汇编语言实现俄罗斯方块游戏项目报告

一、项目介绍

1.1 项目背景

俄罗斯方块是一款经典的益智游戏，由俄罗斯计算机工程师阿列克谢·帕基特诺夫于 1984 年发明。本项目旨在使用汇编语言重新实现这一经典游戏，通过底层编程的方式深入理解计算机系统的工作原理、中断机制以及图形显示技术。

1.2 项目目标

- 完全使用汇编语言实现俄罗斯方块的核心游戏逻辑；
- 通过 DOS 中断实现图形化界面和键盘交互；
- 利用系统时钟中断实现方块自动下落；
- 构建完整的游戏流程，包括开始界面、游戏进行和结束界面；
- 实现计分系统、下一个方块预览等增强功能。

1.3 开发环境

- 汇编器：MASM
- 运行环境：DOSBox 模拟器

二、项目功能

2.1 核心游戏功能

(1) 方块生成与随机化

- 支持 7 种经典方块形状（I、O、T、L、J、S、Z）；
- 随机生成下一个方块，确保游戏不确定性；
- 当前方块掉落时，下一个方块的预览图可以同时显示。

(2) 方块控制与移动

- 左右移动（A/D 键）：方块在游戏区域内水平移动；
- 旋转（W 键）：方块顺时针旋转 90 度；
- 加速下落（S 键）：方块直接下落至底部；
- 边界检测：防止方块移出游戏区域。

(3) 游戏逻辑处理

- 速度选择：进入游戏后，有快/中/慢三级速度选择。
- 碰撞检测：实时检测方块与已固定方块的碰撞；
- 消行判断：当一行被完全填满时自动消除；
- 计分系统：根据消行数量计算得分，支持多位数显示；
- 游戏结束判断：当新方块无法放置即堆满时游戏结束。

2.2 用户界面功能

(1) 游戏主界面

游戏区域位 20×10 的标准俄罗斯方块场地,并且有清晰的游戏区域边界。当前方块为正在下落的可操作方块，右侧显示下一个方块的类型。已落地方块不可操作且位置固定。发生消除后，已落地方块会统一变为紫色。

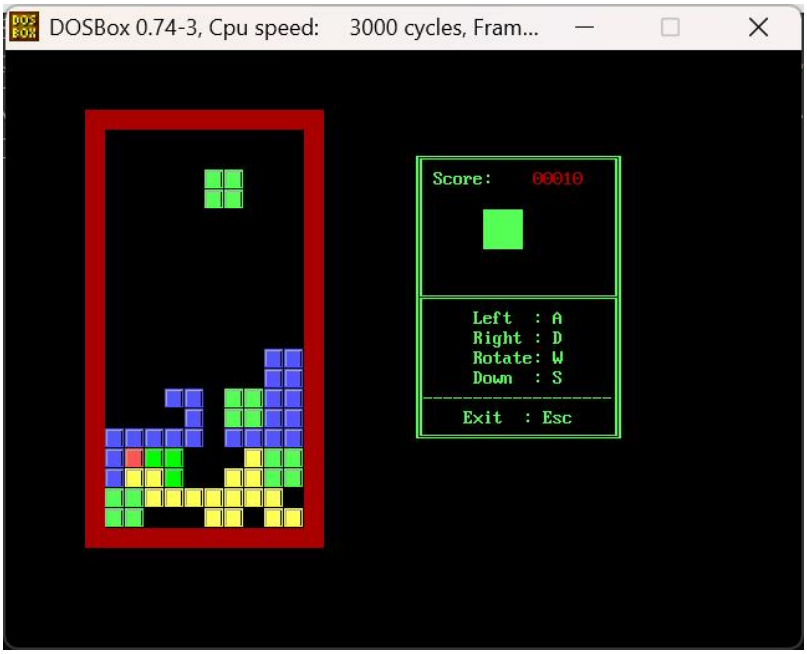

图 1 游戏界面图

(2) 信息显示区域

最上方 score 实时显示当前得分。下方为下一个方块预览，显示即将下落的方块的类型和颜色。同时配有操作指南，显示键盘控制说明。底部为退出说明，按 ESC 键可退出游戏。

图 2 信息显示区域

(3) 菜单系统

开始菜单：游戏速度选择界面。

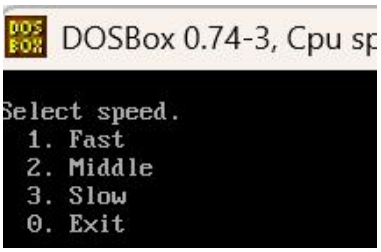

图 3 开始菜单/速度选择界面

游戏结束界面：显示“GAME OVER”和退出提示。

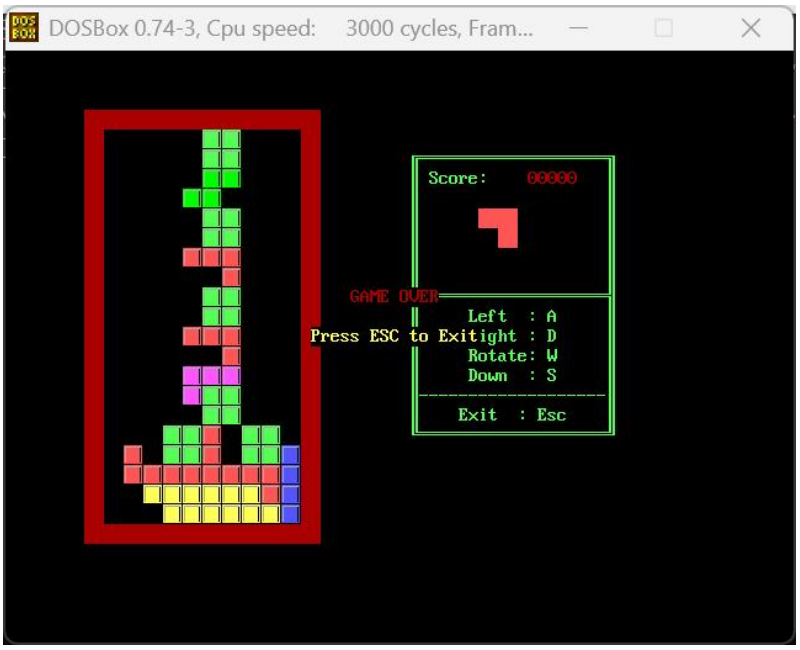

图 4 游戏结束界面

三、项目开发流程

3.1 项目规划阶段

在第 11 周得到期末项目要求后，我开始筹划我的项目。

首先是需求分析与设计。我计划用汇编语言实现一个小型游戏。所以在第 11 周，我阅读了大量有关汇编语言实现小型游戏项目的源码及文档，大部分游戏集中在贪吃蛇、战机对抗、推箱子等单机经典游戏。于是我自然而然地想到了小时候非常喜欢的游戏“俄罗斯方块”，并且在当周确定了游戏的基本规则和功能，计划利用键盘进行用户交互，设计了初步的界面布局。

3.2 设计编码阶段

在第 12 周，我开始了详细的设计和编码工作。

首先需要确定游戏的数据结构。考虑到俄罗斯方块有 7 种不同形状的方块，每个方块有 4 种旋转方向，我决定使用一个 4×4 的二进制矩阵来表示每个方块在每个方向上的形状。为了节省内存，我将每个方块的数据压缩为 4 个字节，每个字节代表一行，每行用 4 位表示。这样，7 种方块×4 个方向×4 行，总共需要 112 字节。

```
; 方块数据存储结构定义
PAD DB 7*4*4 DUP(?)  ; 7种方块 * 4个方向 * 4行 = 112字节

; 方块初始化示例 - I型方块
INIT_PAD_DATA PROC NEAR
    MOV DI, OFFSET PAD

    ; 方块1 - I型
    ; Rotation 0 (Horizontal)
    MOV AL, 00001111b  ; 水平方向：连续4个块
    STOSB
    MOV AL, 00000000b  ; 其余行为空
    STOSB
    MOV AL, 00000000b
    STOSB
    MOV AL, 00000000b
    STOSB

    ; Rotation 1 (Vertical)
    MOV AL, 00100000b  ; 垂直方向：每行一个块，位置居中
    STOSB
    MOV AL, 00100000b
    STOSB
    MOV AL, 00100000b
    STOSB
    MOV AL, 00100000b
    STOSB
    ; ... 其他方块定义省略
INIT_PAD_DATA ENDP
```

接着需要设计游戏板的数据结构。游戏板是一个 20 行×10 列的区域，但为了简化边界检测，我将其设计为 24 行×16 位，实际上每行用 16 位表示，其中中间 10 位用于游戏区域，左右各 3 位用于边界检测。这样，游戏板可以用 24 个 16 位字来表示。

```
; 游戏板数据结构定义
BOARD DW 24 DUP(?),0FFFFH  ; 24行游戏区域，每行16位

; 初始化游戏板边界
INITGAME PROC NEAR
    CLD
    MOV DI,OFFSET BOARD
    MOV CX,24
    MOV AX,0E007H  ; 1110000000000111（0为游戏区域，1为边界）
    REP STOSW      ; 初始化所有行
    RET
INITGAME ENDP
```

之后的时间，我的主要工作集中在方块的形状数据初始化，图形的显示，方块的生成、移动和旋转。这部分函数的编写需要特别注意边界检测和碰撞检测，确保方块不会移出游戏区域或与已固定的方块重叠。

3.3 功能完善阶段

在第 13 周，我继续完善游戏的核心模块。

根据游戏的规则，我先编写了碰撞检测函数（CHECK），该函数通过位运算来检测当前方块在当前位置是否与游戏板上的已有方块发生重叠。

```
; 碰撞检测函数
CHECK PROC NEAR                ; 返回AL=0或F 0为OK F为NO
    MOV AH,0H
    MOV AL,TY
    ADD AL,TY                  ; 行号*2（每个条目是字）
    MOV SI,OFFSET BOARD        ; 把当前的游戏情况加载到SI
    ADD SI,AX                  ; 定位到当前行

    MOV DI,00H
    MOV CX,04H                 ; 检查4行
    CLD
LOOP6:
    LODSW                      ; 将SI中的内容加载到AX
    MOV BX,P2[DI]              ; 获取P2字（当前方块形状）
    AND AX,BX                  ; 与游戏板对应位置进行AND操作
    JNZ SKIP4                  ; 如果结果不为0，说明有重叠
    INC DI
    INC DI                     ; 移动到下一个字
    LOOP LOOP6                 ; 循环判断所有4行

    MOV AL,00H                 ; 返回0表示无碰撞
    RET
SKIP4:
    MOV AL,0FH                 ; 返回0FH表示有碰撞
    RET
CHECK ENDP
```

对于已下落方块，需要实现方块固定到游戏板上的功能（PUT）。当方块无法继续下落时，将其形状合并到游戏板（BOARD）中，并检查是否有完整的行可以消除。消除行的算法是从底部向上扫描，如果某一行被完全填满，即该行的 16 位中，中间 10 位全为 1，则将该行消除，并将上面的所有行下移。同时，根据消除的行数计算得分，消除一行得 10 分。

```
; 方块固定和消行函数
PUT PROC NEAR                   ; 消除整行
    ; 将当前方块合并到游戏板
    MOV AH,0H
    MOV AL,Y
    ADD AL,Y                    ; 行号*2
    MOV SI,OFFSET BOARD
    ADD SI,AX                   ; 定位到当前行
    MOV DI,00H
    MOV CX,04H
    CLD
LOOP15:
    LODSW                       ; 加载游戏板当前行
    MOV BX,P1[DI]               ; 获取当前方块行
    OR AX,BX                    ; 合并到游戏板
    MOV [SI-2],AX               ; 写回游戏板
    INC DI
    INC DI
    LOOP LOOP15

    ; 消行检测（从底部向上）
    MOV SI,OFFSET BOARD
    ADD SI,23*2                 ; 从最后一行开始
    MOV DI,SI
    MOV CX,20                   ; 检查20行（游戏区域）
    MOV FLG,00H                 ; 消行标志清零
    STD                         ; 从后向前处理
LOOP13:
    LODSW                       ; 加载一行
    CMP AX,0FFFFH               ; 检查是否整行填满
    JNE SKIP12                  ; 不是整行跳转
    MOV FLG,0FFH                ; 设置消行标志
    MOV AL,DV
    SAL AL,1                    ; 得分加倍（消行越多得分越高）
    MOV DV,AL
    JMP LOOP13
SKIP12:
    STOSW                       ; 写回当前行（消除后下移）
    ; ... 后续处理省略
    RET
PUT ENDP
```

为了更好地完善用户交互，我设计了分数显示函数（DISPSCORE）

和下一个方块预览函数（DISPNEXT）。分数显示在游戏界面的右侧，下一个方块预览也显示在右侧，以提供更好的游戏体验。

```
; 显示分数函数
DISPSCORE PROC NEAR             ; 显示分数
    MOV AX,@data
    MOV ES,AX
    MOV BP,OFFSET SCORE         ; 分数字符串地址
    MOV CX,05H                  ; 字符串长度
    MOV DX,0635H                ; 屏幕位置（行06H，列35H）
    MOV BH,0H                   ; 页码
    MOV AL,0H                   ; 逐个字符输出
    MOV BL,00110100B            ; 字符属性（颜色）
    MOV AH,13H                  ; 显示字符串功能
    INT 10H                     ; 调用BIOS显示服务
    RET
DISPSCORE ENDP
```

然后，我开始集成这些模块。我设计了游戏的主循环（LOOP1），在主循环中，我通过检测时钟中断产生的计时变量（TIM）来控制方块自动下落的速度，同时通过检测键盘输入来处理用户的操作（左右移动、旋转、加速下落等）。

```
; 游戏主循环
LOOP1:
    STI                         ; 开中断
    MOV AL,TIM
    CMP AL,SPEED                ; 检查是否达到下落时间
    JG TIME
    MOV AH,1
    INT 16H                     ; 检查键盘缓冲区
    JZ LOOP1                    ; 无按键则继续循环
    MOV AH,0
    INT 16H                     ; 读取按键

    ; 按键处理
    CMP AL,1BH                  ; ESC键
    JZ EXIT
    CMP AL,'a'                  ; A键（左移）
    JZ KA
    CMP AL,'w'                  ; W键（旋转）
    JZ KW
    CMP AL,'d'                  ; D键（右移）
    JZ KD
    CMP AL,'s'                  ; S键（加速下落）
    JNZ TIME
    ; ... 按键处理代码省略
```

我还设置了中断处理程序（INT1C），每当时钟中断发生时，将TIM 加 1，从而实现计时功能。

```
; 时钟中断处理程序
INT1C PROC
    PUSH AX
    PUSH DS
    MOV AX, @data
    MOV DS, AX
    INC TIM                    ; 计时变量自增
    POP DS
    POP AX
    IRET                       ; 中断返回
INT1C ENDP
```

3.4 调试与优化阶段

在第 14 周，我进行了初步的调试，发现了一些问题。例如，在方块旋转时，有时旋转后的位置会超出边界或与已有的方块重叠，尤其是长条形方块，W 键旋转为竖向时，方块会直接消失，游戏无法进行，会导致闪退。我通过调整方块数据的初始位置和旋转中心，以及完善碰撞检测函数，解决了这些问题。

我发现游戏在消行时，如果同时消除多行，有时会出现显示错误。经过调试，我发现是因为在消除行后，没有及时更新游戏板的显示。我改进了 PUT 函数，在消除行后，重新绘制游戏板上所有的方块。

我还发现，在游戏进行一段时间后，有时会出现游戏崩溃的情况。通过调试，我发现是因为在移动方块时，没有正确保存和恢复某些寄存器的值，导致后续计算错误。我修改了相关函数，确保在函数调用前后保存和恢复受影响的寄存器。

```
; 改进后的方块旋转函数（保存和恢复寄存器）
ROTATE PROC NEAR
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI
    PUSH DI

    ; 旋转逻辑代码...

    POP DI
    POP SI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
ROTATE ENDP
```

四、项目开发心得

4.1 遇到的问题与解决方法

(1) 初始编译显示 out of memory

最开始时，我的代码没有明显的语法错误，但是编译 asm 文件时，编译器显示 out of memory。经过问题的排查，我发现是一开始的堆栈空间设置过大。我最初设置了.stack 200h，即 512 字节，这通常没问题，但如果 DOSBox/MASM 环境本身可用内存很小，可能会导致溢出，需要减小堆栈空间。于是我改为.stack 100h，这个问题就成功解决了。

(2) 方块不下落和闪退问题

编译成功后，出现了方块不下落的问题。我设置的自动下落依赖于 TIM 变量的自增，而 TIM 是在 INT 1CH 里自增的。而在安装 INT 1CH 时，最初的编码做了如下操作：

```
PUSH DS
MOV AX,SEG INT1C
MOV DS,AX
MOV AL,1CH
MOV AH,25H
INT 21H
POP DS
```

但中断服务程序 INT1C 过程里有 MOV AX,@data 和 MOV DS,AX，这会在中断发生时修改 DS，导致主程序数据段被破坏,无法正确访问并增加计时器变量 TIM。对此，我重写了 INT1C 过程，在其中加入了 PUSH DS 和 POP DS 来保护现场，并显式地将 DS 设置为@data，确保能正确访问 TIM 变量且不影响主程序。

再次运行后，游戏会发生闪退，经过仔细检查，我发现游戏开始时 (BEGIN 过程)，程序将方块数据加载到 P2 变量中。由于原始方块数据是靠右对齐的（低位），而游戏区域的右侧墙壁也是低位（00000111）。这导致 CHECK 函数在第一帧就检测到碰撞，判定游戏结束，从而跳转到 GAMEOVER，最后回到选择速度的界面。而方块不下落是因为游戏一开始就结束了，自然没有机会下落。所以我我在 BEGIN 过程中做了两处修改：在加载方块数据后，立即将 P2 中的数据左移 4 位,使其位于游戏区域的中间,避开右侧墙壁;同时将 XL(左移偏移量) 的初始值设为 4，以匹配移位后的状态，确保后续旋转操作正常。

(3) 方块旋转消失问题

解决其他问题后，游戏终于可以进行。但是我发现，其他方块可以正常旋转，但长条形方块（I 型）按 W 键旋转后，会直接消失。所

以我在 INIT_PAD_DATA 中重新定义了 I 型方块的 4 个旋转状态。之前只定义了水平状态，其他状态为全 0，导致旋转后消失。现在正确定义了水平和垂直状态，使其在旋转时能正确显示。

4.2 开发收获

在项目开发的过程中，我深刻理解了计算机底层硬件的工作原理。之前学习《计算机组成原理》《计算机系统结构》课程时，课本上的知识虽然能够通过做题进行巩固，但是对于这些知识的实际应用，我总是一知半解。在本次项目中，通过直接操作中断向量、端口和内存，我掌握了如何在没有操作系统高级抽象的情况下控制计算机硬件。特别是对中断处理机制的应用，让我明白了实时系统的基本工作原理。

这次项目对于我来说，是一次非常新奇的体验。之前的课程中，我的项目基本上都是用 C++、java 等高级语言完成，这是第一次通过汇编语言构建一整个游戏。汇编语言编程需要极大的耐心和细心，一个字节的错误就可能导致程序崩溃。在连续几周每天数小时的调试过程中，我的耐心和毅力得到了极大的锻炼，这种品质对于解决复杂技术问题至关重要。

当然，项目的推进中我也碰到了很多的问题，需要一点点解决。解决方法有时在课本上是无法找到的，相较于大一大二时专业课课堂高浓度的知识输入，我能感受到本学期的知识是需要自己探索收集的。遇到问题我会利用大模型、CSDN 社区、GitHub，通过互联网来获取问题的解决办法，总有人和我遇到过一样的问题，也总有前辈耐心解决我的提问。与志同道合的人一起解决代码问题，看着自己的项目能

够正常运行，是一件非常有成就感的事情。

五、项目反思与展望

5.1 项目反思

尽管项目基本实现了预定的目标，但仍然存在一些不足之处。

比如，现在一行方块消除之后，相连的方块会变成硬编码的紫色。我试图把这个问题解决掉，让它保持原先的颜色，但是要更改的内容比较多，我修改了很长时间也没有解决这个问题，只能暂时先使用硬编码的紫色来呈现。

现在程序当中的错误处理逻辑相对简单，没有完善的异常处理机制，在某些极端情况下，程序可能会崩溃或者闪退。

后续有时间的时候我会继续改进我的代码，包括进一步减少屏幕刷新的区域，优化一下关键的算法等等。

5.2 未来展望

尽管项目已经完成，我认为但仍有可以扩展的地方：

在图形优化方面，现在的界面略显简陋，后续可以使用更高级的图形技术，如双缓冲、平滑动画等，来提升游戏的视觉效果。

同时可以给游戏添加更丰富的音效和背景音乐，比如消除后可以添加得分音效，按 S 键直接下落可以添加落地音效。使游戏更加生动。

总之，这个项目是一个很好的起点，它不仅巩固了我的汇编语言知识，还提高了我的系统编程能力和解决问题的能力。我相信这些经验将对我未来的学习和工作产生积极的影响。