



**TÉCNICO**  
LISBOA

# ARQUITETURA DE COMPUTADORES

LABORATÓRIO IV

## **PROGRAMAÇÃO ASSEMBLY**

*Pedro Gonalo Bravo Mendes, 81046*

*Joao Guilherme Santos, 81126*

*Professor: Aleksandar Ilic*

*Quinta-feira, 21 e 28 de maio de 2015  
LE3, Lisboa*

## 1. INTRODUÇÃO

Este trabalho laboratorial, intitulado Programação Assembly, realizado para a disciplina de Arquitetura de Computadores, tem como objetivo o desenvolvimento de um programa em assembly, incluindo o uso de periféricos e de rotinas de interrupção no processador P3.

## 2. JOGO PONG – IMPLEMENTAÇÃO DO JOGO

Com a realização deste trabalho laboratorial pretende-se implementar em linguagem de programação assembly o jogo designado “Pong”.

### 2.1. Espaço de jogo

Primeiramente, inicializa-se a janela de texto com uma string que escreve “Prima IO para iniciar o jogo”. Depois que primido o botão de pressão e ativada a interrupção IO é gerado um espaço de jogo limitado na horizontal por duas paredes formadas pelo caracter ‘-’ na linha 0 e 17, e verticalmente por duas paredes formada pelo caracter ‘|’ na coluna 0 e 70 (4Fh).

### 2.2. Raquetes

As raquetes são desenhadas na janela de texto verticalmente nas colunas 5 e 74, respetivamente. São formadas por cinco caracteres ‘#’ dispostos paralelamente na vertical.

As teclas previamente definidas controlam o movimentar as raquetes, não ultrapassando os limites da área de jogo.

### 2.3. A Bola

A bola é gerada numa posição aleatória no centro de jogo e é representada pelo caracter ‘O’. A bola move-se em linha reta na diagonal. Tem apenas quatro direções possíveis, como demonstra a figura 1. Sempre que choca com uma parede horizontal ou com as raquetes, a bola é refletida mudando a direção.

A bola desloca-se com temporizações controladas pela interrupção de temporizador com um período de 0,1 segundo.



Figura 1: Espaço de Jogo com as possíveis direções da bola

## **2.4. Posicionamento aleatório da bola**

Ao iniciar o jogo ou depois da colisão com uma parede vertical a bola é apagada da sua posição e é gerada uma nova bola numa posição aleatória no centro no espaço de jogo. Na rotina “Aleatorio” é gerado um número de acordo com o algoritmo dado no enunciado. Na rotina “DRAWBALL” é desenhada a bola num retângulo gerado entre as linhas 6 e 17 e as colunas 30 e 39. As rotinas descritas encontram-se em anexo.

## **2.5. Pontuação**

Sempre que a bola colide com uma parede vertical, o jogador adversário recebe um ponto. Posteriormente, a bola é redesenhada numa posição aleatória no centro do espaço de jogo.

## **2.6. Fim do Jogo**

Quando um dos jogadores atinge os 5 pontos o jogo termina e é gerado uma mensagem a dizer o vencedor de jogo na janela de texto.

## **2.7. Interrupções**

Para a realização deste jogo são ativas três interrupções diferentes. O botão IA ativa a interrupção 10 e coloca o jogo em pausa. Quando pressionado novamente retira o jogo de pausa. O botão de pressão IO dá (re)início ao jogo, ativando a interrupção 0. A interrupção 15 é responsável pelas interrupções do temporizador a cada 0,1s. Estas interrupções são ativas na mascara de interrupções .



Figura 2: Janela de Texto e da Placa durante o jogo Pong

## 2.8. Fluxograma do programa

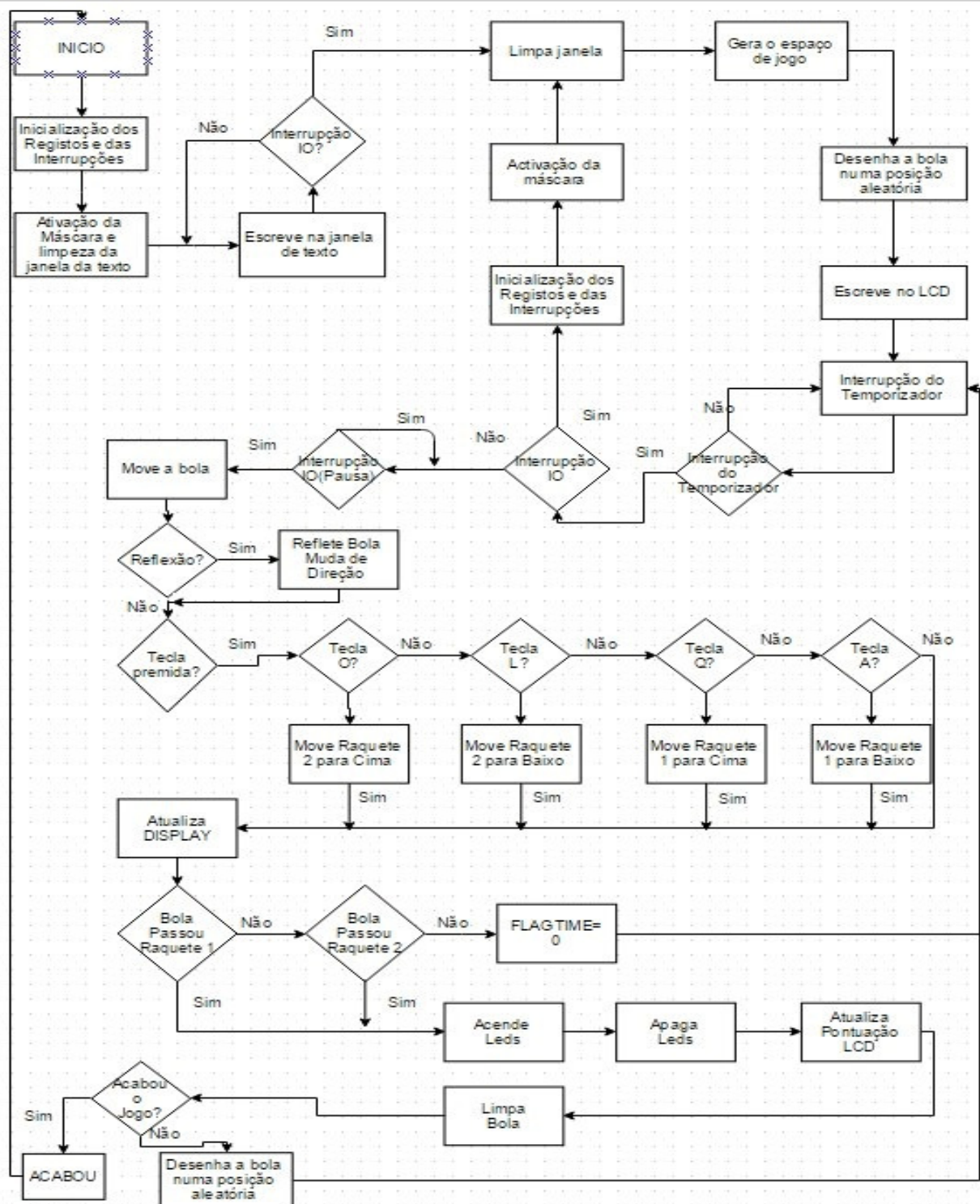


Figura 3: Fluxograma do programa Pong

## 3. CONCLUSÃO

Todos os objetivos do trabalho foram cumpridos. Todas as funções do programa são executadas corretamente de acordo com o enunciado.

```
=====
; Programa PONG lab4
;
; Descricao: Jogo do PONG
;
; Autores:
;   Pedro Mendes, ist 81046
;   Joao Santos, ist 81126
;
;   Arquitetura de Computadores
;   MEEC 2015
;
;   Data: 28/5/2015
;   Instituto Superior Técnico - Lisboa
=====
```

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

```
; TEMPORIZACAO
DELAYVALUE      EQU    0100h
```

```
; STACK POINTER
SP_INICIAL      EQU    FDFh
```

```
; INTERRUPTCOES
TAB_INT0        EQU    FE00h
TAB_INT1        EQU    FE01h
TAB_INTTemp     EQU    FE0Fh
MASCARA_INT     EQU    FFFAh
```

```
; TEMPORIZADOR
TempValor       EQU    FFF6h
TempControloEQU FFF7h
```

```
; 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
LED5           EQU    FFF8h
INTERRUPTORES  EQU    FFF9h
IO_CURSOR      EQU    FFFCh
IO_WRITE       EQU    FFFEh
IO_TEST        EQU    FFFDh
IO_KEYBOARD    EQU    FFFFh
```

```
XY_INICIAL     EQU    0614h
FIM_TEXTO      EQU    '@'
Adhifen_x      EQU    '.'
Adhifen_y      EQU    '|'
Bola           EQU    'O'
Raquete        EQU    '#'
APAGA          EQU    ''
```

```
Mascara        EQU    9C16h
Semente        EQU    F0F5h
ESTADO         EQU    F0F4h
PAUSA          EQU    F0F3h
```

```

=====
; 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 IO para iniciar o jogo',FIM_TEXTO
VarTexto2 STR  'O jogador 1 ganhou o jogo',FIM_TEXTO
VarTexto3 STR  'O jogador 2 ganhou o jogo',FIM_TEXTO
Posicao1   WORD 0A05h
Posicao2   WORD 0A4Ah
Posicao    WORD 0C27H
Direcao   WORD 0

; Relógio
FLAG_TEMPO WORD 9000h
Second      WORD 0d
DezenaSegundo WORD 0d
Minute      WORD 0d
DezenaMinuto WORD 0d

=====
; 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: --
;   Sidas: ---
;   Efeitos: ---
=====
LimpaJanela:  PUSH R2
              MOV  R2, IO_KEYBOARD
              MOV  M[IO_CURSOR], R2
              POP R2
              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"
;   Sidas: ---
;   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
        CALL   EscCar
        INC    R2
        INC    R3
        BR     Ciclo
FimEsc:  POP     R3
        POP     R2
        POP     R1
        RETN   2          ; Actualiza STACK

```

```

;=====
; EscCar: Rotina que efectua a escrita de um caracter para o ecran.
; O caracter pode ser visualizado na janela de texto.
; Entradas: R1 - Caracter a escrever
; Sidas: ---
; Efeitos: alteracao da posicao de memoria M[IO]
;=====

```

```

EscCar:   MOV    M[IO_WRITE], R1
          RET

```

```

;=====
; EscDisplay: Rotina que efectua escrita no DISPLAY de 7 segmentos
; Entradas: R1 - Valor a enviar para o porto do DISPLAY
;           R2 - Porto do DISPLAY a utilizar
; Sidas: ---
; Efeitos: alteracao da posicao de memoria/porto M[R2]
;=====

```

```

EscDisplay: MOV    M[R2], R4
          RET

```

```

EscDisplay2: MOV    M[R2], R3
          RET

```

```

EscDisplay3: MOV    M[R2], R3
          RET

```

```

;=====
; EscDisplay: Rotina que permite gerar um atraso
; Entradas: ---
; Sidas: ---
; Efeitos: ---
;=====

```

```

Delay:    PUSH    R1
          MOV     R1, DELAYVALUE
DelayLoop: DEC     R1
          BR.NZ   DelayLoop
          POP     R1
          RET

```

```

;=====
;
;           LIMPA LCD
; Rotina que limpa o perifério LCD
;=====
LIMPA:    PUSH    R1
          MOV     R1, 8020h
          MOV     M[LCD_CURSOR], R1
          POP     R1
          RET
;=====

```

```

;=====
;          ESCREVE NO LCD
;Rotina que escreve no LCD "J1:0" e "J2_0"
;=====
EscLCD:  PUSH  R1
        PUSH  R2

        CALL  LIMPA

        MOV   R1, 8000h
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 4Ah
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 31h
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 3Ah
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 30h
        MOV   M[LCD_WRITE], R2

        MOV   R1, 8010h
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 4Ah
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 32h
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 3Ah
        MOV   M[LCD_WRITE], R2
        INCR1
        MOV   M[LCD_CURSOR], R1
        MOV   R2, 30h
        MOV   M[LCD_WRITE], R2

        POP   R2
        POP   R1
        RETN  2
;=====

```

```

;=====
;          RESULTADO LCD
;Rotina que atualiza a pontuação do jogo no LCD
;=====
EscRes1: PUSH  R1
        PUSH  R2

        MOV   R1, 8013h
        MOV   M[LCD_CURSOR], R1
        MOV   R2, R5
        MOV   M[LCD_WRITE], R2
        POP   R2
        POP   R1

```



RETN 2

```
EscRes2:  PUSH R1
          PUSH R2
          MOV  R1, 8003h
          MOV  M[LCD_CURSOR], R1
          MOV  R2, R6
          MOV  M[LCD_WRITE], R2

          POP  R2
          POP  R1
          RETN 2
```

---

### GERA MAPA

---

;Rotina que (através das operações mencionadas a baixo) gerar uma janela de texto com as  
;paredes a volta da mesma.

---

```
GeraMapa:  PUSH R1
          PUSH R2
          PUSH R3
          PUSH R5

          MOV  R2, 0000h ;posição inicial da parte de cima da janela
          MOV  R3, 1700h ;posição inicial da parte de baixo da janela
          MOV  R5, 80     ;nºde hifens(-) segundo o eixo dos x
          MOV  R1, Adhifen_x
```

```
GeraParedeX: MOV  M[IO_CURSOR], R2
             CALL  EscCar
             MOV  M[IO_CURSOR], R3
             CALL  EscCar
             INCR2
             INCR3
             DEC  R5
             BR.NZ GeraParedeX ;enquanto não for 0 irá repetir o ciclo
             MOV  R2, 0100h ;posição inicial da janela do lado esquerdo
             MOV  R3, 014Fh ;posição inicial da janela do lado direito
             MOV  R5, 22     ;nºde hifens (|) segundo o eixo dos y
             MOV  R1, Adhifen_y
```

```
GeraParedeY: MOV  M[IO_CURSOR], R2
             CALL  EscCar
             MOV  M[IO_CURSOR], R3
             CALL  EscCar
             ADD  R2, 0100h ;só irá se mover segundo o eixo dos y
             ADD  R3, 0100h ;só irá se mover segundo o eixo dos y
             DEC  R5
             BR.NZ GeraParedeY

             POP  R5
             POP  R3
             POP  R2
             POP  R1
             RET
```

---

```

=====
;
;          DESENHO DA RAQUETE
;Rotina que desenha na janela de texto as duas raquetes
=====
GeraRaquete: PUSH  R1
              PUSH  R2
              PUSH  R3
              PUSH  R4

              MOV   R2, M[Posicao1]    ;posição da raquete 1
              MOV   R3, M[Posicao2]    ;posição da raquete 2
              MOV   R4, 5              ;numero de caracteres #
              MOV   R1, Raquete        ;caracter #

Raquetes:    MOV   M[IO_CURSOR], R2
              CALL  EscCar             ;escreve um # da raquete1
              MOV   M[IO_CURSOR], R3
              CALL  EscCar             ;escreve um # da raquete2
              ADD   R2, 0100h          ;incrementa a coordenada Y da raquete 1 de forma
; a continuar a escrever a raquete.
              ADD   R3, 0100h          ;incrementa a coordenada Y da raquete 2
              DEC   R4                 ;escreve 5 vezes o # na janela de texto
              BR.NZ Raquetes

              POP   R4
              POP   R3
              POP   R2
              POP   R1
              RET
=====

```

```

=====
;
;          VERIFICA TECLADO
;Rotina que verifica se foi primida alguma tecla do teclado
=====
TECLA:       PUSH  R1

;de texto    MOV   R1, M[IO_TEST]    ;porto que verifica se foi primida alguma tecla na janela

              CMP   R1, 0
              JMP.Z NOT_TECLA
              CALL  MOVE              ;rotina que executa o movimento das raquetes
NOT_TECLA:   POP   R1
              RET
=====

```

```

=====
;
;          MOVIMENTO DA RAQUETE
;Rotina responsável pelo movimento das raquetes;
;Dependendo das teclas primidas no teclado é chamada uma subrotina de executa o movimento
=====
MOVE:        PUSH  R1
              PUSH  R2
              PUSH  R3
              PUSH  R4
              PUSH  R5
              PUSH  R6
              PUSH  R7

              MOV   R2, M[IO_KEYBOARD] ;porto que recebe caracteres teclados na janela de
texto
              MOV   R3, M[Posicao1]    ;posição do 1º caracter da raquete1 na janela de texto

```

```

MOV R4, M[Posicao2] ;posição do 1º caracter da raquete2 na janela de texto
MOV R5, 5
MOV R6, 0100h

```

;letra representada em ASCII convertida em hexadecimal

```

CMP R2, 6Fh ;verifica se foi primida a tecla 'o'
JMP.Z TECLAO ;se foi primida a tecla 'o' entra nesta rotina
CMP R2, 6Ch ;verifica se foi primida a tecla 'l'
JMP.Z TECLAL ;se foi primida a tecla 'l' entra nesta rotina
CMP R2, 71h ;verifica se foi primida a tecla 'q'
JMP.Z TECLAQ ;se foi primida a tecla 'q' entra nesta rotina
CMP R2, 61h ;verifica se foi primida a tecla 'a'
JMP.Z TECLAA ;se foi primida a tecla 'a' entra nesta rotina
JMP END ;se na foi primida nenhuma dessas teclas sa desta rotina

```

```

TECLAO: MOV R7, 014Ah ;posição minina da raqueta2 na parte superior da janela de texto
        CMP R4, R7 ;quando se encontra nesta posição impossibilita
        JMP.Z END ;o movimento para cima
        ADD R4, 0400h ;posição do ultimo # da raquete
        MOV R1, APAGA
        MOV M[IO_CURSOR], R4
        CALL EscCar ;apaga esse #
        MOV R4, M[Posicao2]
        MOV R1, Raquete
        SUB R4, R6 ;nova posição para escrever o novo #
WRITE1: MOV M[Posicao2], R4 ;nova posição do início da raquete 2, guardada em memória
        MOV M[IO_CURSOR], R4
        CALL EscCar
        ADD R4, R6 ;desenha na janela de texto a raquete2 com 5 '#'
        DEC R5
        BR.NZ WRITE1
        JMP END

```

```

TECLAL: MOV R7, 164Ah ;posição maxima da raqueta1 na parte inferior da janela de texto
        ADD R4, 0400h ;quando se encontra nesta posição impossibilita
        CMP R4, R7 ;o movimento para baixo
        JMP.Z END
        SUB R4, 0400h
        MOV R1, APAGA
        MOV M[IO_CURSOR], R4
        CALL EscCar ;apaga o 1º '#' da janela de texto
        MOV R1, Raquete
        ADD R4, R6 ;incrementa uma posição ao Y
        MOV M[Posicao2], R4 ;guarda a nova posição inicial da raquete
WRITE2: MOV M[IO_CURSOR], R4
        CALL EscCar
        ADD R4, R6 ;desenha na janela de texto a raquete2 com 5 '#'
        DEC R5
        BR.NZ WRITE2
        JMP END

```

```

TECLAQ: MOV R7, 0105h ;posição minina da raqueta1 na parte superior da janela de texto
        CMP R3, R7 ;quando se encontra nesta posição impossibilita
        JMP.Z END ;o movimento para cima
        ADD R3, 0400h ;posição do ultimo # da raquete
        MOV R1, APAGA
        MOV M[IO_CURSOR], R3
        CALL EscCar ;apaga esse #
        MOV R3, M[Posicao1]
        MOV R1, Raquete
        SUB R3, R6 ;nova posição para escrever o novo #
        MOV M[Posicao1], R3 ;nova posição do início da raquete 1, guardada em memória
WRITE3: MOV M[IO_CURSOR], R3
        CALL EscCar

```

```

        ADD    R3, R6      ;desenha na janela de texto a raquete1 com 5 '#'
        DEC    R5
        BR.NZ  WRITE3
        JMP    END

TECLAA: MOV    R7, 1605h ;posição maxima da raqueta1 na parte inferior da janela de texto
        ADD    R3, 0400h ;quando se encontra nesta posição impossibilita
        CMP    R3, R7      ;o movimento para baixo
        JMP.Z  END
        SUB    R3, 0400h
        MOV    R1, APAGA
        MOV    M[IO_CURSOR], R3
        CALL   EscCar      ;apaga o 1º '#' da janela de texto
        MOV    R1, Raquete
        ADD    R3, R6      ;incrementa uma posição ao Y
        MOV    M[Posicao1], R3 ;guarda a nova posição inicial da raquete
WRITE4: MOV    M[IO_CURSOR], R3
        CALL   EscCar
        ADD    R3, R6      ;desenha na janela de texto a raquete1 com 5 '#'
        DEC    R5
        BR.NZ  WRITE4
        JMP    END

END:    POP    R7
        POP    R6
        POP    R5
        POP    R4
        POP    R3
        POP    R2
        POP    R1
        RET

```

```

;=====
;
;      INTERRUPTÃO TEMPORIZADOR
;Rotina responsavel pela interrupção do tenporizador de 0,1s em 0,1s
;=====
INT_TEMPO: PUSH  R1
          MOV    R1, 1d
          MOV    M[FFF6h], R1 ;porto que indica o intervalo de 0,1s
          MOV    M[FFF7h], R1 ;porto responsável pela ativação do temporizador
          MOV    M[FLAG_TEMPO], R1
          POP    R1
          RTI    ;retorna a interrupção

```

```

;=====
;
;      LimpaBola
;Rotina que apaga da janela de texto a posição antiga da bola
;=====
LimpaBola: PUSH  R1
          PUSH  R2

          MOV    R1, APAGA ;coloca em R1 a constante que contém o espaço
          MOV    R2, M[Posicao]
          MOV    M[IO_CURSOR], R2 ;colocar o endereço de memória da posição na janela
          CALL   EscCar

          POP    R2
          POP    R1
          RET

```

```

=====
;
;           EscreveBola
;Rotina que desenha a bola na janela de texto na sua nova posição
=====
EscreveBola: PUSH  R1
              PUSH  R2

              MOV   R1, Bola
              MOV   R2, M[Posicao]
              MOV   M[IO_CURSOR], R2
              CALL  EscCar

              POP   R2
              POP   R1
              RET

=====

;
;           Movimento Bola
;Rotina que faz com que a bola se desloque irá
;determina a direção em que a bola se move
;0-para cima e para a esquerda
;1-para cima e para a direita,
;2-para baixo e para a direita,
;3-para baixo e para a esquerda
=====
MoveBola: PUSH  R1
           PUSH  R2
           PUSH  R3

           MOV   R1, Bola ;coloca em R1 o caracter 'O'
           MOV   R2, M[Direcao]
;coloca em R2 o que está contido no endereço de memória da direção
           MOV   R3, M[Posicao]
;coloca em R3 o que está contido no endereço de memória da posição
           CALL  LimpaBola

CimEsq:  CMP   R2, R0 ;ve se se move na direção 0
         BR.NZ CimDir ;se não passa para a análise da próxima
         SUB   R3, 0101h ;move decrementando em y e em x
         MOV   M[Posicao], R3
;coloca esta nova posição no endereço de memória
         CALL  EscreveBola
         JMP   Final

CimDir:  CMP   R2, 1 ;analisa se está a mover-se na direção 1
         BR.NZ BaixoEsq
         SUB   R3, 00FFh ;move-se incrementando em x e decrementando em y
         MOV   M[Posicao], R3
         CALL  EscreveBola
         JMP   Final

BaixoEsq: CMP   R2, 2 ;analisa se está a mover-se na direção 2
         BR.NZ BaixoDir
         ADD   R3, 0101h ;move-se incrementando em y e em x
         MOV   M[Posicao], R3
         CALL  EscreveBola
         JMP   Final

BaixoDir: ADD   R3, 00FFh ;move-se na direção 3
         MOV   M[Posicao], R3 ;incrementa segundo y e decrementando em x

```

```

        CALL  EscreveBola
        JMP   Final

Final:   POP    R3
        POP    R2
        POP    R1
        RET

;=====

;=====
;
;           Reflexo
;Rotina que realiza a mudança de direção da bola, quando esta
;choca com as paredes horizontais ou raquetes.
;=====
Reflect:  PUSH   R1
        PUSH   R2
        PUSH   R3
        PUSH   R4
        PUSH   R5

        MOV    R1, M[Direcao] ;R1 guarda a direção da bola
        MOV    R2, M[Posicao]  ;R2 será guardada a posição da bola
        MOV    R3, M[Posicao1] ;R3 guarda a posição da raquete 1
        MOV    R4, M[Posicao2] ;R3 guarda a posição da raquete 2
        MOV    R5, 5d

        AND    R2, 00FFh ;compara a posição das raquetes com a posição da bola
        CMP    R2, 0006h
        JMP.NZ Verifica2
        MOV    R2, M[Posicao]
        SUB    R2, 0001h
ReflectRa1: CMP    R3, R2 ;verifica se a bola embate na raquete 1
        JMP.Z  Reflect5
        ADD    R3, 0100h
        DEC    R5
        JMP.Z  fimRef
        BR     ReflectRa1

Reflect5: CMP    R1, 0d ;muda a direção da bola dependendo da direção anterior
        BR.Z   MUDADIR1
        MOV    R1, 2
        MOV    M[Direcao], R1
        JMP    fimRef

MUDADIR1:MOV    R1, 1
        MOV    M[Direcao], R1
        JMP    fimRef

Verifica2: MOV    R2, M[Posicao]

        AND    R2, 00FFh ;compara a posição das raquetes com a posição da bola
        CMP    R2, 0049h
        JMP.NZ Verifica3
        MOV    R2, M[Posicao]
        ADD    R2, 0001h
ReflectRa2: CMP    R4, R2 ;verifica se a bola embate na raquete 1
        JMP.Z  Reflect6
        ADD    R4, 0100h
        DEC    R5
        JMP.Z  fimRef
        BR     ReflectRa2

```

```

Reflect6:  CMP    R1, 2d      ;muda a direção da bola dependendo da direção anterior
           BR.Z    MUDADIR2
           MOV     R1, 0
           MOV     M[Direcao], R1
           JMP     fimRef

MUDADIR2: MOV    R1, 3
           MOV     M[Direcao], R1
           JMP     fimRef

Verifica3: MOV    R2, M[Posicao]

           AND     R2, FF00h ;bits da componente em y
           CMP     R2, 0100h ;compara posição da bola
           BR.NZ   Reflect2

           CMP     R1, 0
           BR.NZ   Reflect1
           MOV     R1, 3      ;move se na direção zero
           MOV     M[Direcao], R1
           BR      fimRef

Reflect1:  MOV     R1, 2      ;se estiver a mover-se na direção 1 muda para a direção 2
           MOV     M[Direcao], R1
           BR      fimRef

Reflect2:  CMP     R2, 1600h ;analisa se se encontra na posição em y anterior a parede inferior
           BR.NZ   fimRef     ;se não estiver continua o movimento que seguia
           CMP     R1, 2      ;verifica se está a mover-se segundo a direção 2
           BR.NZ   Reflect3    ;desloca-se noutra direção

           MOV     R1, 1      ;desloca-se segundo a direção 2 e muda a direção 1
           MOV     M[Direcao], R1
           BR      fimRef

Reflect3:  MOV     R1, 0      ;mudança da direção 3 para a 0
           MOV     M[Direcao], R1

fimRef:    POP     R5
           POP     R4
           POP     R3
           POP     R2
           POP     R1
           RET

```

```

;=====

```

```

;=====
;
;          Aleatório
;Rotina que gerar um número aleatório entre 0 e M-1, através das operações mencionadas
;=====

```

```

Aleatorio: PUSH    R1
           PUSH    R2
           PUSH    R3

           MOV     R1, M[Semente] ;guarda no registo o valor da seed
           MOV     R2, 1          ;será depois utilizado para fazer uma condição
           MOV     R3, Mascara    ;coloca o valor da mascara no registo
           AND     R2, R1         ;condição para verificar se o ultimo bit da semente
           BR.NZ   nZero

           MOV     R2, R1
;se o ultimo bit for zero realiza as operações a baixo definidas no enunciado

```

```

        ROR    R2, 1
        MOV    M[Semente], R2
        JMP    Finito

nZero:   MOV    R2, R1
;se for 1 o ultimo bit realiza as seguinte operações definidas no enunciado
        XOR    R2, R3
        ROR    R2, 1
        MOV    M[Semente], R2
        MOV    R4, R2 ;coloca em R4 o número aleatório gerado

Finito:  POP    R3
        POP    R2
        POP    R1
        RET

;=====

;
;      DESENHA BOLA ALEATÓRIAMENTE
;Rotina que gera uma posição (aleatória) da bola dentro de uma area no centro de jogo
;=====
DRAWBALL: PUSH  R1
          PUSH  R2
          PUSH  R3
          PUSH  R4

          MOV   R1, M[Posicao]
          MOV   R2, 12d
;(intervalo+1) segundo o eixo dos y onde poderá aparecer a bola
          CALL  Aleatorio
          DIV   R4, R2
;registo que contem o numero aleatorio e faz a divisão com resto pelo (intervalo+1)
          ROL   R2, 8
          ADD   R2, 0600h
;adiciona 6(minimo do intervalo)
;valor após a divisão e rotação será sempre menor que o valor do (intervalo+1)
          MVBH  M[Posicao], R2
; copia os 8 bits mais significativos para o endereço da posição

          MOV   R3, 0014h
;(intervalo+1) segundo o eixo dos x onde poderá aparecer a bola
          CALL  Aleatorio
          DIV   R4, R3
;registo que contem o numero aleatorio e faz a divisão com resto pelo (intervalo+1)
          ADD   R3, 001Eh
;adiciona 30(minimo do intervalo)
;valor após a divisão e rotação será sempre menor que o valor do (intervalo+1)
          MVBL  M[Posicao], R3
; copia os 8 bits menos significativos para o endereço da posição

          POP   R4
          POP   R3
          POP   R2
          POP   R1
          RET

;=====

```



```

=====
;
;      (RE)INICIAR O JOGO
;Rotina responsável pela interrupção que inicia o jogo
=====
GAME_AGAIN: PUSH  R1
             MOV   R1, M[ESTADO]
             MOV   R1, 1d
             MOV   M[ESTADO], R1; Flag de estado que quando se encontra a 1 o jogo é
iniciado
             POP   R1
             RTI
=====

;=====
;
;      PAUSA DO JOGO
;Rotina responsável pela interrupção que pausa o jogo
=====
PAUSE:  PUSH  R1
             MOV  R1, 1d
             CMP  M[PAUSE], R1 ;Verifica o estado da Flag de pausa
             BR.Z ZER
             MOV  R1, 1d
             MOV  M[PAUSE], R1
;se a flag PAUSE está a 0 e é chamada a rotina da interrupção a flag é colocada a 1 e o jogo
;é pausado
             BR   FINITA

ZER:        MOV  M[PAUSE], R0
;se a flag PAUSE está a 1 e é chamada a rotina da interrupção a flag é colocada a 0 e o jogo
;recomeça

FINITA:     POP   R1
             RTI   ;retorna a interrupção
=====

;=====
;
;      DISPLAY
;Rotina responsável pela contagem do tempo no display de 7 segmentos
=====
DISP:      PUSH  R1
             PUSH  R2

             MOV  R1, 1d
             CMP  M[FLAG_TEMPO], R1 ; Verificação se a flag está ativa
             JMP.NZ Endi
             INCR7
             CMP  R7, 10d           ;conta 1 segundo
             JMP.NZ Endi

             MOV  R7, R0
             MOV  R1, M[Second]
             INCR1 ; Incremento dos segundos
             MOV  M[Second], R1
             CMP  R1, 10d           ; Verificação quando chega aos 10 segundos
             BR.NZ Write

             MOV  M[Second], R0
; Atualização dos segundo para 0 para que se recomeçar a contagem
             MOV  R1, M[DezenaSegundo]
             INCR1 ; Incremento das dezenas de segundo
             MOV  M[DezenaSegundo], R1

```

```

        CMP    R1, 6d      ; Verificação de 1 minuto
        BR.NZ  Write

        MOV    M[DezenaSegundo], R0
; Atualização das dezenas de segundo para 0 para que recomece a contagem
        MOV    R1, M[Minute]
        INCR1   ; Incremento dos minutos
        MOV    M[Minute], R1
        CMP    R1, 10d     ; Verificação dos 10 minutos
        BR.NZ  Write

        MOV    M[Minute], R0
; Atualização das unidades de minuto para 0 para recomeçar a contagem
        MOV    R1, M[DezenaMinuto]
        INCR1   ; Incremento das dezenas de minuto
        MOV    M[DezenaMinuto], R1

; Escrita no display de todas as unidades do relógio
Write:   MOV    R1, M[Second]
        MOV    M[DISP7S1], R1
        MOV    R1, M[DezenaSegundo]
        MOV    M[DISP7S2], R1
        MOV    R1, M[Minute]
        MOV    M[DISP7S3], R1
        MOV    R1, M[DezenaMinuto]
        MOV    M[DISP7S4], R1

        MOV    R1, R0

Endi:    POP    R2
        POP    R1
        RET

;=====
;
; Programa principal
;=====
inicio:  MOV    R1, SP_INICIAL
        MOV    SP, R1

        MOV    R5, 48
        MOV    R6, 48
        MOV    R7, 0
        MOV    R4, 0
        MOV    R3, 0
        MOV    M[ESTADO], R0

        MOV    R1, INT_TEMPO ; inicialização das interrupções
        MOV    M[FE0Fh], R1 ; interrupção do temporizador
        MOV    R1, GAME_AGAIN ; interrupção de início do jogo
        MOV    M[FE00h], R1
        MOV    R1, PAUSE      ; interrupção da pausa de jogo
        MOV    M[FE0Ah], R1
        MOV    R1, 8401h      ; interrupções ativas
        MOV    M[FFFAh], R1 ; ativação da máscara
        MOV    R1, 1d
        MOV    M[FFF6h], R1
        MOV    M[FFF7h], R1
        ENI

        CALL   LimpaJanela

        PUSH   VarTexto1
        PUSH   XY_INICIAL

```

```

reinicio: CALL EscString      ;escrita na janela de texto da VarTexto1
          MOV  R2, M[ESTADO]
          CMP  R2, R0
          BR.Z reinicio      ;espera que a interrupção de inicio (I0) seja ativada
                                ;para começar o jogo
          JMP   CA

;rotina para reiniciar o jogo
begin:    MOV  R5, 48
          MOV  R6, 48
          MOV  R7, 0
          MOV  R4, 0
          MOV  R3, 0
          MOV  M[ESTADO], R0

          MOV  R1, INT_TEMPO
          MOV  M[FE0Fh], R1
          MOV  R1, GAME_AGAIN
          MOV  M[FE00h], R1
          MOV  R1, PAUSE
          MOV  M[FE0Ah], R1
          MOV  R1, 8401h
          MOV  M[FFFAh], R1
          MOV  R1, 1d
          MOV  M[FFF6h], R1
          MOV  M[FFF7h], R1
          ENI

CA:        CALL  LimpaJanela
          CALL  GeraMapa      ;rotina gera o espaço de jogo
          CALL  GeraRaquete   ;rotina que gera as raquetes
          CALL  DRAWBALL      ;rotina que desenha a bola numa posição aleatória

          PUSH  R1
          PUSH  8014h
          CALL  EscLCD        ;rotina que escreve no LCD

TEMPO:     MOV  R1, M[FLAG_TEMPO] ;rotina de interrupção do temporizador
          CMP  R1, 1          ;ativa a cada 0,1s
          BR.NZ TEMPO

          MOV  R2, M[ESTADO]   ;rotina de interrupção de reinicio do jogo
          CMP  R2, 1
          JMP.Z begin

pausaa:    MOV  R3, 1d          ;rotina de interrupção da pausa
          CMP  M[PAUSA], R3     ;quando ativa o jogo está parado
          BR.Z pausaa

          CALL  MoveBola       ;rotina responsável pelo movimento da bola
          CALL  Reflect        ;rotina responsável pela mudança de direção da bola
          CALL  TECLA          ;rotina responsável pelo movimento das raquetes
; dependendo das teclas primida

          CALL  DISP          ;rotina que escreve o tempo de jogo no display

          MOV  R2, 00FFh
          AND  R2, M[Posicao] ;verifica se a bola passou a raquete 1
          CMP  R2, 0004h
          BR.Z REINICIA1
          CMP  R2, 004Bh      ;verifica se a bola passou a raquete 2
          JMP.Z REINICIA2

```

```

        MOV    M[FLAG_TEMPO], R0 ;coloca o temporizador a 0 em cada ciclo executado
        JMP    TEMPO             ;repete a rotina

REINICIA1:INC    R5              ;a bola passou pela raquete1
        MOV    R2, FFFFh
        MOV    M[LEDS], R2      ;acende os leds
LUZ:      DEC    R2
        CMP    R2, R0
        BR.NZ  LUZ
        MOV    R2, R0
        MOV    M[LEDS], R2      ;apaga os leds
        PUSH   R1
        PUSH   8014h
        CALL   EscRes1          ;atualiza o resultado do LCD
        CALL   LimpaBola;Apaga a bola
        CMP    R5, 53           ;compara R5 com 5 (53 em ASCII) e
        JMP.Z  ACABOU1          ;verifica se chegou ao final do jogo
        CALL   DRAWBALL         ;se nao é o final volta a desenhar a bola numa posição
        JMP    TEMPO

REINICIA2:INC    R6              ;a bola passou pela raquete2
        MOV    R2, FFFFh
        MOV    M[LEDS], R2      ;acende os leds
LUZ2:     DEC    R2
        CMP    R2, R0
        BR.NZ  LUZ2
        MOV    R2, R0
        MOV    M[LEDS], R2      ;apaga os leds
        PUSH   R1
        PUSH   8014h
        CALL   EscRes2          ;atualiza o resultado do LCD
        CALL   LimpaBola;Apaga a bola
        CMP    R6, 53           ;compara R6 com 5 (53 em ASCII) e
        JMP.Z  ACABOU2          ;verifica se chegou ao final do jogo
        CALL   DRAWBALL         ;se nao é o final volta a desenhar a bola numa posição
        JMP    TEMPO

ACABOU1:PUSH    VarTexto3        ;acabou o jogo
        PUSH    XY_INICIAL
        CALL    EscString         ;escreve na janela de texto a string VarTexto3
        BR      ACABOU

ACABOU2:PUSH    VarTexto2        ;acabou o jogo
        PUSH    XY_INICIAL
        CALL    EscString         ;escreve na janela de texto a string VarTexto3
        BR      ACABOU

ACABOU:  MOV    R2, FFFFh
var:     DEC    R2
        CMP    R2, R0
        BR.NZ  var
        CALL   LimpaJanela
        JMP    inicio
;=====

```