所有字符，即以行为主的顺序及 ASCII 码递增的次序显示对应的字符。每 16 个字符为一行，每行中的相邻两个字符之间用空白符（ASCII 为 0）隔开。					

1. 程序代码如下

```
1.  data segment
2.      ChangeRow db 0DH,0ah,'$'; 输出换行符，以防输出被覆盖
3.  data ends
4.
5.  stack segment
6.          dw 128   dup(0)
7.  stack ends
8.
9.  code segment
10.             assume cs:code, ds:data, ss:stack
11.     start:
12.             mov     ax, data
13.             mov     ds, ax
14.             mov     bx, 0
15.             mov     cx,000fH
16.     ctrl:
17.             push    cx
18.             mov     cx, 0010H
19.     display:
20.             mov     dl,bl
21.             mov     ah,02H
22.             int     21H
23.             mov     dl,20H; 输出每个字符后接一个空格
24.             int     21H
25.             inc     bx
26.             loop    display
27.             call    change;   输出一行后调用子程序进行换行
28.             pop     cx
29.             loop    ctrl
30.     over:
```

```
31.                 mov     ax, 4c00h
32.                 int     21h
33.         ;换行
34.     change PROC
35.                 mov     dx, offset ChangeRow
36.                 mov     ah, 09h
37.                 int     21h
38.                 ret
39.     change ENDP
40.     code ends
41.     end start
```

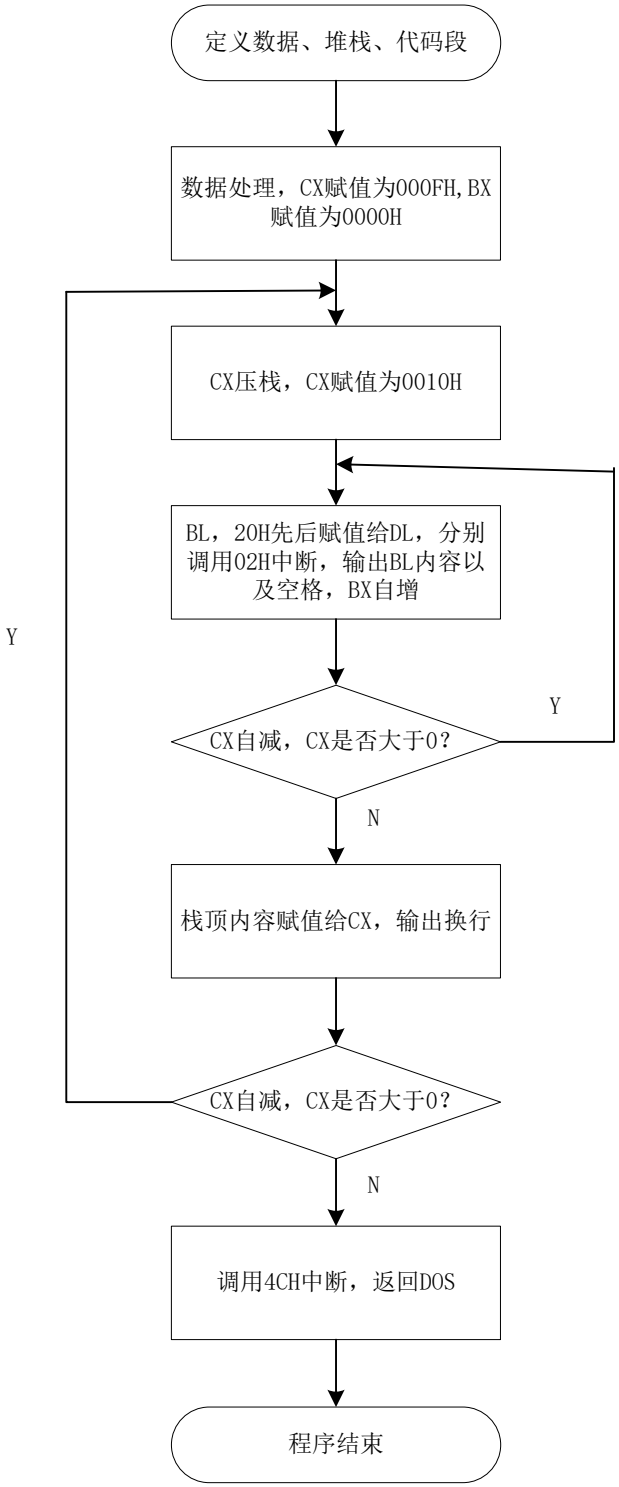
2. 程序分析

实验的基本思路是使用双重循环打印字符，其中外层循环控制输出行数，内层循环控制每行输出的字符数，由于每个字符的 ascii 码呈现递增的趋势，所以可以使用一个寄存器控制每次输出的字符的 ascii 码，输出一个之后就自增。

难点在于怎么合理的控制寄存器 CX 的数值达到两层控制的效果，我们可以在内层循环开始前，使用堆栈存储 CX 的值，并给 CX 赋予新值，新值即为需要每行输出的字符个数，在执行完内层循环后，从堆栈中弹出栈顶值赋值给 CX，然后检测 CX 的值是否大于 0，（CX 此时先自减），如果大于 0，则回到最初的位置开始循环，即又开始将 CX 压栈，否则结束程序。此为循环方式实现，我们可以使用分支结构实现，以下更改后的**循环体**代码。

```
1.      ctrl:
2.              push    cx
3.              mov     cx, 0010H
4.      display:
5.              mov     dl,bl
6.              mov     ah,02H
7.              int     21H
8.              mov     dl,20H
9.              int     21H
10.             inc     bx
11.             sub     cx,1
12.             cmp     cx,0
13.             jne     display
14.             call    change
15.             pop     cx
16.             dec     cx
17.             cmp     cx,0
18.             jne     ctrl
19.     over:
20.             mov     ax, 4c00h
21.             int     21h
```

3. 程序流程图设计



4. 实验体会

通过本次实验，我进一步理解了堆栈段区的作用以及对变址，递增的使用，平时习惯使用高级语言，很自然的就想使用两个变量进行循环控制，但是汇编语言中，只能是由 CX 寄存器来控制循环次数，多层循环下如何找回外层循环的 CX 值就显得尤为重要，通过这个实验也加深了我对状态保存重要性的理解，如果在跳跃执行子程序或者另外一些代码时，在硬件资源有限的情况下，如何合理使用寄存器，对本层次的程序进行保存必要的状态和现场是很重要的，否则就会出错，宕机。合理的应用堆栈段，能解决很多问题，期待下一次的实验。