```
;
;                        _____________________________  __
;                        ___  __/__  __ \_  __ \__  | / /
;                        __  /  __  /_/ /  / / /_   |/ /
;                        _  /   _  _, _// /_/ /_  /|  /
;                        /_/    /_/ |_| \____/ /_/ |_/
;


; ===============================================================================
; 1. Constantes
; ===============================================================================
;       1.1. Interrupcoes
; ===============================================================================

INIT_SP         EQU     FDFFh           ; Valor inicial do SP
INT_MASK_ADDR   EQU     FFFAh           ; Porto da mascara de interrupcoes
; F - Timer, B - P1 Right, 9 - P2 Right, 7 - P2 Left, 1 - Start, 0 - P1 Left
;                       FXXXBX9X7XXXXX10
INT_MASK        EQU     0000000000000010b       ; Interrupcoes (standby)
INT_MASK_INGAME EQU     1000101010000001b       ;       "       (ingame)
INT_MASK_LP1    EQU     1000101010000000b       ; S/ int. I0 (mov. j1 p/ esq.)
INT_MASK_LP2    EQU     1000101000000001b       ; S/ int. I7 (mov. j2 p/ esq.)
INT_MASK_RP1    EQU     1000001010000001b       ; S/ int. IB (mov. j1 p/ dir.)
INT_MASK_RP2    EQU     1000100010000001b       ; S/ int. I9 (mov. j2 p/ dir.)
TIMER_COUNT     EQU     FFF6h                   ; Porto do intervalo de tempo
TIMER_CTRL      EQU     FFF7h                   ; Porto de inicializacao timer
TIME_INTERVAL   EQU     10d                     ; Um segundo (10 ticks)
BASE10          EQU     0Ah

; ===============================================================================
;       1.2. Janela de Texto
; ===============================================================================

INIT_CURSOR     EQU     FFFFh   ; Valor para inicializar o cursor
CURSOR          EQU     FFFCh   ; Porto do cursor
IO_WINDOW       EQU     FFFEh   ; Porto da janela de texto
DIM_WINDOW      EQU     1750h   ; Dimensao da janela de texto (80x24)

; ===============================================================================
;               1.2.1. Texto
; ===============================================================================

STR_END         EQU     '@'     ; Simboliza o fim da escrita
STR_BREAK       EQU     '$'     ; Nova linha
STR_BLANK       EQU     ' '     ; "Vazio"
;POS_WELCOME1   EQU     0B1Fh   ; Coordenadas do pos inicial da msg introdutoria
;POS_WELCOME2   EQU     0C1Bh   ; ---------------------"----------------------
POS_WELCOME1    EQU     0722h   ; Coordenadas do pos inicial da msg introdutoria
POS_TRON        EQU     0818h   ; TRON!!!
POS_WELCOME2    EQU     0E1Bh   ;                           "
POS_GAMEOVER1   EQU     0B22h   ; Coordenadas do pos inicial da msg final
POS_GAMEOVER2   EQU     0C1Ah   ;                        "

; ===============================================================================
;               1.2.2. Campo de jogo
; ===============================================================================

CHAR_CORNER     EQU     '+'     ; Simbolo do canto do campo
POS_CORNER_TOPL EQU     010Fh   ; Coordenadas da pos dos cantos (modo normal)
POS_CORNER_BOTL EQU     160Fh   ;                            "
POS_CORNER_TOPR EQU     0140h   ;                            "
POS_CORNER_BOTR EQU     1640h   ;                            "
```



```
CHAR_VERT       EQU     '|'     ; Simbolo das fronteiras coluna do campo
INC_LINE        EQU     0100h   ; A somar ao cursor para obter a linha seguinte

CHAR_HORI       EQU     '-'     ; Simbolo das fronteiras linha do campo
INC_COL         EQU     0001h   ; A somar ao cursor para obter a coluna seguinte

; ===============================================================================
;               1.2.3. Particulas
; ===============================================================================

CHAR_P1         EQU     'X'     ; Particula do Jogador 1, pos e dir inicial
POS_INI_P1      EQU     0B17h
DIR_INI_P1      EQU     0
CHAR_P2         EQU     '#'     ; Particula do Jogador 2, pos e dir inicial
POS_INI_P2      EQU     0B38h
DIR_INI_P2      EQU     2

; ===============================================================================
;       1.3. Placa
; ===============================================================================

DISPLAY_7SEG    EQU     FFF0h   ; Porto do 7seg menos significativo
NIBBLE          EQU     4       ; Numero de bits num nibble
BYTE            EQU     8       ; Numero de bits num byte
NIBBLE_MASK     EQU     000Fh   ; Para obter o nibble menos significativo

DISPLAY_LEDS    EQU     FFF8h   ; Porto dos LEDs
LVL_1_SPEED     EQU     7d      ; Definicoes para Nivel 1
LVL_1_LEDS      EQU     0000h
LVL_1_TIME      EQU     00h
LVL_2_SPEED     EQU     5d      ; Definicoes para Nivel 2
LVL_2_LEDS      EQU     000Fh
LVL_2_TIME      EQU     10h
LVL_3_SPEED     EQU     3d      ; Definicoes para Nivel 3
LVL_3_LEDS      EQU     00FFh
LVL_3_TIME      EQU     20h
LVL_4_SPEED     EQU     2d      ; Definicoes para Nivel 4
LVL_4_LEDS      EQU     0FFFh
LVL_4_TIME      EQU     40h
LVL_5_SPEED     EQU     1d      ; Definicoes para Nivel 5
LVL_5_LEDS      EQU     FFFFh
LVL_5_TIME      EQU     60h

IO_SWITCHES     EQU     FFF9h   ; Porto de interruptors
BIT_PAUSE       EQU     10000000b

CTRL_LCD        EQU     FFF4h   ; Porto de controlo do LCD
IO_LCD          EQU     FFF5h   ; Porto de escrita do LCD
; Mascara do LCD (F - on/off, 5 - limpa, 4 - linha, 3210 - coluna)
;                       FXXXXXXXXX543210
INIT_LCD        EQU     1000000000100000b
BIT_WIPE_LCD    EQU     0000000000100000b

; ===============================================================================
; 2. Variaveis
; ===============================================================================



                ORIG    8000h
; ===============================================================================
;       2.1. Mensagens (strings)
```

```
; ================================================================================

Welcome1        STR     'Bem-Vindo ao', STR_END ; Mensagem introdutoria
TRON            STR     '_____ _____ _____ _____ _____ _____ _____ ', STR_BREAK
TRON1           STR     '|___ |___ /__ ___ \_ __ \__ | / / /', STR_BREAK
TRON2           STR     '|__ /  __  /_/ /  / / /_  |/ / ', STR_BREAK
TRON3           STR     '|_ /  _ _, _// /_/ /_ /|  /  ', STR_BREAK
TRON4           STR     '/_/   /_/ |_| \____/ /_/ |_/   ', STR_END
Welcome2        STR     'Pressione I1 para comecar', STR_END
LcdContent1     STR     'TEMPO MAX: ', STR_END  ; Informacao no LCD
LcdContent2     STR     'sJ1: ', STR_END
LcdContent3     STR     '    J2: ', STR_END
GameOver1       STR     'Fim do jogo', STR_END  ; Mensagem final
GameOver2       STR     'Pressione I1 para recomecar', STR_END

; ================================================================================
;       2.2. Variaveis de controlo
; ================================================================================

GameState       WORD    0               ; 1 => Jogo a decorrer; !1 => Parado
Time            WORD    0               ; Time = NextLevel? Next level
Counter         WORD    0               ; Counter = 10d? Time <- Time + 1
MaxTime         WORD    0               ; Time > MaxTime? MaxTime = Time
CurrentLevel    WORD    LVL_1_SPEED
Speed           WORD    0               ; Speed = Level? Move

; ================================================================================
;       2.3. Jogadores
; ================================================================================

P1Score         WORD    0       ; Pontuacao+posicao+direcao dos jogadores
P2Score         WORD    0
P1Pos           WORD    0
P2Pos           WORD    0
DirP1           WORD    0       ; 0 - direita, 1 - baixo, 2 - esquerda, 3 - cima
DirP2           WORD    2
Right           WORD    0001h   ; Valores a somar para mover na direcao indicada
Down            WORD    0100h
Left            WORD    FFFFh
Up              WORD    FF00h

WindowMatrix    TAB     1749h   ; Reservar memoria para a janela

; ================================================================================
; 3. Interrupcoes
; ================================================================================
;       3.1. Vector de interrupcoes
; ================================================================================

                ORIG    FE00h
INT0            WORD    LeftP1  ; Altera a direcao do jogador 1 para a esquerda
INT1            WORD    Reset   ; Determina o estado do jogo
                ORIG    FE07h
INT7            WORD    LeftP2  ; Altera a direcao do jogador 2 para a esquerda

                ORIG    FE09h
INT9            WORD    RightP2 ; Altera a direcao do jogador 2 para a direita
                ORIG    FE0Bh
INT11           WORD    RightP1 ; Altera a direcao do jogador 1 para a direita
                ORIG    FE0Fh
INT15           WORD    IntTimer ; Lida com o temporizador
```



```
; ================================================================================
; 4. Programa
; ================================================================================

                ORIG    0000h
                JMP     Main
; ================================================================================
;       4.1. Rotinas de interrupcao
; ================================================================================

; Reset: Permite determinar se o jogo esta ou nao a decorrer. 0 - Nao; !0 - Esta
;  Entradas: ---
;  Saidas: ---
;  Efeitos: Altera o valor de M[GameState]
Reset:          COM     M[GameState]
                RTI

; (Right/Left)P(1/2): Determinam para onde se deslocarao os jogadores
;  Entradas: ---
;  Saidas: ---
;  Efeitos: Altera R5/R6
LeftP1:         DEC     M[DirP1]
                MOV     R7, INT_MASK_LP1        ; Mascara sem interrupcao I0
                MOV     M[INT_MASK_ADDR], R7
                RTI

RightP1:        INC     M[DirP1]
                MOV     R7, INT_MASK_RP1        ; Mascara sem interrupcao IB
                MOV     M[INT_MASK_ADDR], R7
                RTI

LeftP2:         DEC     M[DirP2]
                MOV     R7, INT_MASK_LP2        ; Mascara sem interrupcao I7
                MOV     M[INT_MASK_ADDR], R7
                RTI

RightP2:        INC     M[DirP2]
                MOV     R7, INT_MASK_RP2        ; Mascara sem interrupcao I9
                MOV     M[INT_MASK_ADDR], R7
                RTI

; IntTimer: Incrementa os contadores referentes ao nivel/velocidade e tempo
;       Entradas: ---
;       Saidas: ---
;       Efeitos: Altera o valor de M[Speed], M[Counter] e os portos do timer
IntTimer:       INC     M[Speed]        ; Incrementa o contador da velocidade
                INC     M[Counter]      ; Incrementa o contador do tempo
                MOV     R7, 1
                MOV     M[TIMER_COUNT], R7
                MOV     M[TIMER_CTRL], R7
                RTI

; ================================================================================

;       4.2. Rotinas auxiliares
; ================================================================================
;               4.2.1. Escrita na janela de texto + memeria
; ================================================================================
;                       4.2.1.1. Auxiliares de escrita
; ================================================================================
```

```
; WriteCharW: Escreve um caracter na janela de texto e em memeria
;       Entradas: Caracter a escrever; Posicao do cursor.
;       Saidas: ---
;       Efeitos: Altera M[CURSOR], M[IO_WINDOW] e M[R2+WindowMatrix]
WriteCharW:     PUSH    R1
                PUSH    R2
                MOV     R1, M[SP+5]             ; Caracter a escrever
                MOV     R2, M[SP+4]             ; Posicao do cursor
                MOV     M[CURSOR], R2           ; Apontar para a posicao
                MOV     M[R2+WindowMatrix], R1  ; Escreve em memoria
                MOV     M[IO_WINDOW], R1        ; Escrever na posicao
                POP     R2
                POP     R1
                RETN    2

; WriteStrW: Escreve uma cadeia na janela de texto e em memoria
;       Entradas: Posicao (memoria) do primeiro caracter; Posicao do cursor.
;       Saidas: ---
;       Efeitos: ---
WriteStrW:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R2, M[SP+6]     ; Endereco do 1o. caracter a escrever
                MOV     R3, M[SP+5]     ; Posicao do cursor
NextChar:       MOV     R1, M[R2]       ; Caracter a escrever
                CMP     R1, STR_BREAK   ; Break? Nova linha
                BR.NZ   IsEndStr        ; Nao? Sera que acabou entao?
                MOV     R3, M[SP+5]     ; Posicao inicial da linha em questao
                ADD     R3, INC_LINE    ; Proxima linha
                MOV     M[SP+5], R3     ; Armazena a pos inicial da nova linha
                INC     R2              ; Proximo caracter (salta STR_END)
                MOV     R1, M[R2]
IsEndStr:       CMP     R1, STR_END     ; String terminou? Termina a escrita
                BR.Z    EndWriteStrW    ; Nao? Escreve o caracter
                PUSH    R1
                PUSH    R3
                CALL    WriteCharW
                INC     R2              ; Endereco do proximo caracter
                INC     R3              ; Proxima coluna
                BR      NextChar
EndWriteStrW:   POP     R3
                POP     R2
                POP     R1
                RETN    2

; ========================================================================
;                       4.2.1.2. Escrita do campo de jogo
; ========================================================================

; DrawField: Rotina que chama sub-rotinas para desenhar o campo de jogo
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
DrawField:      CALL    ClearScreen
                CALL    DrawFrontiers
                CALL    DrawCorners
                RET

; ClearScreen: Escreve espacos em toda a janela de texto e em memoria
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
```



```
ClearScreen:    PUSH    R1
                MOV     R1, DIM_WINDOW  ; Coordenadas da posicao final
NextPos:        PUSH    STR_BLANK       ; Espacos (' ')
                PUSH    R1              ; Posicao do cursor
                CALL    WriteCharW
                DEC     R1
                CMP     R1, R0
                BR.NN   NextPos
                POP     R1
                RET

; DrawField: Rotina que desenha as fronteiras do campo
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
DrawFrontiers:  PUSH    CHAR_VERT
                PUSH    POS_CORNER_TOPL
                PUSH    POS_CORNER_BOTL
                PUSH    INC_LINE
                CALL    DrawWall        ; Parede esquerda

                PUSH    CHAR_HORI
                PUSH    POS_CORNER_TOPL
                PUSH    POS_CORNER_TOPR
                PUSH    INC_COL
                CALL    DrawWall        ; Tecto

                PUSH    CHAR_VERT
                PUSH    POS_CORNER_TOPR
                PUSH    POS_CORNER_BOTR
                PUSH    INC_LINE
                CALL    DrawWall        ; Parede direita

                PUSH    CHAR_HORI
                PUSH    POS_CORNER_BOTL
                PUSH    POS_CORNER_BOTR
                PUSH    INC_COL
                CALL    DrawWall        ; Chao
                RET

; DrawCorners: Rotina que desenha os cantos do campo de jogo
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
DrawCorners:    PUSH    R1
                MOV     R1, CHAR_CORNER
                PUSH    R1              ; Canto Superior Esquerdo
                PUSH    POS_CORNER_TOPL
                CALL    WriteCharW

                PUSH    R1              ; Canto Superior Direito
                PUSH    POS_CORNER_TOPR
                CALL    WriteCharW

                PUSH    R1              ; Canto Inferior Esquerdo
                PUSH    POS_CORNER_BOTL
                CALL    WriteCharW

                PUSH    R1              ; Canto Inferior Direito
                PUSH    POS_CORNER_BOTR
                CALL    WriteCharW
                POP             R1
```

```
                RET

; DrawWall: Desenha "paredes" (horizontais ou verticais) na janela de texto
;       Entradas: Caracter; Pos inicial; Pos final; Incremento (1h ou 100h).
;       Saidas: ---
;       Efeitos: ---
DrawWall:       PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1, M[SP+9]     ; Caracter a escrever
                MOV     R2, M[SP+8]     ; Posicao inicial
                MOV     R3, M[SP+7]     ; Posicao final
                MOV     R4, M[SP+6]     ; Incremento
Piece:          CMP     R2, R3          ; Atingiu a posicao final?
                BR.P    EndWall         ; Sim? Terminar a escrita
                PUSH    R1              ; Nao? Escrever o caracter
                PUSH    R2
                CALL    WriteCharW
                ADD     R2, R4          ; Proxima posicao
                BR      Piece
EndWall:        POP     R4
                POP     R3
                POP     R2
                POP     R1
                RETN    4

; ==============================================================================
;                       4.2.1.3. Escrita das particulas + Colisoes
; ==============================================================================

; DrawParticles: Rotina que desenha as duas particulas
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
DrawParticles:  PUSH    CHAR_P1         ; Particula 1
                PUSH    M[P1Pos]        ; Posicao da particula
                CALL    WriteCharW
                PUSH    CHAR_P2         ; Particula 2
                PUSH    M[P2Pos]        ; Posicao da particula
                CALL    WriteCharW
                RET

ResetParticles: MOV     R7, POS_INI_P1  ; Posicao inicial do jogador 1
                MOV     M[P1Pos], R7
                MOV     R7, DIR_INI_P1  ; Direcao inicial do jogador 1
                MOV     M[DirP1], R7
                MOV     R7, POS_INI_P2  ; Repetir para jogador 2
                MOV     M[P2Pos], R7
                MOV     R7, DIR_INI_P2
                MOV     M[DirP2], R7
                RET
; MoveParticles: Rotina que movimenta as particulas para a proxima posicao
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
MoveParticles:  PUSH    R1
                ; DirActual % 4 + PrimDir = DirNova
                MOV     R1, M[DirP1]    ; Direcao da particula
                AND     R1, 3           ; % 4 (pode ser > 3 ou < 0, nao importa)
                ADD     R1, Right       ; Adiciona o endereco da 1a. direcao
                MOV     R1, M[R1]       ; Actualiza a posicao do jogador 1
```



```
                ADD     M[P1Pos], R1

                MOV     R1, M[DirP2]    ; Repetir para jogador 2
                AND     R1, 3
                ADD     R1, Right
                MOV     R1, M[R1]
                ADD     M[P2Pos], R1
                POP     R1
                RET

; Collision: Verifica se houve colisao e determina se o jogo terminou
;       Entradas: ---
;       Saidas: ---
;       Efeitos: ---
Collision:      PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1, M[P1Pos]            ; Posicao do jogador 1
                MOV     R1, M[R1+WindowMatrix]  ; Pos em memoria do jogador 1
                MOV     R2, M[P2Pos]            ; Repete para jogador 2
                MOV     R2, M[R2+WindowMatrix]
                MOV     R3, R0                  ; Flag de colisao do jogador 1
                MOV     R4, R0                  ; Flag de colisao do jogador 2

Player1:        CMP     R1, STR_BLANK   ; Pos seguinte do jogador 1 ocupada?
                BR.Z    Player2         ; Nao? Verifica o jogador 2
                MOV     R3, 1           ; Sim? Jogador 1 colidiu
Player2:        CMP     R2, STR_BLANK   ; Pos seguinte do jogador 2 ocupada?
                BR.Z    Scoring         ; Nao? Ninguem colidiu
                MOV     R4, 1           ; Sim? Jogador 2 colidiu
Scoring:        CMP     R3, R4          ; Compara as flags de colisao
                BR.Z    CheckZero       ; Iguais? Verifica se uma e zero
                BR.N    P1Win           ; R4 = 1? Jogador 1 ganhou
                BR.P    P2Win           ; R3 = 1? Jogador 2 ganhou
CheckZero:      CMP     R3, R0          ; Jogador 1 colidiu?
                BR.NZ   Collided        ; Sim? Termina o jogo em empate
                BR.Z    EndCollision    ; Nao? Resume o jogo
P1Win:          INC     M[P1Score]      ; Venceu o jogador 1
                BR      Collided
P2Win:          INC     M[P2Score]      ; Venceu o jogador 2
Collided:       MOV     M[GameState], R0; Termina o jogo se houve uma colisao
EndCollision:   POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

; ==============================================================================
;       4.3. Escrita na placa
; ==============================================================================
;               4.3.1. 7 Segmentos
; ==============================================================================

; Write7Seg: Rotina que escreve no display de 7 segmentos
;       Entradas: ---
;       Saidas: ---
;       Efeitos: Altera M[Time]
Write7Seg:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R2, DISPLAY_7SEG
```

```
                MOV     R3, NIBBLE
NextSeg:        MOV     R1, M[Time]
                AND     R1, NIBBLE_MASK ; Digito > 9?
                CMP     R1, BASE10
                BR.NZ   ContSeg         ; Se nao, escreve e segue
                PUSH    R1              ; Se sim, "converte" para base 10
                CALL    ComplNib
                POP     R1              ; Recupera o valor convertido da pilha
                ADD     M[Time], R1     ; Adiciona o tempo ao compl. no registo
                MOV     R1, M[Time]
ContSeg:        MOV     M[R2], R1       ; Escreve no segmento em causa
                ROR     M[Time], NIBBLE ; Eleger o proximo nibble
                INC     R2              ; Proximo segmento
                DEC     R3              ; Menos um segmento por esrever
                CMP     R3, R0          ; R3 = 0? Acabou de escrever
                BR.NZ   NextSeg         ; Se nao, escreve no proximo
                POP     R3
                POP     R2
                POP     R1
                RET


; ComplNib: Rotina que complementa o nibble menos significativo
;       Entradas: Tempo em hexadecimal
;       Saidas: Tempo em base b
;       Efeitos: ---
ComplNib:       PUSH    R1
                MOV     R1, M[SP+3]     ; Tempo em hexadecimal
                COM     R1              ; Complemento para 2
                INC     R1
                AND     R1, NIBBLE_MASK ; Preserva apenas o menos significativo
                MOV     M[SP+3], R1     ; Tempo convertido
                POP     R1
                RET

; ===========================================================================
;               4.3.2. LEDs
; ===========================================================================

; VeriLevel: Rotina que verifica se transita de nivel e escreve nos LEDS
;       Entradas: ---
;       Saidas: ---
;       Efeitos: M[CurrentLevel] e escreve na placa (M[DISPLAY_LEDS])
VeriLevel:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, M[Time]             ; Tempo em segundos
                MOV     R2, M[CurrentLevel]     ; Nivel actual (velocidade)
Level5:         CMP     R1, LVL_5_TIME          ; 60 segundos?
                JMP.P   SkipLvl                 ; Se >60s, nao actualiza os LEDs
                BR.N    Level4                  ; Se for inferior, nivel 4
                MOV     R2, LVL_5_SPEED         ; Velocidade do nivel 5
                MOV     R3, LVL_5_LEDS          ; LEDs acesos no nivel 5
                BR      EndLvl                  ; Repete para os outros niveis
Level4:         CMP     R1, LVL_4_TIME          ; Nivel 4 = 40s
                BR.N    Level3
                MOV     R2, LVL_4_SPEED
                MOV     R3, LVL_4_LEDS
                BR      EndLvl
Level3:         CMP     R1, LVL_3_TIME          ; Nivel 3 = 20s
                BR.N    Level2
                MOV     R2, LVL_3_SPEED
```



```
                MOV     R3, LVL_3_LEDS
                BR      EndLvl
Level2:         CMP     R1, LVL_2_TIME          ; Nivel 2 = 10s
                BR.N    Level1
                MOV     R2, LVL_2_SPEED
                MOV     R3, LVL_2_LEDS
                BR      EndLvl
Level1:         MOV     R2, LVL_1_SPEED         ; Nivel = 0s (default)
                MOV     R3, LVL_1_LEDS
EndLvl:         MOV     M[CurrentLevel], R2     ; Actualiza o nivel
                MOV     M[DISPLAY_LEDS], R3     ; e os LEDs
SkipLvl:        POP     R3
                POP     R2
                POP     R1
                RET
; ===========================================================================
;               4.3.3. LCD
; ===========================================================================

; WriteLCD: Escrita dos valor do tempo maximo e pontuacoes no LCD
;       Entradas: ---
;       Saidas: ---
;       Efeitos: Altera M[CTRL_LCD]
; NOTA: A posicao do cursor do LCD e passada atraves da pilha. Quanto as outras
; rotinas, presume-se que o cursor esta na pilha, apenas e removido no fim
WriteLCD:       PUSH    R1
                MOV     R1, INIT_LCD    ; Liga e limpa o LCD
                MOV     M[CTRL_LCD], R1
                SUB     R1, BIT_WIPE_LCD; Remover o bit que limpa o LCD
                PUSH    R1
                PUSH    LcdContent1
                CALL    WriteStrLCD     ; Escreve tempo maximo
                CALL    MaxTimeLCD
                PUSH    LcdContent2
                CALL    WriteStrLCD     ; Escreve a pontuacao do jogador 1

                CALL    ConvScores      ; Converte as pontuacoes
                PUSH    M[P1Score]
                PUSH    2
                CALL    WriteValueLCD
                PUSH    LcdContent3
                CALL    WriteStrLCD     ; Escreve pontuacao do jogador 2
                PUSH    M[P2Score]
                PUSH    2
                CALL    WriteValueLCD
                POP     R0              ; Remove o cursor da pilha
                POP     R1
                RET

; ConvScores: Rotina que converte as pontuacoes para decimal
;       Entradas: ---
;       Saidas: ---
;       Efeitos: Altera M[P1Score] e M[P2Score]
ConvScores:     PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, P1Score     ; Endereco da pontuacao do jogador 1
StartConv:      MOV     R2, 2
ConvLoop:       MOV     R3, M[R1]       ; Pontuacao no endereco R1 (j1 -> j2)
                AND     R3, NIBBLE_MASK ; Codigo semelhante a rotina Write7Seg
                CMP     R3, BASE10
                BR.NZ   DontConv
```

```
                PUSH    R3
                CALL    ComplNib
                POP     R3
                ADD     M[R1], R3
DontConv:       ROR     M[R1], NIBBLE
                DEC     R2
                CMP     R2, R0
                BR.NZ   ConvLoop
                SHR     M[R1], BYTE     ; Preserva apenas o byte com a pontuacao
                CMP     R1, P2Score     ; Segundo jogador?
                BR.Z    Converted       ; Sim? Converteu as pontuacaes
                MOV     R1, P2Score     ; Nao? Passa a pontuacao do jogador 2
                BR      StartConv       ; Recomeca
Converted:      POP     R3
                POP     R2
                POP     R2
                RET

; WriteStrLCD: Escrita de uma string no LCD
;       Entradas: Endereco do primeiro caracter; Cursor do LCD
;       Saidas: Cursor do LCD
;       Efeitos: ---
; Ver WriteStrW (codigo semelhante)
WriteStrLCD:    PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, M[SP+5]
                MOV     R2, M[SP+6]
Char:           MOV     R3, M[R1]
                CMP     R3, STR_END
                BR.Z    EndLCD
                PUSH    R2
                PUSH    R3
                CALL    WriteCharLCD
                POP     R2
                INC     R1
                INC     R2
                BR      Char
EndLCD:         MOV     M[SP+6], R2
                POP     R3
                POP     R2
                POP     R1
                RETN    1

; WriteCharLCD: Escrita de um valor no LCD (formato xxxxh, hexadecimal)
;       Entradas: Cursor, caracter
;       Saidas: Cursor (permanece na pilha)
;       Efeitos: ---
; Ver WriteCharW (codigo semelhante)
WriteCharLCD:   PUSH    R1
                PUSH    R2
                MOV     R1, M[SP+4]
                MOV     R2, M[SP+5]
                MOV     M[CTRL_LCD], R2
                MOV     M[IO_LCD], R1
                POP     R2
                POP     R1
                RETN    1

; WriteValueLCD: Escrita de um valor no LCD (formato xxxxh, hexadecimal)
;       Entradas: Valor, Cursor, numero de digitos (max 4)
;       Saidas: Cursor (permanece na pilha)
```



```
;       Efeitos: ---
WriteValueLCD:  PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1, M[SP+7]     ; Valor a imprimir
                MOV     R2, M[SP+8]     ; Posicao do cursor
                MOV     R3, M[SP+6]     ; Numero de digitos
                MOV     R4, 4           ; Preparar o numero para escrita
                SUB     R4, R3          ; Determina o numero de rotacoes a fazer
Rotate:         CMP     R4, R0          ; para por o nibble -signif no +signif
                BR.Z    NextDig
                ROL     R1, NIBBLE
                DEC     R4
                BR      Rotate
NextDig:        ROL     R1, NIBBLE      ; Rotacao do digito de + para -signif
                PUSH    R1              ; Preserva R1
                AND     R1, NIBBLE_MASK ; Digito a escrever
                ADD     R1, '0'         ; Hex -> ASCII
                PUSH    R2
                PUSH    R1
                CALL    WriteCharLCD    ; Escreve o caracter
                POP     R2
                POP     R1              ; Recupera o valor inalterado da pilha
                INC     R2
                DEC     R3
                CMP     R3, R0
                BR.NZ   NextDig
                MOV     M[SP+8], R2     ; Mantem o cursor na pilha
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RETN    2

; MaxTimeLCD: Escrita apenas do Tempo Maximo ao longo do jogo
;       Entradas: Cursor
;       Saidas: ---
;       Efeitos: Altera M[CTRL_LCD]
MaxTimeLCD:     PUSH    M[SP+2] ; Cursor pre escrita
                PUSH    M[MaxTime]
                PUSH    NIBBLE
                CALL    WriteValueLCD
                POP     M[SP+3] ; Cursor pos escrita (substitui o outro)
                RET

; ===============================================================================

;       4.4. Jogo
; ===============================================================================
;               4.4.1 Auxiliares
; ===============================================================================

SetupBoard:     CALL    Write7Seg
                CALL    VeriLevel
                CALL    WriteLCD
                RET

StartMessage:   PUSH    Welcome1        ; 1a. linha da mensagem introdutoria
                PUSH    POS_WELCOME1
                CALL    WriteStrW
                PUSH    TRON            ; TRON!!!
```

```
                PUSH    POS_TRON
                CALL    WriteStrW
                PUSH    Welcome2        ; 2a. linha da mensagem introdutoria
                PUSH    POS_WELCOME2
                CALL    WriteStrW
                RET

EndMessage:     PUSH    GameOver1       ; 1a. linha da mensagem final
                PUSH    POS_GAMEOVER1
                CALL    WriteStrW
                PUSH    GameOver2       ; 2a. linha da mensagem final
                PUSH    POS_GAMEOVER2
                CALL    WriteStrW
                RET

ResetCtrlVars:  MOV     M[GameState], R0        ; Controlo de jogo
                MOV     M[P1Score], R0          ; Pontuacao do jogador 1
                MOV     M[P2Score], R0          ; Pontuacao do jogador 2
                MOV     M[Speed], R0            ; Velocidade/Nivel
                MOV     M[Time], R0             ; Tempo passado (s)
                MOV     M[MaxTime], R0          ; Tempo maximo
                RET

; ===============================================================================
;               4.4.2 Principal
; ===============================================================================

Main:           MOV     R7, INIT_SP             ; Inicializar pilha
                MOV     SP, R7
                CALL    ResetCtrlVars           ; Reset de todas as vars de ctrl
                MOV     R7, INT_MASK            ; Mascara apenas com I1
                MOV     M[INT_MASK_ADDR], R7
                CALL    SetupBoard              ; Setup da placa (LEDs/7seg/LCD)
                MOV     R7, INIT_CURSOR         ; Inicializar cursor
                MOV     M[CURSOR], R7
                CALL    ClearScreen             ; Apagar ecra
                CALL    StartMessage            ; Escreve a mensagem inicial
                ENI                             ; Activar as interrupcoes
WaitStart:      CMP     M[GameState], R0        ; Game on?
                BR.Z    WaitStart               ; Nao? Reverifica
Setup:          DSI
                MOV     M[Time], R0
                CALL    SetupBoard

                CALL    ClearScreen             ; Limpa o ecra
                CALL    DrawField               ; Desenhar campo+jogadores

                MOV     R7, INT_MASK_INGAME     ; Mascara com as outras ints.
                MOV     M[INT_MASK_ADDR], R7
                ENI
                CALL    ResetParticles          ; Direcoes iniciais dos jogadore
s
                CALL    DrawParticles           ; Desenha os jogadores
                MOV     R7, 1                   ; Inicializa o temporizador
                MOV     M[TIMER_COUNT], R7
                MOV     M[TIMER_CTRL], R7

GameLoop:       BR.NI   GameLoop
                DSI

Paused:         MOV     R7, M[IO_SWITCHES]      ; Interruptor 7? Pausa
                TEST    R7, BIT_PAUSE
```



```
                BR.NZ   Paused

Disp7Seg:       MOV     R7, M[Counter]
                CMP     R7, TIME_INTERVAL       ; Counter = 10d? (1 segundo?)
                BR.N    TimeLCD
                MOV     M[Counter], R0          ; Reset do counter, escreve o
                INC     M[Time]                 ; tempo na placa (7 segmentos)
                CALL    Write7Seg
                CALL    VeriLevel               ; Verif. niveis/escreve nos LEDs
TimeLCD:        MOV     R7, M[Time]
                CMP     R7, M[MaxTime]          ; TMax < Tempo? TMax = Tempo
                BR.N    Particles
                MOV     M[MaxTime], R7
                PUSH    800Bh                   ; Posicao do tempo maximo no LCD
                CALL    MaxTimeLCD              ; Escreve apenas o tempo maximo
                POP     R0                      ; Tira o cursor do LCD da pilha
Particles:      MOV     R7, M[Speed]
                CMP     R7, M[CurrentLevel]     ; A cada N 0.1s, move particulas
                BR.N    IsOver
                MOV     M[Speed], R0            ; Anula o contador do nivel
                CALL    MoveParticles           ; Move as particulas
                CALL    Collision               ; Verifica se houve colisao
                CALL    DrawParticles           ; Escreve as particulas
                MOV     R7, INT_MASK_INGAME     ; Repoe a mascara de ints
                MOV     M[INT_MASK_ADDR], R7

IsOver:         CMP     M[GameState], R0        ; Se o jogo nao acabou, repete
                ENI
                JMP.NZ  GameLoop
GameOver:       DSI
                CALL    WriteLCD                ; Reescreve o LCD inteiro
                CALL    EndMessage              ; Escreve a mensagem final
                MOV     R7, INT_MASK            ; Repoe a mascara so com I1
                MOV     M[INT_MASK_ADDR], R7    ; Previne o accionamento da int1
                ENI                             ; pendente (premida mid-game)
                MOV     M[GameState], R0        ; Garante que o jogo termina
                JMP     WaitStart               ; Espera que recomece
```