

# 打印 ASCII 表实验报告

2251656 于思源

## 一、使用 loop 指令实现

### 1. 基本思路

使用两个循环，外循环用于实现换行，内循环用于实现循环输出字符，同时使用 AL 寄存器实现对字符的记录。

### 2. 核心代码

```
    ; 初始化循环变量
    MOV CX, 2          ; 外循环2次
    MOV AL, 97         ; 'a' 的 ASCII 值是 97, 即0x61

print_line:
    MOV BX, CX         ; 保存外循环次数至DX
    MOV CX, 13         ; 初始化内循环次数

print_loop:
    ; 输出字母
    MOV DL, AL         ; 将当前字母的 ASCII 值放入 DL
    MOV AH, 02h        ; 设置为输出单个字符
    INT 21h            ; 调用 DOS 中断输出字符

    ; 递增字母
    INC AL             ; 增加 AL 中的值, 使其指向下一个字母

    LOOP print_loop    ; 循环, 直到 CX 为 0

    ; 输出换行
    LEA DX, newline
    MOV CX, BX         ; 将外循环次数传回CX
    MOV AH, 09h        ; DOS 中断, 输出字符串
    INT 21h

    LOOP print_line    ; 循环输出换行符
```

### 3. 结果

```
D:\>USELOOP
abcdefghijklmnopqrstuvwxyz
```

## 二、使用条件跳转指令实现

### 1. 基本思路

与循环语句相同，区别只是将循环语句中的 LOOP 指令变成跳转指令，并在前面增加判断。

## 2. 核心代码

```
continue_loop:

    ; 初始化内循环变量
    MOV CX, 13          ; 循环26次，对应26个字母

inside_loop:

    ; 输出字母
    MOV DL, AL          ; 将当前字母的 ASCII 值放入 DL
    MOV AH, 02h         ; 设置为输出单个字符
    INT 21h             ; 调用 DOS 中断输出字符

    ; 递增字母
    INC AL              ; 增加 AL 中的值，使其指向下一个字母
    DEC CX              ; 循环，直到 CX 为 0


    CMP CX, 0
    JNE inside_loop     ; 若count不等于13，继续循环

    ; 输出换行
    LEA DX, newline
    MOV AH, 09h         ; DOS 中断，输出字符串
    INT 21h

    ; 递减外循环
    DEC BX

    CMP BX, 0
    JNE continue_loop  ; 若count不等于0，继续循环
```

## 3. 结果



```
D:\>USEJMP
abcdefghijklmnopqrstuvwxyz
```

## 三、使用 C 语言后查看反汇编代码

### 1. 基本思路

使用循环指令进行执行

### 2. C 语言代码

```

int main(){
    int Out = 'a';

    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 13; j++){
            printf("%c", Out);
            Out++;
        }
        printf("\n");
    }

    return 0;
}

```

### 3. 反汇编代码及备注

0000000000000000 <main>:			; 函数main的入口地址
0: 55	push	%rbp	; 将基址指针寄存器压入栈中, 保存上一个栈帧的基址指针
1: 48 89 e5	mov	%rsp,%rbp	; 将当前的栈指针rsp复制到rbp, 设置新的栈帧基址
4: 48 83 ec 30	sub	\$0x30,%rsp	; 将栈指针rsp向下移动48字节, 为局部变量分配栈空间
8: e8 00 00 00 00	callq	d <main+0xd>	; 占位符
d: c7 45 fc 61 00 00 00	movl	\$0x61,-0x4(%rbp)	; 将ASCII码'a'的值储存在rbp基址偏移-0x4的位置, 即局部变量-4中
14: c7 45 f8 00 00 00 00	movl	\$0x0,-0x8(%rbp)	; 将0储存在rbp基址偏移-0x8的位置, 初始化局部变量-8
1b: eb 2f	jmp	4c <main+0x4c>	; 跳转main至偏移0x4c的位置
1d: c7 45 f4 00 00 00 00	movl	\$0x0,-0xc(%rbp)	; 将0储存在rbp基址偏移-0xc的位置, 初始化局部变量-12
24: eb 12	jmp	38 <main+0x38>	; 跳转main至偏移0x38的位置
26: 8b 45 fc	mov	-0x4(%rbp),%eax	; 将-0x4(%rbp)的值(即局部变量-4, 'a'的ASCII值97)加载到eax中
29: 89 c1	mov	%eax,%ecx	; 将eax的值复制到ecx寄存器中
2b: e8 00 00 00 00	callq	30 <main+0x30>	; 调用printf输出函数
30: 83 45 fc 01	addl	\$0x1,-0x4(%rbp)	; 将局部变量-4的值+1, 即字母递增
34: 83 45 f4 01	addl	\$0x1,-0xc(%rbp)	; 将局部变量-12的值+1, 即增加内循环计数器的值
38: 83 7d f4 0c	cmpl	\$0xc,-0xc(%rbp)	; 比较局部变量12和-12是否相等, 即内循环条件判断
3c: 7e e8	jle	26 <main+0x26>	; 若局部变量-12小于等于12, 则跳转到偏移26, 继续内循环
3e: b9 0a 00 00 00	mov	\$0xa,%ecx	; 将0xa(换行符)存入ecx, 准备输出换行符
43: e8 00 00 00 00	callq	48 <main+0x48>	; 调用printf输出换行符
48: 83 45 f8 01	addl	\$0x1,-0x8(%rbp)	; 将局部变量-8的值+1, 增加外循环计数器的值
4c: 83 7d f8 01	cmpl	\$0x1,-0x8(%rbp)	; 比较局部变量-8与1是否相等, 用于外循环的条件判断
50: 7e cb	jle	1d <main+0x1d>	; 若局部变量-8小于等于1, 则跳转到1d, 继续外循环
52: b8 00 00 00 00	mov	\$0x0,%eax	; 将0存入eax, 用于返回值, 表示程序正常输出
57: 48 83 c4 30	add	\$0x30,%rsp	; 将栈指针rsp加48, 恢复栈空间
5b: 5d	pop	%rbp	; 从栈中弹出rbp, 恢复之前保存的基址指针
5c: c3	retq		; 返回调用main函数的地方
5d: 90	nop		
5e: 90	nop		

### 四、实验总结

通过本次实验, 我掌握了循环指令和跳转指令的基本应用。