源码一：使用 BX 寄存器来保留 CX 的值，完成内循环后恢复 CX 原本的值。

```
MY SEGMENT
   A DB 'a'
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
    MOV CX,2
    MOV AH,2
 L_1:
    MOV BX,CX
    MOV CX,13
    L_2:
    MOV DL,[A]
    INC [A]
    INT 21H
    LOOP L_2
    MOV DL,10
    INT 21H
    MOV CX,BX
    LOOP L_1
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

源码二：使用 JNZ 条件跳转来实现循环。

```
MY SEGMENT
   A DB 'a'
MY ENDS
ASSUME CS:MY
MY SEGMENT
start:
    MOV AX, MY
    MOV DS, AX
    MOV CX,2
    MOV BX,13
    MOV AH,2
 L_1:
    MOV BX,13
    L_2:
    MOV DL,[A]
```

```
    INC [A]
    INT 21H
    DEC BX
    CMP BX,0
    JNZ L_2
    MOV DL,10
    INT 21H
    DEC CX
    CMP CX,0
    JNZ L_1
    MOV AX, 4C00H
    INT 21H
MY ENDS
   END start
```

C 语言实现代码：

```c
#include<stdio.h>

int main()
{
    char a = 'a';
    for(int i=0;i<2;i++)
    {
        for (int j = 0; j < 13; j++)
        {
            printf("%c", a);
            a++;
        }
        printf("\n");
    }
    return 0;
}
```

C 语言反汇编结果：

```
0000000000001149 <main>:
    1149:       f3 0f 1e fa             endbr64
    114d:       55                      push   %rbp
    114e:       48 89 e5                mov    %rsp,%rbp
    1151:       48 83 ec 10             sub    $0x10,%rsp
    1155:       c6 45 f7 61             movb   $0x61,-0x9(%rbp)
    1159:       c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
    1160:       eb 36                   jmp    1198 <main+0x4f>
    1162:       c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
    1169:       eb 19                   jmp    1184 <main+0x3b>
    116b:       0f be 45 f7             movsbl -0x9(%rbp),%eax
    116f:       89 c7                   mov    %eax,%edi
```

```
1171:       e8 da fe ff ff          call   1050 <putchar@plt>
1176:       0f b6 45 f7             movzbl -0x9(%rbp),%eax
117a:       83 c0 01                add    $0x1,%eax
117d:       88 45 f7                mov    %al,-0x9(%rbp)
1180:       83 45 fc 01             addl   $0x1,-0x4(%rbp)
1184:       83 7d fc 0c             cmpl   $0xc,-0x4(%rbp)
1188:       7e e1                   jle    116b <main+0x22>
118a:       bf 0a 00 00 00          mov    $0xa,%edi
118f:       e8 bc fe ff ff          call   1050 <putchar@plt>
1194:       83 45 f8 01             addl   $0x1,-0x8(%rbp)
1198:       83 7d f8 01             cmpl   $0x1,-0x8(%rbp)
119c:       7e c4                   jle    1162 <main+0x19>
119e:       b8 00 00 00 00          mov    $0x0,%eax
11a3:       c9                      leave
11a4:       c3                      ret
```

可以看到 C 语言编译成的汇编代码，首先要通过基指针来给变量分配空间，并初始化变量。然后通过条件跳转的方式来实现循环打印字母表。

比较来看，反汇编查看原本通过汇编语言编写的程序，可以看到汇编语言不需要对变量进行初始化的操作，也不需要一系列的输出流操作来给打印字符，同时他也没有用栈存方式来保留变量。