赵明泽 2151974

汇编语言第一次作业

目录

题目1：按要求打印ASCII表

要求：输出ASCII表中的小写字母部分，要求每行13个字符。

用`loop`指令实现

使用`loop`指令两次，分两行打印字母表。

首先，用C语言写出对应程序，如下

```c
#include <stdio.h>
int main() {
    for (int i = 0; i < 26; i++) {
        printf("%c", 'A'+i);
        if (i == 12) printf("\n");
    }
    return 0;
}
```

用`gcc`编译，生成.o文件

```
gcc -c loop.c
```

输入指令，执行反汇编

```
objdump -S loop.o
```

查看`main`函数反汇编结果

```
0000000000000000 <main>:
```

```
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   48 83 ec 30             sub    $0x30,%rsp
   8:   e8 00 00 00 00          callq  d <main+0xd>
   d:   c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  14:   83 7d fc 19             cmpl   $0x19,-0x4(%rbp)
  18:   7f 23                   jg     3d <main+0x3d>
  1a:   8b 45 fc                mov    -0x4(%rbp),%eax
  1d:   83 c0 41                add    $0x41,%eax
  20:   89 c1                   mov    %eax,%ecx
  22:   e8 00 00 00 00          callq  27 <main+0x27>
  27:   83 7d fc 0c             cmpl   $0xc,-0x4(%rbp)
  2b:   75 0a                   jne    37 <main+0x37>
  2d:   b9 0a 00 00 00          mov    $0xa,%ecx
  32:   e8 00 00 00 00          callq  37 <main+0x37>
  37:   83 45 fc 01             addl   $0x1,-0x4(%rbp)
  3b:   eb d7                   jmp    14 <main+0x14>
  3d:   b8 00 00 00 00          mov    $0x0,%eax
  42:   48 83 c4 30             add    $0x30,%rsp
  46:   5d                      pop    %rbp
  47:   c3                      retq
```

发现`gcc`把C语言代码翻译成汇编时使用了条件跳转。

考虑每行输出13个字母，所以`cx`寄存器设置13，为内圈循环次数。把外圈循环次数2放入`bx`低八位暂存

```
mov bl,2    ;外圈循环次数
mov cx,13   ;内圈循环次数
```

`loop2`主要负责打印完一行后输出回车

```
loop2:              ;外圈循环开始
        mov cx,13   ;第二次要重新初始化次数
loop1:              ;内圈循环开始
        ......
        loop loop1  ;内圈循环结束

        mov bh,dl   ;暂存当前字符
        mov dl,0ah  ;存入回车
        int 21h     ;打印回车
        mov dl,bh   ;恢复当前字符
        mov cl,bl   ;计数器更新外圈循环次数
        sub bl,1    ;外圈循环次数-1

        loop loop2  ;外圈循环结束
```

`loop1`负责打印一行字母

```
loop1:              ;内圈循环开始
        int 21h     ;打印字符
        add dl,1    ;ASCII+1
        loop loop1  ;内圈循环结束
```

完整代码

```
code segment
         assume cs:code
start:
        mov bl,2     ;外圈循环次数
        mov cx,13    ;内圈循环次数
        mov ah,02h
        mov dl,'a'   ;字符a
loop2:               ;外圈循环开始
        mov cx,13    ;第二次要重新初始化次数
loop1:               ;内圈循环开始
        int 21h      ;打印字符
        add dl,1     ;ASCII+1
        loop loop1   ;内圈循环结束
        mov bh,dl    ;暂存当前字符
        mov dl,0ah   ;存入回车
        int 21h      ;打印回车
        mov dl,bh    ;恢复当前字符
        mov cl,bl    ;计数器更新外圈循环次数
        sub bl,1     ;外圈循环次数-1
        loop loop2   ;外圈循环结束
        mov ah,4ch
        int 21h
code ends
       end start
```

用条件跳转指令实现

条件跳转当判断当前输出数量是13个字母时，打印回车，否则跳转继续输出字母。所以开始时，打印数量直接设置为26

```
mov cx,26
```

打印第一行

```
row1:                ;打印第一行
        sub cx,1     ;次数-1
        int 21h      ;打印字符
        add dl,1     ;ASCII+1
        cmp cx,13    ;比较
        jne row1     ;不等跳转
```

打印回车

```
mov bl,dl    ;暂存当前字符
mov dl,0ah   ;存入回车
int 21h      ;打印回车
mov dl,bl    ;恢复当前字符
```

打印第二行

```
row2:               ;打印第二行
        sub cx,1    ;次数-1
        int 21h     ;打印字符
        add dl,1    ;ASCII+1
        cmp cx,0    ;比较
        jne row2    ;不等跳转
```

完整代码

```
code segment
         assume cs:code
start:
        mov cx,26
        mov ah,02h
        mov dl,'a'  ;字符a
row1:               ;打印第一行
        sub cx,1    ;次数-1
        int 21h     ;打印字符
        add dl,1    ;ASCII+1
        cmp cx,13   ;比较
        jne row1    ;不等跳转
        mov bl,dl   ;暂存当前字符
        mov dl,0ah  ;存入回车
        int 21h     ;打印回车
        mov dl,bl   ;恢复当前字符
row2:               ;打印第二行
        sub cx,1    ;次数-1
        int 21h     ;打印字符
        add dl,1    ;ASCII+1
        cmp cx,0    ;比较
        jne row2    ;不等跳转
        mov ah,4ch
        int 21h
code ends
       end start
```

题目2：求和

要求：求1+2+……+100，并将结果“5050”打印到屏幕。

结果放在寄存器中

维护三个寄存器，一个存放当前的加数，一个存放加和，一个存放循环计数器

```
    mov dx,1    ;当前加数
    mov bx,0    ;和
    mov cx,100  ;计数器
```

循环相加

```
loop1:          ;循环开始
    add bx,dx   ;加
    inc dx      ;自增
    loop loop1  ;循环结束
```

计算出的结果为十六进制数，存储在`bx`寄存器中。已知结果为4位数，分别对千位、百位、十位和个位进行进制转换并分解，直接打印到屏幕上。下面仅给出千位的例子

```
    mov cx,1000d;千位
    mov ax,bx   ;拷贝一份
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov bx,dx   ;余数给dx
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
```

完整代码

```
code segment
    assume cs:code
start:
    mov ah,02h  ;输出方式
    mov dx,1    ;当前加数
    mov bx,0    ;和
    mov cx,100  ;计数器
loop1:          ;循环开始
    add bx,dx   ;加
    inc dx      ;自增
    loop loop1  ;循环结束
;以下代码将bx中的值分解成千、百、十、个位，以字符形式输出
    mov cx,1000d;千位
    mov ax,bx   ;拷贝一份
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov bx,dx   ;余数给dx
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,100d ;百位
    mov ax,bx   ;拷贝一份
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov bx,dx   ;余数给dx
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,10d  ;十位
    mov ax,bx   ;拷贝一份
    mov dx,0    ;清空
    div cx      ;ax/cx
```

```
    mov bx,dx    ;余数给dx
    mov dl,al    ;一位数字
    add dl,30h   ;变成ASCII
    mov ah,02    ;输出形式
    int 21h      ;打印
    mov cx,1d    ;个位
    mov ax,bx    ;拷贝一份
    mov dx,0     ;清空
    div cx       ;ax/cx
    mov bx,dx    ;余数给dx
    mov dl,al    ;一位数字
    add dl,30h   ;变成ASCII
    mov ah,02    ;输出形式
    int 21h      ;打印
    mov ah,4ch
    int 21h
code ends
    end start
```

结果放在数据段中

维护一个数据段，在内存中开辟16位的整数倍空间，用来存储数据

```
data segment
    dw 0h
data ends
```

将第一问中的所有寄存器`bx`替换为`ds:[0]`，在内存中存储最终结果

```
mov ds:[0],bx;把和放入数据段
```

其他同上。完整代码

```
data segment
    dw 0h
data ends
code segment
    assume cs:code, ds:data
start:
    mov bx,data ;段地址送入bx
    mov ds,bx   ;存放段地址
    mov ah,02h  ;输出方式
    mov dx,1    ;当前加数
    mov bx,0    ;和
    mov ds:[0],bx;把和放入数据段
    mov cx,100  ;计数器
loop1:          ;循环开始
    add bx,dx   ;加
    mov ds:[0],bx;把和放入数据段
    inc dx      ;自增
    loop loop1  ;循环结束
;以下代码将bx中的值分解成千、百、十、个位，以字符形式输出
    mov cx,1000d;千位
    mov ax,ds:[0];内存直接寻址放入ax
```

```
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov ds:[0],dx;余数给dx，放入数据段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,100d ;百位
    mov ax,ds:[0];内存直接寻址放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov ds:[0],dx;余数给dx，放入数据段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,10d  ;十位
    mov ax,ds:[0];内存直接寻址放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov ds:[0],dx;余数给dx，放入数据段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,1d   ;个位
    mov ax,ds:[0];内存直接寻址放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    mov ds:[0],dx;余数给dx，放入数据段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov ah,4ch
    int 21h
code ends
       end start
```

结果放在栈中

首先，维护一个栈段

```
stk segment
    dw 0h
stk ends
```

初始化栈基址和栈顶指针

```
    mov bx,stk  ;段地址送入bx
    mov ss,bx   ;存放段地址
    mov sp,0    ;栈指针
```

使用 `push` 和 `pop` 方法操作寄存器和内存

```
loop1:          ;循环开始
    pop bx      ;弹出当前和
    add bx,dx   ;加
    push bx     ;把和推入栈段
    inc dx      ;自增
    loop loop1  ;循环结束
```

完整代码

```
stk segment
    dw 0h
stk ends
code segment
    assume cs:code, ss:stk
start:
    mov bx,stk  ;段地址送入bx
    mov ss,bx   ;存放段地址
    mov sp,0    ;栈指针
    mov ah,02h  ;输出方式
    mov dx,1    ;当前加数
    mov bx,0    ;和
    push bx     ;把和推入栈段
    mov cx,100  ;计数器
loop1:          ;循环开始
    pop bx      ;弹出当前和
    add bx,dx   ;加
    push bx     ;把和推入栈段
    inc dx      ;自增
    loop loop1  ;循环结束
;以下代码将栈顶的值分解成千、百、十、个位，以字符形式输出
    mov cx,1000d;千位
    pop ax      ;和放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    push dx     ;余数给dx，推入栈段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,100d ;百位
    pop ax      ;和放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    push dx     ;余数给dx，推入栈段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,10d  ;十位
    pop ax      ;和放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    push dx     ;余数给dx，推入栈段
    mov dl,al   ;一位数字
```

```
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov cx,1d   ;个位
    pop ax      ;和放入ax
    mov dx,0    ;清空
    div cx      ;ax/cx
    push dx     ;余数给dx，推入栈段
    mov dl,al   ;一位数字
    add dl,30h  ;变成ASCII
    mov ah,02   ;输出形式
    int 21h     ;打印
    mov ah,4ch
    int 21h
code ends
       end start
```

用户输入1~100内的任何一个数，完成十进制结果输出

总体思路是读取用户输入，处理为十进制数，输出到屏幕。首先，需要读取用户的输入。初始化内存和寄存器，清空其中的数据，读入第一个字符

```
init:               ;初始化
    xor ax, ax      ;清零
    xor bx, bx      ;清零
    xor cx, cx      ;清零
    xor dx, dx      ;清零
    mov ah, 1       ;读一个字符方式
    int 21h         ;中断
```

未读到非法字符，循环读取

```
input:              ;循环输入开始
    cmp al, 30h     ;判断'0'
    jb check        ;比0小，跳转
    cmp al, 39h     ;判断'9'
    ja check        ;比9大，跳转
    sub al, 30h     ;ASCII处理
    shl bx, 1       ;bx左移
    mov cx, bx      ;赋值
    shl bx, 1       ;bx左移
    shl bx, 1       ;bx左移
    add bx, cx      ;加和
    add bl, al      ;加和
    mov ah, 1       ;读一个字符方式
    int 21h         ;中断
    jmp input       ;循环输入结束
```

读到非法字符，判断一下是不是回车。如果是，跳转至保存结果部分

```
check:              ;检查输入
    cmp al, 0dh     ;输入回车
    je save         ;跳转保存结果
    jmp init        ;输入错误，跳转初始化
save:               ;保存结果
    mov ax, bx      ;ax是结果
    pop dx          ;出栈
    pop cx          ;出栈
    pop bx          ;出栈
    pop bp          ;出栈
```

把寄存器中的值放入内存数据段

```
    mov num, ax     ;存入数据段内存
    push num        ;入栈
```

接下来是输出部分。依然是对十六进制数转换并分解，存入栈中，依次出栈，得到输入的数字。进制转换的部分

```
divide:             ;进制转换
    xor dx, dx      ;清空
    div bx          ;除以10取余数，dx:ax / bx = ax.....dx
    add dl, 30h     ;ASCII转换
    push dx         ;商入栈
    inc cx          ;数字位数+1
    cmp ax, 0       ;商为0，结束
    jne divide      ;商不为0，继续除
```

循环输出

```
output:             ;循环输出开始
    pop dx          ;依次弹出结果
    mov ah, 2       ;输出方式
    int 21h         ;中断
    loop output     ;循环输出结束
```

完整代码

```
stk segment
stk ends
data segment
    num dw ?
data ends
code segment
    assume cs:code, ss:stk, ds:data
start:
    mov ax, data    ;数据段基址
    mov ds, ax      ;赋值给基址寄存器
;---------------------------------------------------------------
;   输入部分
    push bp         ;入栈
    mov bp, sp      ;栈顶指针
    push bx         ;bx入栈
```

```
    push cx          ;cx入栈
    push dx          ;dx入栈
init:                ;初始化
    xor ax, ax       ;清零
    xor bx, bx       ;清零
    xor cx, cx       ;清零
    xor dx, dx       ;清零
    mov ah, 1        ;读一个字符方式
    int 21h          ;中断
input:               ;循环输入开始
    cmp al, 30h      ;判断'0'
    jb check         ;比0小，跳转
    cmp al, 39h      ;判断'9'
    ja check         ;比9大，跳转
    sub al, 30h      ;ASCII处理
    shl bx, 1        ;bx左移
    mov cx, bx       ;赋值
    shl bx, 1        ;bx左移
    shl bx, 1        ;bx左移
    add bx, cx       ;加和
    add bl, al       ;加和
    mov ah, 1        ;读一个字符方式
    int 21h          ;中断
    jmp input        ;循环输入结束
check:               ;检查输入
    cmp al, 0dh      ;输入回车
    je save          ;跳转保存结果
    jmp init         ;输入错误，跳转初始化
save:                ;保存结果
    mov ax, bx       ;ax是结果
    pop dx           ;出栈
    pop cx           ;出栈
    pop bx           ;出栈
    pop bp           ;出栈
;   输入部分结束
;-----------------------------------------------------------------
    mov num, ax      ;存入数据段内存
    push num         ;入栈
;-----------------------------------------------------------------
;   输出部分
    push bp          ;入栈
    mov bp, sp       ;栈顶指针
    push ax          ;ax入栈
    push bx          ;bx入栈
    push cx          ;cx入栈
    push dx          ;dx入栈
    xor cx, cx       ;cx清空
    mov bx, [bp+4]   ;num放入bx
    mov ax, bx       ;赋值
    mov bx, 10       ;十进制
divide:              ;进制转换
    xor dx, dx       ;清空
    div bx           ;除以10取余数，dx:ax / bx = ax.....dx
    add dl, 30h      ;ASCII转换
    push dx          ;商入栈
    inc cx           ;数字位数+1
```

```
    cmp ax, 0        ;商为0，结束
    jne divide       ;商不为0，继续除
output:              ;循环输出开始
    pop dx           ;依次弹出结果
    mov ah, 2        ;输出方式
    int 21h          ;中断
    loop output      ;循环输出结束
    pop dx           ;dx出栈
    pop cx           ;cx出栈
    pop bx           ;bx出栈
    pop ax           ;ax出栈
    pop bp           ;bp出栈
;   输出部分结束
;-------------------------------------------------------------
    mov ah, 4ch
    int 21h
code ends
       end start
```

Q&A

回车怎么打印?

`0DH`是换行，`0AH`是回车，将此十六进制数放入`ax`寄存器并执行中断，将字符输入到屏幕上，就可以完成换行

读和写怎么区分?

`01H`是写，`02H`是读，将此十六进制数放入`ax`寄存器并执行中断，可以向计算机输入字符，或从计算机输出字符

心得

通过本次作业演练，更加加深了对循环和跳转的理解，也对计算机底层寄存器和内存的调用更加熟悉，顺便复习了进制转换的知识

参考资料

1. gcc编译器的使用
2. 数据段
3. 汇编：8086的内存管理方式及数据寻址方式
4. Intel8086处理器-段寄存器ES/DS/CS/SS与寻址
5. 8086汇编 常用指令
6. 8086汇编，十进制转换十六进制
7. 8086系列（20）：十六进制到十进制的转换程序
8. 8086汇编：输入输出数字、字符、字符串功能
9. 8086汇编学习之代码段、数据段、栈段与段地址寄存器
10. 8086CPU中14个寄存器的详解
11. 汇编语言（十）——条件判断指令
12. 汇编语言--cmp指令

13. 汇编语言 CMP指令
14. 汇编->显示26个英文字母
15. 汇编语言（第3版）