TÉCNICO LISBOA

ARQUITETURA DE COMPUTADORES

TRABALHO DE LABORATÓRIO IV

PROGRAMAÇÃO ASSEMBLY

Relatório realizado pelos alunos do curso MEEC:
81048 – Miguel Duarte Serrão Morato Moreira
81293 – Pedro António Gomes Duarte Coimbra

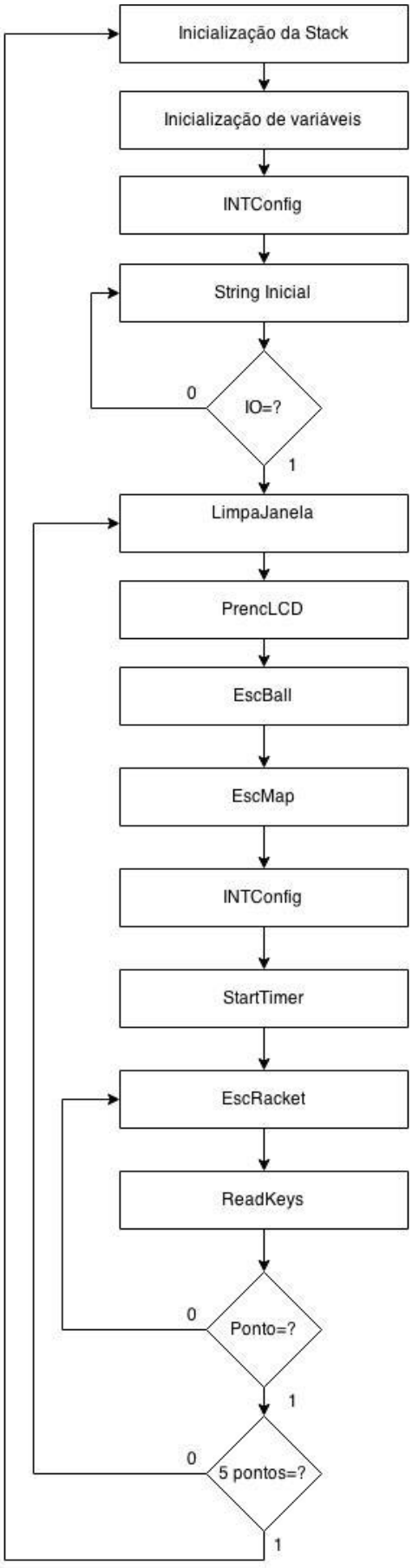
Laboratório L4:
18-05-2015
25-05-2015
Turno de laboratório: Segunda-Feira na Sala LSD1
Professor responsável: Paulo Lopes

Neste trabalho de laboratório foi-nos proposto desenvolver o jogo Pong com o objetivo de compreender a metodologia usada no desenvolvimento de programas em assembly, incluindo o uso de periféricos e de rotinas de interrupção.

Todas as rotinas que foram criadas de forma a responder aos pontos do enunciado de laboratório encontram-se apresentadas no Anexo a este relatório, juntamente com os comentários devidos. Enumeramo-las também a seguir, tendo cada uma sido apresentada com a correspondente descrição:

- LimpaJanela: Rotina que limpa a janela de texto;
- WaitStart: Rotina que aguarda pelo começo/recomeço do jogo;
- INTConfig: Rotina que permite a definição e configuração de uma rotina de interrupção do temporizador e habilitar interrupções de temporizador e pelo botão I/O na mascara de interrupções;
- StartTimer: Rotina que define unidade de contagem e inicia o temporizador do P3;
- Pause: Rotina que define a pausa do jogo;
- MarcacaoJ1: Função responsável por contar e marcar a pontuacao do jogador 1;
- MarcacaoJ2: Função responsável por contar e marcar a pontuação do jogador 2;
- Victory1: Função responsável por apresentar mensagem de vitória para o jogador 1 e reiniciar o jogo;
- Victory2: Função responsável por apresentar mensagem de vitória para o jogador 2 e reiniciar o jogo;
- EscLCD: Função responsável pela escrita no LCD;
- RefreshTime: Função responsável pelo tratamento do tempo ao longo do jogo. Assim incrementa o contador do valor do tempo, converte esse valor em segundos, depois em minutos e, por fim, separa as dezenas e unidades dos minutos e segundos escrevendo-os (atualizando-os) no ecrã LCD;
- EscBall: Rotina que desenha a bola ao início;
- RamdomGen: Rotina que cria um número aleatório entre 0 e M-1;
- EscString: Rotina que efetua a escrita de uma cadeia de caracter, terminada pelo caracter FIM_TEXTO, na janela de texto numa posição especificada. Pode-se definir como terminador qualquer caracter ASCII;
- PrencLCD: Função responsável por escrever a mensagem inicial no LCD. Entende-se por inicial a parte constante da pontuação (J1: e J2:) e a parte variável da pontuação (que nesta função inicializamos a zero);
- EscRacket: Rotina responsável por escrever as raquetes no ecrã;
- EscMap: Rotina responsável por escrever os caracteres que representam o chão, paredes e teto no jogo;
- ReadKeys: Verifica se algum dos jogadores premiu uma tecla;
- INTTemp: Rotina que representa o movimento da bola ao longo da área de jogo e as colisões quer com as raquetes dos jogadores, quer com as paredes horizontais (de onde reflete) e verticais (de forma a marcar pontos.

O fluxograma que descreve o funcionamento do programa principal encontra-se representado a seguir:



CONCLUSÃO

Neste trabalho contactámos novamente com o processador P3 e com a linguagem de programação de baixo-nível Assembly. Em comparação com o laboratório anterior este trabalho foi mais complexo e exigiu mais de nós.

Permitiu-nos a familiarização com o uso de periféricos e rotinas de interrupção, usadas pela primeira vez neste trabalho de laboratório.

Apesar das dificuldades que foram ocorrendo durante as duas semanas, conseguimos completar o jogo Pong, cumprindo todos os objetivos enunciados no guia de laboratório.

Na aula relativa à primeira semana, tal como previsto, tirámos dúvidas com o professor e na segunda semana demonstrámos os resultados obtidos ao docente. Todo o trabalho foi demonstrado com sucesso à exceção do display de 7 segmentos uma vez que não tínhamos implementado essa funcionalidade. No anexo deste relatório segue o código , agora completo, onde é possível verificar a resposta a todos os pontos pedidos no enunciado, incluindo o temporizador no display de 7 segmentos que conseguimos implementar com sucesso.

ANEXO

```
;=================================================================================
; Laboratório 4
; Jogo Pong
;
; Descricao: Implementacao de uma versao do jogo Pong em Assembly para o processador P3.
;           Trata-se de um jogo de dois jogadores em que o objectivo e pontuar. Para alem
;           do controlo das raquetes pelos dois jogadores esta versao permite a pausa do
;           jogo sendo tambem implementados contadores do tempo de jogo decorrido e da
;           pontuacao. O primeiro jogador a atingir os 5 pontos ganha o jogo podendo este
;           ser reiniciado atraves do botao de pressao IO.
;
; Autores: Miguel Moreira e Pedro Coimbra
;=================================================================================

;=================================================================================
; ZONA I: Definicao de constantes
;         Pseudo-instrucao : EQU
;=================================================================================

;STACK POINTER
SP_INICIAL      EQU     FDFFh

;TEMPORIZADOR
TempValor       EQU     FFF6h
TempControlo    EQU     FFF7h
Time            EQU     1

;INTERRUPCOES
TAB_INT0        EQU     FE00h
TAB_INT1        EQU     FE01h
TAB_INTA        EQU     FE0Ah
TAB_INTTemp     EQU     FE0Fh
MASCARA_INT     EQU     FFFAh

;I/O a partir de FF00h
DISP7S1         EQU     FFF0h
DISP7S2         EQU     FFF1h
DISP7S3         EQU     FFF2h
DISP7S4         EQU     FFF3h
LCD_WRITE       EQU     FFF5h
LCD_CURSOR      EQU     FFF4h
LEDS            EQU     FFF8h
INTERRUPTORES   EQU     FFF9h
IO_CURSOR       EQU     FFFCh
IO_TESTE        EQU     FFFDh
IO_WRITE        EQU     FFFEh
IO_READ         EQU     FFFFh
MASK            EQU     8401h
LIMPAR_JANELA   EQU     FFFFh
XY_INICIAL      EQU     0616h
XY_EXTRA        EQU     061Dh
FIM_TEXTO       EQU     '@'

;Variaveis Globais
LRacket         EQU     F0F0h
RRacket         EQU     F0F1h
BPosit          EQU     F0F2h
BDirec          EQU     F0F3h
EstGame         EQU     F0F4h
PontJ1          EQU     F0F8h
PontJ2          EQU     F0F9h
Tempo           EQU     F0FAh

;Variaveis para calculo de numeros aleatorios
Seed            EQU     F0F5h
Mascara         EQU     9C16h
```

```
;=================================================================================
; ZONA II: Definicao de variaveis
;          Pseudo-instrucoes : WORD - palavra (16 bits)
;                              STR  - sequencia de caracteres.
;          Cada caracter ocupa 1 palavra
;=================================================================================
                ORIG    8000h
VarTexto1       STR     '** Prima I0 para iniciar o jogo **',FIM_TEXTO
VarTexto2       STR     '** Ganhou o Jogador 1 **',FIM_TEXTO
VarTexto3       STR     '** Ganhou o Jogador 2 **',FIM_TEXTO

;=================================================================================
; ZONA III: Codigo
;           conjunto de instrucoes Assembly, ordenadas de forma a realizar
;           as funcoes pretendidas
;=================================================================================
                ORIG    0000h
                JMP     Inicio

;=================================================================================
; LimpaJanela: Rotina que limpa a janela de texto.
;              Entradas: --
;              Saidas: ---
;              Efeitos: ---
;=================================================================================
LimpaJanela:    PUSH    R2
                MOV     R2, LIMPAR_JANELA
                MOV     M[IO_CURSOR], R2
                POP     R2
                RET

;=================================================================================
; WaitStart:    Rotina que aguarda pelo comeco/recomeco do jogo
;=================================================================================
WaitStart:      MOV     R1, M[EstGame]
                INC     R1
                MOV     M[EstGame], R1
                RTI

;=================================================================================
;INTConfig: Rotina que permite a definicao e configuracao de uma rotina de interrupcao do temporizador e
;           habilitar interrupcoes de temporizador e pelo botao I/O na mascara de interrupcoes
;=================================================================================
INTConfig:      PUSH    R1
                MOV     R1, WaitStart
                MOV     M[TAB_INT0], R1       ; (Inicializacao da TVI) Escrita do endereco da rotina de
interrupcao que
                MOV     R1, INTTemp           ;incrementa o estado do jogo de forma a inicia-lo
                MOV     M[TAB_INTTemp], R1    ; (Inicializacao da TVI)Escrita do endereco da rotina de
interrupcao que representa
                MOV     R1, Pause             ;o jogo na tabela de vectores de interrupcao na posicao
FEOFh (a do temporizador)
                MOV     M[TAB_INTA], R1       ; (Inicializacao da TVI)Escrita do endereco da rotina de
interrupcao que representa
                MOV     R1, MASK        ;a pausa na tabela de vectores de interrupcao na posicao FEOAh (a
do botao IA)
                MOV     M[MASCARA_INT], R1    ; (Inicializacao da Mascara de Interrupcoes) Permite apenas
a interrupcao 15
                POP     R1                    ;(Temporizador)
                RET

;=================================================================================
;StartTimer:  Rotina que define unidade de contagem e inicia o temporizador do P3
;=================================================================================
StartTimer:     PUSH    R1
                MOV     R1, Time
                MOV     M[TempValor], R1        ; Inicializacao do temporizador do P3 com a definicao de
1 unidade de contagem
```

```
                MOV     R1, 1                   ;de 100ms (ou seja 0,1s) de forma a definir periodo de
deslocamento da bola
                MOV     M[TempControlo], R1     ; Instrucao que da inicio ao temporizador do P3 com a
escrita do valor '1'
                POP     R1                      ;no porto FFF7h
                RET

;================================================================================
;Pause:  Rotina que define a pausa do jogo
;================================================================================
Pause:          PUSH    R1
                MOV     R1, M[TempControlo]
                INC     R1
                MOV     M[TempControlo], R1
                POP     R1
                RTI

;================================================================================
;MarcacaoJ1:  Funcao responsavel por contar e marcar a pontuacao do jogador 1
;================================================================================
MarcacaoJ1:     PUSH    R1
                MOV     R1, M[PontJ1]
                INC     R1
                MOV     M[PontJ1], R1
                CMP     R1, 5
                POP     R1
                JMP.Z   Victory1
                JMP     RestartGame


;================================================================================
;MarcacaoJ2:  Funcao responsavel por contar e marcar a pontuacao do jogador 2
;================================================================================
MarcacaoJ2:     PUSH    R1
                MOV     R1, M[PontJ2]
                INC     R1
                MOV     M[PontJ2], R1
                CMP     R1, 5
                POP     R1
                JMP.Z   Victory2
                JMP     RestartGame

;================================================================================
;Victory1:  Funcao responsavel por apresentar mensagem de vitoria para o jogador 1 e
;           reiniciar o jogo
;================================================================================
Victory1:       PUSH    R1
                MOV     R1, 0F00h
                CALL    LimpaJanela
                PUSH    VarTexto2
                PUSH    XY_EXTRA
                CALL    EscString
                PUSH    5
                PUSH    8005h
                CALL    EscLCD
Wait1:          DEC     R1
                CMP     R1, 0000h
                JMP.NZ  Wait1
                POP     R1
                JMP     Inicio

;================================================================================
;Victory2:   Funcao responsavel por apresentar mensagem de vitoria para o jogador 2 e
;           reiniciar o jogo
;================================================================================
Victory2:       PUSH    R1
```

```
                MOV     R1, 0F00h
                CALL    LimpaJanela
                PUSH    VarTexto3
                PUSH    XY_EXTRA
                CALL    EscString
                PUSH    5
                PUSH    8015h
                CALL    EscLCD
Wait2:          DEC     R1
                CMP     R1, 0000h
                JMP.NZ  Wait2
                POP     R1
                JMP     Inicio

;================================================================================
;EscLCD: Funcao responsavel pela escrita no LCD.
;               Entradas: pilha - posicao para escrita do primeiro carater
;                         pilha - caracter a escrever no LCD
;               Saidas: ---
;================================================================================
EscLCD:         PUSH    R1
                PUSH    R2
                MOV     R1, M[SP+4]
                MOV     R2, M[SP+5]
                MOV     M[LCD_CURSOR], R1
                ADD     R2, 0030h
                MOV     M[LCD_WRITE], R2
                POP     R2
                POP     R1
                RETN    2

;================================================================================
;RefreshTime:  Funcao responsavel pelo tratamento do tempo ao longo do jogo. Assim
;             incrementa o contador do valor do tempo, converte esse valor em segundos, depois
;             em minutos e, por fim, separa as dezenas e unidades dos minutos e segundos,
;             escrevendo-os (actualizando-os) no ecra LCD
;================================================================================
RefreshTime:    PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R1, M[Tempo]            ; Carrega contagem atual do tempo
                INC     R1                      ; Incrementa variavel que vai representar o tempo de
jogo
                MOV     M[Tempo], R1            ; Devolve o novo valor da contagem
                MOV     R2, 10
                DIV     R1, R2                  ; Divide por 10 de forma a ficar com numero de segundos
passados em R2
                MOV     R2, 60
                DIV     R1, R2                  ; Divide por 60 de forma a ficar com o numero de
minutos em R1 e de segundos
                MOV     R3, 10                  ;decorridos em R2
                MOV     R4, 10
                DIV     R1, R3                  ; Divisao por 10 permite ficar com digito das unidades
dos minutos em R3
                MOV     M[DISP7S4], R1          ;e com o digito das dezenas de minutos em R1
                MOV     M[DISP7S3], R3
                DIV     R2, R4                  ; Divisao por 10 permite ficar com digito das unidades
dos segundos em R4
                MOV     M[DISP7S2], R2          ;e com o digito das dezenas de minutos em R2
                MOV     M[DISP7S1], R4
                POP     R4
                POP     R3
                POP     R2
                POP     R1
```

```
                RET

;================================================================================
;EscBall:       rotina que desenha a bola ao inicio
;================================================================================
EscBall:        PUSH    R1
                PUSH    R2
                PUSH    R4
                MOV     R1, M[BPosit]        ; Carrega o valor da variavel global "Posicao da Bola"
iniciado com o valor
                MOV     R4, Ch               ; correspondente a coluna maxima onde a bola se pode
encontrar (17*256+30).
                CALL    RamdomGen            ; Em seguida carregamos o valor de M para ser utilizado pela
rotina de geracao de
                MOV     R2, 100h             ; numeros aleatorios com o valor 11 (maxima diferenca de
linhas entre posicoes onde
                MUL     R4, R2               ; a bola pode ser colocada). O produto de 100h (R2) com o
numero aleatorio calculado
                SUB     R1, R2               ; da-nos o numero de posicoes a decrementar a Bposit de
forma a bola estar entre
                MOV     R4, 14h              ; as linhas devidas.
                CALL    RamdomGen            ; Calculo de numero aleatorio entre 0 e 19 seguidamente
somado a R1, permite-nos
                ADD     R1, R4               ;somar a R1 um numero que coloca a bola entre as colunas 30
e 49 (ja que ao
                MOV     M[BPosit], R1        ;inicio todas as posicoes calculadas para R1 correspondiam à
coluna 30)
                MOV     M[IO_CURSOR], R1
                MOV     R1, 'O'              ; Valor da posicao da bola guardada com variavel global e
escrita do caracter
                MOV     M[IO_WRITE], R1      ;'O' (que representa a bola) na posicao calculada
anteriormente
                MOV     R4, 4h
                CALL    RamdomGen            ; Para calculo da direcao aleatoria da bola, calculo de
numero entre 0 e 3
                MOV     M[BDirec], R4        ;e carregamento do valor na variavel global BDirec
                POP     R4
                POP     R2
                POP     R1
                RET

;================================================================================
;RamdomGen:     Rotina que cria um numero aleatorio entre 0 e M-1
;================================================================================
RamdomGen:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, M[Seed]     ; Em cada invocacao da rotina le-se o valor Ni anterior, que e
carregado em R1
                MOV     R2, 1
                MOV     R3, Mascara
                AND     R2, R1          ; Permite verificar se o bit menos significativo de Ni e igual ou
diferente de zero
                BR.Z    Branch              ; Caso seja igual a zero saltaremos para sub-rotina que fara
um ROR do Ni anterior
                MOV     R2, R1          ; Caso seja diferente de zero, seguiremos o algoritmo fornecido no
guia e iremos
                XOR     R2, R3          ;fazer um XOR do valor de Ni com o valor da Mascara e seguidamente
faremos um
                ROR     R2, 1           ;ROR do valor obtido nesta operacao
                MOV     M[Seed], R2     ; Carregamos este valor em memoria
                BR      Finish
Branch:         MOV     R2, R1
                ROR     R2, 1
                MOV     M[Seed], R2
Finish:         MOV     R1, M[Seed]    ; Carregamos o valor de Ni calculado em R1
```

```
                DIV     R1, R4          ; Apos a operacao de DIV o valor Zi sera encontrado em R4 (resto da
divisao sendo o numero
                POP     R3              ;aleatorio pretendido). Antes da operacao R1 sera o Ni do algoritmo
e R4 (valor carregado
                POP     R2              ;antes de chamar a funcao para gerar numero aleatorio) representara
o M.
                POP     R1
                RET

;=============================================================================
; EscString: Rotina que efectua a escrita de uma cadeia de caracter, terminada
;            pelo caracter FIM_TEXTO, na janela de texto numa posicao
;            especificada. Pode-se definir como terminador qualquer caracter
;            ASCII.
;               Entradas: pilha - posicao para escrita do primeiro carater
;                         pilha - apontador para o inicio da "string"
;               Saidas: ---
;               Efeitos: ---
;=============================================================================
EscString:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R2, M[SP+6]         ; Apontador para inicio da "string"
                MOV     R3, M[SP+5]         ; Localizacao do primeiro carater
Ciclo:          MOV     M[IO_CURSOR], R3
                MOV     R1, M[R2]
                CMP     R1, FIM_TEXTO
                BR.Z    FimEsc
                MOV     M[IO_WRITE], R1
                INC     R2
                INC     R3
                BR      Ciclo
FimEsc:         POP     R3
                POP     R2
                POP     R1
                RETN    2                   ; Actualiza STACK

;=============================================================================
;PrencLCD:  Funcao responsavel por escrever a mensagem inicial no LCD.
;           Entende-se por inicial a parte constante da pontuacao (J1: e J2:) e a parte
;          variavel da pontuacao (que nesta funcao inicializamos a zero)
;=============================================================================
PrencLCD:       PUSH    26
                PUSH    8001h
                CALL    EscLCD          ; Escreve caracter "J" na primeira linha do LCD
                PUSH    1
                PUSH    8002h
                CALL    EscLCD          ; Escreve caracter "1" seguido ao caracter anterior
                PUSH    10
                PUSH    8003h
                CALL    EscLCD          ; Escreve caracter ":" seguido ao caracter anterior
                PUSH    26
                PUSH    8011h
                CALL    EscLCD          ; Escreve caracter "J" na segunda linha do LCD
                PUSH    2
                PUSH    8012h
                CALL    EscLCD          ; Escreve caracter "2" seguido ao caracter anterior
                PUSH    10
                PUSH    8013h
                CALL    EscLCD          ; Escreve caracter ":" seguido ao caracter anterior
                PUSH    R0
                PUSH    8005h
                CALL    EscLCD          ; Escreve caracter "0" na posicao que sera depois carregada
                PUSH    R0              ;com o valor da pontuacao do jogador 1
                PUSH    8015h
```

```
                CALL    EscLCD          ; Escreve caracter "0" na posicao que sera depois carregada
                RET                     ;com o valor da pontuacao do jogador 2

;================================================================================
; EscRacket:    Rotina responsavel por escrever as raquetes no ecra
;================================================================================
EscRacket:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3, M[LRacket]          ; Carregamento da variavel global LRacket (posicao do
primeiro elemento da
                MOV     R2, 5                   ;raquete da esquerda) para R3 e de 5 (numero de
caracteres da raquete) em R2
LeftRacket:     MOV     M[IO_CURSOR], R3
                MOV     R1, '#'                 ; Subrotina imprime um caracter '#' na posicao indicada
por R3, que comeca
                MOV     M[IO_WRITE], R1         ;no lugar do primeiro elemento da raquete e é
incrementado de 256 de forma
                ADD     R3, 100h                ;a representar as posicoes dos caracteres subsequentes
da raquete.
                DEC     R2                      ; Sao impressos 5 cardinais por raquete, ou seja quando
R2 chegar a zero
                BR.NZ   LeftRacket              ;(sendo decrementado a cada impressao) passamos a criar
a raquete da direita
                MOV     R3, M[RRacket]
                MOV     R2, 5                   ; Carregamento da variavel global RRacket (posicao do
primeiro elemento da
RightRacket:    MOV     M[IO_CURSOR], R3        ;raquete da direita) para R3 e de 5 (numero de
caracteres da raquete) em R2
                MOV     R1, '#'                 ; Subrotina imprime um caracter '#' na posicao indicada
por R3, que comeca
                MOV     M[IO_WRITE], R1         ;no lugar do primeiro elemento da raquete e é
incrementado de 256 de forma
                ADD     R3, 100h                ;a representar as posicoes dos caracteres subsequentes
da raquete.
                DEC     R2                      ; Sao impressos 5 cardinais por raquete, ou seja quando
R2 chegar a zero
                BR.NZ   RightRacket             ;(sendo decrementado a cada impressao) terminamos a
rotina.
                POP     R3
                POP     R2
                POP     R1
                RET

;================================================================================
; EscMap:       Rotina responsavel por escrever os caracteres que representam o chao, paredes e teto no
jogo
;================================================================================
EscMap:         PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, 0000h
                MOV     R2, 50h             ; Subrotina permite a escrita dos 80 caracteres '-' que
compoe o teto do jogo.
Roof:           MOV     M[IO_CURSOR], R1    ;R1 corresponde a variavel da posicao onde é impresso o
caracter. Este
                MOV     R3, '-'             ;vai incrementando à medida que os caracteres sao colocados.
                MOV     M[IO_WRITE], R3     ;R2 corresponde à variavel que controla o numero de
caracteres que sao necessarios
                INC     R1                  ;colocar. A cada caracter colocado é decrementada sendo que
quando chega a zero
                DEC     R2                  ;passamos a criar o chao do jogo.
                BR.Z    Floor
                BR      Roof
Floor:          MOV     R1, 1700h
                MOV     R2, 50h
```

```
Floor2:         MOV     M[IO_CURSOR], R1        ; Subrotina permite a escrita dos 80 caracteres '-' que
compoe o chao do jogo.
                MOV     R3, '-'                 ;R1 corresponde a variavel da posicao onde é impresso o
caracter. Este vai
                MOV     M[IO_WRITE], R3         ;incrementando à medida que os caracteres sao colocados
sendo iniciado a (23*256)
                INC     R1                      ;que corresponde à primeira posicao do chao.
                DEC     R2                      ;R2 corresponde à variavel que controla o numero de
caracteres que sao necessarios
                BR.Z    LeftWall                ;colocar. A cada caracter colocado é decrementada sendo que
quando chega a zero
                BR      Floor2                  ;passamos a criar a parede esquerda.
RightWall:      MOV     R1, 004Fh
                MOV     R2, 17h
RightWall2:     ADD     R1, 100h                ; Subrotina que permite a escrita dos 22 caracteres '|' que
representam a
                DEC     R2                      ;parede direita da divisao.
                BR.Z    End                     ;R1 corresponde à posicao onde o caracer sera colocado,
comecando, por isso,
                MOV     M[IO_CURSOR], R1        ;com o valor 79+256 e sendo incrementado de 256 a cada
caracter colocado, até
                MOV     R3, '|'                 ;R2 (onde esta guardado o numero de caracteres necessarios
colocar + 1, ja que
                MOV     M[IO_WRITE], R3         ;decrementamos este antes de o escrever) tomar valor 0.
                BR      RightWall2              ;Nessa altura acabamos de criar o mapa grafico de jogo
LeftWall:       MOV     R1, 0000h
                MOV     R2, 17h
LeftWall2:      ADD     R1, 100h
                DEC     R2                      ; Subrotina que permite a escrita dos 22 caracteres '|' que
representam a
                BR.Z    RightWall               ;parede esquerda da divisao.
                MOV     M[IO_CURSOR], R1        ;R1 corresponde à posicao onde o caracer sera colocado,
comecando, por isso,
                MOV     R3, '|'                 ;com o valor 256 e sendo incrementado de 256 a cada
caracter colocado, até
                MOV     M[IO_WRITE], R3         ;R2 (onde esta guardado o numero de caracteres necessarios
colocar + 1, ja que
                BR      LeftWall2               ;decrementamos este antes de o escrever) tomar valor 0,
altura em que passamos
End:            POP     R3                      ;a criar a parede direita do jogo
                POP     R2
                POP     R1
                RET

;================================================================================
; ReadKeys:     Verifica se algum dos jogadores premiu uma tecla
;================================================================================
ReadKeys:       PUSH    R1
                PUSH    R2
Read:           CMP     M[IO_TESTE], R0         ; Permite testar se houve alguma tecla premida
                BR.Z    Read
                MOV     R1, M[IO_READ]          ; Caso tenha havido guarda a ultima tecla premida em R1
                CMP     R1, 113
                BR.Z    LRUp
                CMP     R1, 97                  ; Os varios CMP's permitem verificar se foram premidas as
teclas
                BR.Z    LRDown                     ;q, a, o ou l, respectivamente e, para cada um dos casos,
                CMP     R1, 111                 ;chama a subrotina apropriada para lidar com o evento.
                BR.Z    RRUp                    ; Caso nao seja nenhuma das teclas referidas nada sera
alterado.
                CMP     R1, 108
                BR.Z    RRDown
                BR      Exit2
                                                ; Raquete esquerda para cima
LRUp:           MOV     R2, 0105h               ; Posicao mais alta possivel para raquete da esquerda
```

```
                MOV     R1, M[LRacket]          ; Move posicao do inicio da raquete para R1
                CMP     R1, R2                  ; Verifica se a raquete esta na posicao mais acima possivel
                BR.Z    Exit2               ; Nao executa movimento da raquete para cima se ja se
encontrar na posicao mais elevada
                MOV     R2, 400h
                ADD     R2, R1                  ; Adiciona 400h a R1 de forma a ficar com a posicao do fim
da raquete em R2
                SUB     R1, 100h                ; Subtrai 100h de forma a raquete subir uma posicao
                MOV     M[LRacket], R1          ; Guarda o novo valor na variavel global
                MOV     M[IO_CURSOR], R2
                MOV     R2, ' '                 ; Escreve um espaco em branco na posicao antiga do final
da raquete
                MOV     M[IO_WRITE], R2
Exit2:          BR      Exit1

RRUp:           BR      RRUp1
RRDown:         BR      RRDown2
                                                ; Raquete esquerda para baixo
LRDown:         MOV     R2, 1205h               ; Posicao mais baixa possivel para inicio da raquete da
esquerda
                MOV     R1, M[LRacket]          ; Move posicao do inicio da raquete para R1
                CMP     R1, R2                  ; Verifica se a raquete esta na posicao mais baixa possivel
                BR.Z    Exit1               ; Nao executa movimento da raquete para baixo se ja se
encontrar na posicao mais baixa
                MOV     R2, R1
                ADD     R1, 100h                ; Adiciona 100h de forma a raquete descer uma posicao
                MOV     M[LRacket], R1          ; Guarda o novo valor na variavel global
                MOV     M[IO_CURSOR], R2
                MOV     R2, ' '
                MOV     M[IO_WRITE], R2         ; Escreve um espaco em branco na posicao antiga do inicio
da raquete
Exit1:          BR      Exit3

RRDown2:        BR      RRDown1
                                                ; Raquete direita para cima
RRUp1:          MOV     R2, 014Ah               ; Posicao mais alta possivel para raquete da direita
                MOV     R1, M[RRacket]          ; Move posicao do inicio da raquete para R1
                CMP     R1, R2                  ; Verifica se a raquete esta na posicao mais alta possivel
                BR.Z    Exit3               ; Nao executa movimento da raquete para cima se ja se
encontrar na posicao mais alta
                MOV     R2, 400h
                ADD     R2, R1                  ; Adiciona 400h a R1 de forma a ficar com a posicao do fim
da raquete em R2
                SUB     R1, 100h                ; Subtrai 100h de forma a raquete subir uma posicao
                MOV     M[RRacket], R1          ; Guarda o novo valor na variavel global
                MOV     M[IO_CURSOR], R2
                MOV     R2, ' '                 ; Escreve um espaco em branco na posicao antiga do final
da raquete
                MOV     M[IO_WRITE], R2
Exit3:          BR      Exit
                                                ; Raquete direita para baixo
RRDown1:        MOV     R2, 124Ah               ; Posicao mais baixa possivel para raquete da direita
                MOV     R1, M[RRacket]          ; Move posicao do inicio da raquete para R1
                CMP     R1, R2                  ; Verifica se a raquete esta na posicao mais baixa possivel
                BR.Z    Exit                ; Nao executa movimento da raquete para baixo se ja se
encontrar na posicao mais baixa
                MOV     R2, R1
                ADD     R1, 100h
                MOV     M[RRacket], R1          ; Adiciona 100h de forma a raquete descer uma posicao
                MOV     M[IO_CURSOR], R2        ; Guarda o novo valor na variavel global
                MOV     R2, ' '
                MOV     M[IO_WRITE], R2         ; Escreve um espaco em branco na posicao antiga do inicio
da raquete
                BR      Exit
Exit:           POP     R2
```

```
                POP     R1
                RET

;================================================================================
;INTTemp:  Rotina que representa o movimento da bola ao longo da area de jogo e as colisoes quer com as
raquetes dos
;         jogadores, quer com as paredes horizontais (de onde reflete) e verticais (de forma a marcar
pontos
;================================================================================
INTTemp:        PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     M[LEDS], R0         ; Apaga os LEDS de "vitoria"
                MOV     R2, 100h            ; Variacao de posicoes correspondente a elementos adjacentes
numa coluna
                MOV     R3, 1h              ; Variacao de posicoes correspondente a elementos adjacentes
numa linha
                MOV     R1, M[BDirec]       ; Carrega a direcao de movimento actual da bola em R1
                CMP     R1, 0               ; Verifica se direcao da bola é "para cima e para a direita"
                JMP.Z   Dir1                ;caso seja chama a sub-rotina apropriada para lidar com o
movimento da bola
                CMP     R1, 1               ; Verifica se direcao da bola é "para cima e para a
esquerda" ...
                JMP.Z   Dir2
                CMP     R1, 2               ; Verifica se direcao da bola é "para baixo e para a
esquerda" ...
                JMP.Z   Dir3
                CMP     R1, 3               ; Verifica se direcao da bola é "para baixo e para a
direita" ...
                JMP.Z   Dir4
Dir1:           MOV     R4, M[BPosit]
                MOV     R5, FF00h
                AND     R4, R5              ; Verifica colisao da bola com a parede horizontal superior do
campo
                CMP     R4, 0100h           ;(caso haja colisao a direcao da bola devera ser invertida)
                JMP.Z   Reflex4
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica colisao da bola com a parede lateral direita
                CMP     R4, 004Eh           ;(caso haja deve reiniciar o jogo e marcar ponto ao jogador 1)
                JMP.Z   MarcacaoJ1
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica se podera haver colisao com a raquete direita neste
movimento
                CMP     R4, 0049h           ;caso nao possa haver sera chamada a sub-rotina para realizar
um movimento
                JMP.NZ  Refresh1                ;para cima e para a direita
                MOV     R4, M[RRacket]
                DEC     R4                  ; Representa o valor da posicao exactamente do lado esquerdo
do topo da raquete
                MOV     R1, 100h
                MOV     R5, M[BPosit]
                CMP     R5, R4              ; Verifica se a bola esta na primeira posicao lateral a
raquete e chama a subrotina
                JMP.Z   Reflex2             ;Reflex2 de forma a fazer a reflexao do movimento da bola
                ADD     R4, R1              ; Soma 100h (256) de forma a passar ao valor da segunda
posicao a contar do topo do
                CMP     R5, R4              ;lado esquerdo da raquete, compara com o valor da posicao da
bola e em caso de
                JMP.Z   Reflex2             ;igualdade chama sub-rotina Reflex2 de forma a fazer a
reflexao da bola
                ADD     R4, R1
```

```
                CMP     R5, R4              ; Mesma verificacao que as anteriores agora para a terceira
posicao lateral esquerda
                JMP.Z   Reflex2                 ;a raquete
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quarta posicao lateral esquerda à raquete
                JMP.Z   Reflex2
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quinta posicao lateral esquerda à raquete
                JMP.Z   Reflex2
                ADD     R4, R1
                CMP     R5, R4
                JMP.Z   Reflex2                 ; Verificacao para a sexta posicao lateral esquerda
(representa a esquina da raquete)
                JMP     Refresh1                ; Caso nenhuma das condicoes anteriores se verifique nao
havera nenhuma colisao com a
Reflex2:        MOV     R5, 0001h           ;raquete pelo que sera chamada a sub-rotina Refresh1 de forma
a realizar movimento normal
                MOV     M[BDirec], R5       ; Subrotina que inverte o movimento da bola para "para cima
e para a esquerda"
                JMP     Refresh2
Dir2:           MOV     R4, M[BPosit]
                MOV     R5, FF00h
                AND     R4, R5              ; Verifica colisao da bola com a parede horizontal superior
do campo
                CMP     R4, 0100h           ;(caso haja colisao a direcao da bola devera ser invertida)
                JMP.Z   Reflex3
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica colisao da bola com a parede lateral esquerda
                CMP     R4, 0001h           ;(caso haja deve reiniciar o jogo e marcar ponto ao jogador
2)
                JMP.Z   MarcacaoJ2
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica se podera haver colisao com a raquete esquerda
neste movimento
                CMP     R4, 0006h           ;caso nao possa haver sera chamada a sub-rotina para
realizar um movimento
                JMP.NZ  Refresh2                ;para cima e para a esquerda
                MOV     R4, M[LRacket]
                INC     R4                  ; Representa o valor da posicao exactamente do lado direito
do topo da raquete
                MOV     R1, 100h
                MOV     R5, M[BPosit]
                CMP     R5, R4              ; Verifica se a bola esta na primeira posicao lateral a
raquete e chama a subrotina
                JMP.Z   Reflex1                 ;Reflex1 de forma a fazer a reflexao do movimento da bola
                ADD     R4, R1              ; Soma 100h (256) de forma a passar ao valor da segunda
posicao a contar do topo do
                CMP     R5, R4              ;lado direito da raquete, compara com o valor da posicao da
bola e em caso de
                JMP.Z   Reflex1                 ;igualdade chama sub-rotina Reflex1 de forma a fazer a
reflexao da bola
                ADD     R4, R1
                CMP     R5, R4              ; Mesma verificacao que as anteriores agora para a terceira
posicao lateral direita
                JMP.Z   Reflex1                 ;a raquete
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quarta posicao lateral direita à raquete
                JMP.Z   Reflex1
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quinta posicao lateral direita à raquete
                JMP.Z   Reflex1
                ADD     R4, R1
```

```
                CMP     R5, R4
                JMP.Z   Reflex1                 ; Verificacao para a sexta posicao lateral direita
(representa a esquina da raquete)
                JMP     Refresh2              ; Caso nenhuma das condicoes anteriores se verifique nao
havera nenhuma colisao com a
Reflex1:        MOV     R5, 0000h             ;raquete pelo que sera chamada a sub-rotina Refresh2 de
forma a realizar movimento normal
                MOV     M[BDirec], R5         ; Subrotina que inverte o movimento da bola para "para cima
e para a direita"
                JMP     Refresh1
Dir3:           MOV     R4, M[BPosit]
                MOV     R5, FF00h
                AND     R4, R5               ; Verifica colisao da bola com a parede horizontal inferior
do campo
                CMP     R4, 1600h            ;(caso haja colisao a direcao da bola devera ser invertida)
                JMP.Z   Reflex2
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5               ; Verifica colisao da bola com a parede lateral esquerda
                CMP     R4, 0001h            ;(caso haja deve reiniciar o jogo e marcar ponto ao jogador 2)
                JMP.Z   MarcacaoJ2
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica se podera haver colisao com a raquete esquerda
neste movimento
                CMP     R4, 0006h           ;caso nao possa haver sera chamada a sub-rotina para realizar
um movimento
                JMP.NZ  Refresh3                ;para esquerda e para baixo
                MOV     R4, M[LRacket]
                INC     R4                  ; Representa o valor da posicao exactamente do lado direito do
topo da raquete
                MOV     R1, 100h
                SUB     R4, R1              ; Posicao superior a do lado direito do topo da raquete para
verificar colisao c/esquina
                MOV     R5, M[BPosit]
                CMP     R5, R4              ; Verifica se a bola esta na esquina superior da raquete e
chama a subrotina
                JMP.Z   Reflex4                 ;Reflex4 de forma a fazer a reflexao do movimento da bola
                ADD     R4, R1              ; Soma 100h (256) de forma a passar ao valor da segunda
posicao a contar do topo do
                CMP     R5, R4              ;lado direito da raquete, compara com o valor da posicao da
bola e em caso de
                JMP.Z   Reflex4                 ;igualdade chama sub-rotina Reflex4 de forma a fazer a
reflexao da bola
                ADD     R4, R1
                CMP     R5, R4              ; Mesma verificacao que as anteriores agora para a terceira
posicao lateral direita
                JMP.Z   Reflex4                 ;a raquete
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quarta posicao lateral direita à raquete
                JMP.Z   Reflex4
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quinta posicao lateral direita à raquete
                JMP.Z   Reflex4
                ADD     R4, R1
                CMP     R5, R4               ; Verificacao para a sexta posicao lateral direita;
                JMP.Z   Reflex4                 ; Caso nenhuma das condicoes anteriores se verifique nao
havera nenhuma colisao com a
                JMP     Refresh3            ;raquete pelo que sera chamada a sub-rotina Refresh3 de forma
a realizar movimento normal
Reflex4:        MOV     R5, 0003h
                MOV     M[BDirec], R5        ;  Subrotina que inverte o movimento da bola  para "para
baixo e para a direita"
                JMP     Refresh4
Dir4:           MOV     R4, M[BPosit]
```

```
                MOV     R5, FF00h
                AND     R4, R5              ; Verifica colisao da bola com a parede horizontal inferior
do campo
                CMP     R4, 1600h           ;(caso haja colisao a direcao da bola devera ser invertida)
                JMP.Z   Reflex1
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica colisao da bola com a parede lateral direita
                CMP     R4, 004Eh           ;(caso haja deve reiniciar o jogo e marcar ponto ao jogador 1)
                JMP.Z   MarcacaoJ1
                MOV     R4, M[BPosit]
                MOV     R5, 00FFh
                AND     R4, R5              ; Verifica se podera haver colisao com a raquete direita
neste movimento,
                CMP     R4, 0049h           ;caso nao possa haver sera chamada a sub-rotina para realizar
um movimento
                JMP.NZ  Refresh4                ;para baixo e para a direita
                MOV     R4, M[RRacket]
                DEC     R4                  ; Representa o valor da posicao exactamente do lado esquerdo
do topo da raquete
                MOV     R1, 100h
                SUB     R4, R1              ; Representa posicao superior a do topo de forma a verificar
colisao com esquina
                MOV     R5, M[BPosit]
                CMP     R5, R4              ; Verifica se a bola esta na esquina da raquete e chama a
subrotina
                JMP.Z   Reflex3                 ;Reflex3 de forma a fazer a reflexao do movimento da bola
                ADD     R4, R1              ; Soma 100h (256) de forma a passar ao valor da segunda
posicao a contar do topo do
                CMP     R5, R4              ;lado esquerdo da raquete, compara com o valor da posicao da
bola e em caso de
                JMP.Z   Reflex3                 ;igualdade chama sub-rotina Reflex3 de forma a fazer a
reflexao da bola
                ADD     R4, R1
                CMP     R5, R4              ; Mesma verificacao que as anteriores agora para a terceira
posicao lateral esquerda
                JMP.Z   Reflex3                 ;a raquete
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quarta posicao lateral esquerda à raquete
                JMP.Z   Reflex3
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a quinta posicao lateral esquerda à raquete
                JMP.Z   Reflex3
                ADD     R4, R1
                CMP     R5, R4              ; Verificacao para a sexta posicao lateral esquerda;
                JMP.Z   Reflex3                 ; Caso nenhuma das condicoes anteriores se verifique nao
havera nenhuma colisao com a
                JMP     Refresh4            ;raquete pelo que sera chamada a sub-rotina Refresh4 de forma
a realizar movimento normal
Reflex3:        MOV     R5, 0002h
                MOV     M[BDirec], R5       ; Subrotina que inverte o movimento da bola para "para baixo e
para a esquerda"
                JMP     Refresh3
Refresh1:       MOV     R4, M[BPosit]       ; Sub-rotina para actualizar posicao da bola em caso de
movimento para cima e para
                MOV     R5, R4              ;a direita ou colisao para baixo e para a direita com a parede
horizontal inferior.
                SUB     R4, R2              ;A posicao atual subtrai R2 (100h) de forma a passar para a
                ADD     R4, R3              ;linha de cima e posteriormente soma R3 (1h) de forma a
avancar uma posicao para
                MOV     M[BPosit], R4       ;a direita (como manda a reflexao).
                JMP     Complete            ; Posteriormente e guardada a nova posicao da bola e chamada
sub-rotina Complete
Refresh2:       MOV     R4, M[BPosit]       ; Sub-rotina para actualizar posicao da bola em caso de
movimento para cima e para
```

```
                MOV     R5, R4              ;a esquerda ou colisao para baixo e para a esquerda com a
parede horizontal inferior.
                SUB     R4, R2              ;A posicao atual subtrai R2 (100h) de forma a passar para a
                SUB     R4, R3              ;linha de cima e posteriormente subtrair R3 (1h) de forma a
retroceder uma posicao
                MOV     M[BPosit], R4       ;para a esquerda (como manda a reflexao).
                JMP     Complete            ; Posteriormente e guardada a nova posicao da bola e chamada
sub-rotina Complete
Refresh3:       MOV     R4, M[BPosit]       ; Sub-rotina para actualizar posicao da bola em caso de
movimento para baixo e para
                MOV     R5, R4              ;a esquerda ou colisao para cima e para a esquerda com a
parede horizontal superior.
                ADD     R4, R2              ;A posicao atual adiciona R2 (100h) de forma a passar para a
                SUB     R4, R3              ;linha de baixo e posteriormente subtrair R3 (1h) de forma a
retroceder uma posicao
                MOV     M[BPosit], R4       ;para a esquerda (como manda a reflexao).
                JMP     Complete            ; Posteriormente e guardada a nova posicao da bola e chamada
sub-rotina Complete
Refresh4:       MOV     R4, M[BPosit]       ; Sub-rotina para actualizar posicao da bola em caso de
movimento para baixo e para
                MOV     R5, R4              ;a direita ou colisao para cima e para a direita com a parede
horizontal superior.
                ADD     R4, R2              ;A posicao atual adiciona R2 (100h) de forma a passar para a
                ADD     R4, R3              ;linha de baixo e posteriormente adiciona R3 (1h) de forma a
avancar uma posicao
                MOV     M[BPosit], R4       ;para a direita (como manda a reflexao).
Complete:       MOV     R1, M[BPosit]               ; Carrega posicao actualizada em R1
                MOV     M[IO_CURSOR], R1            ; Coloca o "cursor" na posicao atualizada para escrita
da bola na nova posicao
                MOV     R2, 'O'
                MOV     M[IO_WRITE], R2             ; Imprime o caracter "0" que representa a bola na
posicao atualizada
                MOV     M[IO_CURSOR], R5            ; Carrega posicao anterior da bola para caracter ser re-
escrito
                MOV     R2, ' '
                MOV     M[IO_WRITE], R2             ; Imprime um espaco em branco por cima do caracter da
posicao anterior da bola
                CALL    RefreshTime
                CALL    StartTimer                  ; Rotina que  define unidade de contagem e re-inicia o
temporizador do P3
                POP     R5                          ;para o movimento seguinte da bola
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RTI

;================================================================================
;                   Programa prinicipal
;================================================================================
Inicio:         MOV     R1, SP_INICIAL              ; Inicializacao do STACK
                MOV     SP, R1
                MOV     R1, 0000h
                MOV     M[Tempo], R1                ; Inicializacao do tempo de jogo
                MOV     M[EstGame], R1              ; Inicializacao do estado do jogo a 0 (espera)
                MOV     M[PontJ1], R1               ; Inicializacao das pontuacoes dos jogadores
                MOV     M[PontJ2], R1
                MOV     R1, 0905h
                MOV     M[LRacket], R1              ; Inicializacao da posicao da raquete esquerda a meio
da janela de jogo
                MOV     R1, 094Ah
                MOV     M[RRacket], R1              ; Inicializacao da posicao da raquete direita a meio da
janela de jogo
                MOV     R1, 7261h
                MOV     M[Seed], R1                 ; Inicializacao da SEED com um valor diferente de zero
```

```
                MOV     R1, 111Eh
                MOV     M[BPosit], R1           ; Inicializacao da Posicao da Bola e da Direcao da bola
antes de chamar-mos
                MOV     R1, 0000h               ;rotina responsavel por fazer render da bola no inicio,
a qual atribui a
                MOV     M[BDirec], R1           ;estas variaveis globais um valor aleatorio dentro dos
limites pedidos
                DSI
                CALL    INTConfig             ; Permite configuracao da rotina de interrupcao pelo
temporizador do P3 e
                ENI                             ;habilitar interrupcoes de temporizador e pelo botao I/
O na mascara de int.
                                                ; ENI permite activar o bit de estado de enable das
instrucoes
                CALL    LimpaJanela             ; Rotina que limpa a janela de jogo

                PUSH    VarTexto1               ; PUSH's do apontador para o inicio da string e da
posicao de escrita do
                PUSH    XY_INICIAL              ;primeiro caracter para serem usados pela funcao
EscString de forma à
                CALL    EscString               ;escrita de "Prima I/O para comecar o jogo"
                MOV     R1, M[EstGame]
WaitBegin:      CMP     R1, 0h                  ; Troço que mantem a string na area de jogo e nao
permite o inicio deste
                BR.Z    WaitBegin               ;(e o render dos elementos do jogo) ate o estado do
jogo ser alterado de 0
                                                ;so possivel atraves da interrupcao pelo botao I/O
Game:           CALL    LimpaJanela
                CALL    PrencLCD              ; Funcao que preenche o LCD com a regiao constante ao
longo do tempo e com as
                CALL    EscBall               ;pontuacoes iniciais
                CALL    EscMap                ; Rotina que LimpaJanela, faz render da bola, escreve o
mapa e configura
                DSI                             ;o temporizador (habilitando tambem as interupcoes de
temporizador e pelo
                CALL    INTConfig             ;botao de I/0). Tambem inicia o temporizador de forma a
iniciar o jogo.
                ENI
                CALL    StartTimer
Game1:          CALL    EscRacket               ; Rotina que permite interacao com o jogo durante o
movimento da bola
                CALL    ReadKeys                ;actualizando as posicoes das raquetes
                BR      Game1
Game2:          CALL    LimpaJanela
                CALL    EscBall             ; Rotina que permite reiniciar o jogo, com limpeza da
janela, re-render da
                CALL    EscMap              ;bola, re-escrita do mapa, re-configuracao das rotina de
interrupcao do
                DSI                         ;temporizador e re-habilitar interrupcoes de temporizador e
pelos botao I/O e
                CALL    INTConfig           ;IA na mascara de interrupcoes. Chama-se StartTimer para
reiniciar jogo
                ENI
                CALL    StartTimer
                BR      Game1
RestartGame:    MOV     R1, 111Eh               ; Permite o reinicio do jogo aquando da chamada desta
sub-rotina. Nesse
                MOV     M[BPosit], R1           ;caso a posicao da bola será colocada na posicao
correspondente ao inicio da
                MOV     R1, FFFFh               ;coluna maxima de forma à rotina de render da bola
funcionar para colocar a
                MOV     M[LEDS], R1             ;bola nos limites pedidos e salta para sub-rotina Game2
de forma a
                MOV     R1, M[PontJ1]           ;reiniciar (e re-escrever) a area de jogo
                PUSH    R1
                PUSH    8005h                   ; MOV R1, M[PontJ1] e posteriores operacoes permitem
```

```
escrever/actualizar
                CALL    EscLCD                  ;as pontuacoes dos jogadores ao longo do jogo (sempre
que o jogo e reiniciado)
                MOV     R1, M[PontJ2]
                PUSH    R1
                PUSH    8015h
                CALL    EscLCD
                JMP     Game2

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