汇编语言：显存直接操作与简单动画入门

如果你只熟悉 `INT 21H` 的字符串输入输出（如 09H 号功能调用），那么接触到“字符弹跳”这种涉及屏幕定位和颜色的程序时，可能会觉得跨度很大。

其实，在 DOS 下做这种“图形”效果（准确说是**彩色文本模式**操作），核心原理非常简单：**屏幕就是一段内存。**

1. 核心概念：屏幕即内存

在 DOS 的 80x25 彩色文本模式下，显卡将一段特定的内存区域映射到了显示器上。
这段内存的起始物理地址是 `B8000H`。

在汇编中，我们通常将 `ES` 段寄存器指向这个段地址：

```
MOV AX, 0B800H
MOV ES, AX
```

一旦 `ES` 指向了 `0B800H`，你往 `ES:[偏移地址]` 里写数据，屏幕上立马就会显示出来，不需要调用任何 DOS 中断。

2. 显存的结构

屏幕被分成了 25 行，每行 80 个字符。
屏幕上的每一个字符，在显存中占用 2 个字节（Byte）。

- **低字节 (偶数地址)**: 存放字符的 ASCII 码。
- **高字节 (奇数地址)**: 存放字符的**属性**（颜色）。

内存布局示意

假设屏幕左上角（第0行，第0列）：

- `ES:[0000]` : 第1个字符的 ASCII 码
- `ES:[0001]` : 第1个字符的 颜色属性
- `ES:[0002]` : 第2个字符的 ASCII 码
- `ES:[0003]` : 第2个字符的 颜色属性
- ...

坐标计算公式

如果你想在 **第 Y 行，第 X 列** 显示一个字符，偏移地址（Offset）怎么算？

1. 一行有 80 个字符。
2. 每个字符占 2 个字节。
3. 所以一行占 $80 \times 2 = 160$ 字节。

$$\text{Offset} = (\text{行号} \times 80 + \text{列号}) \times 2$$

汇编实现套路：

```
; 假设 DH = 行号, DL = 列号
MOV AL, 160         ; 一行的字节数
MUL DH              ; AX = 行号 * 160
MOV DI, AX          ; DI 存放行偏移

MOV AL, 2
MUL DL              ; AX = 列号 * 2
ADD DI, AX          ; DI = 最终偏移地址 (行偏移 + 列偏移)

; 此时 ES:[DI] 就是目标位置
MOV BYTE PTR ES:[DI], 'A'   ; 写入字符 'A'
MOV BYTE PTR ES:[DI+1], 04H ; 写入属性 (红色)
```

3. 颜色属性字节 (Attribute Byte)

属性字节的 8 个位（Bit）决定了字符的颜色和闪烁：

7	6	5	4	3	2	1	0
BL	R	G	B	I	R	G	B
闪烁	**背景色**		高亮	**前景色(文字)**			

- **前景色 (0-2位)**: 文字的颜色。
- **高亮 (3位)**: 让文字变亮。
- **背景色 (4-6位)**: 文字底部的颜色。
- **闪烁 (7位)**: 文字是否闪烁。

常用颜色代码 (十六进制):

- `07H`: 黑底白字 (标准)
- `02H`: 黑底绿字
- `04H`: 黑底红字
- `1EH`: 蓝底黄字
- `87H`: 黑底白字 + 闪烁

4. 动画原理：擦除与重绘

`字符弹跳.asm` 这种程序的动画逻辑其实就是一个死循环，每一步做三件事：

1. **擦除旧的**：在原来的坐标位置写入一个“空格”字符（或者恢复原来的背景）。
2. **计算新的**：根据方向更新坐标（行、列），并处理边界碰撞（如果撞墙就反向）。
3. **绘制新的**：在新的坐标位置写入你的字符和颜色。
4. **延时**：为了让人眼能看清，通常需要加一个延时循环。

伪代码逻辑

```
初始化位置 X, Y
初始化方向 DX, DY

循环:
    1. 计算偏移地址 Offset(X, Y)
    2. ES:[Offset] = ' '        ; 擦除旧的

    3. 更新 X = X + DX
    4. 更新 Y = Y + DY

    5. 检查边界:
       如果 X < 0 或 X > 79 -> DX = -DX (反弹)
       如果 Y < 0 或 Y > 24 -> DY = -DY (反弹)

    6. 计算新偏移地址 Offset(X, Y)
    7. ES:[Offset] = 'O'        ; 绘制新的
    8. ES:[Offset+1] = 颜色

    9. 延时
    跳转回循环
```

5. 总结

相比于 `INT 21H` 的字符串输出，直接写显存的优点是：

1. **速度极快**：没有系统调用的开销。
2. **定位精准**：想在屏幕哪里写就在哪里写。
3. **颜色丰富**：可以控制每个字符的颜色。

这就是为什么做游戏（贪吃蛇、弹球）或者复杂的菜单界面时，都会使用这种方式。