```
;=================================================================
;
;            _______ _______ _______ _______
;           |_     _|   __ \       |     |  |
;             |   | |      <   -   |       |
;             |___| |___|__|_______|__|____|
;
;   Introducao a Arquitetura de Computadores − LEIC−A
;               1o. Sem (2014/2015)
;
;Grupo 64 − Turno 9
;Nuno Vieira    − 81098
;Rodrigo Rato   − 81500
;Patricia Faria − 81611
;=================================================================


;=================================================================
; <INICIO CONSTANTES>
;=================================================================
; PILHA
SP_INICIAL          EQU     FDFFh   ;Inicializacao da pilha

;INTERRUPCOES
INT_MASK_ADDR       EQU     FFFAh   ;Inicializacao da mascara
INT_MASK_JOGO       EQU     1000101010000011b ;Mascara para jogo

;JANELA DE TEXTO
TXT_WRITE           EQU     FFFEh   ;Escrita na janela de texto
TXT_CTRL            EQU     FFFCh   ;Controlo da janela de texto
TXT_CTRL_INIT       EQU     FFFFh   ;Inicializa janela de texto

;CARACTERES
FIM_STR             EQU     '@'     ;Marca o fim das strings
CHR_MAIS            EQU     '+'
CHR_MENOS           EQU     '−'
CHR_ESPACO          EQU     ' '
CHR_BARRA           EQU     '|'
CHR_NEWL            EQU     0Ah     ;ASCII numero 10  −> '\n'
CHR_PLAYER1         EQU     'X'
CHR_PLAYER2         EQU     '#'

;DISPLAY DE SETE SEGMENTOS
                                    ;Digitos do display de 7 segs.
D7S0            EQU     FFF0h   ;menos significativo
D7S1            EQU     FFF1h   ;meio direita
D7S2            EQU     FFF2h   ;meio esquerda
D7S3            EQU     FFF3h   ;mais significativo

;LCD
                                    ;Enderecos do LCD
LCD_CONTROL     EQU     FFF4h   ;controlo
LCD_WRITE       EQU     FFF5h   ;escrita
LCD_L0          EQU     8000h   ;Linha 0 e Posicao 0
LCD_L1          EQU     8010h   ;Linha 1 e Posicao 0
LCD_TEMP        EQU     800Bh   ;Valor o tempo max
LCD_J1          EQU     8014h   ;Pontuacao do Jogador 1
LCD_J2          EQU     801Eh   ;Pontuacao do Jogador 2

;LEDS
LED_PORT        EQU     FFF8h   ;Endereço de escrita nos leds

;TEMPORIZADOR
```



```
TEMP_ONOFF      EQU     FFF7h
TEMP_ADDR       EQU     FFF6h

;INTERRUPTORES
INTR_ADDR       EQU     FFF9h

;=================================================================
; <FIM CONSTANTES>
;=================================================================



;=================================================================
; <INICIO INTERRUPCOES>
;=================================================================

                ORIG    FE01h   ;Menu/Inicio e fim de jogo
INT1            WORD    MenuRedirect

                ORIG    FE00h   ;Botao Esquerda do Player 1
INT0            WORD    J1Esquerda

                ORIG    FE07h   ;Botao Esquerda do Player 2
INT7            WORD    J2Esquerda

                ORIG    FE09h   ;Botao Direita do Player 2
INT9            WORD    J2Direita

                ORIG    FE0Bh   ;Botao Direita do Player 1
INT11           WORD    J1Direita

                ORIG    FE0Fh   ;Interrupcao do temporizador
INT15           WORD    acabaDelay

;=================================================================
; <FIM INTERRUPCOES>
;=================================================================



;=================================================================
; <INICIO VARIAVEIS/STRINGS/MATRIZ EM MEMORIA>
;=================================================================

            ORIG    6000h
MATRIZ      TAB     1100d   ;Declara os espacos necessarios para
                            ;a matriz que vai desenhar o campo
                            ;de jogo para a memoria

            ORIG    8000h
MSG1        STR     'Bem−vindo ao TRON@'
MSG2        STR     'Pressione I1 para comecar@'
MSG3        STR     '||          Fim do jogo          ||@'
MSG4        STR     '||  Pressione I1 para recomecar  ||@'
MSG5        STR     '**PAUSA**@'
MSG6        STR     'JOGADOR 1 GANHOU!@'
MSG7        STR     'JOGADOR 2 GANHOU!@'
MSG8        STR     'EMPATE!@'
FIM_FRAME1  STR     '||                                ||@'
FIM_FRAME2  STR     ' ================================ @'
TEMPOMAX    STR     'TEMPO MAX:     s@'
PONT_JOG    STR     'J1:       J2:@'
```

```
TempJ1J2    TAB     3       ;1ª tempo max, 2º pont j1, 3º pont j2
MOV_J1      TAB     1
MOV_J2      TAB     1
DECIMAS     TAB     1
MENU        TAB     1

;=====================================================================
; <FIM VARIAVEIS/STRINGS/MATRIZ EM MEMORIA>
;=====================================================================


            ;Salta para o inicio do programa
            ORIG    0000h
            JMP     inicio


;=====================================================================
; <DEFINICAO DE ROTINAS/SUBROTINAS/FUNCOES>
;=====================================================================

;   <desenhaMolduraEParticulas>
; Rotina simples que chama as subrotinas de desenho da moldura e
; particulas − apenas deve acontecer quando ha uma interrupcao do
; botao I1 e o jogo nao esta a decorrer
desenhaMolduraEParticulas:  CALL        limpaEcra
                            CALL        drawMoldura
                            CALL        particulasInicio
                            RET
;versao anterior − inicializa porto de controlo da janela de txto



;Rotina que imprime uma string na janela de texto
;centrada na coordenada que vai receber como argumento
;Recebe em R1 o endereco de memoria da primeira posicao da string
;Recebe por R2 as coordenadas onde imprimir a string centrada
;NAO INICIALIZA O CONTROLE DA JANELA DE TEXTO!
;   <prtStringCentrada>
prtStringCentrada:  PUSH    R1
                PUSH        R2
                PUSH        R3
                PUSH        R4

                ;em R4 vamos contar o comprimento da string
                MOV         R4, R0
                MOV         R3, FIM_STR
contaComp:      CMP         M[R1], R3
                ;acabou de contar o comprimento
                BR.Z        centraString
                INC         R4
                INC         R1
                ;ainda nao acabou de contar o comprimento, ciclo
                BR          contaComp

centraString:   MOV         R3, R4
                SHR         R3, 1       ;divide por dois sem DIV
                ;calcula a posicao onde vai ficar escrito o
                ;ultimo caracter da string, vai escrever
                ;do fim para o inicio
                ADD         R2, R3

loopEscritaStr: DEC         R2
```



```
                ;especial atencao para nao
                ;imprimir FIM_STR (constante)
                DEC         R1
                DEC         R4
                BR.N        prtStringFim

                MOV         R3, M[R1]
                MOV         M[TXT_CTRL], R2
                MOV         M[TXT_WRITE], R3
                BR          loopEscritaStr
                ;esta a escrever a string,
                ;char a char, do fim para o inicio

prtStringFim:   POP         R4
                POP         R3
                POP         R2
                POP         R1
                RET


; Rotina que desenha o ecra de inicio do jogo:
; Desenha as duas frases definidas em MSG1 e MSG2
;centradas na janela de texto − Nao recebe nem devolve argumentos
;   <drawInicio>
drawInicio:     PUSH    R1
                PUSH    R2
                PUSH    R3

                ;R1 <− String a escrever, MSG1 (constante)
                MOV     R1, MSG1
                MOV     R2, 0B29h
                ;R2 <− Coordenada onde centrar a string,
                ;neste caso a coordenada e = (12,40)
                CALL    prtStringCentrada

                ;R1 <− String a escrever, MSG2 (constante)
                MOV     R1, MSG2
                MOV     R2, 0C29h
                ;R2 <− Coordenada onde centrar a string,
                ;neste caso a coordenada e = (13,40)
                CALL    prtStringCentrada

                POP     R3
                POP     R2
                POP     R1
                RET

; Rotina que desenha o ecra de inicio do jogo:
; Desenha as duas frases definidas em MSG1 e MSG2
;centradas na janela de texto − Nao recebe nem devolve argumentos
; a nao ser as strings a desenhar (constantes)
;   <drawInicio>
drawFim:        PUSH    R1
                PUSH    R2
                PUSH    R3

                MOV     R1, FIM_FRAME2
                MOV     R2, 0829h
                CALL    prtStringCentrada

                MOV     R1, FIM_FRAME1
                MOV     R2, 0929h
```

```
                CALL    prtStringCentrada

                MOV     R1, FIM_FRAME1
                MOV     R2, 0A29h
                CALL    prtStringCentrada

                CMP     R4,R6
                BR.Z    Empate

                CMP     R4,R0
                BR.NZ   P1ganhou

                CMP     R6,R0
                BR.NZ   P2ganhou

Empate:         MOV     R1,MSG8
                BR      drawFimaposJ

P1ganhou:       MOV     R1,MSG6
                BR      drawFimaposJ

P2ganhou:       MOV     R1,MSG7

drawFimaposJ:   CALL    prtStringCentrada
                ;R1 <- String a escrever, MSG3 (constante)
                MOV     R1, MSG3
                MOV     R2, 0B29h
                ;R2 <- Coordenada onde centrar a string,
                ;neste caso a coordenada e = (12,40)
                CALL    prtStringCentrada

                ;R1 <- String a escrever, MSG4 (constante)
                MOV     R1, MSG4
                MOV     R2, 0C29h
                ;R2 <- Coordenada onde centrar a string,
                ;neste caso a coordenada e = (13,40)
                CALL    prtStringCentrada

                MOV     R1, FIM_FRAME1
                MOV     R2, 0D29h
                CALL    prtStringCentrada

                MOV     R1, FIM_FRAME2
                MOV     R2, 0E29h
                CALL    prtStringCentrada

                POP     R3
                POP     R2
                POP     R1
                RET


;Rotina que desenha uma linha 'qualquer' da moldura
;Comeca a desenhar na coord R3
;Um caracter igual ao Registo 1, 48 iguais ao Registo 2
;e o caracter do Registo 1 outra vez
;Recebe em R1 um caracter
;Recebe em R2 outro caracter
;   <drawMolReg>
drawMolReg:     PUSH    R1
                PUSH    R2
                PUSH    R3
```



```
                PUSH    R4
                MOV     M[TXT_CTRL], R3
                MOV     M[TXT_WRITE], R1
                MOV     R4, 48d
drawMolRegCiclo:    INC     R3
                MOV     M[TXT_CTRL], R3
                MOV     M[TXT_WRITE], R2
                DEC     R4
                BR.NZ   drawMolRegCiclo
                INC     R3
                MOV     M[TXT_CTRL], R3
                MOV     M[TXT_WRITE], R1
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET


;   <drawMoldura>
; Rotina que desenha a moldura do jogo a partir de uma tela vazia
; nao recebe nem devolve nada
drawMoldura:    PUSH        R1
                PUSH        R2
                PUSH        R3
                PUSH        R4
                MOV         R3, 010Fh
                MOV         R1, CHR_MAIS
                MOV         R2, CHR_MENOS
                CALL        drawMolReg

                MOV         R4, 20d
                MOV         R1, CHR_BARRA
                MOV         R2, CHR_ESPACO
drawMolduraCiclo:   ADD         R3, 0100h
                CALL        drawMolReg
                DEC         R4
                BR.NZ       drawMolduraCiclo

                ADD         R3, 0100h
                MOV         R1, CHR_MAIS
                MOV         R2, CHR_MENOS
                CALL        drawMolReg


                POP         R4
                POP         R3
                POP         R2
                POP         R1
                RET


;   <particulasInicio>
;Rotina que desenha as duas particulas nas suas posicoes iniciais
;# <-> (8, 10)  da area de jogo
;X <-> (40,10) da area de jogo
particulasInicio:       PUSH        R1
                PUSH        R2
                MOV         R1, 0C18h
                MOV         R2, CHR_PLAYER1
                MOV         M[TXT_CTRL], R1
                MOV         M[TXT_WRITE], R2
```

```
                MOV         R1, 0C37h
                MOV         R2, CHR_PLAYER2
                MOV         M[TXT_CTRL], R1
                MOV         M[TXT_WRITE], R2
                POP         R2
                POP         R1
                RET

;   <limpaLinha>
; Rotina que desenha um caracter espaco (' ') em
; todas as posicoes da linha que recebe por registo no R3
; O valor da linha tem de vir em formato normal(hexadecimal)
limpaLinha:     PUSH        R1
                PUSH        R2
                PUSH        R3
                MOV         R1, CHR_ESPACO
                SHL         R3, 8
                MOV         R2, R3
                ADD         R3, 80d

cicloLinha:     MOV         M[TXT_CTRL], R2
                ;escreve na posicao da janela
                ;de texto apontada por R2 um espaco
                MOV         M[TXT_WRITE], R1
                INC         R2


                CMP         R2, R3
                BR.NZ       cicloLinha
                POP         R3
                POP         R2
                POP         R1
                RET

;   <limpaEcra>
; Rotina que desenha espacos (' ') em todas as
; posicoes da janela de texto
; usa a rotina limpaLinha para limpar linha a linha
limpaEcra:      PUSH    R3

                MOV     R3, R0
cicloEcra:      CALL    limpaLinha
                INC     R3
                CMP     R3, 24d
                BR.NZ   cicloEcra

                POP     R3
                RET



;   <escreveMatriz>
;Rotina que escreve a matriz correspondente ao espaco de jogo
;(com moldura) em memoria a partir da posicao apontada por MATRIZ
;escreve já o campo e as particulas na posição inicial
escreveMatriz:  PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1, MATRIZ
                MOV     R2, CHR_MAIS
                MOV     R3, R2
```



```
                CALL    encheLinhaMatriz
                INC     R1

                MOV     R4, 20d
                ;Numero de linhas do tipo b a desenhar
                ;as com espacos no meio, tantas quantas
                ;linhas tiver o campo de jogo, neste caso 20
                MOV     R3, CHR_ESPACO
escreveMCiclo:  CALL    encheLinhaMatriz
                INC     R1
                DEC     R4
                BR.NZ   escreveMCiclo

                MOV     R3, R2
                CALL    encheLinhaMatriz

                MOV     R3, MATRIZ
                ADD     R3, 559d
                ;fica a apontar para a posicao
                ;onde esta o char do P1
                MOV     R1, CHR_PLAYER1
                MOV     M[R3], R1
                MOV     R3, MATRIZ
                ADD     R3, 590d
                ;fica a apontar para a posicao
                ; onde esta o char do P2
                MOV     R1, CHR_PLAYER2
                MOV     M[R3], R1


                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

;   <encheLinhaMatriz>
; Rotina que escreve uma linha na matriz com
; os caracteres que recebe por registos:
; R1:   Aponta para o primeiro elemento da linha a preencher
; R2:   Caracter para primeira e ultima posicao
; R3:   Caracter para as posicoes intermedias
; O R1 SAI ALTERADO PARA O VALOR QUE APONTA
; PARA O FINAL DA LINHA QUE ESCREVEU EM MEMORIA!
encheLinhaMatriz:   PUSH    R2
                    PUSH    R3
                    PUSH    R4

                    MOV     M[R1], R2
                    MOV     R4, 48d
                    ;Numero de colunas a desenhar sem
                    ;contar com a moldura, ou seja o numero
                    ;de colunas do campo de jogo, 48
                    INC     R1
encheLinhaCiclo:    MOV     M[R1], R3
                    INC     R1
                    DEC     R4
                    BR.NZ   encheLinhaCiclo
                    MOV     M[R1], R2

                    POP     R4
                    POP     R3
```

```
                    POP     R2
                    RET

;   <converteEcraMatriz>
; Rotina que recebe uma coordenada do ecra,
; do formato das que o porto de controlo da janela de texto
; recebe e devolve a posicao de memoria da posicao correspondente
; a essa coordenada.
; R1:   Recebe coordenada do ecra
; R1:   Devolve o valor da posicao de memoria pretendida
; Assume que a matriz a trabalhar esta
; declarada com o simbolo MATRIZ
; Exemplo: 0210h devolve 6033h (coordenada do
; canto superior esquerdo do campo de jogo)
converteEcraMatriz: PUSH    R2
                    PUSH    R3
                    SUB     R1, 010Fh
                    MOV     R2, R1
                    MOV     R3, R1
                    SHR     R2, 8
                    ;O R2 AGORA GUARDA A LINHA
                    ;EM FORMATO INTERMEDIO
                    SHL     R3, 8
                    SHR     R3, 8
                    ;O R3 AGORA GUARDA A COLUNA
                    ;EM FORMATO INTERMEDIO
                    CALL    multiplicaCoordenada
                    ;ALTEROU R2<-R2*50
                    ADD     R2, R3
                    ADD     R2, 6000h
                    MOV     R1, R2
                    POP     R3
                    POP     R2
                    RET

; <multiplicaCoordenada>
; Recebe em R2 um valor que vai multiplicar
; por 50 e devolver por R2
; R2 <- R2*50
; devolve R2
multiplicaCoordenada:   PUSH    R1
                        PUSH    R3
                        CMP     R2, R0
                        BR.Z    fim_mult
                        MOV     R3, R0
ciclo_mult:             ADD     R3, 50d
                        DEC     R2
                        BR.NZ   ciclo_mult
                        MOV     R2, R3
fim_mult:               POP     R3
                        POP     R1
                        RET

; <escreveCharNaMatriz>
; Recebe em R1 uma coordenada do formato das coordenadas
; do porto de controlo da janela de texto e recebe por
; R2 um valor do que vai escrever na posicao correspondente a R1
; na matriz em memoria.
escreveCharNaMatriz:    PUSH    R1
                        CALL    converteEcraMatriz
                        ;R1 agora tem pos. de memoria a escrever
                        MOV     M[R1], R2
```



```
                        POP     R1
                        RET

; <pausa>
;Funcao que atua durante a pausa
;Atenção que os jogadores podem alterar a direcção durante a pausa
pausa:      PUSH        R1
            PUSH        R2
            PUSH        R3
            BR          cicloPausa
initPausa:  MOV         R1, MSG5
            MOV         R2, 0129h
            CALL        prtStringCentrada
cicloPausa: MOV         R1, M[INTR_ADDR]
            SHR         R1, 7
            CMP         R1, 1
            BR.NZ       fimPausa
            BR          initPausa
fimPausa:   MOV         R3, 010Fh
            MOV         R1, CHR_MAIS
            MOV         R2, CHR_MENOS
            CALL        drawMolReg
            POP         R3
            POP         R2
            POP         R1
            RET

;===============================================================
; <PARA CIMA ESTAO AS FUNCOES ASSOCIADAS A ECRA/MEMORIA>
;
;===============================================================
;
; <PARA BAIXO ESTAO AS FUNCOES ASSOCIADAS A PERIFERICOS>
;===============================================================


;LimpaLCD função que limpa o LCD sem o desligar
;Não recebe nada nem devolve nada
LimpaLCD:       PUSH    R1
                MOV     R1,8020h ;5º bit a 1 para limpar
                MOV     M[LCD_CONTROL],R1
                POP     R1
                RET


;Escreve_LCD : função que vai escrever algo na posição do lcd
;Recebe por R1 o caracter a escrever
;Recebe por R2 a posição a escrever , atenção que o bit mais
;significativo tem de estar sempre a um(LCD ligado)
;e o 5 nunca a um (limpa LCD)
;Não altera nenhum dos registos que recebe
;Não devolve nada
Escreve_LCD:    PUSH    R1
                PUSH    R2
                MOV     M[LCD_CONTROL],R2
                MOV     M[LCD_WRITE],R1
                POP     R2
                POP     R1
                RET
```

```
;Inicia_LCD: função que inicia o lcd e escreve Tempo max, J1 e J2
;Não recebe nada
;Não devolve nada
Inicia_LCD:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R4,FIM_STR
                MOV     R2, LCD_L0
                ;R2 <- Valor do controlo lcd
                MOV     R3,TEMPOMAX
                ;R3 <- Posição de memoria da string
                MOV     R1,M[R3]
                ;R1 <- Caracter da string
IniLCDEscTemp:  CALL    Escreve_LCD
                ;Escreve no LCD
                INC     R2
                ;Inc R2 para escrever no espaço asseguir
                INC     R3
                ;Inc R3 para a posição do char a seguir da string
                MOV     R1,M[R3]
                ;Mov para R1 o caracter
                CMP     R1,R4
                ;Compara se R1 = @
                BR.NZ   IniLCDEscTemp
                ;caso nao seja igual, repete o ciclo
                MOV     R2,LCD_L1
                MOV     R3,PONT_JOG
                MOV     R1,M[R3]
IniLCDEscJ1J2:  CALL    Escreve_LCD
                ;Escreve no LCD
                INC     R2
                ;Inc R2 para escrever no espaço asseguir
                INC     R3
                ;Inc R3 para a posição do char a seguir da string
                MOV     R1,M[R3]
                CMP     R1,R4
                BR.NZ   IniLCDEscJ1J2
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

;Função que incrementa em decimal
;Recebe em R1 o numero a incrementar
;Devolve em R1 o numero já incrementado
;PS: se incrementar 9999 devolve R1 a 0
IncDeci:        PUSH    R2
                MOV     R2,R1
                SHL     R2,12
                SHR     R2,12
                INC     R2
                CMP     R2,000Ah
                BR.NZ   IncDeci1
                SUB     R1,0009h
                MOV     R2,R1
                SHL     R2,8
                SHR     R2,12
                INC     R2
                CMP     R2,000Ah
                BR.NZ   IncDeci2
```



```
                SUB     R1,0090h
                MOV     R2,R1
                SHL     R2,4
                SHR     R2,12
                INC     R2
                CMP     R2,000Ah
                BR.NZ   IncDeci3
                SUB     R1,0900h
                MOV     R2,R1
                SHR     R2,12
                INC     R2
                CMP     R2,000Ah
                BR.NZ   IncDeci4
                MOV     R1,R0
                BR      IncDeciFim
IncDeci1:       INC     R1
                BR      IncDeciFim
IncDeci2:       ADD     R1,0010h
                BR      IncDeciFim
IncDeci3:       ADD     R1,0100h
                BR      IncDeciFim
IncDeci4:       ADD     R1,1000h
                BR      IncDeciFim
IncDeciFim:     POP     R2
                RET


;Função que converte um numero decimal para ascii
;Recebe por R1 o numero de 0 a 9 ou seja 000x , x= numero
;0 em ascii tem o valor 48
;Devolve em R1 o numero em ascii
DecToAscii:     ADD     R1,48
                RET

;Função que vai escrever no lcd o
;tempo max, a pontuação do J1 e J2
;Vai buscar à memoria os valores a escrever
;Não recebe nada
;Não devolve nada

ActualizaLCD:   CALL    ActualizaLCDT
                CALL    ActualizaLCDJ
                RET


;Função que actualiza o temp max do lcd
ActualizaLCDT:  PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3,M[TempJ1J2]
                MOV     R1,R3
                ;R1 <-- o valor do tempo maximo
                SHR     R1,12
                ;R1 <-- o digito mais significativo de R1 (10^3)
                CALL    DecToAscii
                ;Converte para ascii R1
                MOV     R2,LCD_TEMP
                ;R2 <-- Endereço para começar a escrever
                CALL    Escreve_LCD
                ;Escreve no LCD
                MOV     R1,R3
                ;R1 <-- o valor do tempo maximo
```

```
                SHL     R1,4
                ;transformações para ficar no R1
                ;só o numero pretendido (centenas)
                SHR     R1,12
                CALL    DecToAscii
                ;converte o numero para ascii
                INC     R2
                ;Incrementa R2 , posição onde vai escrever
                CALL    Escreve_LCD
                ;repete para os 4 numeros
                MOV     R1,R3
                SHL     R1,8
                ;transformações para ficar com as dezenas
                SHR     R1,12
                CALL    DecToAscii
                INC     R2
                CALL    Escreve_LCD
                MOV     R1,R3
                SHL     R1,12
                ;transformações para ficar com as unidades
                SHR     R1,12
                CALL    DecToAscii
                INC     R2
                CALL    Escreve_LCD
                POP     R3
                POP     R2
                POP     R1
                RET

;Função que actualiza no LCD pontuação dos jogadores
;Se um dos jogadores chegar aos 99, a pontuação retorna a 0
ActualizaLCDJ:  PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3,TempJ1J2
                ;R3 <-- Sitio de memoria onde tao
                ;os tempos e as pontuações
                INC     R3
                ;R3 <-- Agora aponta para a pontuação
                ; do jogador um
                MOV     R1,M[R3]
                ;R1 <-- numero a escrever
                SHL     R1,8
                ;Transformações para ficar so com as dezenas
                SHR     R1,12
                CALL    DecToAscii
                ;converte para ascii
                MOV     R2,LCD_J1
                ;R2 <-- Sitio para escrever o numero
                CALL    Escreve_LCD
                ;Escreve no LCD
                MOV     R1,M[R3]
                ;Move para R1 outra vez o numero
                SHL     R1,12
                ;transformações para ficar so
                ;com o digito das unidades
                SHR     R1,12
                CALL    DecToAscii
                ;converte para ascii
                INC     R2
                ;Incrementa a posição a escrever
                CALL    Escreve_LCD
```



```
                ;Escreve no LCD
                INC     R3
                ;R3 agora aponta para a pontuação do jogador 2
                MOV     R1,M[R3]
                ;R1 <-- a pontuação do jogador 2
                SHL     R1,8
                ;Transformações para ficar com as dezenas
                SHR     R1,12
                CALL    DecToAscii
                ;converte para ascii
                MOV     R2,LCD_J2
                ;R2 <-- posição para escrever no LCD
                CALL    Escreve_LCD
                ;Escreve
                MOV     R1,M[R3]
                SHL     R1,12
                ;Transformações para ficar com as unidades
                SHR     R1,12
                CALL    DecToAscii
                INC     R2
                ;prox sitio para escrever no LCD
                CALL    Escreve_LCD
                POP     R3
                POP     R2
                POP     R1
                RET


;Escreve os numeros de R1 no display 7 segmentos
;Recebe em R1 o valor a escrever no display
;Nao altera nada
EscreveD7S:     PUSH    R2
                MOV     R2,R1 ;digito menos significativo é o que é escrito
                ;logo nao precisa de ser modificado
                MOV     M[D7S0],R2
                SHL     R2,8
                SHR     R2,12
                MOV     M[D7S1],R2
                MOV     R2,R1
                SHL     R2,4
                SHR     R2,12
                MOV     M[D7S2],R2
                MOV     R2,R1
                SHR     R2,12
                MOV     M[D7S3],R2
                POP     R2
                RET

;===============================================================
; <PARA CIMA ESTAO AS FUNCOES ASSOCIADAS A PERIFERICOS>
;
;===============================================================
;
; <PARA BAIXO ESTAO AS FUNCOES DE CINEMATICA DE PARTICULAS>
;===============================================================

;Função chamada quando é pressionado o botao I0
;Função que muda o vector do jogador um para a esquerda
;       ^1
;<- 2       0->
;       \/3
J1Esquerda:     CMP     M[MOV_J1], R0
```

```
                BR.NZ   J1Esquerda2
                INC     R5
                PUSH    R1
                MOV     R1, 1
                MOV     M[MOV_J1], R1
                POP     R1
                CMP     R5,0004h
                BR.Z    J1Esquerda1
                ENI     ;teste
                RTI
J1Esquerda1:    MOV     R5,R0
J1Esquerda2:    ENI     ;teste
                RTI

;Função que é chamada quando o botao IB é pressionado
;Função que muda o vector do jogador um para a direita
J1Direita:      CMP     M[MOV_J1], R0
                BR.NZ   J1Direita2
                DEC     R5
                PUSH    R1
                MOV     R1, 1
                MOV     M[MOV_J1], R1
                POP     R1
                CMP     R5,FFFFh
                ;FFFF = -1, em complemento para dois
                BR.Z    J1Direita1
                ENI     ;teste
                RTI
J1Direita1:     MOV     R5,0003h
J1Direita2:     ENI     ;teste
                RTI

;Função que é chamada quando o botao I7 é pressionado
;Função que muda o vector do jogador dois para a esquerda
J2Esquerda:     CMP     M[MOV_J2], R0
                BR.NZ   J2Esquerda2
                INC     R7
                PUSH    R1
                MOV     R1, 1
                MOV     M[MOV_J2], R1
                POP     R1
                CMP     R7,0004h
                BR.Z    J2Esquerda1
                ENI     ;teste
                RTI
J2Esquerda1:    MOV     R7,R0
J2Esquerda2:    ENI     ;teste
                RTI



;Função que é chamada quando o botao I9 é pressionado
;Função que muda o vector do jogador dois para a direita
J2Direita:      CMP     M[MOV_J2], R0
                BR.NZ   J2Direita2
                DEC     R7
                PUSH    R1
                MOV     R1, 1
                MOV     M[MOV_J2], R1
                POP     R1
                CMP     R7,FFFFh
                ;FFFF = -1, em complemento para dois
```



```
                BR.Z    J2Direita1
                ENI     ;teste
                RTI
J2Direita1:     MOV     R7,0003h
J2Direita2:     ENI     ;teste
                RTI

;Funcao que gera um delay com o valor em periodos de 1/10s
;Entra em R2 o numero de periodos de decimas de segundo para o
;delay a gerar.
;ALTERA R2 PARA ZERO
delay:          PUSH    R1
delayciclo:     MOV     R1,0
                PUSH    R1
                CALL    contaSegundos
                MOV     R1,1
                MOV     M[TEMP_ADDR], R1
                MOV     M[TEMP_ONOFF], R1
                POP     R1
esperaFimDelay: ENI ;teste
                CMP     R1, 1
                BR.NZ   esperaFimDelay
                DEC     R2
                CMP     R2,R0
                BR.NZ   delayciclo
                POP     R1
                RET

;Funcao que acaba com o delay gerado pela funcao homonima
;e chamada quando ha uma interrupcao gerada pelo temporizador
;   MUDA R1!
acabaDelay:     MOV     R1, 1
                RTI

;Funcao que gera um delay dependendo do nivel de dificuldade
;Recebe o tempo actual por R3
delayDificuldade:   PUSH    R1
                    PUSH    R2
                    JMP     delayDifInicio
delayDifiN0:        MOV     R1, 0000h
                    MOV     M[LED_PORT], R1
                    MOV     R2, 7
                    CALL    delay
                    JMP     fimDifi
delayDifiN1:        MOV     R1, 000Fh
                    MOV     M[LED_PORT], R1
                    MOV     R2, 5
                    CALL    delay
                    JMP     fimDifi
delayDifiN2:        MOV     R1, 00FFh
                    MOV     M[LED_PORT], R1
                    MOV     R2, 3
                    CALL    delay
                    JMP     fimDifi
delayDifiN3:        MOV     R1, 0FFFh
                    MOV     M[LED_PORT], R1
                    MOV     R2, 2
                    CALL    delay
                    JMP     fimDifi
delayDifiN4:        MOV     R1, FFFFh
                    MOV     M[LED_PORT], R1
                    MOV     R2, 1
```

```
                    CALL    delay
                    JMP     fimDifi
delayDifInicio:     CMP     R3, 60h
                    JMP.NN  delayDifiN4
                    CMP     R3, 40h
                    JMP.NN  delayDifiN3
                    CMP     R3, 20h
                    JMP.NN  delayDifiN2
                    CMP     R3, 10h
                    JMP.NN  delayDifiN1
                    CMP     R3, R0
                    JMP.NN  delayDifiN0
fimDifi:            CALL    pausa
                    POP     R2
                    POP     R1
                    RET

;Funcao que aplica o vetor que recebe por R1
;a coordenada que recebe pelo registo R2
;       ^1
;<- 2       0->
;       \/3
aplicaVetor:        CMP     R1, R0
                    BR.NZ   aplicaUm
                    INC     R2
                    JMP     aplicaVetorFim
aplicaUm:           CMP     R1, 1
                    BR.NZ   aplicaDois
                    SUB     R2, 100h
                    JMP     aplicaVetorFim
aplicaDois:         CMP     R1, 2
                    BR.NZ   aplicaTres
                    DEC     R2
                    JMP     aplicaVetorFim
aplicaTres:         ADD     R2, 100h
aplicaVetorFim:     RET     ;tinha ENI

;Funcao que é chamada pelo delay e adiciona 1 à DECIMAS
;calcula e compara o numero de decimas de segundo que ja passaram
;e se necessario incrementa o valor(em R3) de segundos no D7S
contaSegundos:      PUSH    R1
                    MOV     R1,DECIMAS
                    INC     M[R1]
                    PUSH    R1
                    MOV     R1,M[R1]
                    CMP     R1,0Ah
                    POP     R1
                    BR.N    contaSegFim
                    MOV     M[DECIMAS],R0
                    MOV     R1,R3
                    CALL    IncDeci
                    CALL    EscreveD7S
                    MOV     R3,R1
contaSegFim:        POP     R1
                    RET

;Funcao que testa colisao da posicao que vem de R2
;compara com valores da matriz em memoria declarada pelo simbolo
;MATRIZ
;Se nao houve colisao nao altera nada
;Se houve colisao devolve em R2 o valor 0000 (R0), marca que o jogador
;asoociado a este registo perdeu (sabemos que esse registo nunca
```



```
;pode chegar ao valor 0000h pois isso implica escrever fora da
;matriz)
testaColisoes:      PUSH    R1
                    PUSH    R3
                    MOV     R1, R2
                    CALL    converteEcraMatriz
                    ;em R1 temos coordenada de memoria da posicao
                    ;correspondente a R1 inicial do ecra
                    MOV     R3, M[R1]
                    SUB     R3, 20h ;Valor de ' '
                    BR.Z    testaFim
                    MOV     R2, R0
testaFim:           POP     R3
                    POP     R1
                    RET

;=================================================================
; <FIM DA DEFINICAO DE ROTINAS/SUBROTINAS/FUNCOES>
;=================================================================

inicio:         MOV         R6, SP_INICIAL
                MOV         SP, R6          ;Inicializa a pilha
                MOV         R6, INT_MASK_JOGO
                ;Inicializa a mascara de interrupcoes com I1
                MOV         M[INT_MASK_ADDR], R6
                ;inicializa o porto de controle da janela de txto
                MOV         R3, TXT_CTRL_INIT
                ;no simulador apaga toda a janela de texto
                ;, na placa nao ATENCAO!!
                MOV         M[TXT_CTRL], R3

                MOV         R1, 0000h
                MOV         M[LED_PORT], R1
                CALL        EscreveD7S

                CALL        limpaEcra
                CALL        drawInicio
                CALL        LimpaLCD
                CALL        Inicia_LCD          ;Inicializa LCD
                ;Actualiza o LCD (tempo max, pontos jogador um e
                ;pontos jogador dois, que começam a zero
                CALL        ActualizaLCD
                MOV         M[MENU],R0
                ENI
NaoComecou:     BR          NaoComecou

;=================================================================
; <FUNCAO MAIN/JOGO>
;=================================================================

;Função que é chamada quando o botao I1 é pressionado
;ve se há jogo a decorrer e decide o que faz
;se M[MENU] == 0 , não há jogo a decorrer e aponta para as
;inicializações
;Se M[MENU] != 0 dá return Interrupt e volta para onde estava
MenuRedirect:   PUSH        R1
                MOV         R1,MENU
                CMP         M[R1],R0
                POP         R1
                JMP.Z       jogo
                RTI
```

```
jogo:           MOV         R3,1
                MOV         M[MENU],R3
                ;Indica que há um jogo a decorrer

                CALL        escreveMatriz   ;desenha campo em mem
                CALL        desenhaMolduraEParticulas ;campo ecra

                MOV         R1, R0
                MOV         M[LED_PORT], R1
                CALL        EscreveD7S  ;limpa leds just in case

                ;As direccoes iniciais definidas abaixo seguem
                ;valores definidos pelo tipo:
                ;       ^1
                ;<- 2       0->
                ;       \/3
                MOV         R3, R0      ;R3 = Tempo decorrido
                MOV         R4, 0C18h   ;R4 = Pos. Inicial do J1
                MOV         R5, R0      ;R5 = Vetor Inicial J1
                MOV         R6, 0C37h   ;R6 = Pos. Inicial do J2
                MOV         R7, 2       ;R7 = Vetor Inicial J2


                ;Jogada do jogador 1 (X)
JogadaP1:       DSI
                MOV         M[MOV_J1], R0
                MOV         M[MOV_J2], R0
                MOV         R1, R5      ;R1 tem Vetor inicial J1
                MOV         R2, R4      ;R2 tem Pos. inicial J1
                CALL        aplicaVetor
                CALL        testaColisoes
                MOV         R4, R2
                CMP         R4, R0
                BR.Z        JogadaP2 ;Se perdeu, saltar para J2
                    ;desenha no ecra a jogada do P1
                MOV         M[TXT_CTRL], R4
                MOV         R2, CHR_PLAYER1
                MOV         M[TXT_WRITE], R2
                    ;desenha na memoria a jogada do P1
                MOV         R1, R4
                CALL        converteEcraMatriz
                MOV         M[R1], R2
                ;Se este jogador perdeu:
                ;R4 = R0


                ;Jogada do jogador 2 (#)
JogadaP2:       MOV         R1, R7      ;R1 tem Vetor inicial J2
                MOV         R2, R6      ;R2 tem Pos. inicial J2
                CALL        aplicaVetor
                CALL        testaColisoes
                MOV         R6, R2
                CMP         R6, R0
                BR.Z        condicoesFim
                CMP         R4,R0
                BR.Z        condicoesFim
                    ;desenha no ecra a jogada do P2
                MOV         M[TXT_CTRL], R6
                MOV         R2, CHR_PLAYER2
                MOV         M[TXT_WRITE], R2
                    ;desenha na memoria a jogada do P2
```



```
                MOV         R1, R6
                CALL        converteEcraMatriz
                MOV         M[R1], R2
                ;Se este jogador perdeu:
                ;R6 = R0


                ;Ninguem perdeu? Mais uma jogada
                ENI
                CALL        delayDificuldade
                JMP         JogadaP1

                ;Alguem perdeu? Quem? Quem damos pontos?
                ;Esta aqui em baixo.
condicoesFim:   DSI
                CMP         R4, R0
                BR.NZ       condicoesFimP2
                PUSH        R4
                MOV         R4, TempJ1J2
                ADD         R4, 2
                ;R4 aponta para os pontos do P2
                ;Como P1 perdeu queremos dar pontos a P2
                PUSH        R1
                MOV         R1, M[R4]
                CALL        IncDeci
                MOV         M[R4], R1
                POP         R1
                POP         R4
condicoesFimP2: CMP         R6, R0
                BR.NZ       FimDesteJogo
                PUSH        R6
                MOV         R6, TempJ1J2
                INC         R6
                ;R6 aponta para os pontos do P1
                ;Como P2 perdeu queremos dar pontos a P1
                PUSH        R1
                MOV         R1, M[R6]
                CALL        IncDeci
                MOV         M[R6], R1
                POP         R1
                POP         R6


                ;Este jogo em particular acabou. Ja demos pontos.
FimDesteJogo:   CMP         R3, M[TempJ1J2]
                BR.N        FimDesteJogo1
                MOV         M[TempJ1J2], R3

FimDesteJogo1:  CALL        ActualizaLCD
                CALL        drawFim
                MOV         M[MENU],R0
                ;Indica que não há jogo a decorrer
                ENI
Fim:            BR          Fim
```