```
;+--------------------------------------------------+
;|                      TRON                        |
;|                                                  |
;|                   Constantes                     |
;+--------------------------------------------------+


;Pilha
SP_START                EQU     FDFFh

;Janela de Texto e cursor
JT_OUT                  EQU     FFFEh
JT_CUR_START            EQU     FFFFh
JT_CURSOR               EQU     FFFCh

;Controlo de fim de string
END_STR                 EQU     '@'

; Display 7 seguementos
DISPLAY_7SEG            EQU     FFF0h

;Interruptores
INT_CONTROL             EQU     FFF9h

;Temporizador
TimerValue              EQU     FFF6h ; endereÃ§o do Temporizador
TimerControl            EQU     FFF7h ; endereÃ§o de controlo do temporizador

;LEDS
LEDS_CONTROL            EQU     FFF8h ; endereÃ§o dos leds

;LCD
LCD_CONTROL             EQU     FFF4h
LCD_ESCRITA             EQU     FFF5h
PONTUA_MASK             EQU     0100h

;Tabela de colisÃµes
COLS                    EQU     80

;Texto_inicial
FRASE_UM                EQU     0B1Ah
FRASE_DOIS              EQU     0C1Ah

;Janela de Texto
WALL_START              EQU     0110h
ONE_LINE                EQU     0100h
ONE_COL                 EQU     0001h
WALL_END                EQU     1610h

;Posicoes iniciais

I_LOCAL1                EQU     0A09h
I_LOCAL2                EQU     0A29h


;-----------------Mascaras-----------------

COLS_MASK               EQU     00FFh                   ;mascara para o
numero de colunas
LINE_MASK               EQU     FF00h                   ;mascara para o
numero de linhas
```



```
;Mascaras de interrupcoes
INT_MASK_ADDR           EQU     FFFAh                   ;morada da masca
ra de interrupcoes

INT_MASK_INI            EQU     0000000000000010b       ;mascara interrupcoes pa
ra o I1
INT_MASK_PAUSA          EQU     0000101010000001b       ;mascara interrupcoes pa
ra pausa
INT_MASK_JOGO           EQU     1000101010000001b       ;mascara interrupcoes pa
ra jogo

;Mascaras de interruptores
PAUSA_MASK              EQU     0000000010000000b       ;mascara para o interrup
tor 7
HARDCORE_MASK           EQU     0000000000000001b       ;mascara para o interrup
tor 0

;+--------------------------------------------------+
;|                     Variaveis                    |
;+--------------------------------------------------+

                ORIG    8001h
DIR_MASK1               WORD    1100h
DIR_MASK2               WORD    0001h
DIR_MASK3               WORD    0100h
DIR_MASK4               WORD    1001h

JOG1_COL                WORD    0
JOG2_COL                WORD    0

;Texto inicial
ComecaStr               STR     '     Bem-vindo ao TRON @'
ComecaStr2              STR     'Pressione I1 para comecar @'

;Texto final
moldura                 STR     '                          @'
FimStr                  STR     '         Fim do Jogo      @'
FimStr2                 STR     ' Pressione I1 para recomecar @'

;Janela de jogo
TB_Wall                 STR     '+----------------------------------------------
-+ @'
RL_Wall                 STR     '|
 | @'

;Particulas
JOG1_PART               STR     'X@'
JOG2_PART               STR     '#@'

;Posicoes das particulas
LOCAL1                  WORD    0A09h
LOCAL2                  WORD    0A29h



;Variavel de controlo de jogo
GAME_ON                 WORD    0
PAUSA                   WORD    0
HARDCORE                WORD    0

;Temporizador
SPEED                   WORD    0
```

```
TIME                    WORD    0
SECONDS                 WORD    0
TMP_MAX                 WORD    0
OLD_TIME                WORD    0

;Niveis
SPEED_ATUAL             WORD    7

;LEDS

LED_MASK                WORD    0h

;Pontuacoes

POINT1                  WORD    0000h
POINT2                  WORD    0000h

;Strings para LCD
TEXTO_LCD               STR     'TEMPO MAX: ',END_STR
TEXTO2_LCD              STR     'sJ1: ',END_STR
TEXTO3_LCD              STR     '    J2: ',END_STR

;Direcoes das particulas

DIR1_ATUAL              WORD    1
DIR2_ATUAL              WORD    3

DIR1_PROX               WORD    1
DIR2_PROX               WORD    3

;Matriz para verificacao de colisoes
Colisoes                TAB             1920d


;+--------------------------------------------------+
;|                Tabela de interrupcoes            |
;+--------------------------------------------------+

                ORIG    FE00h
INT_0                   WORD    ViraEsquerda1
INT_1                   WORD    intRoutI1

                ORIG    FE0Bh
INT_B           WORD    ViraDireita1

                ORIG    FE07h
INT_7                   WORD    ViraEsquerda2

                ORIG    FE09h
INT_9                   WORD    ViraDireita2



                ORIG    FE0Fh
INT_TIMER       WORD    intRoutTimer

;+--------------------------------------------------+
;|           Fim de tabela de interrupcoes          |
;+--------------------------------------------------+

                ORIG    0000h
                JMP     Inicializar
```



```
;+--------------------------------------------------+
;|             Rotinas de interrupcao               |
;+--------------------------------------------------+


; intRoutI1: Rotina que lida com as interrupcao I0
; Input:  ---
; Efeito: altera M[GAME_ON].
; Output: ---
intRoutI1:              INC     M[GAME_ON]      ;altera o estado de jogo
                RTI

; intRoutTimer: Rotina que lida com as interrupcao do temporizador
; Input:  ---
; Efeito: altera M[SPEED] e M[TIME].
; Output: ---

intRoutTimer:   INC             M[SPEED]
                INC             M[TIME]
                MOV             R7,     1
                MOV             M[TimerValue], R7
                MOV             M[TimerControl], R7             ;reinicia o temp
orizador
                RTI

; ViraEsquerda1: Rotina que lida com as interrupcao I0
; Input:  ---
; Efeito: altera M[DIR1_PROX].
; Output: ---
ViraEsquerda1:  DEC     M[DIR1_PROX]    ;incrementa a direcao da particula 1
                RTI

; ViraEsquerda2: Rotina que lida com as interrupcao I7
; Input:  ---
; Efeito: altera M[DIR2_PROX].
; Output: ---
ViraEsquerda2:  DEC     M[DIR2_PROX]    ;incrementa a direcao da particula 2

                RTI

; ViraDireita1: Rotina que lida com as interrupcao IB
; Input:  ---
; Efeito: altera M[DIR1_PROX].
; Output: ---
ViraDireita1:   INC     M[DIR1_PROX]    ;decrementa a direcao da particula 1
                RTI

; ViraDireita2: Rotina que lida com as interrupcao I9
; Input:  ---
; Efeito: altera M[DIR2_PROX].
; Output: ---
ViraDireita2:   INC     M[DIR2_PROX]    ;decrementa a direcao da particula 2
                RTI

;+--------------------------------------------------+
;|               Inicializacao do jogo              |
;+--------------------------------------------------+

; Inicializar: Rotina que trata da inicializacoes
; Input:  ---
; Efeito: ---
```

```
; Output: ---

Inicializar:    MOV     R7, SP_START            ;inicializa a pilha
                MOV     SP, R7
                MOV     R7, JT_CUR_START        ;inicializa o cursor
                MOV     M[JT_CURSOR], R7
                MOV     R7,INT_MASK_INI         ;seleciona apenas a interrupcao
I1
                MOV     M[INT_MASK_ADDR],R7
                MOV     R7,8020h                ;limpa LCD
                MOV     M[LCD_CONTROL], R7
                CALL    Texto_inicial           ;escreve o texto de boas-vindas
                ENI                             ;ativa a interrupcao
                JMP     Waitgamestart

; Texto_inicial: Rotina que escreve a mensagem de boas vindas na janela de texto
; Input:  ---
; Efeito: ---
; Output: ---
Texto_inicial:  PUSH    R1                      ;salvaguarda valores de registos
                PUSH    R2
                MOV     R2, FRASE_UM            ;envia para o R2 a posicao incia
l do cursor onde vai ser escrita a primeira frase
                MOV     R1, ComecaStr           ;envia para o R1, a posicao de m
emoria onde comeca a primeira frase
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                MOV     R1, ComecaStr2          ;envia para o R1, a posicao de m
emoria onde comeca a segunda frase
                MOV     R2, FRASE_DOIS          ;envia para o R2 a posicao incia
l do cursor onde vai ser escrita a segnda frase
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                POP             R2
                POP     R1
                RET




; Waitgamestart: Loop que aguarda o premir de I1
; Input:  ---
; Efeito: ---
; Output: ---

Waitgamestart:  CALL    VeriHardcore            ;verifica se o modo harcore sera
 ativo
                CMP     M[GAME_ON], R0          ;verifica se a interrupcao I1 fo
i premida
                BR.Z    Waitgamestart
                CMP     M[HARDCORE], R0
                CALL.NZ LigaHardcore            ;ativa o modo hardcore se M[HARD
CORE] == 1
                DSI
                JMP     Jogo

; VeriHardcore: Rotina que verifica se o modo hardcore vai ser ativo ou nao
; Input:  ---
; Efeito: altera M[HARDCORE].
; Output: ---
```



```
VeriHardcore:   MOV     R1, M[INT_CONTROL]      ;passa para R1, o valor dos inte
rruptores
                AND     R1, HARDCORE_MASK
                CMP     R1, 1                   ;compara o valor do interruptor
0 , com 1
                BR.Z    IfHarcore               ;se for igual, a variavel de con
trol passa a 1,
                BR.NZ   ElseHardcore            ;se nÃ£o passa a 0
IfHarcore:      MOV     R1, 1
                MOV     M[HARDCORE], R1
                BR      FimVeriHarcore
ElseHardcore:   MOV     M[HARDCORE], R0
FimVeriHarcore: RET

; LigaHardcore: Rotina que ativa o modo hardcore
; Input:  ---
; Efeito: altera M[LEDS_CONTROL] e M[SPEED_ATUAL].
; Output: ---

LigaHardcore:   MOV     R1, FFFFh
                MOV     M[LEDS_CONTROL], R1     ;liga os leds todos
                MOV     R1, 1
                MOV     M[SPEED_ATUAL],R1       ;altera a velocidade de jogo par
a 1 em vez de 7
                RET
;+--------------------------------------------------+
;|                      Jogo                        |
;+--------------------------------------------------+

; Jogo: Rotina que inicia o jogo
; Input:  ---
; Efeito: altera M[DIR1_ATUAL], [DIR1_PROX], M[DIR2_ATUAL] e [DIR2_PROX].
; Output: ---

Jogo:           CALL    Desenhacampo                    ;desenha o campo
                CALL    EscrevePart                     ;desenha as particulas n
a posicao inicial
                MOV     R7,INT_MASK_JOGO                ;interrupcoes
                MOV     M[INT_MASK_ADDR],R7
                ENI
                MOV     R7, 1                           ;inicia temporizador
                MOV     M[TimerValue], R7
                MOV     M[TimerControl], R7
                MOV     M[DIR1_ATUAL],R7                ;reinicia direcoes
                MOV     M[DIR1_PROX],R7
                MOV     R7,3
                MOV     M[DIR2_ATUAL],R7
                MOV     M[DIR2_PROX],R7
                JMP     IngameLoop

; IngameLoop: Loop de jogo
; Input:  ---
; Efeito: altera M[SPEED] e M[SECONDS].
; Output: ---

IngameLoop:     MOV     R7, M[TIME]
                CMP     R7, M[OLD_TIME]                 ;se a variavel TIME nÃ£o
 se tiver alterado
                BR.Z    IngameLoop                      ;da-se um salto para o i
nicio da rotina
                DSI
```

```
                CMP     M[HARDCORE], R0                 ;se o modo harcore estiv
er ativo, nao e permitido
                BR.NZ   SkipPausa                       ;fazer-se pausa
                CALL    VerificaPausa                   ;verifica pausa
SkipPausa:      MOV     R7, M[TIME]
                CMP     R7, 10d                         ;se R7 for igual a 10d,
significa que passou um segundo
                BR.N    SkipSegs                        ;logo tem de ser chamada
s rotinas relacionadas com a placa,
                INC     M[SECONDS]                      ;incrementado os segundo
s que passaram
                CALL    Escreve_7seg                    ;sete seguementos
                CALL    LedSpeed                        ;leds e velocidades
                CALL    VerificaTMAX                    ;verifica o tempo maximo

SkipSegs:       MOV     R7, M[SPEED]
                CMP     R7, M[SPEED_ATUAL]              ;se o R7 for igual a vel
ocidade atual, sao executadas rotinas de movimento
                BR.N    EndgameLoop
                MOV     M[SPEED], R0
                CALL    Particulas
                CALL    VerificaFimJogo
EndgameLoop:    ENI
                PUSH    M[TIME]                                 ;atualiza o cont
ador anterior, para efeitos de comparacao
                POP     M[OLD_TIME]
                JMP             IngameLoop

; VerificaPausa: Rotina que ativa e destiva pausa
;   Input:  ---
;       Efeito: ---
;   Output: ---

VerificaPausa:  PUSH    M[DIR1_PROX]                    ;salvaguarda as direcoes
 antes da pausa
                PUSH    M[DIR2_PROX]
    LoopPausa:  MOV             R1, M[INT_CONTROL]
                TEST    R1, PAUSA_MASK                  ;verifica se o interrupt
or 7 esta ligado
                BR.NZ   LoopPausa                               ;se sim executa
o loop, ate se desligar
                MOV     R7,INT_MASK_PAUSA               ;atualizar interrupcoes
para o relogio nao avancar
                MOV     M[INT_MASK_ADDR],R7
                ENI
                POP             M[DIR2_PROX]                    ;volta a colocar
 as direcoes corretas
                POP     M[DIR1_PROX]
                DSI
                MOV     R7,INT_MASK_JOGO                ;volta a ativar o relogi
o interrupcoes
                MOV     M[INT_MASK_ADDR],R7
                RET




; Particulas: Rotina que trata dos movimentos das particulas
;   Input:  ---
;       Efeito: ---
```



```
;   Output: ---
Particulas:     CALL    VerificaInt
                CALL    VerificaDir
                CALL    MoveParticulas
                RET



; VerificaFimJogo: Rotina verifica se o jogo acabou
;   Input:  ---
;       Efeito: ---
;   Output: ---

VerificaFimJogo:MOV     R1, M[JOG1_COL]
                MOV     R2, M[JOG2_COL]
                OR              R1, R2                  ;se uma das particulas c
olidiu
                CMP     R1, 1                   ;o jogo vai ser reiniciado
                BR.Z    ReiniciaJogo
                RET


; ReiniciaJogo: Rotina que reinicia o jogo
;   Input:  ---
;       Efeito: altera M[GAME_ON], M[LOCAL1], M[LOCAL2], M[SPEED], M[TIME], M[SE
CONDS],
;               M[SPEED_ATUAL], M[LED_MASK], M[JOG1_COL], M[JOG2_COL], M[DIR1
_ATUAL],
;               M[DIR1_PROX], M[DIR2_ATUAL] e [DIR2_PROX].
;   Output: ---
ReiniciaJogo:   DSI
                CALL    Texto_final                     ;escreve o texto final
                CALL    Escrita_LCD                     ;atualiza o lcd
                POP     R7
                MOV     R7, SP_START            ;reinicia a pilha
                MOV     SP, R7
                MOV     M[GAME_ON], R0          ;desliga o controlo de jogo
                MOV     R7, I_LOCAL1
                MOV     M[LOCAL1], R7           ;reinicia o cursor das particula
s
                MOV     R7, I_LOCAL2
                MOV     M[LOCAL2], R7
                MOV     M[SPEED], R0            ;reinicia os contadores
                MOV     M[TIME], R0
                MOV     M[SECONDS], R0
                CALL    Escreve_7seg            ;mete o tempo a 0 no display de
sete seguementos
                MOV     R7,7
                MOV     M[SPEED_ATUAL], R7      ;reinicia a velocidade
                MOV     M[LED_MASK], R0         ;desliga os leds
                MOV             M[LEDS_CONTROL], R0

                MOV     M[JOG1_COL], R0         ;coloca o numero de colisoes a 0
                MOV     M[JOG2_COL], R0

                MOV             M[TimerControl], R0     ;desliga o temporizador

                MOV     R7, INT_MASK_JOGO
                MOV     M[INT_MASK_ADDR], R7
                ENI
                MOV     R7, INT_MASK_INI        ;interrupcoes iniciais novamente
                MOV     M[INT_MASK_ADDR], R7
```

```
                MOV     M[GAME_ON], R0
                MOV     R7,1                            ;reinicia as direcoes da
s particulas
                MOV     M[DIR1_ATUAL],R7
                MOV     M[DIR1_PROX],R7
                MOV     R7,3
                MOV     M[DIR2_ATUAL],R7
                MOV     M[DIR2_PROX],R7
                JMP     Waitgamestart


; Texto_final: Rotina que escreve na janela de texto as freses de fim de jogo
;   Input:  ---
;       Efeito: ---
;   Output: ---

Texto_final:    PUSH    R1
                PUSH    R2
                MOV     R1, moldura                     ;desenha a moldura super
ior do texto final na janela de texto
                MOV     R2, 0A1Ah
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                MOV     R1, FimStr                      ;desenha a primeira fras
e final na janela de texto
                MOV     R2, FRASE_UM
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                MOV     R1, FimStr2             ;desenha a segunda frase final n
a janela de texto
                MOV     R2, FRASE_DOIS
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                MOV     R1, moldura             ;desenha a moldura inferior do t
exto final na janela de texto
                MOV     R2, 0D1Ah
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                POP             R2
                POP     R1
                RET
;----------------------Colisoes------------------

; Vericolisoes: Rotina que verifica se houve colisoes
;   Input:  ---
;       Efeito: altera M[POINT1], M[POINT2]
;   Output: ---

Vericolisoes:   MOV     R1, M[LOCAL1]
                PUSH    R1
                CALL    Ler_matriz              ;coloca no R1, o  caracter da po
sicao que estava em R1
                POP     R1
                CMP     R1,20h                  ;se o caracter nao for um espaco
 houve colisao do jogador1
                CALL.NZ Colide1
                MOV     R1, M[LOCAL2]
                PUSH    R1
```



```
                CALL    Ler_matriz
                POP     R1
                CMP     R1 , 20h
                CALL.NZ Colide2
                MOV     R2, M[JOG1_COL]
                AND     R1, R2
                CMP     R1,1                    ;se ambos os jogadores colidiram
 vai ser subtraido o ponto que outrora foi adicionado
                BR.NZ   FimColisoes
                MOV     R1, PONTUA_MASK
                SUB     M[POINT1],R1
                SUB     M[POINT2],R1
FimColisoes:    MOV     R1, M[POINT1]           ;vai fazer o complemento para 2
de cada pontuacao
                PUSH    R1
                CALL    VeriPontuacoes
                POP     R1
                MOV     M[POINT1], R1
                MOV     R1, M[POINT2]
                PUSH    R1
                CALL    VeriPontuacoes
                POP     R1
                MOV     M[POINT2], R1
                RET

; Colide1: Rotina que incrementa os pontos do jog2 se o jog1 colidiu
; Input:  ---
; Efeito: altera M[POINT2] e M[JOG1_COL]
; Output: ---

Colide1:        MOV     R1, PONTUA_MASK         ;adiciona um ponto ao adversario

                ADD     M[POINT2],R1            ;e coloca na variavel de colisao

                MOV     R1, 1                   ;do jogador1 o valor 1
                MOV     M[JOG1_COL], R1
                RET

; Colide2: Rotina que incrementa os pontos do jog1 se o jog2 colidiu
;   Input:  ---
;       Efeito: altera M[POINT1] e M[JOG2_COL]
;   Output: R1 - o estado da veriavel de controlo da colisao do jogador2

Colide2:                MOV     R1, PONTUA_MASK         ;adiciona um ponto ao ad
versario
                ADD             M[POINT1],R1            ;e coloca na variavel de
 colisao
                MOV     R1, 1                           ;do jogador1 o valor 1
                MOV     M[JOG2_COL], R1
                RET

; VeriPontuacoes: Rotina que converte a pontuacao de hexadecimal para uma repres
entacao decimal
;   Input:  pilha- Pontuacao(R1)
;       Efeito: altera M[POINT1] e M[JOG2_COL]
;   Output: pilha- Pontuacao convertida(R1)
VeriPontuacoes: POP     R4
                POP     R1
                MOV     R2, R1
                AND     R2, 0F00h               ;verifica se o digito Ã  direita
 da pontuacao e 'A'
                CMP     R2, 0A00h               ;e se o for subtrai A e soma 10,
```

```
 para simular o sistema decimal
                BR.NZ   FimVeriPont
                ADD     R1, 1000h
                SUB     R1, 0A00h
FimVeriPont:    PUSH    R1
                PUSH    R4
                RET



;-----------------Verifica Direcoes--------------

; VerificaDir: Rotina que compara as direcoes com 5 e 0 e as atualiza para direc
oes validas
;   Input:  ---
;       Efeito: altera M[DIR1_PROX] e M[DIR2_PROX]
;   Output: ---
VerificaDir:    MOV     R1,M[DIR1_PROX]
                CALL    ComparaDir5
                MOV     M[DIR1_PROX], R1
                MOV     R1,M[DIR2_PROX]
                CALL    ComparaDir5
                MOV     M[DIR2_PROX], R1
                RET

ComparaDir5:    CMP     R1,5                            ;se a direcao for 5 volt
a a 1
                BR.NZ   ComparaDir0
                MOV     R1,1
                BR              FimComparaDir
ComparaDir0:    CMP     R1,0                            ;se a direcao for 0 volt
a a 4
                BR.NZ   FimComparaDir
                MOV     R1,4
FimComparaDir:  RET


;--------------Verifica InterrupÃ§Ãµes-------------

; VerificaInt: Rotina que compara a proxima direcao com a anterior e atualiza a
proxima direcao dependendo do resultado
;   Input:  ---
;       Efeito: altera M[DIR1_PROX] e M[DIR2_PROX]
;   Output: ---

VerificaInt:    MOV     R1, M[DIR1_PROX]
                MOV     R2, M[DIR1_ATUAL]
                CMP     R1, R2                                  ;se a proxima di
recao foi maior que a anterior,
                BR.Z    NextVerificaInt                 ;a proxima direcao passa
 a ser igual a anterior +1
                CALL.P  Aumentaproxdir                  ;se for menor, em vez de
 incrementar, decremente.
                CALL.N  Decreproxdir                    ;se nao se verificar nen
huma das opcoes, mante se igual
                MOV     M[DIR1_PROX], R1

NextVerificaInt:MOV     R1, M[DIR2_PROX]                ;para esta particula rep
ete-se o procedimento da anterior
                MOV     R2, M[DIR2_ATUAL]
                CMP     R1, R2
                BR.Z    FimVerificaInt
```



```
                CALL.P  Aumentaproxdir
                CALL.N  Decreproxdir
                MOV     M[DIR2_PROX], R2
FimVerificaInt: RET

; Aumentaproxdir: Rotina que incrementa a direcao
;   Input:  R2-direcao atual
;       Efeito: ---
;   Output: R1-proxima direcao
Aumentaproxdir: INC     R2                      ;incrementa direcao atual
                MOV     R1, R2          ;coloca o valor em R1
                RET
; Decreproxdir: Rotina que decrementa a direcao
;   Input:  R2-direcao atual
;       Efeito: ---
;   Output: R1-proxima direcao
Decreproxdir:   DEC     R2                      ;incrementa direcao atual
                MOV     R1, R2          ;coloca o valor em R1
                RET


;-----------------Move Particulas----------------

; MoveParticulas: Rotina que altera o cursor para cada particula.
;   Input:  ---
;       Efeito: altera M[LOCAL1], M[LOCAL2], M[DIR2_ATUAL] e M[DIR1_ATUAL]
;   Output: ---

MoveParticulas: MOV     R2, M[DIR1_PROX]
                MOV     M[DIR1_ATUAL], R2       ;atualiza a direcao atual
                MOV     R3, M[R2+8000h]         ;retira o valor da soma/subtraca
o da memoria
                MOV     R4, R3
                MOV     R2, M[LOCAL1]
                AND     R3, F000h                       ;se o bit  mais signific
ativo for 1 a
                AND     R4, 0FFFh                       ;operacao a ser executad
a sera a subtracao
                CMP     R3, 1000h                       ;caso contrario sera a a
dicao
                CALL.Z  Subtraidirecao          ;quanto ao que sera somado, essa
 informacao
                CMP     R3, 1000h                       ;esta nos 3 bits menos s
ignificativos dessa palavra
                CALL.NZ Aumentadirecao          ;retirada da posicao de memoria
 8000h + R2
                MOV     M[LOCAL1], R2
                MOV     R2, M[DIR2_PROX]
                MOV     M[DIR2_ATUAL], R2
                MOV     R3, M[R2+8000h]
                MOV     R4, R3
                MOV     R2, M[LOCAL2]
                AND     R3, F000h
                AND     R4, 0FFFh
                CMP     R3, 1000h
                CALL.Z  Subtraidirecao
                CMP     R3, 1000h
                CALL.NZ Aumentadirecao
                MOV     M[LOCAL2], R2
                CALL    Vericolisoes            ;verifica colisoes apos atualiza
r os as novas posicoes das particulas
                CALL    EscrevePart                     ;escreve ambas as partic
```

```
ulas nas novas posicoes
                RET

; Subtraidirecao: Rotina que subtrai valores a antiga posicao da particula.
;   Input:  R2-Antiga posicao da particula, R4-Elemento a subtrair
;       Efeito: ---
;   Output: R2-Nova posicao da particula
Subtraidirecao: SUB     R2, R4
                RET

; Aumentadirecao: Rotina que adiciona valores a antiga posicao da particula.
;   Input:  R2-Antiga posicao da particula, R4-Elemento a adicionar
;       Efeito: ---
;   Output: R2-Nova posicao da particula
Aumentadirecao: ADD     R2, R4
                RET




;-----------------------------------------------

; VerificaTMAX: Rotina que atualiza o tempo maximo, por comparacao ao tempo atua
l
;   Input:  ---
;       Efeito: altera M[TMP_MAX]
;   Output: ---

VerificaTMAX:   PUSH    R1
                MOV             R1, M[SECONDS]          ;compara o tempo atual c
om o maximo
                CMP             R1, M[TMP_MAX]          ;se o tempo atual for ma
ior, o tempo maximo
                BR.N    FimVerificaTMAX         ;assume o seu valor
                MOV             M[TMP_MAX], R1
                CALL    Escrita_LCD                     ;atualiza o lcd
FimVerificaTMAX:POP     R1
                RET



;+--------------------------------------------------+
;|               Alteracoes da Placa                |
;+--------------------------------------------------+


;-----------Sete seguementos-----------------
; Escreve_7seg: Rotina que escreve o tempo decorrido nos display de 7 seguemento
s
;   Input:  ---
;       Efeito: altera M[TIME], M[SECONDS]
;   Output: ---


Escreve_7seg:   PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV             M[TIME], R0                     ;reset ao contad
or
                MOV             R1, M[SECONDS]
                MOV             R2, 4                           ;numero de digit
```



```
o, e o numero de vezes que o loop vai repetir
                MOV             R3, DISPLAY_7SEG
Loop7Seg:               PUSH    R1                                      ;coloca
na pilha o numero de segundos
                AND             R1, 0Fh                         ;se o ultimo dig
ito for 'A',
                CMP             R1, 0Ah                         ;vai ser feito o
 complemento a dois do digito,
                CALL.Z  ComplementaDez          ;transformando-se o mesmo em 0 e
 somando 1 ao digito a esquerda
                POP             R1                                      ;do mesm
o. o tempo e extraido da pilha
                MOV             M[R3], R1                       ;coloca esse dig
ito num nibble do display
                ROR             R1, 4                           ;roda-se para qu
e o digito menos significativo passe ao da esquerda
                INC             R3                                      ;muda-se
 de nibble
                DEC             R2
                CMP             R2, R0                          ;quando ja tiver
em sido analisados os 4 digitos o loop para
                BR.NZ   Loop7Seg
                MOV             M[SECONDS], R1          ;atualizacao dos segundo
s
                POP             R3
                POP             R2
                POP             R1
                RET

;-----------------------------------------------

;--------------------LEDS-------------------

; LedSpeed: Rotina que altera a velocidade de jogo e que acende os leds na passa
gem de nivel
; Input:  ---
; Efeito: altera M[SPEED_ATUAL], M[LED_MASK]
; Output: ---

LedSpeed:       PUSH    R1
                PUSH    R2
                MOV     R1, M[HARDCORE]
                CMP     R1, 1                   ;se o modo hardcore estiver ativ
o
                JMP.Z   FimLedSpeed             ;nada e executado no que diz res
peito
                MOV     R1, M[SECONDS]          ;a alteracoes de velocidades ou
leds
                CMP     R1, 10h                 ;dependendo do tempo pass que pa
ssou
                BR.Z    Lvl2                    ; e escolhida uma velocidade
                CMP     R1, 20h
                BR.Z    Lvl3
                CMP     R1, 40h
                BR.Z    Lvl4
                CMP     R1, 60h
                BR.Z    Lvl5
                BR      FimLedSpeed
Lvl2:           MOV     R2,5
                BR      EscreveLed
Lvl3:           MOV     R2,3
                BR      EscreveLed
```

```
Lvl4:           MOV     R2,2
                BR      EscreveLed
Lvl5:           MOV     R2,1
                BR      EscreveLed
EscreveLed:     MOV     M[SPEED_ATUAL], R2      ;atualiza leds, ligado mais quat
ro leds
                MOV     R2, M[LED_MASK]
                SHL     R2, 4
                ADD     R2,     Fh
                MOV     M[LED_MASK], R2
                MOV     M[LEDS_CONTROL], R2
FimLedSpeed:    POP     R2
                POP     R1
                RET
;------------------------------------------------
;--------------------LCD--------------------

; Escrita no LCD

; Escrita_LCD: Rotina que escreve no lcd toda ainformacao pretendida
; Input:  ---
; Efeito: ---
; Output: ---

Escrita_LCD:    PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1,TEXTO_LCD            ;vai buscar a posicao de memoria
 da primeira string do lcd
                MOV     R3, M[R1]               ;tira-lhe o  primeiro caracter
                MOV     R2,8000h                ;mete no R2 o controlo do lcd
                CALL    CaracteresLCD           ;escreve no lcd. o proesso repet
e-se
                MOV     R1, M[TMP_MAX]
                MOV     R4,4                    ;para valores numericos, mete-se
 o R4 o numero de digitos do numero
                CALL    EscreveNumLCD
                MOV     R1, TEXTO2_LCD
                MOV     R3,M[R1]
                CALL    CaracteresLCD
                MOV     R1, M[POINT1]
                MOV     R4,2
                CALL    EscreveNumLCD
                MOV     R1,TEXTO3_LCD
                MOV     R3, M[R1]
                CALL    CaracteresLCD
                MOV     R1, M[POINT2]
                MOV     R4,2
                CALL    EscreveNumLCD
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

; CaracteresLCD: Rotina que escreve strings no lcd
; Input:  R2-Posicao do cursor do lcd, R3-Primeiro caracter a escrever
; Efeito: ---
; Output: R2-Posicao do cursor do lcd

CaracteresLCD:  MOV     M[LCD_CONTROL], R2              ;atualiza o cursor
```



```
                MOV     M[LCD_ESCRITA], R3              ;escreve o caracter
                INC     R1
                MOV     R3, M[R1]                       ;passa ao caracter segui
nte
                INC     R2                              ;avanca uma coluna
                CMP     R3, END_STR
                BR.NZ   CaracteresLCD
                RET

; EscreveNumLCD: Rotina que escreve numeros no lcd
; Input: R1-Numero a escrever, R2-Posicao do cursor do lcd, R4-Numero de digitos
 do numero
; Efeito: ---
; Output: R2-Posicao do cursor do lcd

EscreveNumLCD:  ROL     R1,4                            ;passa o primeiro digito
 para o fim para ser o primeiro a ser escrito
                MOV     R3, R1
                AND     R3, 0Fh
                ADD     R3,30h                          ;soma 30 para que se tor
ne no digito ascii correspondente
LoopValores:    MOV     M[LCD_CONTROL], R2
                MOV     M[LCD_ESCRITA], R3
                ROL     R1 , 4                          ;passa ao proximo digito
                MOV     R3, R1
                AND     R3, 0Fh                         ;seleciona o proximo dig
ito e repete-se o procedimento
                ADD     R3,30h
                INC     R2
                DEC     R4
                CMP     R4,R0                           ;ve se ja se escreveram
 todos os digitos
                BR.NZ   LoopValores
                RET
;+--------------------------------------------------+
;|         Rotinas auxiliares gerais                |
;+--------------------------------------------------+

; Desenhacampo: Rotina que desenha os limites de jogo na janela de texto
; Input:        ---
; Efeito: ---
; Output: ---

Desenhacampo:   MOV     R3,WALL_START                   ;move a origem do cursor
 para R3
                MOV     R1,TB_Wall                      ;move a string correspon
dente a parte superior da janela de jogo
                MOV     R2,R3
    Desenhalat: PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt                      ;escreve a string com os
 dados que estavam em R1, e R2
                MOV     R1,RL_Wall                      ;passa para R1 a string
 correspondente as paredes laterais
                ADD     R3,ONE_LINE                     ;passa o cursor para a l
inha seguinte
                MOV     R2,R3
                CMP     R2, WALL_END
                BR.NZ   Desenhalat                      ;quando o cursor chegar
a posicao deermindada o loop pÃ¡ra
                MOV     R1,TB_Wall                      ;e de seguida escreev-se
 a parede inferior.
```

```
                PUSH    R1
                PUSH    R2
                CALL    Escreve_Jt
                RET

; EscrevePart: Rotina que desenha ambas as particulas na janela de texto
; Input:        ---
; Efeito: ---
; Output: ---

EscrevePart:    MOV     R1, JOG1_PART                   ;coloca o simbolo da par
ticula em R1.
                MOV     R2, M[LOCAL1]                   ;coloca o local onde vai
 ser desenhada a particula em R2.
                CALL    Desenhapart                     ;escreve a particula com
 esses parametros e repete para a outra particula.
                MOV     R1, JOG2_PART
                MOV     R2, M[LOCAL2]
                CALL    Desenhapart
                RET

; Desenhapart: Rotina que desenha uma particula na janela de texto
; Input: R1-o valor em hexadecimal da particula, R2-a posicao do cursor onde se
vai escrever
; Efeito: ---
; Output: ---

Desenhapart:    ADD     R2, WALL_START                  ;como as 'coordenadas' d
a particula sao medidas para dentro da janela de jogo,
                PUSH    R1                              ;soma-se Ã s mesmas a po
sicao do canto superior esquerdo
                PUSH    R2
                CALL    Escreve_Jt
                RET

; Escreve_Jt: Rotina que desenha caracteres na janela de texto
; Input: pilha- posicao do cursor(R2), caracter a escrever(R1)
;       Efeito: ---
; Output: ---

Escreve_Jt:     POP     R4
                POP     R2
                POP     R1
                PUSH    R3
                MOV     R3,M[R1]                        ;seleciona apenas um car
acter da string.
    Escreve:    MOV     M[JT_CURSOR], R2                ;atualiza o cursor.
                MOV     M[JT_OUT],R3                    ;escreve o caracter sele
cionado
                CALL    Escreve_matriz                  ;escreve na matriz esse
caracter, na mesma posicao que na janela de texto
                INC     R1
                MOV     R3,M[R1]
                INC     R2
                CMP     R3, END_STR                     ;quando atinge o caracte
r '@' para de escrever.
                BR.NZ   Escreve
                POP     R3
                PUSH    R4
                RET

; Escreve_matriz: Rotina que escreve os caracteres em posicoes de memoria da mat
```



```
riz colisoes
; Input: pilha- posicao do cursor(R2), caracter a escrever(R1)
; Efeito: ---
; Output: ---

Escreve_matriz: PUSH    R1                              ;salvaguardar valores Ã©
 muit importante aqui,
                PUSH    R2                              ;para que a escrita na j
anela de texto funcione com normalidade
                PUSH    R3
                PUSH    R4
                MOV     R1,M[SP+2]                      ;coloca em R1 o caracter
                MOV     R2,M[SP+3]                      ;coloca em R2 a posicao
onde se irÃ¡ escrever
                MOV     R3, R2
                MOV     R4, COLS                        ;move para R4 o numero d
e colunas
                AND     R2, LINE_MASK                   ;seleciona o numero da l
inha
                SHR     R2, 8
                AND     R3, COLS_MASK                   ;seleciona o numero da c
oluna
                MUL     R4, R2                          ;mulltiplica o numero de
 colunas da matriz o numero da linha onde se vai escrever
                ADD     R2, R3                          ;soma desse resultado co
m o numero da coluna onde vamos escrever
                MOV     M[R2+Colisoes], R1              ;escreve a particula na
posicao de memoria calculada
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

; Ler_matriz: Rotina que devolve o caracteres em posicoes de memoria da matriz c
olisoes
; Input: pilha- posicao do cursor(R2)
; Efeito: ---
; Output: pilha- caracter que esta na posicao correspondente(R1)

Ler_matriz:     POP     R4
                POP     R2                              ;R2 tem a posicao da jan
ela de texto onde se quer procurar caracter
                PUSH    R3
                ADD     R2, WALL_START                  ;o procedimento Ã© equiv
alente ao da rotina de escrita na matriz
                MOV     R3, R2                          ;no entanto, no final, e
m vez de se escrever o caracter que estÃ¡ em R1,
                MOV     R1, COLS                        ;na posicao calculada, c
oloca-se em R1 o caracter que estÃ¡ na posicao calculada.
                AND     R2, LINE_MASK
                SHR     R2, 8
                AND     R3, COLS_MASK
                MUL     R1, R2
                ADD     R2, R3
                MOV     R1, M[R2+Colisoes]
                POP     R3
                PUSH    R1
                PUSH    R4
                RET
```

```
; ComplementaDez: Rotina que devolve um numero cujo digito final Ã© transformado
 no seu complemento para 2
; Input:        pilha- digito a converter(R1)
;       Efeito: ---
;   Output: pilha- digito convertido(R1)

ComplementaDez: POP     R4
                POP     R1                      ;R1 contÃ©m o numero para o qual
 iremos fazer o complemento do ultimo digito
                PUSH    R2                      ;salvaguarda o conteudo de R2
                MOV     R2, R1
                COM     R1                      ;faz o complemento de R1,
                INC     R1                      ;soma um ao ultimo digito
                AND     R1, 0Fh
                ADD     R1 , R2                 ;e soma apenas esse digito ao nu
mero inicial
                POP     R2
                PUSH    R1
                PUSH    R4
                RET
```