```
;/\/\/\/\/\/\/\/\/\/\/\/\//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\\/\/
;|---------------------CORRIDA BICICLETAS-------------------|
;\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\\/\/\/

;-------------------------------------------------------------
;Grupo 44:
;Eduardo Rodrigues n.78431
;Alcino Albuquerque n.78753
;Rui Matos n.79100
;-------------------------------------------------------------

;-------------------------------------------------------------
; -----------CONSTANTES DA JANELA DE TEXTO--------------------
;-------------------------------------------------------------

LINHAS_FIM_ESQ  EQU     181Dh   ; Última  Linha
LINHAS_FIM_DIR  EQU     1836h   ; Última  Linha
MUDA_LINHA      EQU     0100h   ; Avança uma linha
IO_CONTROL      EQU     FFFCh   ; Recebe coordenadas
IO_WRITE        EQU     FFFEh   ; Escreve um determinado valor numa coordenada indicada anteriormente
SIMBOLO         EQU    '+'
SP_INICIAL      EQU     FDFFh
LIMPA_J         EQU     FFFFh
BARRA           EQU     '|'
RODA            EQU     'O'
ESPACO          EQU     ' '
LIM_LINHAS      EQU     1800h ; Corresponde à ultima linha
COLUNAS         EQU     0001h ; Adiciona uma coluna
INTADDR         EQU     FFFAh
INTMASK         EQU     1000110000000111b  ;Activa as interrupções F , A , B , 2  , 1 e  0
MASK            EQU     1000000000010110b  ;usada na sequência aleatória
DISP7s1         EQU     FFF0h
DISP7s2         EQU     FFF1h
DISP7s3         EQU     FFF2h
DISP7s4         EQU     FFF3h
lcd_cursor      EQU     FFF4h
lcd_write       EQU     FFF5h
io_led          EQU     FFF8h
LIM_BIC_ESQ     EQU     151Fh  ;Limite que a bicicleta não pode ultrapassar à esquerda
LIM_BIC_DIR     EQU     1534h  ;Limite que a bicicleta não pode ultrapassar à direita
TIMER_COUNT     EQU     FFF6h  ;Recebe um determinado tempo no temporizador
TIMER_ONOFF     EQU     FFF7h  ;Activa ou desactiva o temporizador
OBS             EQU     '*'
NIVEL1          EQU     0005h  ;Velocidade 2m/s
NIVEL2          EQU     0004h  ;Velocidade 2.5m/s
NIVEL3          EQU     0003h  ;Velocidade 3.3m/s
NIVELT          EQU     0002h  ;Velocidade 5m/s - TURBO
COLISAO         EQU     0005h  ;INDICA QUE HOUVE COLISÃO
BIC_INIT        EQU     1529h  ;Posição Inicial da Bicicleta

;-------------------------------------------------------------
;--------------------------VARIAVEIS-------------------------
;-------------------------------------------------------------

                ORIG    8000h
POSICAOBIC      WORD    1529h   ;Posição da Bicicleta que vai variando
POSICAO_OBS1    WORD    0000h   ;Variável das posições dos obstáculos
POSICAO_OBS2    WORD    0000h
POSICAO_OBS3    WORD    0000h
POSICAO_OBS4    WORD    0000h
CONT1           WORD    0006h   ;Primeiro valor que indica que tem de ser criado mais um obstáculo
CONT2           WORD    000Ch   ;Segundo valor que indica que tem de ser criado mais um obstáculo
CONT3           WORD    0012h   ;Terceiro valor que indica que tem de ser criado mais um obstáculo
```

```
BV1             STR     'Bem-Vindo a corrida de bicicleta!'     ;Mensagem inicial do jogo1
BV2             STR     'Prima o interruptor I1 para comecar'   ;Mensagem inicial do jogo2
END1            STR     'Fim do Jogo'   ;Mensagem final do jogo1
END2            STR     'Prima o interruptor I1 para comecar'   ;Mensagem final do jogo2
contatemp       WORD    0000h   ;variável incrementada pelo temporizador
VEL             WORD    0005h   ;Variável que indica ao temporizador a velocidade a que o obstáculo vai
cair
COLISAOCHECK    WORD    0001h   ;Indica se existe ou não colisão
ONOFF           WORD    0001h   ;Activa ou desactiva o Temporizador
DIST            STR     'Distancia:00000m'   ;Frase do LCD 1
MAX             STR     'Maximo:00000m'   ; Frase do LCD 2
OBS_COUNT       WORD    0000h   ;conta o número de obstáculos passados
DECIMAL         WORD    0000h   ;Guarda o valor em decimal para o lcd
OBS_COUNT_DECIMAL       WORD    0000h   ;Guarda o valor em decimal do número de obstáculos
MAX_COUNT       WORD    0000h   ;Guarda o valor máximo presente no LCD até ao momento do jogo
NVAR1           WORD    0000h   ;Usado no Anexo A para coordenada aleatória
POS             WORD    0000h   ;Gerada pela sequência aleatória uma coordenada
TURBOONOFF      WORD    0000h   ;verifica se o turbo está ou não ligado
PAUSASTR        STR     '-PAUSA-'

;-------------------------------------------------------------
;------------------------Interrupções-------------------------
;-------------------------------------------------------------
                ORIG FE00h
Int0            WORD IntE
Int1            WORD IntI
Int2            WORD IntTurbo

                ORIG FE0Ah
IntA            WORD IntPausa
IntB            WORD IntD

                ORIG FE0Fh
IntF            WORD Contador
;-------------------------------------------------------------
;----------Interrupções - Ordens + Inicializador--------------
;-------------------------------------------------------------
                ORIG    0000h
                JMP     progprincipal
IntI:           MOV     R7, 0001h       ;Caso se carregue no botão I1 o valor 0001h passa para R7
                RTI
IntE:           MOV     R7, 0004h       ;Caso se carregue no botão I0 o valor 0004h passa para R7
                RTI
IntD:           MOV     R7, 0003h       ;Caso se carregue no botão IB o valor 0003h passa para R7
                RTI
Contador:       INC     R1      ;Correponde ao temporizador, que incrementa o registo R1.
                MOV     R7, M[VEL]
                MOV     M[TIMER_COUNT], R7
                MOV     R7, M[ONOFF]
                MOV     M[TIMER_ONOFF], R7
                RTI
IntPausa:       MOV     R7, 0023h ;Caso se carregue no botão IA o valor 0098 passa para R7
                RTI
IntTurbo:       MOV     R7, 0098h  ; Caso se carregue no botão I2 o valor 0098 passa para R7
                RTI
;-------------------------------------------------------------
;------------------------LIMPA JANELA-------------------------
;--Substitui todos as posições por espaços, apagando, assim a-
;--------------janela - Necessário para a Placa--------------
;-------------------------------------------------------------

LIMPA:          PUSH    R1
```

```
                PUSH    R2
                PUSH    R3
                MOV     R1, 0000h ; Começa a limpa a primeira coordenada
limpa_cont:     MOV     M[IO_CONTROL],R1
                MOV     R2, ESPACO ; Enche as posições com espaços
                MOV     M[IO_WRITE], R2
                ADD     R1, MUDA_LINHA
                MOV     R3, R1
                AND     R3, FF00h ; "Apaga" as colunas
                CMP     R3, 1800h ; E compara se já se chegou à última linha.
                BR.Z    muda_coluna ;Caso se tenha chegado muda-se de coluna
                BR      limpa_cont
muda_coluna:    CMP     R1, 184Fh ; compara-se se se chegou à ultima coluna da última linha
                BR.Z    fim_limpa
                AND     R1, 00FFh ; Renova-se as linhas
                INC     R1 ; Adiciona-se uma coluna.
                BR      limpa_cont
fim_limpa:      POP     R3
                POP     R2
                POP     R1
                RET

;---------------------------------------------------------------
;----------------------WELCOME JOGO-----------------------------
;--------Escreve a mensagem inicial no ecrã de jogo-------------
;---------------------------------------------------------------

Texto_Init:     PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1,BV1
                MOV     R2, 0B17h ; primeira posição onde é colocado o primeiro caracter linha = 11
coluna = 23
Continua_esc:   MOV     R3,M[R1] ;
                MOV     M[IO_CONTROL], R2
                MOV     M[IO_WRITE], R3
                ADD     R2, COLUNAS
                CMP     R2, 0B38h ; Verifica-se se já se chegou à coordenada final. x = 11 y = 56
                BR.Z    esc_prox_linha ; Caso se tenha verificado a condiçao anterior escreve-se a
próxima linha
                INC     R1 ; Incrementa-se o indice para se obter o próximo caracter
                BR      Continua_esc

;Comentários idênticos aos anteriores

esc_prox_linha: MOV     R1,BV2
                MOV     R2, 0D16h ;x=13 y =22
Continua_esc_2: MOV     R3,M[R1]
                MOV     M[IO_CONTROL], R2
                MOV     M[IO_WRITE], R3
                ADD     R2, COLUNAS
                CMP     R2, 0D39h ;x=13 y=57
                BR.Z    fim_esc_texto
                INC     R1
                BR      Continua_esc_2
fim_esc_texto:  POP     R3
                POP     R2
                POP     R1
                RET

;---------------------------------------------------------------
;-------------------------FIM DO JOGO---------------------------
```

```
;--------Escreve a mensagem final no ecrã de jogo-------------
;-------------------------------------------------------------

;O processo é idêntico ao da rotina anterior, pelo que se excluiu
;os comentários de forma a facilitar a leitura.
Texto_Init2:    PUSH    R1
                PUSH    R2
                PUSH    R3
                CALL    LIMPA
                MOV     R1,END1
                MOV     R2, 0B21h ;x=11 y=33
Continua_esc2:  MOV     R3,M[R1]
                MOV     M[IO_CONTROL], R2
                MOV     M[IO_WRITE], R3
                ADD     R2, COLUNAS
                CMP     R2, 0B2Ch ;x=11 y = 44
                BR.Z    esc_prox_linha2
                INC     R1
                BR      Continua_esc2
esc_prox_linha2:MOV     R1,END2
                MOV     R2, 0D16h ;x=13 y =22
Continua_esc2_: MOV     R3,M[R1]
                MOV     M[IO_CONTROL], R2
                MOV     M[IO_WRITE], R3
                ADD     R2, COLUNAS
                CMP     R2, 0D39h ;x=13 y=57
                BR.Z    fim_esc_texto2
                INC     R1
                BR      Continua_esc2_
fim_esc_texto2: POP     R3
                POP     R2
                POP     R1
                RET

;-------------------------------------------------------------
;---------------------------CENARIO---------------------------
;--------------Desenha o Cenário de jogo = Paredes------------
;-------------------------------------------------------------

Inicio:         PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                CALL    LIMPA ; Limpa a janela
Escreve_Esq:    MOV     R1, 001Dh ; Coloca no registo a coordenada correspondente à primeira posição onde
vai ser escrito o '+'
                MOV     M[IO_CONTROL], R1
                MOV     R2, SIMBOLO
                MOV     M[IO_WRITE], R2
                MOV     R3, 001Eh ; Coloca no registo a coordenada correspondente à posicao onde vai ser
escrita a '|'
                MOV     M[IO_CONTROL],R3
                MOV     R4,BARRA
                MOV     M[IO_WRITE], R4
novalinha_e:    ADD     R3, MUDA_LINHA ;Adiciona uma linha
                ADD     R1, MUDA_LINHA
                CMP     R1, LINHAS_FIM_ESQ ; Verifica se a escrita chegou à última linha do lado esquerdo
                BR.Z    Escreve_Dir ; Caso se verifique escreve a parede do lado direito
                MOV     M[IO_CONTROL], R1
                MOV     M[IO_WRITE], R2
                MOV     M[IO_CONTROL],R3
```

```
                MOV     M[IO_WRITE],R4
                BR      novalinha_e ; Volta a incrementar as linhas e a escrever

;Comentários idênticos aos anteriores.
Escreve_Dir:    MOV     R1, 0036h
                MOV     M[IO_CONTROL], R1
                MOV     R2, SIMBOLO
                MOV     M[IO_WRITE], R2
                MOV     R3, 0035h
                MOV     M[IO_CONTROL],R3
                MOV     R4,BARRA
                MOV     M[IO_WRITE], R4
novalinha_d:    ADD     R3, MUDA_LINHA
                ADD     R1, MUDA_LINHA
                CMP     R1, LINHAS_FIM_DIR
                BR.Z    fim_d
                MOV     M[IO_CONTROL], R1
                MOV     M[IO_WRITE], R2
                MOV     M[IO_CONTROL],R3
                MOV     M[IO_WRITE],R4
                BR      novalinha_d

fim_d:          POP     R4 ; Termina o escreve cenário
                POP     R3
                POP     R2
                POP     R1
                RET

;--------------------------------------------------------------
;--------------------------BICICLETA---------------------------
;Desenha a bicicleta e faz as suas movimentacoes,começando por
;--por apagar a posição actual e escrevendo na nova posição.--
;--------------------------------------------------------------

desenha_bicicleta_INIT: PUSH    R1
                PUSH    R2
                MOV     R1, M[POSICAOBIC] ; Posicao correspondente à variável para o registo 1
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2 ; Escreve 'O' na determinada posição
                ADD     R1, MUDA_LINHA ; Adiciona linha
                MOV     M[IO_CONTROL], R1
                MOV     R2, BARRA
                MOV     M[IO_WRITE], R2 ; Escreve uma barra na determinada posição
                ADD     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2
                SUB     R1, MUDA_LINHA
                SUB     R1, MUDA_LINHA ; Volta devolve o valor da linha correcta à bicicleta, visto terem
sido adicionada duas colunas
                MOV     M[POSICAOBIC],R1 ; Adiciona-se o valor à Memória da posicao da BIC
                MOV     R7,R0 ; Repoe-se R7 para não influenciar as interrupções
                POP     R2
                POP     R1
                RET
;Substitui todas as posiçoes da bic por espaços usando o mesmo racíocinio explicado anteriormente
apaga_bic_:     PUSH    R1
                PUSH    R2
                MOV     R1, M[POSICAOBIC]
                MOV     M[IO_CONTROL], R1
```

```
                MOV     R2, ESPACO
                MOV     M[IO_WRITE], R2
                ADD     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, ESPACO
                MOV     M[IO_WRITE], R2
                ADD     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, ESPACO
                MOV     M[IO_WRITE], R2
                MOV     M[POSICAOBIC], R1
                POP     R2
                POP     R1
                RET
;Raciocinio Igual ao anterior sendo explicado os limites da BIC
desenha_esq:    PUSH    R1
                PUSH    R2
                MOV     R1, M[POSICAOBIC]
                CMP     R1, LIM_BIC_ESQ ; Só desenha a bicicleta caso a bicicleta não se encontre no
limite esquerdo definido
                BR.Z    fim_esq
                CALL    apaga_bic_ ; Caso a condição atrás o permita apaga a actual posicao e escreve na
nova posição uma coluna à esquerda
                MOV     R1, M[POSICAOBIC]
                SUB     R1, COLUNAS
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2
                SUB     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, BARRA
                MOV     M[IO_WRITE], R2
                SUB     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2
                MOV     M[POSICAOBIC], R1
                MOV     R7,R0
fim_esq:        POP     R2
                POP     R1
                RET
;Raciocinio Igual ao anterior sendo explicado os limites da BIC
desenha_dir:    PUSH    R1
                PUSH    R2
                MOV     R1, M[POSICAOBIC]
                CMP     R1, LIM_BIC_DIR; Só desenha a bicicleta caso a bicicleta não se encontre no
limite esquerdo definido
                BR.Z    fim_dir
                CALL    apaga_bic_ ; Caso a condição atrás o permita apaga a actual posicao e escreve na
nova posição uma coluna à esquerda
                MOV     R1, M[POSICAOBIC]
                ADD     R1, COLUNAS
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2
                SUB     R1, MUDA_LINHA
                MOV     M[IO_CONTROL], R1
                MOV     R2, BARRA
                MOV     M[IO_WRITE], R2
                SUB     R1, MUDA_LINHA
```

```
                MOV     M[IO_CONTROL], R1
                MOV     R2, RODA
                MOV     M[IO_WRITE], R2
                MOV     M[POSICAOBIC], R1
                MOV     R7,R0
fim_dir:        POP     R2
                POP     R1
                RET

;---------------------------------------------------------------
;-------------------------OBSTACULO-----------------------------
;----Desenha o obstaculo e faz a sua movimentacao (apaga a sua
;posicao atual desenha o obstaculo na posicao abaixo, verifica
;se existem colisões entre os obstáculos e as posições da bic.
;---------------------------------------------------------------
;Rotina que escreve da esquerda para a direita o primeiro obstáculo consoante a posição que recebe em R1
referente a cada obstáculo
Esc_geral_obs:  MOV     R3, R1
                ADD     R3, 0003h ; Cria um limite lateral para o obstáculo
esc_new:        MOV     M[IO_CONTROL],R1
                MOV     R2,OBS
                MOV     M[IO_WRITE], R2
                ADD     R1, COLUNAS
                CMP     R1, R3 ; Verifica se a escrita se encontra nesse limite
                BR.NZ   esc_new
                SUB     R3, COLUNAS ; Devolve ao obstáculo a posição inicial.
                SUB     R1, COLUNAS ; Devolve ao obstáculo a posição inicial.
                RET

;Rotina que apaga da direita para a esquerda o obstáculo
Apaga_geral_obs:MOV     R3,R1 ;Recebe uma posição em R1 e subtitui por espaços todos os outros asteriscos
                SUB     R3, 0003h
apaga_obs_geral:MOV     M[IO_CONTROL],R1
                MOV     R2, ESPACO
                MOV     M[IO_WRITE],R2
                SUB     R1, COLUNAS
                CMP     R1,R3
                BR.NZ   apaga_obs_geral
                ADD     R1, COLUNAS ; Repoe a posicao do obstáculo
                RET
;Rotina que reescreve o obstáculo
Esc_geral_2:    ADD     R1, MUDA_LINHA
                MOV     R3,R1
                ADD     R3, 0003h
conti_2:        MOV     M[IO_CONTROL],R1
                MOV     R2, OBS
                MOV     M[IO_WRITE],R2
                ADD     R1, COLUNAS
                MOV     R4,R1 ; Coloca no registo a posicao a ser escrita de forma a comparar se existem
colisoes
                CALL    colisao
notcolisao:     CMP     R1,R3
                BR.NZ   conti_2
                SUB     R1, COLUNAS
fim_esc_2:      RET

;                   VERIFICA SE EXISTE COLISÃO
;---------------------------------------------------------------
colisao:        MOV     R6, R4
                AND     R6, FF00h  ; verifica se alguma das linhas em que está o obstáculo este pode
colidir com a BIC
                CMP     R6, 1400h ; Caso alguma desta condições se verifique, verificam-se as colunas
```

```
                BR.P    colidecolunas
                BR      fimc ; Caso nenhuma das linhas satisfaça a condição não é necessário verificar as
colunas
colidecolunas:  MOV     R5, M[POSICAOBIC]
                MOV     R6, R4
                DEC     R6
                AND     R6, 00FFh ;Verifica somente as colunas anulando as linhas
                AND     R5, 00FFh
                CMP     R6,R5 ; Caso se verifique existe colisão
                CALL.Z  Hacolisao
fimc:           RET

Hacolisao:      PUSH    R1
                MOV     R1, COLISAO     ;move o valor que indica que houve colisão para uma variável que
será posteriormente testada
                MOV     M[COLISAOCHECK],R1
                POP     R1
                RET
;-------------------------------------------------------------

Obs1_esc:       PUSH    R1
                PUSH    R2
                PUSH    R3
                CALL    random ; Chama a rotina que cria uma coordenada aleatória
                MOV     R1, M[POS] ; Coloca essa coordenada aleatória no registo
                MOV     M[POSICAO_OBS1],R1 ; Posteriormente coloca essa coordenada na variável na posição
do obstáculo
                CALL    Esc_geral_obs ; Escreve o obstáculo pela primeira vez
                MOV     M[POSICAO_OBS1], R3 ; Actualiza a posição do obstáculo da memória
                POP     R3
                POP     R2
                POP     R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs2_esc:       PUSH    R1
                PUSH    R2
                PUSH    R3
                CALL    random
                MOV     R1, M[POS]
                MOV     M[POSICAO_OBS2],R1
                CALL    Esc_geral_obs
                MOV     M[POSICAO_OBS2], R3
                POP     R3
                POP     R2
                POP     R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs3_esc:       PUSH    R1
                PUSH    R2
                PUSH    R3
                CALL    random
                MOV     R1, M[POS]
                MOV     M[POSICAO_OBS3],R1
                CALL    Esc_geral_obs
                MOV     M[POSICAO_OBS3], R3
                POP     R3
                POP     R2
                POP     R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs4_esc:       PUSH    R1
```

```
                PUSH    R2
                PUSH    R3
                CALL    random
                MOV     R1, M[POS]
                MOV     M[POSICAO_OBS4],R1
                CALL    Esc_geral_obs
                MOV     M[POSICAO_OBS4], R3
                POP     R3
                POP     R2
                POP     R1
                RET

Obs1_apg:       MOV     R1, M[POSICAO_OBS1] ; coloca a posição do obstáculo no registo
                CALL    Apaga_geral_obs ; Chama a rotina que apaga o obstáculo
                MOV     M[POSICAO_OBS1], R1 ; Repoe o valor da posição do obstáculo
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs2_apg:       MOV     R1, M[POSICAO_OBS2]
                CALL    Apaga_geral_obs
                MOV     M[POSICAO_OBS2], R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs3_apg:       MOV     R1, M[POSICAO_OBS3]
                CALL    Apaga_geral_obs
                MOV     M[POSICAO_OBS3], R1
                RET

;Idêntico ao anterior só variando a posição do obstáculo
Obs4_apg:       MOV     R1, M[POSICAO_OBS4]
                CALL    Apaga_geral_obs
                MOV     M[POSICAO_OBS4], R1
                RET

Obs1_esc_2:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                CALL    Obs1_apg ; Apaga a actual posição do obstáculo
                MOV     R1, M[POSICAO_OBS1] ; Coloca a posicao do obstáculo no registo
                CALL    muda_posicao ; Verica se o obstáculo está na última linha
                CMP     R1, M[POSICAO_OBS1]
                BR.NZ   new ; Verifica se a posição do obstáculo se alterou
                CALL    Esc_geral_2 ;Caso não se tenha alterado, escreve o obstáculo numa próxima linha
                JMP     fimd ; Termina o escreve obstáculos
new:            CALL    Esc_geral_obs ; Caso exista uma nova posição, escreve-se o obstáculo na primeira
linha referente à posição gerada
                CALL    Obs_count_esc ; Visto que o obstáculo anterior tinha chegado à última linha,
então conta-se um obstáculo para o contador
fimd:           MOV     M[POSICAO_OBS1], R1 ; Repõe-se a posição do obstáculo
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET

;VERIFICA SE É NECESSÁRIA UMA NOVA POSIÇÃO PARA O OBSTÁCULO E GERA-A
;--------------------------------------------------------------------
muda_posicao:   MOV     R4, R1
                AND     R4, FF00h; Verifica se apesar das colunas o obstáculo está na última linha
                CMP     R4, 1700h
                BR.Z    novaposicao ; Caso se verifique a condição é gerada uma nova posição
```

```
                RET
novaposicao:    CALL    random ; nova posição
                MOV     R1, M[POS]
                RET
;-----------------------------------------------------------------
;Idêntico ao anterior só variando a posição do obstáculo
Obs2_esc_2:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                CALL    Obs2_apg
                MOV     R1, M[POSICAO_OBS2]
                CALL    muda_posicao
                CMP     R1, M[POSICAO_OBS2]
                BR.NZ   new2
                CALL    Esc_geral_2
                JMP     fimd2
new2:           CALL    Esc_geral_obs
                CALL    Obs_count_esc
fimd2:          MOV     M[POSICAO_OBS2], R1
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs3_esc_2:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                CALL    Obs3_apg
                MOV     R1, M[POSICAO_OBS3]
                CALL    muda_posicao
                CMP     R1, M[POSICAO_OBS3]
                BR.NZ   new3
                CALL    Esc_geral_2
                JMP     fimd3
new3:           CALL    Esc_geral_obs
                CALL    Obs_count_esc
fimd3:          MOV     M[POSICAO_OBS3], R1
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET
;Idêntico ao anterior só variando a posição do obstáculo
Obs4_esc_2:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                CALL    Obs4_apg
                MOV     R1, M[POSICAO_OBS4]
                CALL    muda_posicao
                CMP     R1, M[POSICAO_OBS4]
                BR.NZ   new4
                CALL    Esc_geral_2
                JMP     fimd4
new4:           CALL    Esc_geral_obs
                CALL    Obs_count_esc
fimd4:          MOV     M[POSICAO_OBS4], R1
```

```
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET
;---------------------------------------------------------
;-------------------------Repõe---------------------------
;Repõe todos os valores necessários a um novo jogo, e chama a
;rotina que actualiza o valor da da pontuação máxima atingida
;pelo o utilizador.--------------------------------------
;---------------------------------------------------------

repoe:          DSI     ; Enibe as interrupções
                CALL    mantem_max ; Verifica se é necessário actualizar o LCD para a posição máxima
atingida
                MOV     R1, 0000h
                MOV     M[ONOFF], R1 ; Desliga o contador
                MOV     M[io_led], R1 ; Desliga os leds
                MOV     M[OBS_COUNT],R1 ; Repoe o valor dos obstáculos ultrapassados
                MOV     M[contatemp], R1 ; Repoe o valor do contador
                MOV     M[TURBOONOFF], R1 ; Repoe o valor do turbo
                MOV     R1, 0001h
                MOV     M[COLISAOCHECK], R1 ; Repoe o valor que indica que não existe colisão
                MOV     R1, NIVEL1
                MOV     M[VEL], R1 ; Repoe o valor da velocidade para o nível 1
                MOV     R1, BIC_INIT
                MOV     M[POSICAOBIC], R1 ; Repoe o valor inicial da bicicleta
                MOV     R1, R0 ; Repoe R1
                MOV     R7, R0 ; Repoe R7
                RET
;---------------------------------------------------------
;-------------------------LCD WRITER----------------------
;Escreve no LCD a pontuação actual bem como a posição máxima
;que o utilizador já atingiu. Converte o número de hexadecimal
;para decimal de forma a ser escrito no LCD.-------------------
;---------------------------------------------------------
lcd:            CALL    linha_cima ; Escreve a primeira linha do LCD
                PUSH    R1
                MOV     R1, R0
                CMP     R1, M[MAX_COUNT]  ; se o contador for superior a 0 então não se volta a escrever
a linha de baixo
                BR.NZ   lcd2
                CALL    linha_de_baixo ; Escreve a segunda linha do LCD
lcd2:           CALL    vel_led ;verifica-se se é necessário actualizar os leds e a velocidade
                CMP     M[contatemp], R0
                CALL.P  Aumenta_dist ;se o valor do contatemp for maior que 0 entao ele actualiza a
distancia percorrida.
                POP     R1
                RET

PAUSA_LCD:      PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, PAUSASTR    ;coloca-se no R1 a string da pausa
                MOV     R3, 8004h       ;coordenada em R3 do primeiro caracter
continua_pausa_lcd:MOV  R2, M[R1]       ;passa-se o primeiro caracter para R2
                MOV     M[lcd_cursor], R3
                MOV     M[lcd_write], R2
                CMP     R3, 800Ah       ;ve se já escreveu o último caracter
                BR.Z    Fim_lcd_pausa
                INC     R1              ;incrementa-se o indice de forma a obter o proximo caracter
                INC     R3              ;passa para a seguinte coluna
```

```
                BR      continua_pausa_lcd
Fim_lcd_pausa:  POP     R3
                POP     R2
                POP     R1
                RET

APAGA_LCD_1:    PUSH    R1
                PUSH    R2
                MOV     R1, 8000h
                MOV     R2, ESPACO      ;substitui-se todas as posições da primeira linha por espaços
apaga_lcd_cont: MOV     M[lcd_cursor], R1
                MOV     M[lcd_write], R2
                CMP     R1, 800Fh
                BR.Z    fim_apaga_lcd
                INC     R1
                JMP     apaga_lcd_cont
fim_apaga_lcd:  POP     R2
                POP     R1
                RET

Aumenta_dist:   PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3,R0
                MOV     R1, M[contatemp]
                MOV     R2, 000Ah       ;passa o valor 10 para o R2
                DIV     R1, R2  ;divide o valor em hex do contatemp por R2 de forma a converter para
decimal posteriormente
                MOV     R3, R2  ;guarda o valor em decimal no R3, correspondente ao resto da divisão
                CMP     R1, 0000h ; verifica se o valor da divisão por 10 é 0
                BR.Z    fim_conversao ;caso seja termina a conversao
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 4 ; multiplica o resultado de forma a colocar o algarismo na posição
correcta, neste caso das dezenas
                ADD     R3, R2 ;Soma-se esse valor ao R3
                CMP     R1, 0000h
                BR.Z    fim_conversao
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 8 ;multiplica o resultado de forma a colocar o algarismo na posição correcta,
neste caso das centenas
                ADD     R3, R2
                CMP     R1, 0000h
                BR.Z    fim_conversao
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 12  ;multiplica o resultado de forma a colocar o algarismo na posição
correcta, neste caso dos milhares
                ADD     R3, R2
fim_conversao:  MOV     M[DECIMAL], R3 ; guarda o valor de R3 numa variável
                CALL    converte_str_e ; traduz o número para ASCII de forma a poder escreve-lo no
contador como se fosse um caracter
                POP     R3
                POP     R2
                POP     R1
                RET

converte_str_e: PUSH    R1;
                PUSH    R2
                PUSH    R3
```

```
                MOV     R3, M[DECIMAL] ; coloca o valor em decimal no registo R3
                MOV     R1, R3; escreve o 1º digito
                AND     R1, 000Fh ; Obtem-se somente o algarismo das unidades
                ADD     R1, '0' ; converte-se para um caracter
                MOV     R2, 800Eh ; Coloca-se na posição do LCD correspondente às unidades
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 2º digito
                AND     R1, 00F0h ; Obtem-se somente o algarismo das dezenas
                SHR     R1, 4 ; coloca-se esse algarismo nas unidades
                ADD     R1, '0'
                DEC     R2 ; Coloca-se na coluna à esquerda do algarismo anteriormente colocado
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 3º digito
                AND     R1, 0F00h ; Obtem-se somente o algarismo das centenas
                SHR     R1, 8 ; Coloca-se esse algarismo nas unidades
                ADD     R1, '0'
                DEC     R2 ; Coloca-se na coluna à esquerda do algarismo anteriormente colocado
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 4ºdigito
                AND     R1, F000h ;Obtem-se o algarismo dos milhares
                SHR     R1, 12 ; Coloca-se esse algarismo nas unidades
                ADD     R1, '0'
                DEC     R2 ; Coloca-se na coluna à esquerda do algarismo anteriormente colocado
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                POP     R3
                POP     R2
                POP     R1
                RET

linha_cima:     PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3, DIST ; Move-se a string para o R3
                MOV     R1, 8000h ; Primeira posicação a escrever no LCD
conti_lcd:      MOV     R2, M[R3];coloca-se o primeiro caracter da string em R2
                MOV     M[lcd_cursor],R1
                MOV     M[lcd_write], R2
                CMP     R1, 800Fh
                BR.Z    fim_lcd
                INC     R1 ; Coloca-se mais uma coluna para escrever o próximo caracter
                INC     R3 ; Incrementa-se o Indice de forma a obter o próximo caracter
                JMP     conti_lcd ;Continua a escrever-se
fim_lcd:        POP     R3
                POP     R2
                POP     R1
                RET

;Igual ao código anterior só mudando a string a escrever
linha_de_baixo: PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R3, MAX
                MOV     R1, 8010h
lb_conti:       MOV     R2, M[R3]
                MOV     M[lcd_cursor],R1
                MOV     M[lcd_write], R2
```

```
                CMP     R1, 801Ch
                BR.Z    fim_lcd_lb
                INC     R1
                INC     R3
                BR      lb_conti
fim_lcd_lb:     POP     R3
                POP     R2
                POP     R1
                RET

;Verifica se é necessário actualizar a pontuação máxima obtida no final do jogo para o LCD
mantem_max:     PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                MOV     R3, M[DECIMAL] ; Move-se o número decimal obtido no final do jogo para R3
                MOV     R4, M[MAX_COUNT] ; Move-se o número decimal que corresponde à pontuação máxima
atingida até ao momento
                CMP     R3, R4 ; Verifica-se se a actual é maior que a anterior
                JMP.NP  fim_max ; não actualiza
                MOV     R1, R3; Converte o número decimal para caracteres  e escreve-os nas posições
correctas como já indicado anteriormente no converte_str_e
                AND     R1, 000Fh
                ADD     R1, '0'
                MOV     R2, 801Bh
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 2º digito
                AND     R1, 00F0h
                SHR     R1, 4
                ADD     R1, '0'
                DEC     R2
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 3º digito
                AND     R1, 0F00h
                SHR     R1, 8
                ADD     R1, '0'
                DEC     R2
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     R1,R3;começa a escrever o 4ºdigito
                AND     R1, F000h
                SHR     R1, 12
                ADD     R1, '0'
                DEC     R2
                MOV     M[lcd_cursor],R2
                MOV     M[lcd_write], R1
                MOV     M[MAX_COUNT], R3; actualiza o valor correspondente à pontuação obtida até ao
momento
fim_max:        POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET
;---------------------------------------------------------------
;-------------------CONTADOR DE OBSTÁCULOS--------------------
; - Escreve o número de obstáculos ultrapassados pelo user no
;display de 7 segmentos. Converte também o número para decimal.
;---------------------------------------------------------------
;Coloca todos os displays a 0
```

```
Obs_Init_esc:   PUSH    R1
                MOV     R1, 0
                ADD     R1, '0' ; converte para caracter
                MOV     M[DISP7s1], R1
                MOV     M[DISP7s2], R1
                MOV     M[DISP7s3], R1
                MOV     M[DISP7s4], R1
                POP     R1
                RET

Obs_count_esc:  PUSH    R1
                PUSH    R2
                PUSH    R3
                MOV     R1, M[OBS_COUNT] ; Move o contador de obstáculos para R1
                INC     R1 ; Adiciona o valor a R1 pois existe um obstáculo que passou
                MOV     M[OBS_COUNT], R1 ; Actualiza o contador de obstáculos
                MOV     R3,R0
                MOV     R2, 000Ah ; Converte-se o número para decimal como já explicado anteriormente
                DIV     R1, R2
                MOV     R3, R2
                CMP     R1, 0000h
                BR.Z    fim_conversao2
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 4
                ADD     R3, R2
                CMP     R1, 0000h
                BR.Z    fim_conversao2
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 8
                ADD     R3, R2
                CMP     R1, 0000h
                BR.Z    fim_conversao2
                MOV     R2, 000Ah
                DIV     R1, R2
                SHL     R2, 12
                ADD     R3, R2
fim_conversao2: MOV     M[OBS_COUNT_DECIMAL], R3 ; actualiza-se o valor o contador de obstáculos em
decimal
                CALL    converte_str_e_OBS ; converte-se o valor dos obstáculos de forma a poder escrever
no display de 7 seg.
                POP     R3
                POP     R2
                POP     R1
                RET
;Os comentários que são idênticos a rotinas anteriores não estão apresentados de forma a facilitar a
leitura do código
converte_str_e_OBS:PUSH R1;
                PUSH    R2
                PUSH    R3
                MOV     R3, M[OBS_COUNT_DECIMAL] ; Move-se o valor para o registo 3
                MOV     R1, R3;escreve o 1º digito
                AND     R1, 000Fh
                ADD     R1, '0' ;converte-se num caracter
                MOV     M[DISP7s1], R1;Adiciona-se ao primeiro display
                MOV     R1,R3;começa a escrever o 2º digito
                AND     R1, 00F0h
                SHR     R1, 4
                ADD     R1, '0'
```

```
                MOV     M[DISP7s2], R1
                MOV     R1,R3;começa a escrever o 3° digito
                AND     R1, 0F00h
                SHR     R1, 8
                ADD     R1, '0'
                MOV     M[DISP7s3], R1
                MOV     R1,R3;começa a escrever o 4°digito
                AND     R1, F000h
                SHR     R1, 12
                ADD     R1, '0'
                MOV     M[DISP7s4], R1
                POP     R3
                POP     R2
                POP     R1
                RET

;--------------------------------------------------------------
;------------SEQUENCIA ALEATORIA DE COORDENADAS---------------
;--Gera de forma Pseudoaleatoria segundo o algoritmo em Anexo
;uma uma coordenada para o objecto cair.----------------------
;--------------------------------------------------------------

random:         PUSH    R1
                PUSH    R2
                MOV     R1, M[NVAR1]
                MOV     R2, R1
                TEST    R1, 0001h ;verifica se o bit menos signficativo está a 1
                BR.NZ   random_1 ;caso não esteja salta para a segunda opção que gera uma coordenada
aleatória
                SHR     R1, 1 ;
                MOV     M[NVAR1], R1 ; actualiza-se o valor de NVAR1
                BR      gen_coord ; gera uma nova coordenada dentro dos limites
random_1:       XOR     R1, MASK ; Gera-se um XOR de R1 com a MASK indicada
                SHR     R1, 1
                MOV     M[NVAR1], R1
gen_coord:      MOV     R2, 0014h ; move-se 20 para R2
                DIV     R1, R2 ; divide-se o valor obtido de forma aleatória por 20; de forma a obter um
valor dentro das primeira 20 colunas
                ADD     R2, 001Fh ; adiciona-se ao resto o valor 31 de forma a que a primeira posição do
obstáculo fique dentro dos limites das paredes
                MOV     M[POS], R2 ; coloca-se essa coordenada na variável POS que será posteriormente
usada quando é necessário escrever um obs
                POP     R2
                POP     R1
                RET

;--------------------------------------------------------------
;-------------ALTERA LEDS + VELOCIDADES DE JOGO--------------
;Altera os leds e a velocidade do jogo consoante o número de
;obstáculos ultrapassados.-----------------------------------
;--------------------------------------------------------------

vel_led:        PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R7
                MOV     R3, M[TURBOONOFF]
                CMP     R3, 0001h ; verifica se o turbo está ou não ligado
                JMP.Z   fim_leds ; se tiver não altera nem velocidades nem leds
                MOV     R1, M[OBS_COUNT]
                CMP     R1, 0003h ; verica-se se já foram passados 4 obstáculos
                BR.NP   nivel1 ; se não, mantem nivel
```

```
                CMP     R1, 0004h ; verifica se foram passados os 4 obstaculos
                BR.Z    nivel2 ;se sim, activa nivel 2
                CMP     R1, 0007h ; verifica-se se já passados 8 obstáculos
                BR.NP   nivel2 ;se não, nivel 2
                BR      nivel3 ; activa nivel 3
                JMP     fim_leds
nivel2:         MOV     R7, FF00h ; Colocam-se os 8 bits mais significativos a 1 e os restantes a zero
                MOV     M[io_led], R7 ; acende-se os 8 primeiros leds do lado esq
                MOV     R2, NIVEL2 ; Coloca-se a velocidade correspondente ao nivel2 no R2
                MOV     M[VEL], R2 ; coloca-se esse valor na variável VEL
                JMP     fim_leds
nivel1:         MOV     R7, F000h ; acende-se os 4 primeiros leds
                MOV     M[io_led], R7
                MOV     R2, NIVEL1 ; coloca-se a velocidade correspondente ao nivel1 no R2
                MOV     M[VEL], R2 ; coloca-se esse valor na variavel VEL
                JMP     fim_leds
nivel3:         MOV     R7, FFF0h ;Colocam-sem os 12 bits mais significativos a 1 e os restantes a zero
                MOV     M[io_led], R7 ; acende-se os 12 primeiros leds do lado esq
                MOV     R2, NIVEL3 ;Coloca-se a velocidade correspondente ao nivel2 no R2
                MOV     M[VEL], R2 ; coloca-se esse valor na variável VEL
fim_leds:       POP     R7
                POP     R3
                POP     R2
                POP     R1
                RET

;---------------------------------------------------------------
;----------------------------PAUSA------------------------------
;Coloca o jogo em pausa desligando o temporizador.------------
;---------------------------------------------------------------

Pausa:          MOV     R7, R0 ; repoe o valor de R7 a 0
                MOV     M[ONOFF], R0 ; desliga o temporizador
                CALL    APAGA_LCD_1
                CALL    PAUSA_LCD
conti_pausa:    CMP     R7, 0023h ; verifica se foi acionada a interrupão
                BR.NZ   conti_pausa ; se não, continua em pausa
                MOV     R7, M[VEL] ; volta a colocar-se a velocidade no R7
                MOV     M[TIMER_COUNT], R7 ; adiciona-se ao timer_count para indicar ao temporizador a
velocidade
                MOV     R7, 0001h
                MOV     M[TIMER_ONOFF], R7 ; activa-se o temporizador
                MOV     M[ONOFF], R7 ; activa-se o ONOFF para ser utilizado na interrupção do temporizador
                MOV     R7, R0 ; repoe-se R7 a 0
                RET


;---------------------------------------------------------------
;----------------------------TURBO------------------------------
;----------------Coloca o jogo no modo TURBO ----------------
;---------------------------------------------------------------
Turbo:          PUSH    R1
                PUSH    R3
                MOV     R7, R0
                MOV     R1, NIVELT ; transfere a velocidade do turbo para turbo
                CMP     M[VEL], R1 ; vê se a velocidade presente no turbo é a indicada anteriormente
                BR.Z    desactiva ; caso seja repoe os valores e desliga turbo
                MOV     R3, 0001h
                MOV     M[TURBOONOFF], R3
                MOV     M[VEL], R1 ; actualiza a velocidade para turbo
                MOV     R1, FFFFh ; 16 bits a 1
```

```
                MOV     M[io_led], R1 ; acende todos os LEDS
                JMP     fim_turbo
desactiva:      MOV     R3, 0000h
                MOV     M[TURBOONOFF], R3 ;Desliga-se o turno
fim_turbo:      MOV     R7, R0 ; Repoe-se R7
                POP     R3
                POP     R1
                RET

;-----------------------------------------------------------------
;----------------------PROGRAMA PRINCIPAL-----------------------
;--Funcionamento principal do jogo + Rotinas Mov de Obstáculos
;-----------------------------------------------------------------

progprincipal:  MOV     R7, SP_INICIAL
                MOV     SP, R7
                MOV     R7, INTMASK ; move a intmask com as inteerrupções que estarão activas para o R7
                MOV     M[INTADDR],R7 ; move o r7 para o INTADDR ficando assim as interrupções activas
                MOV     R7, LIMPA_J ;
                MOV     M[IO_CONTROL], R7 ;Inicia-se o cursor da janela
                CALL    Texto_Init ; Desenha a mensagem de boas vindas
                ENI ; Activa as interrupções
START:          MOV     R2, M[NVAR1]; move-se o valor da variável para R2
                INC     R2 ; incrementa-se R2
                MOV     M[NVAR1], R2 ; Volta-se a colocar o valor já incrementado na variável
                CMP     R7, 0001h ; verifica-se se foi pressionado o botão I1
                BR.NZ   START ; caso não tenha sido volta-se ao ciclo de forma a criar um NVAR1 que
depende do tempo de espera em que se carrega no I1
NEWGAME:        CALL    Inicio ; Desenha as paredes
                CALL    Obs_Init_esc ; Desenha no display de 7 seg '0000'
                CALL    desenha_bicicleta_INIT ; Desenha a Bicicleta
                CALL    Obs1_esc ; Cria o primeiro obstáculo
                JMP     ACTIVATEMPORIZADOR ;Inicia o temporizador
MODOPLAY:       CMP     R7, 0023h
                CALL.Z  Pausa ; Verifica se o botão IA foi pressionado, se sim, chama a pausa
                CMP     R7, 0098h
                CALL.Z  Turbo; Verifica se o botão I2 foi pressionado, se sim, chama o turbo
                CMP     R1,M[contatemp]
                BR.P    MOVEOBSTACULOS  ;se o R1 for maior do que o seu valor anterior chama-se o
moveobstáculos
                CMP     R7, 0004h
                BR.NZ   MOVEDIREITA ; caso se carregue no botão I0 move-se a bic para a esquerda
                CALL    desenha_esq
                BR      MODOPLAY
MOVEDIREITA:    CMP     R7, 0003h
                BR.NZ   MODOPLAY
                CALL    desenha_dir ; caso se carregue no botão IB move-se a bic para a direita
                BR      MODOPLAY ; Volta-se ao decorrer do jogo
;Faz os obstaculos cairem continuamente
MOVEOBSTACULOS: MOV     M[contatemp], R1;
                CALL    lcd ;actualiza o LCD
                CMP     M[CONT1], R1
                JMP.P   MOVOBS1 ;Caso o valor seja inferior à posicao de onde cai o segundo obstaculo
continua a mover obs1
                CMP     M[CONT1], R1
                JMP.Z   MOVOBS1ESC2 ;Caso o valor seja igual à posicao de onde cai o segundo obstaculo
move obs1 e cria um novo obs2
                CMP     M[CONT2], R1
                BR.P    MOVOBS12
                CMP     M[CONT2], R1
                JMP.Z   MOVOBS12ESC3 ;Caso o valor seja igual à posicao de onde cai o terceiro obstaculo
```

```
move obs1/2 e cria um novo obs3
                CMP     M[CONT3], R1
                JMP.P   MOVOBS123
                CMP     M[CONT3], R1
                JMP.Z   MOVOBS123ESC4 ;Caso o valor seja igual à posicao de onde cai o quardo obstaculo
move obs1/2/3 e cria um novo obs4
                CALL    Obs1_esc_2
                CALL    Obs2_esc_2
                CALL    Obs3_esc_2
                CALL    Obs4_esc_2 ;caso seja superior a qualquer posição, então move cada obstáculo,
visto que estas rotinas já têm implementadas a função de criarem novos obstáculos caso o obstáculo em
causa chegue ao fim da última linha
                MOV     R2, 0001h
                CMP     R2, M[COLISAOCHECK] ; verifica se existiu alguma colisão
                JMP.Z   MODOPLAY ;caso não exista continua o jogo
                JMP     Fim_do_Jogo ; caso exista termina o jogo
MOVOBS12:       CALL    Obs1_esc_2
                CALL    Obs2_esc_2
                JMP     MODOPLAY
MOVOBS1:        CALL    Obs1_esc_2
                JMP     MODOPLAY
MOVOBS1ESC2:    CALL    Obs1_esc_2
                CALL    Obs2_esc
                JMP     MODOPLAY
MOVOBS12ESC3:   CALL    Obs1_esc_2
                CALL    Obs2_esc_2
                CALL    Obs3_esc
                JMP     MODOPLAY
MOVOBS123:      CALL    Obs1_esc_2
                CALL    Obs2_esc_2
                CALL    Obs3_esc_2
                JMP     MODOPLAY
MOVOBS123ESC4:  CALL    Obs1_esc_2
                CALL    Obs2_esc_2
                CALL    Obs3_esc_2
                CALL    Obs4_esc
                JMP     MODOPLAY

;TERMINA O JOGO E REPÕE OS VALORES DE ORIGEM PARA UM NOVO JOGO
Fim_do_Jogo:    CALL    repoe ; repoe os valores de origem
                CALL    Texto_Init2 ; cria a mensagem de fim do jogo
                ENI ; activa as interrupções
                MOV     R7, R0 ;repoe o R7
NewGame:        CMP     R7, 0001h ; verifica se o R1 foi pressionado
                BR.NZ   NewGame
                JMP     NEWGAME ; quando for pressionado começa um novo jogo

;INICIA O TEMPORIZADOR
ACTIVATEMPORIZADOR:MOV  R7, 0001h
                MOV     M[ONOFF], R7;activa o temporizador
                MOV     R7, F000h ;liga os primeiros 4 leds da esquerda
                MOV     M[io_led], R7
                MOV     R7, M[VEL]
                MOV     M[TIMER_COUNT], R7;inicia o temporizador com a velocidade definida pela variável
                MOV     R7, M[ONOFF]
                MOV     M[TIMER_ONOFF], R7;Liga o temporizador
                JMP     MODOPLAY ; Começa o jogo
```