实验名称：X86 汇编实现双人贪吃蛇游戏

学　　院： 睿信书院

专　　业： 计算机科学与技术

班　　级： 07812201

姓　　名： 曾洳睫（1820221053）、

丘绎楦（1820221050）、

孙将斌（1820221055）、

陈家豪（1820221047）、

陈文盛（1820221069）

指导教师： 张全新

目录

1 实验目的

编写一个游戏程序，如俄罗斯方块、贪吃蛇、扫雷、简单射击类游戏等，题目自拟，要求具备一定功能难度，代码性能高。

2 游戏功能简介

本组采用汇编语言实现了一个经典的贪吃蛇小游戏，并且采用 VS Code 环境下的 DOSBOX 环境运行，我们在传统的游戏模式下引入双人游玩的模式，提高了游戏的趣味性也增加了游戏实现逻辑的复杂度。我们采用控制台程序的方式开发此游戏，设计精简的游戏画面。

2.1 游戏玩家操作

本实验是一款双人对战贪吃蛇游戏。游戏中，两条颜色不同的蛇分别由两名玩家操控。若玩家未进行操作，蛇将沿着之前的移动方向继续前行。一名玩家通过“上”、“下”、“左”、“右”方向键控制蛇的移动，另一名玩家使用“W”、“A”、“S”、“D”键进行方向调整

2.2 输赢规则

在我们的双人贪吃蛇游戏中，若任意一名玩家的蛇发生以下情况之一，则游戏结束：撞到墙壁、撞上自己的身体或对方的身体。撞上墙壁、自己或别人，则视为对方获胜。并且我们采用了抢分机制，任意一方先到达 10 分，则该方获得胜利。

2.3 界面及道具说明

界面内容主要包括蛇形、食物、分数记录。界面左、右上角分别为双方分数记录，可以使玩家清晰地看到自己当前分数。

3 程序实现

3.1 数据定义

在本实验的双人贪吃蛇游戏中，数据定义部分是程序实现的基础，为游戏运行提供了必要的变量和初始值。以下是数据定义的详细说明：

```
.data
        snakeHead       dw 0
        snakeBody       dw 1001 dup (0)                 ; 蛇身体坐标数据
        body            db 0dbh                         ; 身体图案
        snakeColor      db 08h                          ; 身体颜色
        snakeHead2      dw 0
        snakeBody2      dw 1001 dup (0)                 ; 蛇身体坐标数据
        body2           db 0dbh                         ; 身体图案
        snakeColor2     db 02h                          ; 身体颜色
        food            dw 3 dup(0)                     ; 食物坐标位置
        foodColor       db 04h                          ; 食物颜色存
        oldint9         dw 2 dup (0)                    ; 原来 int9 中断地址暂
        gameStatus      db 4                            ; 状态：0 游戏中 1 退出 2 暂停 3
        initTarget      db 'R'                          ; 初始方向
        initTarget2     db 'L'
        target          db 'R'                          ; 方向
        target2         db 'L'
        speed           dw 02h                          ; 速度
        speed2          dw 02h
        initLen         dw 4                            ; 初始长度
        len             dw 4                            ; 长度
        len2            dw 4
        score           dw 0                            ; 得分
        score2          dw 0
        gameoverStr     db "GAME OVER!", 0              ; 游戏结束提示
        gamewinnerStr   db "Player 1 Wins!", 0
        gamewinner2Str  db "Player 2 Wins!", 0
        scoreStr        db "Player 1 Score: ", 0        ; 游戏得分提示
        scoreStr2       db "Player 2 Score: ", 0
```

图：数据定义

3.1.1 蛇与食物相关数据

说明数据定义中的变量含义：

1) snakeHead 和 snakeBody 分别用于存储玩家 1 的蛇头位置及身体坐标，snakeBody 使用数组存储最多 1001 节蛇身。snakeHead2 和 snakeBody2 为玩家 2 的蛇头位置及身体坐标，定义方式与玩家 1 相同。
2) body 和 snakeColor 分别定义玩家 1 的蛇身图案和颜色，其中 body 的值为 0DBh，表示蛇身字符的 ASCII 编码，snakeColor 的值为 08h，表示蛇的颜色。对应地，body2 和 snakeColor2 定义玩家 2 的蛇身图案和颜色，snakeColor2 的值为 02h。
3) food 定义为一个 3 元数组，用于存储食物的坐标位置。foodColor 定义为 04h，用于表示食物的显示颜色。

3.1.2 游戏状态与控制数据

1) 游戏状态：gameStatus 表示游戏的当前状态，其值可以为以下几种：0：游戏中、1：退出、2：暂停、3：游戏失败、4：准备状态
2) 初始方向：initTarget 和 initTarget2 分别定义玩家 1 和玩家 2 的蛇初始移动方向，默认为 'R'（右）和 'L'（左）。target 和 target2 用于实时存储玩家当前的移动方向。player 为该物体属于哪一个玩家，因为绘制的物体中有玩家 1，2 的蛇部分还有玩家 1，2 碰到物体的显示的表情，需要保存蛇和表情是属于玩家 1，还是玩家 2 。

3.1.3 游戏速度与长度

1) 速度：speed 和 speed2 定义玩家 1 和玩家 2 的蛇移动速度，初始值均为 02h，表示游戏刷新速度。
2) 长度：initLen 表示初始蛇身长度，默认值为 4。len 和 len2 分别存储玩家 1 和玩家 2 的蛇身长度，在游戏过程中动态变化。

3.1.4 分数与提示信息

1) 分数:score 和 score2 分别存储玩家 1 和玩家 2 的得分,初始值为 0。
2) 提示信息：gameoverStr 存储 "GAME OVER!" 提示字符串，用于游戏失败时显示。gamewinnerStr 和 gamewinner2Str 分别存储玩家 1 和玩家 2 获胜时的提示信息。scoreStr 和 scoreStr2 分别存储玩家 1 和玩家 2 的得分提示文字。restartStr 存储 "Press the R key to restart!" 提示信息，用于游戏结束后的重玩提示。

3.1.5 随机数种子与其他数据

1) 随机数种子：seed 和 seed2 分别为玩家 1 和玩家 2 的随机数种子，用于生成食物的随机坐标位置。
2) 其他数据：winner 用于表示游戏的赢家，其值为 0 时表示平局，为 1 或 2 时分别表示玩家 1 或玩家 2 获胜。startStr 定义了游戏开始界面的

ASCII 图形与提示信息，包括欢迎语、操作指南等，增强了程序的交互性。

3.2 功能实现

本部分介绍显示功能的实现，主要包括 **单个字符的显示** 和 **单个方块的显示**，它们共同构成了游戏画面展示的核心。

3.2.1 单个字符的显示

功能描述：

display1 函数用于在指定的控制台位置显示一个字符。输入参数包括行号（dh）、列号（dl）、字符颜色（ch）以及字符内容（cl）。

实现原理：

控制台的字符显示依赖于内存段 0xB800，每个字符占用 2 个字节（一个用于字符本身，一个用于颜色属性）。通过计算 行号 × 160 和 列号 × 2 的偏移量，确定显示字符的内存位置。字符与颜色设置将字符内容和颜色组合成一个 word 数据写入指定位置，实现字符的颜色显示。

```
; 显示单个字符函数
; @ dh 行
; @ dL 列
; @ ch 颜色
; @ cL 内容
display1:       push  es
                push  di
                push  ax

                mov   ax, 0b800h
                mov   es, ax
                mov   di, 0

                mov   al, 160
                mul   dh
                add   di, ax

                mov   al, 2
                mul   dl
                add   di, ax

                mov   word ptr es:[di], cx

                pop   ax
                pop   di
                pop   es
                ret
```

3.2.2 单个方块的显示

功能描述：

display2 函数用于显示一个 2 列宽的方块，常用于绘制游戏中的蛇身和其他图形元素。输入参数包括行号（dh）、列号（dl）、颜色和内容（cx）。

实现原理：

相较于 display1，该函数在指定位置显示一个宽度为 2 的方块，占用 4 个字节的内存。通过连续写入两个 word 数据，实现方块的显示效果。

```
; 显示单个方块函数
; @ dh 行
; @ dl 列
display2:
            push  es
            push  di
            push  ax
            push  ds

            mov   ax, @data
            mov   ds, ax

            mov   ax, 0b800h
            mov   es, ax
            mov   di, 0

            mov   al, 160
            mul   dh
            add   di, ax

            mov   al, 4
            mul   dl
            add   di, ax

            mov   word ptr es:[di], cx
            mov   word ptr es:[di + 2], cx

            pop   ds
            pop   ax
            pop   di
            pop   es
            ret
```

3.2.3 安装按键中断程序

功能描述：

按键中断程序是游戏的重要模块，用于响应用户键盘输入，通过自定义中断处理程序实现玩家对游戏的控制。此部分代码包含 中断处理程序的安装 和 恢复 功能。

实现原理：

键盘中断 int9h 的入口地址存储在中断向量表中，通过保存地址 (es:[9 * 4] 和 es:[9 * 4 + 2]) 将其备份到变量 oldint9，以便后续恢复。将自定义按键处理程序（do0）复制到内存地址 200h，确保程序能够正确运行。将键盘中断向量指向新程序起始地址 200h。

```
push  es:[9 * 4]
pop   oldint9[0]
push  es:[9 * 4 + 2]
pop   oldint9[2]

mov   ax, seg do0
mov   ds, ax
mov   si, offset do0

mov   ax, 0
mov   es, ax
mov   di, 200h

mov   cx, offset do0end - offset do0
cld
rep   movsb

mov   word ptr es:[9 * 4], 200h
mov   word ptr es:[9 * 4 + 2], 0
```

3.2.4 运动模块实现

功能描述：

运动模块是游戏的核心功能之一，负责蛇在游戏中的移动、食物显示、以及游戏状态更新。以下代码实现了两条蛇根据当前方向运动的逻辑，同时包含对食物的绘制和碰撞检查。

功能实现分析：

两条蛇的运动逻辑：

根据当前方向（target[0] 和 target2[0]），更新蛇头坐标（snakeHead 和 snakeHead2）。运动方向通过比较方向字符（'U', 'D', 'L', 'R'）进行判断，并调整对应坐标值。

核心代码片段：

```
cmp   target[0], 'U'    ; 判断方向是否为向上
jne   run_D             ; 如果不是，跳转到向下的逻辑
dec   dh                ; 向上移动，行号减 1
jmp   run_U2            ; 跳转到统一处理逻辑
```

统一坐标更新：

```
mov   snakeHead, dx     ; 更新蛇 1 的头部位置
mov   snakeHead2, dx    ; 更新蛇 2 的头部位置
```

调用功能模块

- move：更新蛇身体的整体位置。
- disAll：显示整条蛇的身体在屏幕上。
- check：检测蛇是否碰撞、吃食物或其他特殊情况。

绘制食物

使用循环显示食物坐标（food），并调用 display2 绘制方块图案：

```
food_dis:
mov   dx, food[si]    ; 读取当前食物的坐标
mov   cl, body[0]     ; 设置食物显示的字符
mov   ch, foodColor[0] ; 设置食物颜色
call  display2        ; 调用显示函数
add   si, 2           ; 处理下一块食物
cmp   si, 6           ; 检查是否绘制完成
jne   food_dis        ; 未完成则继续
```

清理尾部块

- 使用 clearend 函数清除蛇移动后的尾部块，保持屏幕整洁。

延时

- 通过 delay 控制移动速度，确保游戏节奏适中。

3.2.5 蛇身体移动模块实现

功能描述：

move 模块负责更新蛇的身体位置，使其实现动态移动的效果。通过整体位移的逻辑，将新的蛇头坐标插入身体列表的首位，原有的身体块依次向后移动，模拟出蛇移动的行为。

实现原理：

蛇身体位置更新：每次调用函数时，从尾部到头部依次更新蛇身体的位置。新的蛇头坐标 snakeHead 插入身体列表首位（snakeBody[0]）。

```
mov   ax, snakeBody[bx - 2]  ; 获取当前块的前一个块位置
mov   snakeBody[bx], ax      ; 更新当前位置为前一个块的坐标
sub   bx, 2                  ; 移动到上一个块
cmp   bx, 0                  ; 判断是否处理到蛇头
jne   move_s                 ; 如果未到头部，则继续循环
```

3.2.6 游戏校验模块实现

功能描述：

check 模块负责校验游戏中的各种状态，包括蛇与墙壁、其他蛇、自身体积或食物的碰撞情况，并作出相应的处理。这是游戏逻辑的核心模块，决定了游戏结束或状态的改变。

实现原理：

边界检测：检查蛇头是否超出游戏区域边界。若超出边界，则触发相应的游戏结束逻辑，并调用 win_1 或 win_2 确定获胜方。

关键代码：

```
cmp    al, 40 - 1   ; 检查是否超出右侧墙壁
ja     g_o          ; 若超出，跳转至结束逻辑
cmp    ah, 25 - 1   ; 检查是否超出下方墙壁
ja     g_o
```

蛇与蛇碰撞：比较两条蛇的头部位置，若相同则触发结束逻辑。检查蛇 1 的头部是否与蛇 2 的身体块重合，以及蛇 2 的头部是否与蛇 1 的身体块重合。

关键代码：

```
cmp    ax, bx       ; 检查两条蛇头部是否重合
je     g_o3
```

咬到自身检测：分别从蛇身体的第二节开始循环检查，判断蛇头是否与其自身身体重合。

关键代码：

```
cmp    ax, snakeBody[0] ; 蛇头与身体某块比较
je     g_o              ; 若重合，则触发结束逻辑
```

食物碰撞检测：检查蛇头是否与当前食物位置重合，若重合则触发 levelup 或 levelup2，增加身体长度和分数，同时重新生成食物。

关键代码：

```
cmp    ax, food[si]  ; 检查蛇头是否碰到食物
je     levelup
```

分数与胜利判断：每次吃到食物后增加分数，若分数达到 9，则当前蛇玩家获胜。调用 win_1 或 win_2 标记胜利方，并显示结束信息。

关键代码：

```
cmp    score[0], 9    ; 检查分数是否达到获胜条件
jna    notWin         ; 若未达到，继续游戏
```

游戏结束逻辑：触发不同的结束分支 (g_o, g_o2, g_o3)。调用 dis_g_o 显示游戏结束信息。

3.2.7 随机生成食物模块实现

功能描述：

random 模块负责为游戏中的蛇生成随机位置的食物。通过结合系统时间和预设种子 (seed 和 seed2) 来生成伪随机数，并确保生成的食物不会与蛇的身体重合。

实现原理：

功能实现分析：

时间驱动的伪随机数生成：通过读取 CMOS 实时时钟寄存器的秒数 (70h, 71h) 作为随机种子。对种子进行简单的位操作（如右移）和数学运算，生成伪随机数。

关键代码：

```
mov    al, 0          ; CMOS 秒数寄存器
out    70h, al        ; 选择秒数寄存器
in     al, 71h        ; 读取秒数
mov    ah, al         ; 将秒数存入 AH
add    ax, seed[0]    ; 加入种子
mov    cl, 3
shr    ax, cl         ; 右移 3 位，生成伪随机数
```

坐标生成：随机生成一个坐标点 (x, y)，范围为游戏区域的宽度和高度：x ∈ [0, 40)、y ∈ [0, 25)，使用除法取余的方法将随机数限制在合法范围内。

关键代码：

```
mov    dl, 40         ; 游戏宽度
```

```
div   dl                ; 生成 x 坐标
mov   byte ptr food[si], ah
mov   dl, 25            ; 游戏高度
div   dl                ; 生成 y 坐标
mov   byte ptr food[si + 1], ah
```

避免食物与蛇重合：检查生成的食物位置是否与蛇 1 或蛇 2 的身体重合：如果重合，重新生成随机坐标。使用循环依次检查蛇的每个身体块。

关键代码：

```
mov   cx, len[0]        ; 获取蛇 1 长度
rd_s:                   ; 循环检查
mov   ax, snakeBody[di] ; 获取蛇身体坐标
cmp   ax, dx            ; 比较是否重合
je    re_create         ; 重合则重新生成
```

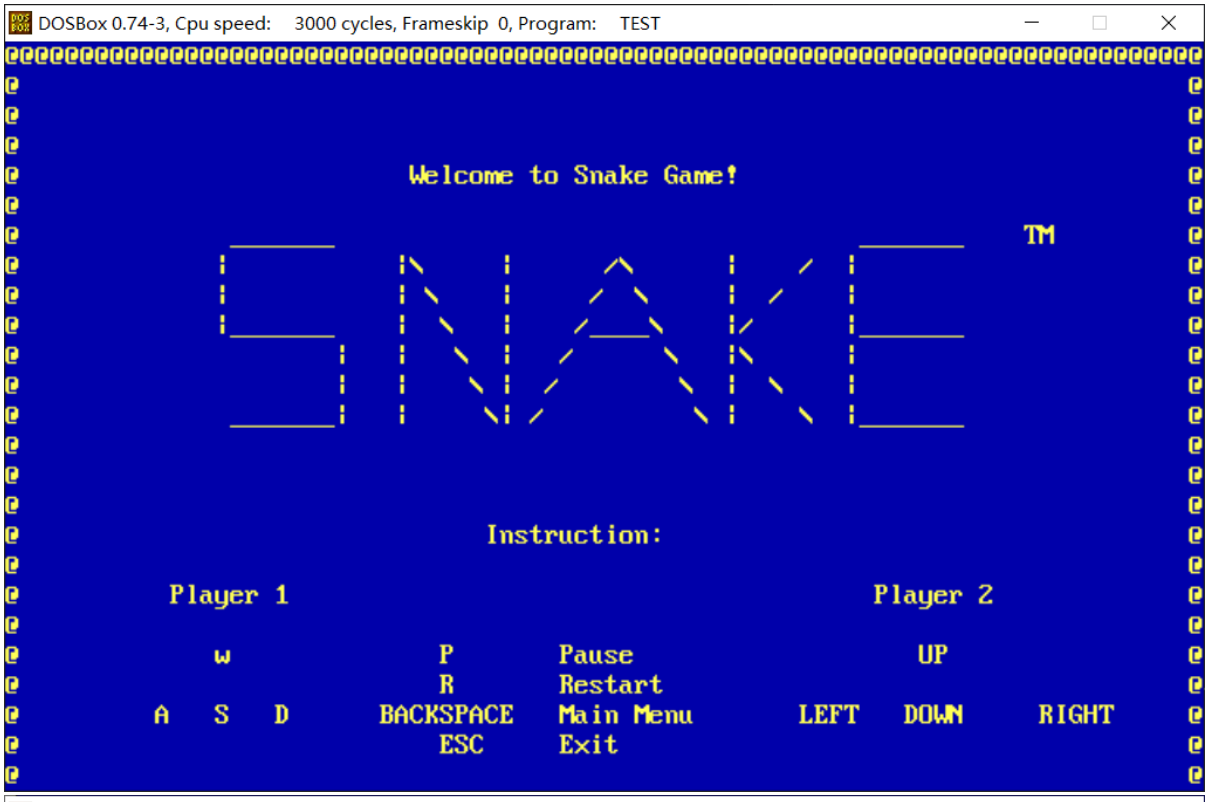
填充多个食物位置：每次生成两个字节（x, y）表示一个食物的坐标。生成 3 个食物位置（共 6 字节）。

关键代码：

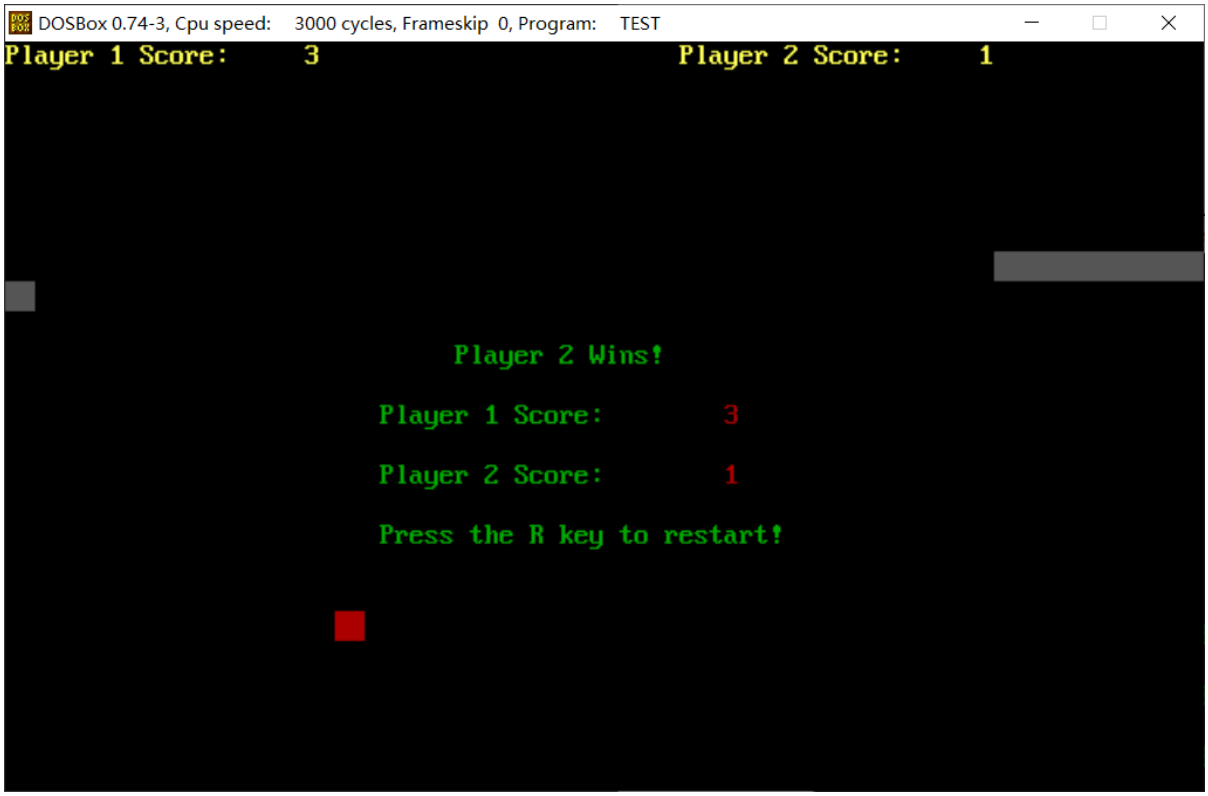
```
add   si, 2             ; 生成下一个食物
cmp   si, 6             ; 检查是否完成 3 个食物
jne   re_create         ; 未完成则继续生成
```

循环结束与返回：确保所有寄存器和堆栈被正确恢复，程序可以继续其他操作。

4 运行截图


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST
Welcome to Snake Game!
TM
Instruction:
Player 1
Player 2
W
A S D
P Pause
R Restart
BACKSPACE Main Menu
ESC Exit
UP
LEFT DOWN RIGHT


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST
Player 1 Score: 2
Player 2 Score: 1



5 难点及解决方案

5.1 按键冲突问题

问题：

在控制蛇移动时，如果玩家同时按下相反方向的按键（比如 W 和 S 或 ↑ 和 ↓），会导致蛇运动逻辑混乱，例如停滞不动或直接游戏结束。这会影响游戏体验。

解决方法：

限制反向按键：

当蛇正在向某个方向移动时，忽略与当前方向相反的按键输入。例如，如果蛇正在向上移动（W），就忽略向下（S）的输入。

实现示例：

```
cmp    al, 11h                ; 检测 W 键
jne    dkn_a
cmp    target[0], 'D'         ; 当前方向是向下?
je     ignore_key             ; 忽略反向按键
```

```
mov    target[0], 'U'              ; 更新为向上
jmp    do0over
ignore_key:                        ; 忽略无效按键
jmp    do0over
```

6 组员及分工

小组包含 5 位成员，分别是：

1. 曾泇睫（1820221053）（组长）：代码功能实现
2. 丘绎楦（1820221050）（组员）：代码功能实现
3. 孙将斌（1820221055）（组员）：报告撰写
4. 陈文盛（1820221069）（组员）：代码功能实现
5. 陈家豪（1820221047）（组员）：报告撰写