输出 ASCII 表中的小写字母部分，每 13 个一行

一．使用 LOOP 指令实现

数据段：初始化字母'a'，并定义换行符。

```
czwSEG SEGMENT
    MSG2 DB "a"
    NEWLINE DB 0DH, 0AH, '$'
czwSEG ENDS
```

假设段：当前代码段使用 czwSEG 段。

```
ASSUME CS:czwSEG
```

代码段：

```
czwSEG SEGMENT
czw:
    MOV AX,czwSEG
    MOV DS,AX

    MOV CX, 2          ; 外层循环，控制行数，共两行
RowLoop:
    MOV BX, CX         ; 将外层循环的计数值保存到 BX 寄存器

    MOV CX, 13         ; 内层循环，控制每行打印 13 个字母
    MOV AH, 2          ; 设置中断功能号，显示字符
ColLoop:
    MOV AL, [MSG2]     ; 从 MSG2 加载当前字母
    MOV DL, AL         ; 把当前字母放到 DL
    INT 21H            ; 调用中断 21H，显示字符
    INC AL             ; 字符递增
    MOV [MSG2], AL     ; 更新 MSG2，保存递增后的字母
    LOOP ColLoop       ; 内层循环结束，CX减1，若不为0跳回ColLoop

    ; 打印换行符
    MOV DX, OFFSET NEWLINE  ; 加载换行符地址
    MOV AH, 09H             ; 设置 DOS 中断功能号 09H（显示字符串）
    INT 21H                 ; 调用中断 21H，显示换行符

    MOV CX, BX         ; 恢复外层循环的计数值
    LOOP RowLoop       ; 外层循环控制，CX减1，若不为0跳回RowLoop

    MOV AX,4C00H
    INT 21H
czwSEG ENDS

END czw
```

程序通过双重循环实现了字符的逐行输出，外层循环控制输出行数，内层循环控制每行输出的字母数量。

二．使用跳转指令实现

数据段和假设段与方法一相同，代码段有所区别。

代码段:

```
czwSEG SEGMENT
czw:
    MOV AX,czwSEG
    MOV DS,AX

    MOV BX, 2          ; 外层循环计数器，设置为 2（两行）
RowLoop:
    MOV CX, 13         ; 内层循环计数器，设置为 13（每行打印 13 个字母）
    MOV AH, 2          ; 设置中断功能号，显示字符
ColLoop:
    MOV AL, [MSG2]     ; 从 MSG2 加载当前字母
    MOV DL, AL         ; 把当前字母放到 DL
    INT 21H            ; 调用中断 21H，显示字符
    INC AL             ; 字符递增
    MOV [MSG2], AL     ; 更新 MSG2，保存递增后的字母

    DEC CX             ; 内层循环计数减 1
    JNZ ColLoop        ; 如果 CX 不为 0，跳回 ColLoop

    ; 打印换行符
    MOV DX, OFFSET NEWLINE  ; 加载换行符地址
    MOV AH, 09H             ; 设置 DOS 中断功能号 09H（显示字符串）
    INT 21H                 ; 调用中断 21H，显示换行符

    DEC BX                  ; 外层循环计数减 1
    JNZ RowLoop             ; 如果 BX 不为 0，跳回 RowLoop

    MOV AX, 4C00H           ; 正常退出程序
    INT 21H
czwSEG ENDS

END czw
```

在此程序中，使用 DEC 和 JNZ 来实现循环控制。DEC 减去 1 并设置标志位，JNZ 根据标志位判断是否跳回循环。这种方法更灵活，可以根据需要实现更复杂的循环条件（例如，基于其他寄存器或变量的值）。

三．用 C 语言实现并反汇编

C 语言代码:

```c
#include <stdio.h>

int main() {
    char letter = 'a'; // 从 'a' 开始
    for (int i = 0; i < 2; i++) {  // 外层循环控制行数
        for (int j = 0; j < 13; j++) {  // 内层循环控制每行打印的字母数
            printf("%c", letter);  // 打印当前字母
            letter++;          // 字母递增
        }
        printf('\n');  // 打印换行符
    }
    return 0;  // 程序正常退出
}
```

反汇编结果:

00007FF6A3E03BA0 push rbp

00007FF6A3E03BA2 push rdi

00007FF6A3E03BA3 sub rsp,148h

00007FF6A3E03BAA lea rbp,[rsp+20h]

00007FF6A3E03BAF lea rcx,[__843D2DC9_第二次作业\print_letters@c (07FF6A3E11008h)]

00007FF6A3E03BB6 call __CheckForDebuggerJustMyCode (07FF6A3E01357h)

00007FF6A3E03BBB nop

保存当前基指针到栈中，并为局部变量分配空间。

00007FF6A3E03BBC mov byte ptr [letter],61h

初始化变量 letter。

00007FF6A3E03BC0 mov dword ptr [rbp+24h],0

00007FF6A3E03BC7 jmp main+31h (07FF6A3E03BD1h)

00007FF6A3E03BC9 mov eax,dword ptr [rbp+24h]

00007FF6A3E03BCC inc eax

00007FF6A3E03BCE mov dword ptr [rbp+24h],eax

```
00007FF6A3E03BD1    cmp             dword ptr [rbp+24h],2
00007FF6A3E03BD5    jge             main+79h (07FF6A3E03C19h)
```

初始化 i 为 0，并通过条件跳转控制外层循环，判断 i 是否小于 2。

```
00007FF6A3E03BD7    mov             dword ptr [rbp+44h],0
00007FF6A3E03BDE    jmp             main+48h (07FF6A3E03BE8h)
00007FF6A3E03BE0    mov             eax,dword ptr [rbp+44h]
00007FF6A3E03BE3    inc             eax
00007FF6A3E03BE5    mov             dword ptr [rbp+44h],eax
00007FF6A3E03BE8    cmp             dword ptr [rbp+44h],0Dh
00007FF6A3E03BEC    jge             main+6Ch (07FF6A3E03C0Ch)
```

初始化 j 为 0，并通过条件跳转控制内层循环，判断 j 是否小于 13。

```
00007FF6A3E03BEE    movsx           eax,byte ptr [letter]
00007FF6A3E03BF2    mov             edx,eax
00007FF6A3E03BF4    lea             rcx,[string "%c" (07FF6A3E09BD8h)]
00007FF6A3E03BFB    call            printf (07FF6A3E013BBh)
00007FF6A3E03C00    nop
```

将 letter 的值传递给 printf 函数进行打印。

```
00007FF6A3E03C01    movzx           eax,byte ptr [letter]
00007FF6A3E03C05    inc             al
00007FF6A3E03C07    mov             byte ptr [letter],al
}
00007FF6A3E03C0A    jmp             main+40h (07FF6A3E03BE0h)
```

将 letter 的值递增。

```
00007FF6A3E03C0C    mov             ecx,0Ah
00007FF6A3E03C11    call            printf (07FF6A3E013BBh)
00007FF6A3E03C16    nop
```

}

00007FF6A3E03C17 jmp main+29h (07FF6A3E03BC9h)

打印换行符。

00007FF6A3E03C19 xor eax,eax

将 0 返回给操作系统，表示正常退出。

00007FF6A3E03C1B lea rsp,[rbp+128h]

00007FF6A3E03C22 pop rdi

00007FF6A3E03C23 pop rbp

00007FF6A3E03C24 ret

恢复栈指针和基指针，结束函数。

C 语言反汇编生成的代码，首先需要保存基指针，为局部变量分配空间，然后初始化变量，再通过条件跳转指令实现循环打印字母表。