```
;                       PROJECTO DE IAC: JOGO DA BICICLETA
;
;                       Elaborado por:  Lidia Freitas           n 78559
;                                       Joao Jorge              n 73779


;-------------------------------------------------------------------------------
;|                       ZONA I: DEFINICAO DE CONSTANTES                       |
;|                                                                             |
;|      Pseudo-instrucao : EQU                                                 |
;-------------------------------------------------------------------------------


FIM_TEXTO       EQU     '@'     ; caracter de fim de texto
LINHAS          EQU     0018h   ; numero de linhas
COLUNAS         EQU     004Fh   ; numero de colunas
ESPACO          EQU     ' '     ; caracter espaco
CLEAR           EQU     FFFFh   ; codigo que limpa a janela de texto
IO_WRITE        EQU     FFFEh   ; porto de escrita da Janela de Texto
IO_CONTROL      EQU     FFFCh   ; porto do cursor de escrita da Janela
LCD_WRITE       EQU     FFF5h   ; porto de escrita do LCD
LCD_CURSOR      EQU     FFF4h   ; porto do cursor de escrita do LCD
ACT_REL         EQU     FFF7h   ; porto de activacao do relogio
REL             EQU     FFF6h   ; porto de inicio de contagem do relogio
INT_MASK_ADDR   EQU     FFFAh   ; endereco das mascaras
INT_MASK1       EQU     0000000000000010b       ; mascara 1
INT_MASK_PRIN   EQU     1000110000001101b       ; mascara principal
INT_MASK_PAUSA  EQU     1000010000000000b       ; mascara so de pausa
INT_MASK_TURB   EQU     1000110000000101b       ; mascara so de turbo
INT_MASK_ULTURB EQU     1000110000001001b       ; mascara so de ultra turbo
INT_MASK_REL    EQU     1000000000000000b       ; mascara apenas do relogio
RAND_MASK       EQU     1000000000010110b       ; mascara da rotina random
SP_INICIAL      EQU     FDFFh   ; posicao inicial da pilha
MUDALINHA       EQU     0100h   ; adicionar para ir para a linha seguinte
PRIM_RODA       EQU     1500h   ; linha da primeira roda da bicicleta
COL_INI_PISTA   EQU     29      ; coluna inicial da pista
COL_INI_BIC     EQU     41      ; coluna inicial da bicicleta
LEDS            EQU     FFF8h   ; porto de escrita dos leds
DISPLAY_0       EQU     FFF0h   ; display mais a direita
DISPLAY_1       EQU     FFF1h   ; segundio display
DISPLAY_2       EQU     FFF2h   ; terceiro display
DISPLAY_3       EQU     FFF3h   ; display mais a esquerda

;-------------------------------------------------------------------------------
;|                       ZONA II: DEFINICAO DE VARIAVEIS                       |
;|                                                                             |
;|              Pseudo-instrucoes :     WORD - palavra (16 bits)               |
;|                                      STR  - sequencia de caracteres.        |
;|              Cada caracter ocupa 1 palavra                                  |
;-------------------------------------------------------------------------------

                ORIG    8000h

VarTexto1_Ini   STR     'Bem-vindo a Corrida de Bicicleta!', FIM_TEXTO
VT1_Vazia       STR     '                                  ', FIM_TEXTO
VarTexto2       STR     'Prima o interruptor I1 para comecar', FIM_TEXTO
VT2_Vazia       STR     '                                   ', FIM_TEXTO
Txt_Pausa       STR     '     Pausa     ', FIM_TEXTO
VarTexto1_Fim   STR     'Fim Jogo', FIM_TEXTO
LCD_text1       STR     'Distancia:XXXXXm', FIM_TEXTO   ; texto LCD linha 1
LCD_text2       STR     'Maximo:YYYYYm', FIM_TEXTO      ; texto LCD linha 2
LinPista        STR     '+|                     |+', FIM_TEXTO ; linha pista
```



```
Bicicleta       STR     'O|O', FIM_TEXTO        ; bicicleta
BicObsVazia     STR     '   ', FIM_TEXTO        ; bicicleta e obstaculo vazio
Obstaculo       STR     '***', FIM_TEXTO        ; obstaculo

TempEspera      WORD    0005h   ; indica temp de espera em 100 milisec

Obstaculo1      WORD    0000h   ; posicao do obstaculo 1
Obstaculo2      WORD    0000h   ; posicao do obstaculo 2
Obstaculo3      WORD    0000h   ; posicao do obstaculo 3
Obstaculo4      WORD    0000h   ; posicao do obstaculo 4
ObsPassados     WORD    0000h   ; indica numero de obstaculos passados
ObsPass_0       WORD    0000h   ; valor em display 0
ObsPass_1       WORD    0000h   ; valor em display 1
ObsPass_2       WORD    0000h   ; valor em display 2
ObsPass_3       WORD    0000h   ; valor em display 3

DistPercorrida  WORD    0000h   ; indica a distancia percorrida
DistMaxima      WORD    0000h   ; indica a distancia maxima
D_0             WORD    0000h   ; digito 0 da distancia percorrida
D_1             WORD    0000h   ; digito 1 da distancia percorrida
D_2             WORD    0000h   ; digito 2 da distancia percorrida
D_3             WORD    0000h   ; digito 3 da distancia percorrida
D_4             WORD    0000h   ; digito 4 da distancia percorrida
D_0_max         WORD    0000h   ; digito 0 da distancia maxima
D_1_max         WORD    0000h   ; digito 1 da distancia maxima
D_2_max         WORD    0000h   ; digito 2 da distancia maxima
D_3_max         WORD    0000h   ; digito 3 da distancia maxima
D_4_max         WORD    0000h   ; digito 4 da distancia maxima

RandomNum       WORD    3A9Eh   ; numero random inicial

I1Press         WORD    0000h   ; se a 1, indica que I1 foi pressionado
I0Press         WORD    0000h   ; se a 1, indica que I0 foi pressionado
IBPress         WORD    0000h   ; se a 1, indica que IB foi pressionado
Colisao         WORD    0000h   ; se a 1, indica que houve colisao
Relogio         WORD    0000h   ; se a 1, indica que o relogio chegou a zero
Turbo           WORD    0000h   ; se a 1, indica se turbo esta activo
UltraTurbo      WORD    0000h   ; se a 1, indica se ultra turbo esta activo
VelTurbo        WORD    0002h   ; velocidade do turbo em 100 milisec
VelUltraTurbo   WORD    0001h   ; velocidade do turbo em 100 milisec
Pausa           WORD    0000h   ; se a 1, indica que está em modo de pausa

;-------------------------------------------------------------------------------
;|                       ZONA III: TABELA DE INTERRUPCOES                      |
;|                                                                             |
;-------------------------------------------------------------------------------

                ORIG    FE00h
INT0            WORD    I0              ; interrupcao de mover bic - esquerda
INT1            WORD    I1              ; interrupcao de iniciar o jogo
INT2            WORD    I_Turbo         ; interrupcao do iniciar turbo
INT3            WORD    I_UltraTurbo    ; interrupcao do iniciar ultraturbo

                ORIG    FE0Ah
INTA            WORD    I_Pausa         ; interrupcao de colocar em pausa
INTB            WORD    IB              ; interrupcao de mover bic - direita

                ORIG    FE0Fh
IF              WORD    ZerRel          ; interrupcao de termino do relogio

;-------------------------------------------------------------------------------
;|                                ZONA IV: CODIGO                              |
```

```
;|                                                                              |
;|      conjunto de instrucoes Assembly ordenadas de forma a realizar           |
;|      as funcoes pretendidas                                                  |
;------------------------------------------------------------------------------

                ORIG    0000h
                JMP     Jogo            ; salta para o programa principal

;______________________________________________________________________________
;       ZONA IV.I       ROTINAS DAS INTERRUPCOES - INCREMENTAM VARIAVEIS
;______________________________________________________________________________

I0:     INC     M[I0Press]              ; passa valor de I0Press para 1
        RTI

I1:     INC     M[I1Press]              ; passa valor de I1Press para 1
        RTI

IB:     INC     M[IBPress]              ; passa valor de IBPress para 1
        RTI

ZerRel: INC     M[Relogio]              ; passa valor de Relogio para 1
        RTI

I_Pausa:INC     M[Pausa]                ; passa valor de Pausa para 1 ou 2
        RTI

I_Turbo:INC     M[Turbo]                ; passa valor de Turbo para 1 ou 2
        RTI

I_UltraTurbo:   INC     M[UltraTurbo]   ; passa valor de UltraTurbo para 1 ou 2
        RTI


;______________________________________________________________________________
;                               CountCar
;       rotina que conta os caracteres de uma string e guarda o resultado
;       na pilha.
;               Entradas:       pilha - sitio onde guardar o resultado
;                               pilha - local de inicio da string
;               Saidas:         pilha - sitio onde guardar o resultado
;               Efeitos:        ---

CountCar:       PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                PUSH    R3              ; salvaguarda R3
                MOV     R1, R0          ; R1 - numero de caracteres
                MOV     R2, M[SP+5]     ; R2 indica inicio da string
CicloCC:        MOV     R3, M[R2]       ; R3 - caracter relativo a R2
                CMP     R3, FIM_TEXTO   ; ve se chegou ao fim
                BR.Z    EscCountCar     ; se sim, salta
                INC     R1              ; se nao, incrementa num de caracteres
                INC     R2              ; incrementa posicao da string
                BR      CicloCC         ; repete ciclo
EscCountCar:    DEC     R1              ; decrementa 1 caracter
                MOV     M[SP+6], R1     ; guarda resultado na pilha
                POP     R3              ; retoma valor de R3
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    1               ; retorna e retira 1 val da pilha

;______________________________________________________________________________
```



```
;                               EscEspacos
;       rotina que escreve numero de espacos indicados na pilha
;               Entradas:       pilha - num de espacos
;                               registo - R4 indica pos do cursor
;               Saidas:         ---
;               Efeitos:        muda R4

EscEspacos:     PUSH    R1                      ; salvaguarda R1
                PUSH    R2                      ; salvaguarda R2
                MOV     R1, M[SP+4]             ; R1 indica num de espacos
CicloEEsp:      MOV     M[IO_CONTROL], R4       ; coloca cursor no sitio
                MOV     R2, ESPACO              ; R2 tem o caracter espaco
                MOV     M[IO_WRITE], R2         ; escreve espaco
                INC     R4                      ; incrementa posica a escrever
                DEC     R1                      ; decrementa num de espacos
                BR.NZ   CicloEEsp               ; se nao for zero repete
                POP     R2                      ; retoma valor de R2
                POP     R1                      ; retoma valor de R1
                RETN    1                       ; retorna, retira da pilha 1 val

;______________________________________________________________________________
;                               Centra
;       rotina que escreve a pista
;               Entradas:       pilha - linha onde vai centrar
;                               pilha - pos de string que vai centrar
;               Saidas:         ---
;               Efeitos:        ---

Centra:         PUSH    R1                      ; salvaguarda R1
                PUSH    R2                      ; salvaguarda R2
                PUSH    R3                      ; salvaguarda R3
                PUSH    R4                      ; salvaguarda R4
                MOV     R4, M[SP+7]             ; R4 e linha onde vai centrar
                MOV     M[IO_CONTROL], R4       ; coloca cursor em R4
                MOV     R2, M[SP+6]             ; R2 e pos da str a centrar
                PUSH    R0                      ; guarda R0
                PUSH    R2                      ; guarda R2
                CALL    CountCar                ; conta caracteres
                POP     R1                      ; R1 e num de caracteres
                MOV     R3, COLUNAS             ; R3 e num de colunas
                SUB     R3, R1                  ; subtrai caracteres a colunas
                SHRA    R3, 1                   ; divide por dois
                PUSH    R3                      ; coloca na pilha num de espacos
                CALL    EscEspacos              ; escreve num de espacos
                PUSH    R2                      ; coloca na pilha pos da string
                PUSH    R4                      ; pilha - pos de escrita
                CALL    EscString               ; escreve string
                POP     R4                      ; retoma valor de R4
                POP     R3                      ; retoma valor de R3
                POP     R2                      ; retoma valor de R2
                POP     R1                      ; retoma valor de R1
                RETN    2                       ; retorna, retira da pilha 2 val


;______________________________________________________________________________
;                               EscL12L14
;       rotina que escreve strings centradas na linha 12 e 14
;               Entradas:       pilha - primeira frase
;                               pilha - segunda frase
;               Saidas:         ---
;               Efeitos:        ---
```

```
EscL12L14:      CALL    IniPortoCtrl            ; limpa janela
                PUSH    0b00h                   ; coloca na pilha linha 12
                PUSH    M[SP+4]                 ; coloca na pilha pos da 1 frase
                CALL    Centra                  ; centra
                PUSH    0d00h                   ; coloca na pilha linha 12
                PUSH    M[SP+3]                 ; coloca na pilha pos da 2 frase
                CALL    Centra                  ; centra
                RETN    2                       ; retorna, retira 2 val da pilha


;_____________________________________________________________________________
;                               EscPista
;       rotina que escreve a pista
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        R6 - indica a coluna da bicicleta

EscPista:       PUSH    R1                      ; salvaguarda R1
                PUSH    R2                      ; salvaguarda R2
                PUSH    R3                      ; salvaguarda R3
                CALL    LimpaL12L14             ; limpa linha 12 e 14
                MOV     R1, LINHAS              ; R1 e o numero de linhas
                MOV     R3, R0                  ; R3 e zero
                ADD     R3, COL_INI_PISTA       ; R3 - col de inicio da pista
CicloEPis:      PUSH    LinPista                ; pos da string da pista
                PUSH    R3                      ; coluna onde pista comeca
                CALL    EscString               ; escreve linha da pista
                ADD     R3, MUDALINHA           ; R3 e valor da linha seguinte
                AND     R3, FF00h               ; volta ao inicio da linha
                ADD     R3, COL_INI_PISTA       ; coloca na coluna certa
                DEC     R1                      ; dec o num de linhas do ecra
                BR.NZ   CicloEPis               ; se nao for zero repete
                MOV     R6, COL_INI_BIC         ; R6 e a col inicial da bic
                POP     R3                      ; retoma valor de R3
                POP     R2                      ; retoma valor de R2
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               IniPortoCtrl
;       rotina que inicia o porto de escrita
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

IniPortoCtrl:   PUSH    R1                      ; salvaguarda R1
                MOV     R1, CLEAR               ; R1 fica com valor de CLEAR
                MOV     M[IO_CONTROL], R1       ; limpa janela
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               LimpaPista
;       rotina que limpa a pista
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LimpaPista:     PUSH    R1                      ; salvaguarda R1
                PUSH    R4                      ; salvaguarda R4
                MOV     R1, 29                  ; R1 e inicio da pista
                MOV     R4, R0                  ; R4 e zero
```



```
LimpaPista_aux: ADD     R4, R1                  ; R4 e inicio da pista
                PUSH    26                      ; pilha - comprimento da pista
                CALL    EscEspacos              ; escreve espacos
                AND     R4, FF00h               ; R4 e linha do cursor
                ADD     R4, MUDALINHA           ; R4 e linha seguinte
                CMP     R4, 1800h               ; ve se chegou ao fim
                BR.NZ   LimpaPista_aux          ; se nao repete
                POP     R4                      ; retoma valor de R4
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna
;_____________________________________________________________________________
;                               LimpaL12L14
;       rotina que limpa a linha 12 e 14
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LimpaL12L14:    PUSH    VT1_Vazia               ; coloca na pilha frase 1
                PUSH    VT2_Vazia               ; coloca na pilha frase 2
                CALL    EscL12L14               ; escreve na linha 12 e 14
                RET                             ; retorna


;_____________________________________________________________________________
;                               EscString
;       rotina que efectua a escrita de uma cadeia de caracteres
;       terminada pelo caracter FIM_TEXTO
;               Entradas:       pilha - inicio da cadeia de caracteres
;                               pilha - local onde vai ser escrita a string
;               Saidas:         ---
;               Efeitos:        ---

EscString:      PUSH    R1                      ; salvaguarda R1
                PUSH    R2                      ; salvaguarda R2
                PUSH    R3                      ; salvaguarda R3
                MOV     R2, M[SP+6]             ; indica o inicio da frase
                MOV     R3, M[SP+5]             ; apontador para ini da string
CicloEscStr:    MOV     R1, M[R2]               ; R1 e o caracter a escrever
                CMP     R1, FIM_TEXTO           ; ve se a string chegou ao fim
                BR.Z    FimEsc                  ; se sim salta para FimEsc
                MOV     M[IO_CONTROL], R3       ; coloca cursor na pos de R3
                MOV     M[IO_WRITE], R1         ; escreve caracter de R1
                INC     R2                      ;  posicao da str seguinte
                INC     R3                      ; cursor para a pos seguinte
                BR      CicloEscStr             ; continua o ciclo
FimEsc:         POP     R3                      ; retoma valor de R3
                POP     R2                      ; retoma valor de R2
                POP     R1                      ; retoma valor de R1
                RETN    2                       ; retorna, elimina 2 val

;_____________________________________________________________________________
;                               LCD_EscStr
;       rotina que efectua a escrita de uma cadeia de caracteres
;       terminada pelo caracter FIM_TEXTO
;               Entradas:       pilha - inicio da cadeia de caracteres
;                               pilha - linha do LCD onde vai ser escrita
;               Saidas:         ---
;               Efeitos:        ---

LCD_EscStr:     PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                PUSH    R3              ; salvaguarda R3
```

```
                MOV     R1, M[SP+6]       ; R1 e inicio da cadeia de caracteres
                MOV     R2, M[SP+5]       ; R2 e linha onde sera escrita a str
LCD_ES_Ciclo:   MOV     R3, M[R1]         ; R3 e caracter a escrever
                CMP     R3, FIM_TEXTO     ; compara com caracter de fim
                BR.Z    LCD_ES_Fim        ; se for salta para o fim
                MOV     M[LCD_CURSOR], R2 ; coloca cursor no sitio
                MOV     M[LCD_WRITE], R3  ; escreve caracter
                INC     R1                ; inc pos da string
                INC     R2                ; inc pos do lcd
                BR      LCD_ES_Ciclo      ; repete
LCD_ES_Fim:     POP     R3                ; retoma valor de R3
                POP     R2                ; retoma valor de R2
                POP     R1                ; retoma valor de R1
                RETN    2                 ; retorna e elemina 2 val da pilha

;_____________________________________________________________________________
;                               LCD_EscStr
;       rotina que efectua a escrita das mensagens de dist: e maximo:
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LCD_EscTexto:   PUSH    LCD_text1       ; pilha - pos do texto LCD_text1
                PUSH    8000h           ; pilha - pos da primeira linha do LCD
                CALL    LCD_EscStr      ; escreve string LCD_text1
                PUSH    LCD_text2       ; pilha - pos do texto LCD_text2
                PUSH    8010h           ; pilha - pos da segunda linha do LCD
                CALL    LCD_EscStr      ; escreve string LCD_text2
                RET                     ; retorna

;_____________________________________________________________________________
;                               EscApagBic
;       rotina que escreve a bicicleta se na pilha estiver Bicicleta,
;       apaga a bicicleta se na pilha estiver BicObsVazia
;               Entradas:       R6 - coluna da bicicleta
;                               pilha - Bicicleta ou BicObsVazia
;               Saidas:         ---
;               Efeitos:        ---

EscApagBic:     PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                ADD     R6, PRIM_RODA   ; R6 indica pos da pri roda
                MOV     R1, M[SP+4]     ; indica string a escrever
CicloEAB:       MOV     M[IO_CONTROL], R6 ; coloca cursor em R6
                MOV     R2, M[R1]       ; R2 e primeiro caracter
                CMP     R2, FIM_TEXTO   ; ve se ja chegou ao fim
                BR.Z    EscApagTermina  ; se sim salta
                MOV     M[IO_WRITE], R2 ; se nao escreve
                INC     R1              ; passa para caracter seguint
                ADD     R6, MUDALINHA   ; muda de linha
                BR      CicloEAB        ; repete
EscApagTermina: AND     R6, 00FFh       ; limpa linha da bicicleta
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    1               ; retorna e elimina

;_____________________________________________________________________________
;                               EscBic
;       rotina que escreve a bicicleta
;               Entradas:       R6 - coluna da bicicleta
;               Saidas:         ---
;               Efeitos:        ---
```



```
EscBic:         PUSH    Bicicleta       ; indica pos da str bicicleta
                CALL    EscApagBic      ; escreve bicicleta
                RET                     ; retorna

;_____________________________________________________________________________
;                               ApagBic
;       rotina que apaga a bicicleta
;               Entradas:       R6 - coluna da bicicleta
;               Saidas:         ---
;               Efeitos:        ---
ApagBic:        PUSH    BicObsVazia     ; indica pos da str vazia
                CALL    EscApagBic      ; apaga bicicleta
                RET                     ; retorna

;_____________________________________________________________________________
;                               MoveEsq
;       rotina que move a bicicleta para a esquerda
;               Entradas:       R6 - coluna da bicicleta
;               Saidas:         ---
;               Efeitos:        Muda R6 se possivel mover bic

MoveEsq:        MOV     M[I0Press], R0  ; coloca valor indicativo a 0
                CMP     R6, 31          ; ve se esta na coluna 31
                BR.Z    MoveEsqFim      ; se sim, salta para o fim
                CALL    ApagBic         ; se nao, apaga bic
                DEC     R6              ; decrementa coluna
                CALL    EscBic          ; escreve bicicleta
MoveEsqFim:     RET                     ; retorna

;_____________________________________________________________________________
;                               MoveDir
;       rotina que move a bicicleta para a direita
;               Entradas:       R6 - coluna da bicicleta
;               Saidas:         ---
;               Efeitos:        Muda R6 se possivel mover bic

MoveDir:        MOV     M[IBPress], R0  ; coloca valor indicativo a 0
                CMP     R6, 52          ; ve se esta na coluna 52
                BR.Z    MoveDirFim      ; se sim, salta para o fim
                CALL    ApagBic         ; se nao, apaga bic
                INC     R6              ; decrementa coluna
                CALL    EscBic          ; escreve bicicleta
MoveDirFim:     RET                     ; retorna

;_____________________________________________________________________________
;                               EscInicio
;       escreve as mensagens iniciais
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

EscInicio:      PUSH    R1              ; salvaguarda R1
                MOV     R1, INT_MASK1   ; R1 e mascara da INT1
                MOV     M[INT_MASK_ADDR], R1 ; activa mascara
                PUSH    VarTexto1_Ini   ; coloca na pilha frase 1
                PUSH    VarTexto2       ; coloca na pilha frase 2
                CALL    EscL12L14       ; escreve na linha 12 e 14
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_____________________________________________________________________________
;                               EscFim
```

```
;       escreve as mensagens finais
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---
EscFim:         PUSH    R1                      ; salvaguarda R1
                CALL    LimpaPista              ; limpa pista
                MOV     R1, INT_MASK1           ; R1 e mascara da INT1
                MOV     M[INT_MASK_ADDR], R1    ; activa mascara
                PUSH    VarTexto1_Fim           ; coloca na pilha frase 1
                PUSH    VarTexto2               ; coloca na pilha frase 2
                CALL    EscL12L14               ; escreve na linha 12 e 14
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna
;_____________________________________________________________________________
;                               Espera1
;       rotina que espera até I1 ser pressionado
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

Espera1:        ENI                             ; activa interrupcoes
                CMP     M[I1Press], R0          ; I1 pressionado?
                BR.Z    Espera1                 ; se nao, espera
                MOV     M[I1Press], R0          ; se sim, coloca I1 a zero
                RET                             ; retorna

;_____________________________________________________________________________
;                               EscApagObstaculo
;       rotina que escreve a bicicleta
;               Entradas:       pilha - posicao do obstaculo
;                               pilha - Obstaculo ou Vazio
;               Saidas:         ---
;               Efeitos:        ---

EscApagObs:     PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                PUSH    R3              ; salvaguarda R3
                MOV     R1, M[SP+6]     ; R1 e pos do obstaculo
                MOV     R2, M[SP+5]     ; R2 e obs ou vazio
CicloEscObs:    MOV     M[IO_CONTROL], R1 ; coloca cursor na pos do obs
                MOV     R3, M[R2]       ; R3 e caracter
                CMP     R3, FIM_TEXTO   ; chegou ao fim?
                BR.Z    FimEscObs       ; se sim, salta
                MOV     M[IO_WRITE], R3 ; se nao, escreve caracter
                INC     R1              ; inc pos do obstaculo
                INC     R2              ; inc pos da string obs ou vazio
                BR      CicloEscObs     ; repete
FimEscObs:      POP     R3              ; retoma valor de R3
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    2               ; retorna e retira 2 val da pilha



;_____________________________________________________________________________
;                               CriaObstaculo
;       rotina que cria um obstaculo novo
;               Entradas:       pilha - obstaculo
;               Saidas:         ---
;               Efeitos:        ---

CriaObstaculo:  PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
```



```
                PUSH    R3              ; salvaguarda R3
                MOV     R1, M[SP+5]     ; R1 e obstaculo
                CALL    Random          ; faz random
                MOV     R3, M[RandomNum]; R3 e numero random 16 bits
                AND     R3, 000Fh       ; R3 e numero random ate 15
                CALL    Random          ; faz random
                MOV     R2, M[RandomNum]; R2 e numero random 16 bits
                AND     R2, 0003h       ; R2 e numero random ate 3
                ADD     R2, R3          ; R2 e numero random ate 18
                CALL    Random          ; faz random
                MOV     R3, M[RandomNum]; R3 e numero random 16 bits
                AND     R3, 0001h       ; R3 e numero random 0 ou 1
                ADD     R2, R3          ; R2 e numero random ate 19
                CALL    Random          ; faz random
                MOV     R3, M[RandomNum]; R3 e numero random 16 bits
                AND     R3, 0001h       ; R3 e numero random 0 ou 1
                ADD     R2, R3          ; R2 e numero random ate 20
                ADD     R2, 31          ; R2 agora e pos random de obs
                MOV     M[R1], R2       ; move R2 para pos do obstaculo R1
                PUSH    R2              ; coloca pos obs na pilha
                PUSH    Obstaculo       ; coloca obs na pilha
                CALL    EscApagObs      ; escreve obs na posicao de R2
                POP     R3              ; retoma valor de R3
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    1               ; retorna e elimina 1 val da pilha

;_____________________________________________________________________________
;                               Inicializa
;       rotina que inicia variaveis
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

Inicializa:     PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_PRIN       ; R1 e mascara principal
                MOV     M[INT_MASK_ADDR], R1    ; muda para mascara principal
                MOV     R1, F000h               ; R1 com valor de leds_nivel1
                MOV     M[LEDS], R1             ; escreve nos led
                CALL    LimpaDisplays           ; limpa displays de 7 segmentos
                MOV     M[Colisao], R0          ; indica que nao houve colisao
                CALL    EscPista                ; escreve a pista
                CALL    EscBic                  ; escreve bicicleta
                PUSH    Obstaculo1              ; coloca obstaculo1 na pilha
                CALL    CriaObstaculo           ; escreve obs inicial
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               LimpaDisplays
;       rotina que coloca todos os displays de 7 segmentos a 0
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LimpaDisplays:  MOV     M[DISPLAY_0], R0 ; coloca Display 0 a 0
                MOV     M[DISPLAY_1], R0 ; coloca Display 1 a 0
                MOV     M[DISPLAY_2], R0 ; coloca Display 2 a 0
                MOV     M[DISPLAY_3], R0 ; coloca Display 3 a 0
                RET                      ; retorna
;_____________________________________________________________________________
;                               IncDist
```

```
;       rotina que incrementa a distancia percorrida
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

IncDist:        PUSH    R1              ; salvaguarda R1
                INC     M[DistPercorrida]; incrementa distancia percorrida
                MOV     R1, 10          ; R1 tem valor de dez
                INC     M[D_0]          ; incrementa val de D_0
                CMP     M[D_0], R1      ; ve se e 10
                BR.Z    Ced_1           ; se sim, salta para Ced_1
                BR      FimIncDist      ; se nao, salta para fim
Ced_1:          MOV     M[D_0], R0      ; coloca anterior a zero
                INC     M[D_1]          ; incrementa val de D_1
                CMP     M[D_1], R1      ; ve se e 10
                BR.Z    Ced_2           ; se sim, salta para Ced_2
                BR      FimIncDist      ; se nao, salta para fim
Ced_2:          MOV     M[D_1], R0      ; coloca anterior a zero
                INC     M[D_2]          ; incrementa val de D_2
                CMP     M[D_2], R1      ; ve se e 10
                BR.Z    Ced_3           ; se sim, salta para Ced_3
                BR      FimIncDist      ; se nao, salta para fim

FimIncDist:     POP     R1              ; retoma R1
                RET                     ; retorna

Ced_3:          MOV     M[D_2], R0      ; coloca anterior a zero
                INC     M[D_3]          ; incrementa val de D_3
                CMP     M[D_3], R1      ; ve se e 10
                BR.Z    Ced_4           ; se sim, salta para Ced_4
                BR      FimIncDist      ; se nao, salta para fim
Ced_4:          MOV     M[D_3], R0      ; coloca anterior a zero
                INC     M[D_4]          ; incrementa val de D_4
                CMP     M[D_4], R1      ; ve se e 10
                CALL.Z  LimpaDist       ; se sim, salta para LimpaCED
                BR      FimIncDist      ; se nao, salta para fim

;_______________________________________________________________________________
;                               LimpaDist
;       rotina que limpa os displays da distancia percorrida
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LimpaDist:      MOV     M[D_0], R0      ; reinicia D_0
                MOV     M[D_1], R0      ; reinicia D_1
                MOV     M[D_2], R0      ; reinicia D_0
                MOV     M[D_3], R0      ; reinicia D_0
                MOV     M[D_4], R0      ; reinicia D_4
                RET                     ; retorna
;_______________________________________________________________________________
;                               EscQQRDist
;       rotina que escreve a distancia em determinada pos do LCD
;               Entradas:       pilha - pos de inicio da distancia
;               Saidas:         ---
;               Efeitos:        ---

EscQQRDist:     PUSH    R1              ; salvaguarda R1
                MOV     R1, M[SP+3]     ; R1 e pos cursor LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_4]          ; pilha - valor de D_4
                CALL    LCD_EscNum      ; escreve no LCD
```



```
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_3]          ; pilha - valor de D_3
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_2]          ; pilha - valor de D_2
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_1]          ; pilha - valor de D_1
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_0]          ; pilha - valor de D_0
                CALL    LCD_EscNum      ; escreve no LCD
                POP     R1              ; retoma valor de R1
                RETN    1               ; retorna

;_______________________________________________________________________________
;                               EscDist
;       rotina que escreve a distancia percorrida no LCD
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

EscDist:        PUSH    800Ah           ; pos inicio da distancia
                CALL    EscQQRDist      ; escreve distancia
                RET                     ; retorna

;_______________________________________________________________________________
;                               EscDist
;       rotina que escreve a distancia maxima no LCD
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

EscDistMax:     PUSH    R1              ; salvaguarda R1
                MOV     R1, 8017h       ; R1 e pos cursor LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_4_max]      ; pilha - valor de D_4_max
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_3_max]      ; pilha - valor de D_3_max
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_2_max]      ; pilha - valor de D_2_max
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_1_max]      ; pilha - valor de D_1_max
                CALL    LCD_EscNum      ; escreve no LCD
                INC     R1              ; incrementa pos LCD
                PUSH    R1              ; pilha - R1
                PUSH    M[D_0_max]      ; pilha - valor de D_0_max
                CALL    LCD_EscNum      ; escreve no LCD
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_______________________________________________________________________________
```

```
;                               CLRLCD
;       rotina para limpar LCD
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

CLRLCD:         PUSH    R1              ; salvaguarda R1
                MOV     R1, 8020h       ; move ordem de limpeza para R1
                MOV     M[LCD_CURSOR], R1 ; limpa LCD
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_____________________________________________________________________________
;                               LCD_EscNum
;       rotina que escreve um numero no LCD
;               Entradas:       pilha - posicao do cursor
;                               pilha - posicao do numero
;               Saidas:         ---
;               Efeitos:        ---

LCD_EscNum:     PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                MOV     R1, M[SP+5]     ; R1 e pos do cursor
                MOV     R2, M[SP+4]     ; R2 e pos do numero
                ADD     R2, 0030h       ; passa num para ASCII
                MOV     M[LCD_CURSOR], R1 ; posiciona cursor
                MOV     M[LCD_WRITE], R2  ; escreve numero
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    2               ; retorna e elimina 2 val da pilha

;_____________________________________________________________________________
;                               Espera2
;       segunda rotina de espera: incrementa distancia percorrida, termina
;       quando temp relogio chega ao fim, testa modo pausa e modo turbo,
;       testa movimento da bicicleta para a esquerda e para a direita.
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

Espera2:        PUSH    R1              ; salvaguarda R1
                CALL    IncDist         ; incrementa distancia
                CALL    EscDist         ; escreve distancia
Espera_aux:     CMP     M[Relogio], R0  ; ve se tempo chegou ao fim
                JMP.NZ  FimEspera2      ; se sim, salta para fim
                CMP     M[Pausa], R0    ; testa modo de pausa
                CALL.NZ ModoPausa       ; se sim, chama ModoPausa
                MOV     R1, 1           ; R1 e valor 1
                CMP     M[Turbo], R1    ; testa se botao turbo press 1 vez
                CALL.Z  MudaVelTurbo    ; se sim, chama MudaVelTurbo
                CMP     M[UltraTurbo], R1 ; testa se botao ultra turbo press 1
                CALL.Z  MudaVelU_Turbo  ; se sim, chama MudaVelTurbo
                INC     R1              ; R1 e valor 2
                CMP     M[Turbo], R1    ; testa se botao turbo press 2 vez
                CALL.Z  RepoeVelTurbo   ; se sim, chama RepoeVelTurbo
                CMP     M[UltraTurbo], R1 ; testa se botao turbo press 2 vez
                CALL.Z  RepoeVelU_Turbo ; se sim, chama RepoeVelTurbo
                CMP     M[I0Press], R0  ; ve se I0 foi pressionado
                CALL.NZ MoveEsq         ; se sim, move esquerda
                CMP     M[IBPress], R0  ; ve se IB foi pressionado
                CALL.NZ MoveDir         ; se sim, move direita
                JMP     Espera_aux      ; repete
```



```
FimEspera2:     MOV     M[Relogio], R0  ; coloca var de Relogio a 0
                MOV     R1, 1           ; R1 e 1
                MOV     M[ACT_REL], R1  ; activa relogio
                MOV     R1, M[TempEspera] ; R1 e tempo de espera
                MOV     M[REL], R1      ; coloca temp de espera no relogio
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_____________________________________________________________________________
;                               LCD_EscPausa
;       rotina que escreve no LCD a mensagem de pausa
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

LCD_EscPausa:   PUSH    R1              ; salvaguarda R1
                MOV     R1, 8020h       ; R1 e ordem de limpeza do LCD
                MOV     M[LCD_CURSOR], R1 ; limpa o LCD
                PUSH    Txt_Pausa       ; pilha - string que de mensagem pausa
                PUSH    8000h           ; pilha - pos da primeira linha
                CALL    LCD_EscStr      ; escreve string no lcd
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_____________________________________________________________________________
;                               ModoPausa
;       rotina que para o jogo ate que o botao pausa chega carregado pela
;       segunda vez e por isso tem que desactivar os outros botoes enquanto
;       espera pelo botao pausa.
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

ModoPausa:      PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_PAUSA      ; R1 e mascara da pausa
                MOV     M[INT_MASK_ADDR], R1    ; activa mascara da pausa
                MOV     R1, 0002h               ; R1 e 2
                CALL    LCD_EscPausa            ; escreve mensagem de pausa
Ciclo_MP:       CMP     M[Pausa], R1            ; botao pressionado segunda vez?
                BR.NZ   Ciclo_MP                ; se nao, repete ate ser
                MOV     R1, INT_MASK_PRIN       ; se sim, R1 e mascara principal
                MOV     M[INT_MASK_ADDR], R1    ; activa mascara principal
                MOV     M[Pausa], R0            ; coloca var Pausa a zero
                CALL    LCD_EscTexto            ; escreve mensag. de dist no LCD
                CALL    EscDist                 ; escreve distancia percorrida
                CALL    EscDistMax              ; escreve distancia maxima
                POP     R1                      ; retoma R1
                RET                             ; retorna


;_____________________________________________________________________________
;                               MoveObs
;       rotina que move obstaculo
;               Entradas:       pilha - obstaculo a ser movido
;               Saidas:         ---
;               Efeitos:        ---

MoveObs:        PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                PUSH    R3              ; salvaguarda R3
                MOV     R1, M[SP+5]     ; R1 e obstaculo
                MOV     R2, M[R1]       ; R2 e pos do obstaculo
```

```
                CALL    HaColisao       ; testa colisao
                CMP     M[Colisao], R0  ; ha colisao?
                JMP.NZ  MoveObs_Fim     ; se sim, salta para fim
MoveObsCont:    PUSH    R2              ; pilha - pos do obstaculo
                PUSH    BicObsVazia     ; pilha - obstaculo vazio
                CALL    EscApagObs      ; apaga obstaculo
                MOV     R3, R2          ; R3 e pos do obstaculo
                AND     R3, FF00h       ; R3 e linha do obstaculo
                CMP     R3, 1700h       ; ve se obs esta na ultima linha
                BR.Z    UltLin          ; se sim salta para UltLin
                ADD     R2, MUDALINHA   ; R2 passa para prox linha
                BR      MoveObs_aux     ; salta para MoveObs_aux
UltLin:         CALL    Random          ; faz random
                MOV     R2, M[RandomNum]; R2 e numero random de 16bits
                MOV     R3, R2          ; R3 e valor de R2
                AND     R3, 000Fh       ; R3 e numero random ate 15
                CALL    Random          ; faz random
                MOV     R2, M[RandomNum]; R2 e numero random de 16 bits
                AND     R2, 0003h       ; R2 e numero random ate 3
                ADD     R2, R3          ; R2 e numero random ate 18
                ADD     R2, 31          ; R2 e nova pos do obstaculo
                CALL    IncObsPassados  ; incrementa obstaculos passados
MoveObs_aux:    PUSH    R2              ; pilha - pos do obstaculo
                PUSH    Obstaculo       ; pilha - obstaculo
                CALL    EscApagObs      ; escreve obstaculo
                MOV     M[R1], R2       ; actualiza pos do obstaculo
MoveObs_Fim:    POP     R3              ; retoma valor de R3
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RETN    1               ; retorna e elimina um valor da pilha

;_____________________________________________________________________________
;                               HaColisao
;       rotina que move obstaculo
;               Entradas:       R2 - pos do obstaculo a ser movido
;               Saidas:         ---
;               Efeitos:        ---

HaColisao:      PUSH    R1              ; salvaguarda R1
                PUSH    R2              ; salvaguarda R2
                PUSH    R3              ; salvaguarda R3
                MOV     R1, R2          ; R1 e pos do obstaculo a ser movido
                AND     R1, 00FFh       ; R1 e coluna do obstaculo
                AND     R2, FF00h       ; R2 e linha do obstaculo
                CMP     R1, R6          ; ve se col do obs e col da bic
                BR.Z    HaCol_aux       ; se sim, vai ver linha
                INC     R1              ; se nao, continua
                CMP     R1, R6          ; ve se segundo * e col da bic
                BR.Z    HaCol_aux       ; se sim, vai ver linha
                INC     R1              ; se nao, continua
                CMP     R1, R6          ; ve se terceiro * e col da bic
                BR.Z    HaCol_aux       ; se sim, vai ver linha
                BR      HCFim           ; se nao, vai para o fim
HaCol_aux:      MOV     R3, PRIM_RODA   ; linha da primeira roda
                SUB     R3, MUDALINHA   ; linha anterior
                CMP     R2, R3          ; ve se esta na linha de R3
                BR.Z    Booom           ; se sim, ha colisao
                ADD     R3, MUDALINHA   ; se nao, passa para linha seg
                CMP     R2, R3          ; ve se esta na linha de R3
                BR.Z    Booom           ; se sim, ha colisao
                ADD     R3, MUDALINHA   ; se nao, passa para linha seg
                CMP     R2, R3          ; ve se esta na linha de R3
```



```
                BR.Z    Booom           ; se sim, ha colisao
                ADD     R3, MUDALINHA   ; se nao, passa para ultima linha
                CMP     R2, R3          ; ve se esta na linha de R3
                BR.Z    Booom           ; se sim, ha colisao
                BR      HCFim           ; se nao, salta para fim
Booom:          MOV     R1, 1           ; R1 e 1
                MOV     M[Colisao], R1  ; Muda var Colisao para 1
HCFim:          POP     R3              ; retoma valor de R3
                POP     R2              ; retoma valor de R2
                POP     R1              ; retoma valor de R1
                RET                     ; retorna


;_____________________________________________________________________________
;                               Random
;       rotina que cria um valor random de 16 bits
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[Random]

Random:         PUSH    R1              ; salvaguarda R1
                MOV     R1, M[RandomNum] ; R1 e valor random
                TEST    R1, 0001h       ; testa ultimo bit de R1
                BR.Z    Random_aux      ; se for zero salta para Random_aux
                XOR     R1, RAND_MASK   ; se nao, faz XOR com a mascara random
Random_aux:     ROR     R1, 3           ; roda3 bits para a direita
                MOV     M[RandomNum], R1 ; actualiza valor random
                POP     R1              ; retoma valor de R1
                RET                     ; retorna

;_____________________________________________________________________________
;                               IncObsPassados
;       rotina que incrementa os obstaculos passados
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[ObsPassados], M[TempEspera], M[LEDS],
;                               M[ObsPass_0], M[ObsPass_1], M[ObsPass_2],
;                               M[ObsPass_3] e displays de 7 segmentos

IncObsPassados: PUSH    R1              ; salvaguarda valor de R1
                PUSH    R2              ; salvaguarda valor de R2
                INC     M[ObsPassados]  ; incrementa obstaculos passados
                CMP     M[Turbo], R0    ; ve se turbo esta activo
                BR.NZ   IncObs_aux_0    ; se estiver activo salta
                CMP     M[UltraTurbo], R0 ; ve se turbo esta activo
                BR.NZ   IncObs_aux_0    ; se estiver activo salta
                MOV     R1, 0004h       ; se nao, R1 e 4
                CMP     M[ObsPassados], R1; ve se obstaculos passados e 4
                BR.Z    Nivel2          ; se sim salta para Nivel2
                ADD     R1, 0004h       ; R1 e 8
                CMP     M[ObsPassados], R1; ve se obstaculos passados e 8
                BR.Z    Nivel3          ; se sim salta para Nivel3
                BR      IncObs_aux_0    ; se nao, salta para IncObs_aux_0
Nivel2:         DEC     M[TempEspera]   ; decrementa tempo de espera
                MOV     R1, FF00h       ; R1 e leds do nivel 2
                MOV     M[LEDS], R1     ; coloca leds do nivel 2
                BR      IncObs_aux_0    ; salta para IncObs_aux_0
Nivel3:         DEC     M[TempEspera]   ; decrementa tempo de espera
                MOV     R1, FFF0h       ; R1 e leds do nivel 3
                MOV     M[LEDS], R1     ; coloca leds do nivel 3
IncObs_aux_0:   INC     M[ObsPass_0]    ; incrementa val de obsPass_0
                MOV     R1, 10          ; R1 e 10
```

```
                CMP     M[ObsPass_0], R1  ; compara com 10
                BR.Z    IncObs_aux_1      ; se for salta
                MOV     R2, M[ObsPass_0]  ; se nao R2 e valor de ObsPass_0
                MOV     M[DISPLAY_0], R2  ; escreve R2 no display_0
                JMP     IncObsFim         ; salta para o fim
IncObs_aux_1:   MOV     M[ObsPass_0], R0  ; coloca val de ObsPass_0 a zero
                MOV     M[DISPLAY_0], R0  ; coloca display_0 a zero
                INC     M[ObsPass_1]      ; incrementa val de ObsPass_1
                CMP     M[ObsPass_1], R1  ; ve se chegou a 10
                BR.Z    IncObs_aux_2      ; se sim salta
                MOV     R2, M[ObsPass_1]  ; se nao actualiza valor R2
                MOV     M[DISPLAY_1], R2  ; actualiza display
                JMP     IncObsFim         ; salta para o fim
IncObs_aux_2:   MOV     M[ObsPass_1], R0  ; coloca val a zero
                MOV     M[DISPLAY_1], R0  ; coloca display a 0
                INC     M[ObsPass_2]      ; incrementa valor de M[ObsPass_2]
                CMP     M[ObsPass_2], R1  ; ve se chegou a 10
                BR.Z    IncObs_aux_3      ; se sim, salta
                MOV     R2, M[ObsPass_2]  ; se nao, actualiza valor R2
                MOV     M[DISPLAY_2], R2  ; actualiza display
                BR      IncObsFim         ; salta para o fim
IncObs_aux_3:   MOV     M[ObsPass_2], R0  ; coloca val a zero
                MOV     M[DISPLAY_2], R0  ; coloca display a zero
                INC     M[ObsPass_3]      ; incrementa valor de M[ObsPass_3]
                CMP     M[ObsPass_3], R1  ; ve se chegou a 10
                BR.Z    IncObs_aux_4      ; se sim, salta
                MOV     R2, M[ObsPass_3]  ; se nao, actualiza valor de R3
                MOV     M[DISPLAY_3], R2  ; actualiza display
                BR      IncObsFim         ; salta para o fim
IncObs_aux_4:   MOV     M[ObsPass_3], R0  ; coloca val a zero
                MOV     M[DISPLAY_3], R0  ; coloca display a zero
IncObsFim:      POP     R2                ; retoma valor de R2
                POP     R1                ; retoma valor de R1
                RET                       ; retorna

;_____________________________________________________________________________
;                               MudaVelTurbo
;       rotina que muda as variaveis para colocar o turbo activo
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[LEDS] e M[TempEspera]

MudaVelTurbo:   PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_TURB       ; desactiva int do ultraturbo
                MOV     M[INT_MASK_ADDR], R1    ; desactiva int do ultraturbo
                MOV     R1, FFFFh               ; R1 e leds do turbo
                MOV     M[LEDS], R1             ; actualiza leds para turbo
                MOV     R1, M[VelTurbo]         ; R1 e velocidade turbo
                MOV     M[TempEspera], R1       ; actualiza tempespera para R1
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               MudaVelU_Turbo
;       rotina que muda as variaveis para colocar o turbo activo
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[LEDS] e M[TempEspera]

MudaVelU_Turbo: PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_ULTURB     ; desactiva int do turbo
                MOV     M[INT_MASK_ADDR], R1    ; desactiva int do turbo
```



```
                MOV     R1, 0101010101010101b   ; R1 e leds do turbo
                MOV     M[LEDS], R1             ; actualiza leds para turbo
                MOV     R1, M[VelUltraTurbo]    ; R1 e velocidade turbo
                MOV     M[TempEspera], R1       ; actualiza tempoespera para R1
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               RepoeVelTurbo
;       rotina que repoe a velocidade e os leds quando turbo e inactivo
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[LEDS] e M[TempEspera]

RepoeVelTurbo:  PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_PRIN       ; R1 e mascara principal
                MOV     M[INT_MASK_ADDR], R1    ; volta a colocar mascara prin
                MOV     M[Turbo], R0            ; coloca val de turbo a zero
                MOV     R1, 0008h               ; R1 e 8
                CMP     M[ObsPassados], R1      ; compara obsPassados com 8
                BR.NN   RVT_Nivel3              ; se maior que 8 salta
                MOV     R1, 0004h               ; R1 e 4
                CMP     M[ObsPassados], R1      ; compara obsPassados com 4
                BR.NN   RVT_Nivel2              ; se maior que 4 salta
                MOV     R1, 0005h               ; R1 e 5
                MOV     M[TempEspera], R1       ; TempEspera act. p velNivel1
                MOV     R1, F000h               ; R1 e leds nivel 1
                MOV     M[LEDS], R1             ; actualiza leds para nivel 1
                BR      RVT_Fim                 ; salta para fim
RVT_Nivel2:     MOV     R1, FF00h               ; R1 e leds nivel 2
                MOV     M[LEDS], R1             ; actualiza leds para nivel 2
                MOV     R1, 0004h               ; R1 e 4
                MOV     M[TempEspera], R1       ; actualiza tempo de espera
                BR      RVT_Fim                 ; salta para fim
RVT_Nivel3:     MOV     R1, FFF0h               ; R1 e leds nivel 3
                MOV     M[LEDS], R1             ; actualiza leds para nivel 3
                MOV     R1, 0003h               ; R1 e 3
                MOV     M[TempEspera], R1       ; actualiza tempo de espera
                BR      RVT_Fim                 ; salta para fim
RVT_Fim:        POP     R1                      ; retoma valor de R1
                RET                             ; retorna


;_____________________________________________________________________________
;                               RepoeVelU_Turbo
;       rotina que repoe a velocidade e os leds quando turbo e inactivo
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        muda M[LEDS] e M[TempEspera]

RepoeVelU_Turbo:PUSH    R1                      ; salvaguarda R1
                MOV     R1, INT_MASK_PRIN       ; R1 e mascara principal
                MOV     M[INT_MASK_ADDR], R1    ; volta a colocar mascara prin
                MOV     M[UltraTurbo], R0       ; coloca val de turbo a zero
                MOV     R1, 0008h               ; R1 e 8
                CMP     M[ObsPassados], R1      ; compara obsPassados com 8
                BR.NN   RVT_UNivel3             ; se maior que 8 salta
                MOV     R1, 0004h               ; R1 e 4
                CMP     M[ObsPassados], R1      ; compara obsPassados com 4
                BR.NN   RVT_UNivel2             ; se maior que 4 salta
                MOV     R1, 0005h               ; R1 e 5
                MOV     M[TempEspera], R1       ; TempEspera act. p velNivel1
```

```
                MOV     R1, F000h               ; R1 e leds nivel 1
                MOV     M[LEDS], R1             ; actualiza leds para nivel 1
                BR      RVT_UFim                ; salta para fim
RVT_UNivel2:    MOV     R1, FF00h               ; R1 e leds nivel 2
                MOV     M[LEDS], R1             ; actualiza leds para nivel 2
                MOV     R1, 0004h               ; R1 e 4
                MOV     M[TempEspera], R1       ; actualiza tempo de espera
                BR      RVT_UFim                ; salta para fim
RVT_UNivel3:    MOV     R1, FFF0h               ; R1 e leds nivel 3
                MOV     M[LEDS], R1             ; actualiza leds para nivel 3
                MOV     R1, 0003h               ; R1 e 3
                MOV     M[TempEspera], R1       ; actualiza tempo de espera
                BR      RVT_UFim                ; salta para fim
RVT_UFim:       POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;_____________________________________________________________________________
;                               CriaObs2
;       rotina que cria obstaculo 2
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

CriaObs2:       PUSH    Obstaculo2              ; pilha - obstaculo 2
                CALL    CriaObstaculo           ; cria obstaculo 2
                RET                             ; retorna

;_____________________________________________________________________________
;                               CriaObs3
;       rotina que cria obstaculo 3
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

CriaObs3:       PUSH    Obstaculo3              ; pilha - obstaculo 3
                CALL    CriaObstaculo           ; cria obstaculo 3
                RET                             ; retorna

;_____________________________________________________________________________
;                               CriaObs4
;       rotina que cria obstaculo 4
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

CriaObs4:       PUSH    Obstaculo4              ; pilha - obstaculo 4
                CALL    CriaObstaculo           ; cria obstaculo 4
                RET                             ; retorna

;_____________________________________________________________________________
;                               MoveObstaculos
;       rotina que move os obstaculos
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

MoveObstaculos: PUSH    Obstaculo1              ; pilha - Obstaculo1
                CALL    MoveObs                 ; move Obstaculo1
                PUSH    Obstaculo2              ; pilha - Obstaculo2
                CALL    MoveObs                 ; move Obstaculo2
                PUSH    Obstaculo3              ; pilha - Obstaculo3
                CALL    MoveObs                 ; move Obstaculo3
```



```
                PUSH    Obstaculo4              ; pilha - Obstaculo4
                CALL    MoveObs                 ; move Obstaculo4
                RET                             ; retorna

;_____________________________________________________________________________
;                               CicloInicial
;       rotina do ciclo que corre até estarem criados os 4 obstaculos
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

CicloInicial:   MOV     R1, 0006h               ; R1 e 6
                CALL    Espera2                 ; rotina espera2
                PUSH    Obstaculo1              ; pilha - Obstaculo1
                CALL    MoveObs                 ; move obstaculo 1
                CMP     M[DistPercorrida], R1   ; ve se ja percorreu 6
                BR.Z    CicloIni_2              ; se sim salta para CicloIni_2
                CMP     M[Colisao], R0          ; se nao, ve se houve colisao
                BR.Z    CicloInicial            ; se nao, repete ciclo
CicloIni_2:     CALL    CriaObs2                ; cria obstaculo 2
                ADD     R1, 0006h               ; incrementa 6 a R1
CicloIni_2_aux: CALL    Espera2                 ; rotina espera2
                PUSH    Obstaculo1              ; pilha - Obstaculo1
                CALL    MoveObs                 ; move obstaculo 1
                PUSH    Obstaculo2              ; pilha - Obstaculo2
                CALL    MoveObs                 ; move obstaculo 2
                CMP     M[DistPercorrida], R1   ; compara R1 com DistPerc.
                BR.Z    CicloIni_3              ; se forem iguais salta
                CMP     M[Colisao], R0          ; se nao, ve se houve colisao
                BR.Z    CicloIni_2_aux          ; se nao, repete ciclo
CicloIni_3:     CALL    CriaObs3                ; cria obstaculo 3
                ADD     R1, 0006h               ; incrementa 6 a R1
CicloIni_3_aux: CALL    Espera2                 ; rotina espera2
                PUSH    Obstaculo1              ; pilha - Obstaculo1
                CALL    MoveObs                 ; move obstaculo 1
                PUSH    Obstaculo2              ; pilha - Obstaculo2
                CALL    MoveObs                 ; move obstaculo 2
                PUSH    Obstaculo3              ; pilha - Obstaculo3
                CALL    MoveObs                 ; move obstaculo 3
                CMP     M[DistPercorrida], R1   ; compara R1 com DistPerc.
                BR.Z    CicloIni_4              ; se forem iguais salta
                CMP     M[Colisao], R0          ; se nao, ve se houve colisao
                BR.Z    CicloIni_3_aux          ; se nao, repete ciclo
CicloIni_4:     CALL    CriaObs4                ; cria obstaculo 4
                RET                             ; retorna

;_____________________________________________________________________________
;                               ActDistMax
;       rotina que actualiza a distancia maxima
;               Entradas:       ---
;               Saidas:         ---
;               Efeitos:        ---

ActDistMax:     PUSH    R1                      ; salvaguarda R1
                MOV     R1, M[DistPercorrida]   ; R1 e distancia percorrida
                MOV     M[DistMaxima], R1       ; val de DistMaxima e DistPerc.
                MOV     R1, M[D_0]              ; actualiza distancia maxima
                MOV     M[D_0_max], R1
                MOV     R1, M[D_1]              ; actualiza distancia maxima
                MOV     M[D_1_max], R1
                MOV     R1, M[D_2]              ; actualiza distancia maxima
                MOV     M[D_2_max], R1
```

```
                MOV     R1, M[D_3]              ; actualiza distancia maxima
                MOV     M[D_3_max], R1
                MOV     R1, M[D_4]              ; actualiza distancia maxima
                MOV     M[D_4_max], R1
                CALL    EscDistMax              ; escreve distancia maxima
                POP     R1                      ; retoma valor de R1
                RET                             ; retorna

;-------------------------------------------------------------------------------
;|                          ZONA IV: PROGRAMA PRINCIPAL                        |
;|                                                                             |
;|                    Programa que corre o jogo da bicicleta                   |
;-------------------------------------------------------------------------------

Jogo:           MOV     R1, SP_INICIAL          ; R1 e posicao inicial da pilha
                MOV     SP, R1                  ; inicializa a pilha na pos R1
                CALL    IniPortoCtrl            ; inicia porto de controlo
                CALL    EscInicio               ; escreve frases iniciais
                CALL    LCD_EscTexto            ; escreve texto das dist no LCD
                CALL    EscDistMax              ; escreve zero na distancia max
                CALL    EscDist                 ; escreve zero na distancia perc
ParteI:         CALL    Espera1                 ; espera por I1
                CALL    Inicializa              ; inicializa variaveis
                MOV     R1, 1                   ; R1 e 1
                MOV     M[ACT_REL], R1          ; activa relogio
                MOV     R1, M[TempEspera]       ; R1 e tempo a esperar
                MOV     M[REL], R1              ; inicia tempo de espera
                CALL    CicloInicial            ; corre ciclo inicial
CicloPrinc:     CALL    Espera2                 ; rotina espera2
                CALL    MoveObstaculos          ; move obstaculos
                CMP     M[Colisao], R0          ; ve se houve colisao
                BR.Z    CicloPrinc              ; repete se nao houver colisao
FimJogo:        CALL    EscFim                  ; escreve mensagens finais
                MOV     M[ObsPass_0], R0        ; coloca ObsPass_0 a zero
                MOV     M[ObsPass_1], R0        ; coloca ObsPass_1 a zero
                MOV     M[ObsPass_2], R0        ; coloca ObsPass_2 a zero
                MOV     M[ObsPass_3], R0        ; coloca ObsPass_3 a zero
                MOV     M[Turbo], R0            ; coloca turbo inactivo
                MOV     M[UltraTurbo], R0       ; coloca ultra turbo inactivo
                MOV     R1, M[DistPercorrida]   ; R1 e distpercorrida
                SUB     R1, M[DistMaxima]       ; subtrai DistPerc a DistMax
                CALL.NN ActDistMax              ; call se DPerc maior que DMax
                MOV     M[DistPercorrida], R0   ; coloca DistPercorrida a zero
                MOV     M[ObsPassados], R0      ; coloca ObsPassados a zero
                CALL    LimpaDist               ; limpa distancia
                CALL    LimpaDisplays           ; limpa displays de 7 segmentos
                CALL    EscDist                 ; escreve distancia
                MOV     R1, 0005h               ; R1 e TempEspera inicial
                MOV     M[TempEspera], R1       ; reinicia TempEspera
                JMP     ParteI                  ; salta para parteI
```