```
; Programa "Corridas de Bicicleta"

;                                   ZONA REGISTOS
;
; R7:   Posicao atualizada da bicicleta.

;_____________________________________________________________________________

;                                   ZONA EQU
;

INITIAL_POS     EQU     152Ch                   ; Linha 21, coluna 44
INT_MASK_ADDR   EQU     FFFAh
INT_MASK        EQU     1000110000000111b       ;1000 1100 0000 0111b
IO_DISPLAY      EQU     FFF0h
IO_INIC         EQU     FFFFh
IO_READ         EQU     FFFCh
IO_WRITE        EQU     FFFEh
LCD_CONTROL     EQU     FFF4h
LCD_WRITE       EQU     FFF5h
LCD_0_0         EQU     8000h                   ; Linha 0, coluna 0 do LCD
LCD_1_0         EQU     8010h
LCD_0_15        EQU     800Eh
LCD_0_16        EQU     800Fh
LCD_1_11        EQU     801Bh
LCD_1_12        EQU     801Ch
LEDS            EQU     FFF8h
LEFT_BORDER     EQU     1521h                   ; linha 21, coluna 33
LINE            EQU     0100h
MASK_RANDOM     EQU     1000000000010110b       ;1000 0000 0001 0110b
F_NOTE1_POS     EQU     0C28h                   ; linha 12, coluna 40
F_NOTE2_POS     EQU     0E1Ch                   ; linha 14, coluna 28
METERS          EQU     'm'
NOTE1_POS       EQU     0C1Ch                   ; linha 12, coluna 27
NOTE2_POS       EQU     0E1Bh                   ; linha 14, coluna 26
OBST_DIFF       EQU     0006h
PAUSE_POS       EQU     0D2Ah                   ; linha 13, coluna 42
RECT_BOTTOM     EQU     1100h                   ; linha 17
RECT_TOP        EQU     0900h                   ; linha 9
RIGHT_BORDER    EQU     1538h                   ; linha 21, coluna 56
SP_INITIAL      EQU     FDFFh
STRING_END      EQU     '@'
TIMER_ACTIV     EQU     FFF7h
TIMER_VALUE     EQU     FFF6h

;_____________________________________________________________________________

;                                   ZONA WORD
;
                ORIG    8000h
BIKE_POS        WORD    0000h
COLISION_IND    WORD    0000h
CREATE_FLAG     WORD    0001h
DISTANCE        WORD    0000h
DIST_DECIMAL    WORD    0000h
MAX_DIST        WORD    0000h
NEW_OBST_POS    WORD    0000h
OBST_COUNTER    WORD    0000h
OBSTACLE_FLAG   WORD    0000h
OVERCOMED       WORD    0000h
PACE            WORD    0000h
PAUSE_FLAG      WORD    0000h
```



```
PSEUDO          WORD    0000h
START_FLAG      WORD    0000h
TURBO_FLAG      WORD    0000h
TURBO_STATE     WORD    FFFFh

;_____________________________________________________________________________

;                                   ZONA STR
;

BIKE            STR     'O|O', STRING_END
DEL_BIKE        STR     '   ', STRING_END
DEL_PAUSE       STR     '     ', STRING_END
DEL_STRING      STR     '
                         ', STRING_END
DEL_OBSTACLE    STR     '   ', STRING_END
F_NOTE1         STR     'Fim do Jogo', STRING_END
F_NOTE2         STR     'Prima o interruptor I1 para recomecar', STRING_END
LCD_DIST        STR     'Distancia:     m', STRING_END
LCD_MAX         STR     'Maximo:', STRING_END
LINE_WALL       STR     '||||||||||||||||||||||', STRING_END
NOTE1           STR     'Bem-vindo a Corrida de Bicicleta!', STRING_END
NOTE2           STR     'Prima o interruptor I1 para comecar', STRING_END
OBSTACLE        STR     '***', STRING_END
PAUSE_NOTE      STR     'PAUSE', STRING_END
PLUS_WALL       STR     '++++++++++++++++++++++++', STRING_END

;_____________________________________________________________________________

;                                   ZONA TAB
;

OBSTACLE_POS    TAB     4               ; Numero maximo de obstaculos = 4
DIST_DEC_DIG    TAB     4
MAX_DEC_DIG     TAB     4

;_____________________________________________________________________________

;                               Tabela de Interrupcoes
;
                ORIG    FE00h
INT0            WORD    TURN_LEFT
INT1            WORD    START_MAIN
INT2            WORD    TURBO
                ORIG    FE0Ah
INTA            WORD    PAUSE
INTB            WORD    TURN_RIGHT
                ORIG    FE0Fh
TEMP            WORD    OBSTACLE_IND


;_____________________________________________________________________________

                ORIG    0000h
                JMP     MAIN

;                                   INTERRUPCOES
;

; OBSTACLE_IND: Rotina que indica quando mover os obstaculos.
;               Entradas: ---
;               Saídas: ---
```

```
;               Efeitos: Altera valor na posicao OBSTACLE_FLAG
;
OBSTACLE_IND:   INC     M[OBSTACLE_FLAG]
                CALL    TIMER_INIT
                RTI


; PAUSE: Rotina que permite efetuar uma pausa no jogo.
;               Entradas: ---
;               Saídas: ---
;               Efeitos: Altera valor na posicao PAUSE_FLAG
;
PAUSE:          INC     M[PAUSE_FLAG]
                RTI


; START_MAIN: Rotina que começa o jogo.
;               Entradas: ---
;               Saídas:
;               Efeitos: Altera valor na posicao START_FLAG
;
START_MAIN:     INC     M[START_FLAG]
                RTI


; TURBO: Rotina que permite entrar no modo TURBO.
;               Entradas: ---
;               Saídas: ---
;               Efeitos: Altera valor na posicao TURBO_FLAG
;
TURBO:          INC     M[TURBO_FLAG]
                RTI


; TURN_LEFT:    Rotina que indica que a bicicleta deve virar a esquerda.
;               Entradas: ---
;               Saídas: ---
;               Efeitos: Decrementa R7
;
TURN_LEFT:      DEC     R7
                RTI


; TURN_RIGHT:   Rotina que indica que a bicicleta deve virar a direita.
;               Entradas: ---
;               Saídas: ---
;               Efeitos: Incrementa R7
;
TURN_RIGHT:     INC     R7
                RTI

;_____________________________________________________________________________

;                                   ROTINAS
;


; ASCII_CONVERT: Rotina que converte um digito decimal em ASCII.
;               Entradas: Pilha - Recebe o valor da posicao onde colocar o
; resultado, e o valor a ser convertido
;               Saidas: Memoria - valor convertido
;               Efeitos: ---
```



```
;
ASCII_CONVERT:  PUSH    R1
                PUSH    R2

                MOV     R1, M[SP+4]
                MOV     R2, M[SP+5]

                ADD     R1, '0'
                MOV     M[R2], R1

                POP     R2
                POP     R1
                RETN    2


; COLISION: Rotina que verifica se ocorreu colisao da bicicleta com
; um obstaculo.
;               Entradas: Memoria - OBSTACLE_POS
;               Saidas: ---
;               Efeitos: Pode alterar o valor da posicao COLISION_IND
;
COLISION:       PUSH    R1
                PUSH    R2
                PUSH    R3

                MOV     R3, 3
                MOV     R1, M[OBSTACLE_POS]

FC_CICLE2:      MOV     R2, 3

FC_CICLE:       CMP     R7, R1
                BR.Z    COLIDED

                INC     R1              ; Percorrer caracteres do obstaculo.
                DEC     R2

                CMP     R2, R0
                BR.NZ   FC_CICLE

                SUB     R1, LINE        ; Com isto, a rotina permite verificar
                SUB     R1, 3           ; choques laterais em toda a bicicleta.
                DEC     R3              ;

                CMP     R3, R0
                BR.NZ   FC_CICLE2

COLI_END:       POP     R3
                POP     R2
                POP     R1
                RET

COLIDED:        INC     M[COLISION_IND] ; Ativacao de indicador de colisao
                BR      COLI_END        ; com obstaculo.


; CREATE: Rotina que verifica se se deve criar um novo obstaculo.
;               Entradas: Memoria - OBST_COUNTER
;               Saidas: ---
;               Efeitos: Pode alterar os valores da posicao CREATE_FLAG e da
; tabela OBSTACLE_POS
;
CREATE:         PUSH    R1
```

```
                PUSH    R2

                DEC     M[OBST_COUNTER]         ; Começa com o valor de
                CMP     M[OBST_COUNTER], R0     ; OBST_DIFF. Quando chegar a
                BR.NZ   C_END                   ; zero e gerado um novo
                                                ; obstaculo.
                MOV     R1, OBST_DIFF
                MOV     M[OBST_COUNTER], R1

                MOV     R2, OBSTACLE_POS
                MOV     R1, M[R2]

                MOV     R1, M[R2+1]             ; As posicoes dos obstaculos sao
                MOV     M[R2], R1               ; deslocados uma posicao na
                                                ; tabela, eliminando-se o
                MOV     R1, M[R2+2]             ; primeiro obstaculo e
                MOV     M[R2+1], R1             ; criando-se espaço para um novo
                                                ;
                MOV     R1, M[R2+3]             ;
                MOV     M[R2+2], R1             ;
                                                ;
                MOV     M[R2+3], R0             ;

                INC     M[CREATE_FLAG]          ; Ativa a indicacao de criacao
                                                ; de novo obstaculo.
C_END:          POP     R2
                POP     R1
                RET


; CREATE_OBSTACLE: Rotina que cria um novo obstaculo na primeira linha da janela
;               Entradas: Memoria - NEW_OBST_POS
;               Saidas: ---
;               Efeitos: Altera o valor da posicao OBSTACLE_POS+3
;
CREATE_OBSTACLE:PUSH    R1
                PUSH    R2

                MOV     R1, M[NEW_OBST_POS]
                MOV     R2, OBSTACLE_POS

                MOV     M[R2+3], R1

                PUSH    M[R2+3]
                CALL    WRITE_OBSTACLE

                POP     R2
                POP     R1
                RET


; DELETE_BIKE: Rotina que apaga a bicicleta da janela.
;       Entradas: Memoria - BIKE_POS
;       Saidas: ---
;       Efeitos: ---
;
DELETE_BIKE:    PUSH    M[BIKE_POS]
                PUSH    DEL_BIKE
                CALL    WRITE_STR_COL
                RET
```



```
; DELETE_OBSTACLE: Rotina que apaga um obstaculo.
;       Entradas: Pilha - posicao na janela do obstaculo a apagar
;       Saidas: ---
;       Efeitos: ---
;
DELETE_OBSTACLE:PUSH    M[SP+2]
                PUSH    DEL_OBSTACLE
                CALL    WRITE_STR_LINE
                RETN    1


; DELETE_PAUSE: Rotina que apaga a mensagem de pausa.
;               Entradas: Memoria - PAUSE_POS
;               Saidas: ---
;               Efeitos: ---
;
DELETE_PAUSE:   PUSH    PAUSE_POS
                PUSH    DEL_PAUSE
                CALL    WRITE_STR_LINE
                CALL    WRITE_LANE
                RET


; GAME_BEGGINING: Rotina que da inicio ao jogo.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: Altera R7, BIKE_POS, DISTANCE, OVERCOMED, COLISION_IND,
; TURBO_STATE e OBST_COUNTER
;
GAME_BEGINNING: MOV     R7, INITIAL_POS
                MOV     M[BIKE_POS], R7
                CALL    RESET_BOARD
                CALL    WRITE_LANE
                CALL    WRITE_BIKE
                CALL    LEVEL_UP
                CALL    W_DISTS_STR
                CALL    W_SEG_DISPLAY
                CALL    RESET_OBST_POS
                MOV     M[DISTANCE], R0
                MOV     M[OVERCOMED], R0
                MOV     M[COLISION_IND], R0
                MOV     R1, FFFFh
                MOV     M[TURBO_STATE], R1
                MOV     M[TURBO_FLAG], R0
                MOV     R1, OBST_DIFF
                MOV     M[OBST_COUNTER], R1
                RET


; GEN_RANDOM: Rotina que determina a coluna em que sera gerado o proximo obstacu
lo.
;               Entradas: Memoria - PSEUDO
;               Saidas: ---
;               Efeitos: Altera valores das posicoes PSEUDO e NEW_OBST_POS
;
GEN_RANDOM:     PUSH    R1
                PUSH    R2

                MOV     R1, M[PSEUDO]

GR_CICLE:       TEST    R1, 0001
                BR.NZ   GR_JUMP
```

```
                ROR     R1, 1
                BR      VERIFICATION

GR_JUMP:        XOR     R1, MASK_RANDOM
                ROR     R1, 1

VERIFICATION:   MOV     M[PSEUDO], R1

                MOV     R1, LEFT_BORDER
                MOV     R2, RIGHT_BORDER

                AND     R1, 00FFh
                AND     R2, 00FFh

                SUB     R2, R1
                DEC     R2              ; R2 = Numero de colunas da pista - 2

                MOV     R1, M[PSEUDO]

                DIV     R1, R2

                ADD     R2, LEFT_BORDER
                AND     R2, 00FFh       ; E necessaria esta instrucao porque,
                                        ; LEFT_BORDER inclui um valor de linha.

                MOV     M[NEW_OBST_POS], R2

                POP     R2
                POP     R1
                RET


; LEVEL: Rotina que permite iniciar cada nivel do jogo.
;               Entradas: Pilha - valor correspondente ao intervalo de tempo
; que se quer colocar no temporizador
;                         - LEDS que se pretende ligar
;               Saidas: ---
;               Efeitos: Altera valor da posicao PACE
;
LEVEL:          PUSH    R1

                MOV     R1, M[SP+4]
                MOV     M[PACE], R1

                PUSH    M[SP+3]
                CALL    TURN_LEDS

                POP     R1
                RETN    2


; LEVEL_UP: Rotina que verifica quando se deve incrementar o nivel do jogo.
;               Entradas: Memoria - OVERCOMED
;               Saidas: ---
;               Efeitos: ---
;
LEVEL_UP:       PUSH    R1

                PUSH    5                       ; Nivel 1.
                PUSH    F000h                   ;
                CALL    LEVEL                   ;
```



```
                MOV     R1, 4
                CMP     M[OVERCOMED], R1
                BR.N    LU_END

                PUSH    4                       ; Nivel 2.
                PUSH    FF00h                   ;
                CALL    LEVEL                   ;

                MOV     R1, 8
                CMP     M[OVERCOMED], R1
                BR.N    LU_END

                PUSH    3                       ; Nivel 3.
                PUSH    FFF0h                   ;
                CALL    LEVEL                   ;

LU_END:         POP     R1
                RET


; MOVE_ALL_OBST: Rotina que provoca o movimento de todos os obstaculos.
;               Entradas: Memoria - OBSTACLE_POS
;               Saidas: ---
;               Efeitos: Altera valores das posicoes DISTANCE e OBSTACLE_FLAG
;
MOVE_ALL_OBST:  PUSH    R1
                PUSH    R2

                MOV     R1, OBSTACLE_POS
                ADD     R1, 3
                MOV     R2, 4

MAO_CICLE:      CMP     M[R1], R0
                BR.Z    MAO_JUMP

                PUSH    R1
                CALL    MOVE_OBSTACLE

                DEC     R1              ; Percorre todos as posicoes na tabela
                DEC     R2              ; de posicoes dos obstaculos.

                CMP     R2, R0
                BR.NZ   MAO_CICLE

MAO_JUMP:       CALL    CREATE

                INC     M[DISTANCE]
                MOV     M[OBSTACLE_FLAG], R0

                POP     R2
                POP     R1
                RET


; MOVE_OBSTACLE: Rotina que provoca o movimento de um obstaculo.
;               Entradas: Pilha - posicao de memoria que contem a posicao do
; obstaculo
;               Saidas: ---
;               Efeitos: Pode alterar o valor da posicao OVERCOMED
;
MOVE_OBSTACLE:  PUSH    R1
```

```
                PUSH    R2
                PUSH    R3

                MOV     R1, M[SP+5]
                MOV     R2, M[R1]

                PUSH    R2
                CALL    DELETE_OBSTACLE

                MOV     R3, R2          ; Isto impede que na placa ocorra
                AND     R3, FF00h       ; wrap_around quando os obstaculos sao
                CMP     R3, 1700h       ; ultrapassados.
                BR.Z    OBST_OVERCOMED

                ADD     R2, LINE

                PUSH    R2
                CALL    WRITE_OBSTACLE

                MOV     M[R1], R2

MO_END:         POP     R3
                POP     R2
                POP     R1
                RETN    1

OBST_OVERCOMED: INC     M[OVERCOMED]
                CALL    W_SEG_DISPLAY
                BR      MO_END


; PAUSE_MODE: Rotina que permite entrar no modo pausa.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: Altera PAUSE_FLAG e TURBO_FLAG
;
PAUSE_MODE:     PUSH    R7

                MOV     M[PAUSE_FLAG], R0
                CALL    WRITE_PAUSE

PAUSE_CICLE:    CMP     M[PAUSE_FLAG], R0
                BR.Z    PAUSE_CICLE

                CALL    DELETE_PAUSE

                MOV     M[PAUSE_FLAG], R0
                MOV     M[TURBO_FLAG], R0

                POP     R7
                RET


; RECTANGLE: Rotina que apaga linhas centrais no formato de um rectangulo de
; forma a permitir a impressao de mensagens no centro da janela.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;
RECTANGLE:      PUSH    R1
                PUSH    R2
```



```
                MOV     R1, RECT_TOP
                MOV     R2, RECT_BOTTOM

RECT_CICLE:     PUSH    R1
                PUSH    DEL_STRING
                CALL    WRITE_STR_LINE

                ADD     R1, LINE

                CMP     R2, R1
                BR.NN   RECT_CICLE

                POP     R2
                POP     R1
                RET


; RESET_BOARD: Rotina que apaga a janela de texto.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;
RESET_BOARD:    PUSH    R1
                PUSH    R2

                MOV     R1, R0
                MOV     R2, 80

RB_CICLE:       PUSH    R1
                PUSH    DEL_STRING
                CALL    WRITE_STR_COL

                INC     R1
                CMP     R1, R2
                BR.NP   RB_CICLE

                POP     R2
                POP     R1
                RET


; RESET_OBST_POS: Rotina que faz reset as posicoes dos obstaculos.
;               Entradas: Memoria - OBSTACLE_POS
;               Saidas: ---
;               Efeitos: ---
;
RESET_OBST_POS: PUSH    R1

                MOV     R1, OBSTACLE_POS
                ADD     R1, 3

ROP_CICLE:      MOV     M[R1], R0
                DEC     R1

                CMP     R1, OBSTACLE_POS
                BR.NN   ROP_CICLE

                POP     R1
                RET


; TIMER_INIT: Rotina que inicializa o temporizador.
```

```
;               Entradas: Memoria - PACE
;               Saidas: ---
;               Efeitos: Coloca valores nos portos de controlo e de valor do
; temporizador.
;
TIMER_INIT:     PUSH    R1

                MOV     R1, M[PACE]
                MOV     M[TIMER_VALUE], R1
                MOV     R1, 1
                MOV     M[TIMER_ACTIV], R1

                POP     R1
                RET


; TURBO_MODE: Rotina que inicia o modo TURBO.
;               Entradas: TURBO_STATE
;               Saidas: ---
;               Efeitos: Altera os valores das posicoes TURBO_STATE e TURBO_FLAG
;
TURBO_MODE:     PUSH    R1

                NEG     M[TURBO_STATE]

                CMP     M[TURBO_STATE], R0
                BR.P    TURBO_ON

                CALL    LEVEL_UP
                BR      TBM_END

TURBO_ON:       PUSH    2
                PUSH    FFFFh
                CALL    LEVEL

TBM_END:        MOV     M[TURBO_FLAG], R0

                POP     R1
                RET


; TURN_BIKE: Rotina que permite mover a bicicleta para os lados.
;               Entradas: Registo R7
;               Saidas: ---
;               Efeitos: Altera o registo R7 e o valor da posicao BIKE_POS
;
TURN_BIKE:      CALL    DELETE_BIKE

                CMP     R7, LEFT_BORDER
                BR.P    TB_RIGHT

                MOV     R7, LEFT_BORDER

                BR      TB_WRITE

TB_RIGHT:       CMP     R7, RIGHT_BORDER
                BR.N    TB_WRITE

                MOV     R7, RIGHT_BORDER

TB_WRITE:       CALL    WRITE_BIKE
                MOV     M[BIKE_POS], R7
```



```
                RET


; TURN_LEDS: Rotina que permite ligar/desligar os LEDS.
;               Entradas: Pilha - bits dos LEDS a ficarem ligados
;               Saidas: ---
;               Efeitos: Coloca valor recebido pela pilha no porto de escrita
; dos LEDS.
;
TURN_LEDS:      MOV     R1, M[SP+2]
                MOV     M[LEDS], R1
                RETN    1


; WRITE_BIKE: Rotina que escreve a bicicleta na pista de jogo.
;               Entradas: Registo R7
;               Saidas: ---
;               Efeitos: ---
;
WRITE_BIKE:     PUSH    R7
                PUSH    BIKE
                CALL    WRITE_STR_COL
                RET


; WRITE_CAR: Rotina que escreve um caracter na janela de texto.
;               Entradas: Pilha - posicao do cursor e caracter a escrever
;               Saidas: ---
;               Efeitos: Coloca a posicao do cursor e o caracter a escrever nos
; portos respetivos da janela de texto
;
WRITE_CAR:      PUSH    R1

                MOV     R1, M[SP+4]
                MOV     M[IO_READ], R1          ; Posiciona cursor na janela.
                MOV     R1, M[SP+3]
                MOV     M[IO_WRITE], R1         ; Escreve caracter recebido.

                POP     R1
                RETN    2


; WRITE_DISTS: Rotina que escreve no LCD tanto a distancia percorrida no momento
; como a distancia maxima alguma vez percorrida pela bicicleta, fazendo a
; conversao necessaria para ASCII.
;               Entradas: DISTANCE, MAX_DIST
;               Saidas: ---
;               Efeitos: Pode alterar o valor da posicao MAX_DIST
;
WRITE_DISTS:    PUSH    R1

                PUSH    LCD_0_15
                PUSH    M[DISTANCE]
                PUSH    DIST_DEC_DIG
                CALL    W_DISTS_AUX

                MOV     R1, M[DISTANCE]
                CMP     R1, M[MAX_DIST]
                BR.NP   WD_END

                MOV     M[MAX_DIST], R1
```

```
                PUSH    LCD_1_11
                PUSH    M[MAX_DIST]
                PUSH    MAX_DEC_DIG
                CALL    W_DISTS_AUX

                PUSH    LCD_1_12
                PUSH    METERS
                CALL    W_LCD_CAR

WD_END:         POP     R1
                RET


; W_DISTS_AUX: Rotina auxiliar a rotina WRITE_DISTS.
;               Entradas:  Pilha − tabela onde colocar os digitos a converter
; para ASCII, valor a ser escrito, posicao a escrever no LCD
;               Saidas: ---
;               Efeitos: ---
;
W_DISTS_AUX:    PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                PUSH    R5

                MOV     R1, M[SP+8] ; Valor a ser escrito.
                MOV     R2, 100
                MOV     R3, 10
                MOV     R4, M[SP+7] ; Tabela.
                MOV     R5, M[SP+9] ; Posicao no LCD.

                DIV     R1, R2
                DIV     R2, R3

                MOV     M[R4], R3
                PUSH    R4
                PUSH    M[R4]
                CALL    ASCII_CONVERT ; Conversao para ASCII.

                PUSH    R5
                PUSH    M[R4]
                CALL    W_LCD_CAR     ; Escrita no LCD do caracter convertido.

                INC     R4
                DEC     R5

                MOV     M[R4], R2
                PUSH    R4
                PUSH    M[R4]
                CALL    ASCII_CONVERT

                PUSH    R5
                PUSH    M[R4]
                CALL    W_LCD_CAR

                MOV     R3, 10
                DIV     R1, R3

                INC     R4
                DEC     R5
```



```
                MOV     M[R4], R3
                PUSH    R4
                PUSH    M[R4]
                CALL    ASCII_CONVERT

                PUSH    R5
                PUSH    M[R4]
                CALL    W_LCD_CAR

                INC     R4
                DEC     R5

                MOV     M[R4], R1
                PUSH    R4
                PUSH    M[R4]
                CALL    ASCII_CONVERT

                PUSH    R5
                PUSH    M[R4]
                CALL    W_LCD_CAR

                POP     R5
                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RETN    3


; W_DISTS_STR: Rotina que escreve as palavras 'Distancia:' e 'Maximo:' nas 1ª e
; 2ª linha, respetivamente.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;
W_DISTS_STR:    PUSH    LCD_0_0
                PUSH    LCD_DIST
                CALL    WRITE_LCD

                PUSH    LCD_1_0
                PUSH    LCD_MAX
                CALL    WRITE_LCD

                RET


; W_FINAL_NOTES: Rotina que escreve as notas finais de jogo.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;
W_FINAL_NOTES:  CALL    RECTANGLE

                PUSH    F_NOTE1_POS
                PUSH    F_NOTE1
                CALL    WRITE_STR_LINE

                PUSH    F_NOTE2_POS
                PUSH    F_NOTE2
                CALL    WRITE_STR_LINE

                RET
```

```
; W_INITIAL_NOTES: Rotina que escreve as mensagens iniciais do jogo na janela.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
;
W_INITIAL_NOTES:PUSH    NOTE1_POS
                PUSH    NOTE1
                CALL    WRITE_STR_LINE

                PUSH    NOTE2_POS
                PUSH    NOTE2
                CALL    WRITE_STR_LINE

                RET


; WRITE_LANE: Rotina que escreve a pista de jogo.
;               Entradas: ---
;               Saidas: ---
;               Efeitos:
;
WRITE_LANE:     PUSH    R1

                MOV     R1, LEFT_BORDER
                AND     R1, 00FFh
                SUB     R1, 2

                CALL    W_PLUS_WALL

                INC     R1

                CALL    W_LINE_WALL

                MOV     R1, RIGHT_BORDER
                AND     R1, 00FFh
                INC     R1

                CALL    W_LINE_WALL

                INC     R1

                CALL    W_PLUS_WALL

                POP     R1
                RET


; WRITE_LCD: Rotina que permite escrever uma cadeia de caractres no LCD.
;               Entradas: Pilha - Posicao do cursor e string a escrever
;               Saidas: ---
;               Efeitos: ---
;
WRITE_LCD:      PUSH    R1
                PUSH    R2
                PUSH    R3

                MOV     R2, M[SP+5]             ; O que se pretende escrever.
                MOV     R3, M[SP+6]             ; Onde se pretende escrever.

WL_CICLE:       MOV     R1, M[R2]
```



```
                CMP     R1, STRING_END
                BR.Z    WL_END

                PUSH    R3
                PUSH    R1
                CALL    W_LCD_CAR

                INC     R2
                INC     R3
                BR      WL_CICLE

WL_END:         POP     R3
                POP     R2
                POP     R1

                RETN    2


; W_LCD_CAR: Rotina que permite escrever um caracter no LCD.
;               Entradas: Pilha - Posicao do cursor e caracter a escrever
;               Saidas: ---
;               Efeitos: ---
;
W_LCD_CAR:      PUSH    R1

                MOV     R1, M[SP+4]
                MOV     M[LCD_CONTROL], R1

                MOV     R1, M[SP+3]
                MOV     M[LCD_WRITE], R1

                POP     R1
                RETN    2


; W_LINE_WALL: Rotina que escreve uma coluna de linhas verticais.
;               Entradas: Registo R1
;               Saidas: ---
;               Efeitos: ---
;
W_LINE_WALL:    PUSH    R1
                PUSH    LINE_WALL
                CALL    WRITE_STR_COL
                RET


; WRITE_OBSTACLE: Rotina que escreve um obstaculo.
;               Entradas: Pilha - posicao do cursor
;               Saidas: ---
;               Efeitos: ---
;
WRITE_OBSTACLE: PUSH    M[SP+2]
                PUSH    OBSTACLE
                CALL    WRITE_STR_LINE
                RETN    1


; WRITE_PAUSE: Rotina que escreve a mensagem do modo pausa.
;               Entradas: ---
;               Saidas: ---
;               Efeitos: ---
```

```
;
WRITE_PAUSE:    CALL    RECTANGLE

                PUSH    PAUSE_POS
                PUSH    PAUSE_NOTE
                CALL    WRITE_STR_LINE

                RET


; W_PLUS_WALL: Rotina que escreve uma coluna de sinais de adicao.
;               Entradas: Registo R1
;               Saidas: ---
;               Efeitos: ---
;
W_PLUS_WALL:    PUSH    R1
                PUSH    PLUS_WALL
                CALL    WRITE_STR_COL
                RET


; W_SEG_DISPLAY:Rotina que permite escrever no display de 7 segmentos.
;               Entradas: Memoria - OVERCOMED
;               Saidas: ---
;               Efeitos: ---
;
W_SEG_DISPLAY:  PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4

                MOV     R1, M[OVERCOMED]
                MOV     R2, 100
                MOV     R3, 10
                MOV     R4, IO_DISPLAY

                DIV     R1, R2
                DIV     R2, R3

                MOV     M[R4], R3

                INC     R4
                MOV     M[R4], R2

                MOV     R3, 10
                DIV     R1, R3

                INC     R4
                MOV     M[R4], R3

                INC     R4
                MOV     M[R4], R1

                POP     R4
                POP     R3
                POP     R2
                POP     R1
                RET


; WRITE_STR_COL: Rotina que escreve uma string numa coluna da janela de texto.
;               Entradas: Pilha - posicao do cursor e string a escrever
```



```
;               Saidas: ---
;               Efeitos: ---
;
WRITE_STR_COL:  PUSH    R1
                PUSH    R2
                PUSH    R3

                MOV     R2, M[SP+5]             ; O que se pretende escrever.
                MOV     R3, M[SP+6]             ; Onde se pretende escrever.

WSC_CICLE:      MOV     R1, M[R2]
                CMP     R1, STRING_END
                BR.Z    WSC_END

                PUSH    R3
                PUSH    R1
                CALL    WRITE_CAR

                ADD     R3, LINE
                INC     R2
                BR      WSC_CICLE

WSC_END:        POP     R3
                POP     R2
                POP     R1
                RETN    2


; WRITE_STR_LINE: Rotina que escreve uma string numa linha da janela de texto.
;               Entradas: Pilha - posicao do cursor e string a escrever
;               Saidas: ---
;               Efeitos: ---
;
WRITE_STR_LINE: PUSH    R1
                PUSH    R2
                PUSH    R3

                MOV     R2, M[SP+5]             ; O que se pretende escrever.
                MOV     R3, M[SP+6]             ; Onde se pretende escrever.

WSL_CICLE:      MOV     R1, M[R2]
                CMP     R1, STRING_END
                BR.Z    WSL_END

                PUSH    R3
                PUSH    R1
                CALL    WRITE_CAR

                INC     R2
                INC     R3
                BR      WSL_CICLE

WSL_END:        POP     R3
                POP     R2
                POP     R1
                RETN    2

;_____________________________________________________________________________

;                           Programa Principal

MAIN:           MOV     R1, SP_INITIAL          ;{ Pre-jogo
```

```
                MOV     SP, R1                  ;
                                                ;
                MOV     R1, INT_MASK            ;
                MOV     M[INT_MASK_ADDR], R1    ;
                                                ;
                MOV     R1, IO_INIC             ;
                MOV     M[IO_READ], R1          ;
                                                ;
                CALL    W_INITIAL_NOTES         ;
                                                ;
                ENI                             ; }

START_CICLE:    INC     M[PSEUDO]               ;
                CMP     M[START_FLAG], R0       ;
                BR.Z    START_CICLE             ;

RESTART:        CALL    GAME_BEGINNING          ;{Inicio do jogo}
                CALL    TIMER_INIT              ;{Inicializacao do temporizador}

NEW_OBSTACLE:   CALL    GEN_RANDOM              ;{Jogo
                CALL    CREATE_OBSTACLE         ;
                MOV     M[CREATE_FLAG], R0      ;
                                                ;
BIKE_RUN:       CMP     M[PAUSE_FLAG], R0       ;
                CALL.NZ PAUSE_MODE              ;
                                                ;
                CMP     M[TURBO_FLAG], R0       ;
                CALL.NZ TURBO_MODE              ;
                                                ;
                CMP     R7, M[BIKE_POS]         ;
                BR.Z    M_NEXT                  ;
                                                ;
                CALL    TURN_BIKE               ;
                                                ;
M_NEXT:         CMP     M[OBSTACLE_FLAG], R0    ;
                BR.Z    M_NEXT2                 ;
                                                ;
                CALL    MOVE_ALL_OBST           ;
                                                ;
                CMP     M[TURBO_STATE], R0      ;
                BR.P    M_NEXT2                 ;
                                                ;
                CALL    LEVEL_UP                ;
                                                ;
M_NEXT2:        CALL    COLISION                ;
                CMP     M[COLISION_IND], R0     ;
                JMP.P   END_GAME                ;
                                                ;
                CALL    WRITE_DISTS             ;
                                                ;
                CMP     M[CREATE_FLAG], R0      ;
                JMP.P   NEW_OBSTACLE            ;
                                                ;
                JMP     BIKE_RUN                ; }

END_GAME:       CALL    W_FINAL_NOTES           ;{Fim do jogo
                MOV     M[START_FLAG], R0       ; }

END_GAME_CICLE: CMP     M[START_FLAG], R0       ;
                BR.Z    END_GAME_CICLE          ;
                                                ;
                JMP     RESTART                 ;
```