

# 代码的直接定址表

贺利坚 主讲



汇编语言程序设计  
Assembly Language

# 使用代码的直接定址表解决问题

## 直接定址表法

- 📁 用查表的方式，通过依据数据，直接计算出所要找的元素的位置

## 直接定址表分类

- 📁 数据的直接定址表
- 📁 代码的直接定址表

### 子程序入口参数说明：

- (1) 用ah 寄存器传递功能号：
  - 0 表示清屏，
  - 1表示设置前景色，
  - 2 表示设置背景色，
  - 3 表示向上滚动一行；
- (2) 对2、3号功能，用al传送颜色值， $(al) \in \{0, 1, 2, 3, 4, 5, 6, 7\}$

## 要解决的问题

实现一个子程序setscreen，为显示输出提供如下功能

- (1) 清屏。
- (2) 设置前景色。
- (3) 设置背景色。
- (4) 向上滚动一行

## 解决方案

- 📁 将4个功能写成4个子程序
- 📁 将这些功能子程序的入口地址存储在一个表中，它们在表中的位置和功能号相对应。
- 📁 对应关系为：功能号 $\times 2$ =对应的功能子程序在地址表中的偏移。

# 各种功能的实现

( 1 ) 清屏：将显存中当前屏幕中的字符设为空格符

```
sub1:
    push bx      (ah): 功能号
    push cx
    push es

    mov bx,0b800h
    mov es,bx
    mov bx,0
    mov cx,2000

sub1s:
    mov byte ptr es:[bx], ' '
    add bx,2
    loop sub1s

    pop es
    pop cx
    pop bx
    ret ;sub1 ends
```

( 2 ) 设置前景色：设置显存中奇地址的属性字节的第0、1、2位

```
sub2:
    push bx      (ah): 功能号
    push cx      (al): 颜色
    push es

    mov bx,0b800h
    mov es,bx
    mov bx,1
    mov cx,2000

sub2s:
    and byte ptr es:[bx],11111000b
    or es:[bx],al
    add bx,2
    loop sub2s

    pop es
    pop cx
    pop bx
    ret ;sub2 ends
```

( 3 ) 设置背景色：设置显存中奇地址的属性字节的第4、5、6位

```
sub3:
    push bx      (ah): 功能号
    push cx      (al): 颜色
    push es

    mov cl,4
    shl al,cl
    mov bx,0b800h
    mov es,bx
    mov bx,1
    mov cx,2000

sub3s:
    and byte ptr es:[bx],10001111b
    or es:[bx],al
    add bx,2
    loop sub3s

    pop es
    pop cx
    pop bx
    ret ; sub3 ends
```

# 各种功能的实现(续)

(4) 向上滚动一行：依次将第 n+1 行的内容复制到第 n 行处：最后一行为空。

```
sub4:
    push cx
    push si
    push di
    push es
    push ds
```

;准备

;复制前24行

;清空最后一行

```
    pop ds
    pop es
    pop di
    pop si
    pop cx
    ret ;sub4 ends
```

```
mov si,0b800h
mov es,si
mov ds,si
mov si,160    ;ds:si指向第n+1行
mov di,0      ;es:di指向第n行
cld
mov cx,24;共复制24行
```

```
sub4s:
    push cx
    mov cx,160
    rep movsb    ;复制
    pop cx
    loop sub4s
```

```
mov cx,80
mov si,0
sub4s1:
    mov byte ptr es:[160*24+si],' '
    add si,2
    loop sub4s1
```

```
;主程序
assume cs:code
code segment
start:
```

;主程序

;setscreen子程序

;各功能的实现

```
code ends
end start
```

```
sub1: ... ..
sub2: ... ..
sub3: ... ..
sub4: ... ..
```

```
mov ah, 2
mov al, 5
call screenset
mov ax, 4c00h
int 21h
```

```
setscreen:
    jmp short set
table dw sub1,sub2,sub3,sub4
set:
    push bx
    cmp ah,3
    ja sret
    mov bl,ah
    mov bh,0
    add bx,bx
    call word ptr table[bx]
sret:
    pop bx
    ret
```

# 直接写址表的优势

```
mov ah, 2
mov al, 5
call setscreen
mov ax, 4c00h
int 21h
```

; 主程序

assume cs:code

code segment

start:

; 主程序

; setscreen子程序

; 各功能的实现

code ends

end start

```
sub1: ... ..
sub2: ... ..
sub3: ... ..
sub4: ... ..
```

```
setscreen:
    jmp short set
    table dw sub1,sub2,sub3,sub4
set:
    push bx
    cmp ah,3
    ja sret
    mov bl,ah
    mov bh,0
    add bx,bx
    call word ptr table[bx]
sret:
    pop bx
    ret
```

要在 setscreen 中再加入新功能，只需要在地址表中加入它的入口地址就可以了。结构清晰，便于扩充。

要在 setscreen 中再加入新功能，则需要修改程序的逻辑，加入新的比较、转移指令。

```
setscreen:
    cmp ah,0
    je do1
    cmp ah,1
    je do2
    cmp ah,2
    je do3
    cmp ah,3
    je do4
    jmp short sret
do1:call sub1
    jmp short sret
do2:call sub2
    jmp short sret
do3:call sub3
    jmp short sret
do4:call sub4
sret:ret
```