```
; *=============================================*
; *                                             *
; * PROJETO IAC 1º SEMESTRE: PASSARO BAMBOLEANTE *
; *                                             *
; *=============================================*
; *============================*
; * Trabalho realizado por:    *
; * Amândio Faustino   Nº83422 *
; * João Sousa         Nº83487 *
; * Pedro Lopes        Nº83540 *
; *               Grupo nº 48 *
; *============================*
;########################################
;#             CONSTANTES               #
;########################################
;Posicao inicial do SP
SP_INICIAL      EQU     FDFFh

;Constantes para escrita na Janela de Texto
CONTROLO        EQU     FFFFh                   ;Porto que permite inici
ar escrita na Janela de Texto
ESCRITA_JT      EQU     FFFEh                   ;Porto escrita Janela de
 Texto
CURSOR_JT       EQU     FFFCh                   ;Porto que permite posic
ionar o cursor Janela de Texto

;Constantes para escrita LCD
ESCRITA_LCD     EQU     FFF5h
CURSOR_LCD      EQU     FFF4h

;Constantes displays
IO_LED          EQU     FFF8h                   ;Porto dos LED
IO_DISPLAY_1    EQU     FFF0h                   ;Portos do Display7s
IO_DISPLAY_2    EQU     FFF1h
IO_DISPLAY_3    EQU     FFF2h
IO_DISPLAY_4    EQU     FFF3h

;Constantes das Interrupcoes
INT_MASK_ADDR   EQU     FFFAh
INT_MASK        EQU     1000000000001111b

;Constantes para o temporizador
EST_RELOG_M     EQU     FFF7h                   ;Porto que permite ligar
/desligar o Temporizador
EST_RELOG       EQU     0001h
DIF_TEMPO_M     EQU     FFF6h                   ;Porto que permite selec
ionar o tempo da proxima int
DIF_TEMPO       EQU     1

;Constantes Construtoras
COUNT_TRACO     EQU     79                      ;Count_traco e o numero
de vezes que o traco e escrito
TRACO           EQU     '-'
ESPACO          EQU     ' '
CABECA          EQU     'O'
BICO            EQU     '>'
OBS             EQU     'X'
FIM_STR         EQU     '@'

;Constantes Colunas e Linhas
GET_LINHA       EQU     FF00h
GET_COL         EQU     00FFh
```



```
INC_LINHA       EQU     0100h
LINHA_22        EQU     1600h
COL_78          EQU     004Eh
ESPACAMENTO     EQU     0500h                   ;Espacamento entre as co
lunas verticais
COLPASSARO      EQU     0014h                   ;Colpassaro tem a coluna
 do passaro

;Mascaras
RANDOM_MASK     EQU     1000000000010110b

;Constantes Random
BIT_MENOR       EQU     0001h
DIVISOR         EQU     17                      ;Para gerar um valor ent
re 0 e 16 ao qual vamos incrementar mais 1 de forma a ficar entre 1 e 17

;Posicoes Texto
POS_LCD_L1_1    EQU     8000h
POS_LCD_L1_2    EQU     800Ah
POS_TEXTO_M1_L1 EQU     0314h
POS_TEXTO_M1_L2 EQU     0414h
POS_TEXTO_L1    EQU     0C21h
POS_TEXTO_L2    EQU     0A14h
POS_PONTUACAO   EQU     0E1Fh
POS_SCORE       EQU     0E2Ah

;Constantes Tubos
NUM_MAX_TUBOS   EQU     7                       ;Esta constante tem de s
er sempre igual ao nr TAB TUBOS -1
ESPACO_TUBOS    EQU     10                      ;Espaço entre tubos
NO_NEED         EQU     0012h                   ;Coluna a partir do qual
 já não interessa verificar a Colisão com um determinado obstáculo

;Constantes Passaro
POS_PASSARO     EQU     0C14h                   ;Primeira posição do pás
saro

;Constantes Gravidade
DURACAOTICK     EQU     1                       ;Refresh rate
ACELERACAO      EQU     5

;Constantes Salto
ALTURA_SALTO    EQU     3

;Constantes Niveis
POS_STR_NIVEL   EQU     0C31h
NIVEL_1         EQU     6
NIVEL_2         EQU     4
NIVEL_3         EQU     2
DISPLAY_N1      EQU     F000h
DISPLAY_N2      EQU     FF00h
DISPLAY_N3      EQU     FFF0h

;Constantes Distancia/Obstaculos
DIST_SEM_OBS    EQU     003Bh                   ;Distancia percorrida at
é encontrar o primeiro obstáculo.
CONV_ASCII      EQU     0030h

;Constantes sobre Variáveis
ZONA_VARS       EQU     8000h
NUM_VARS        EQU     32
MUDA_NIVEL_VAR  EQU     2
```

```
ESPACO_ATUALVAR EQU     10                              ;É importante esta const
ante estar igual ESPACO_TUBOS.

;#########################################
;#              VARIAVEIS                #
;#########################################
                ORIG    8000h
COORD_TRACO     WORD    0000h                           ;Memoria onde vai guarda
r a coordenada do proximo traco a escrever
SALTOU          WORD    0000h
TEMPO_SALTO     WORD    0000h
TEMPO_GRAVIDADE WORD    0000h
CONTADOR_TEMPO  WORD    0000h
NUMRANDOM       WORD    0000h
COL_DESE        WORD    0000h
NIVEL           WORD    0000h
ASS_NIVEL       WORD    0000h
I1              WORD    0000h
I3              WORD    0000h
NEW_COL         WORD    0000h
POS_TUBOS       WORD    0000h
Y0              WORD    0000h
POINTER_TUBO    WORD    0000h
DISTANCIA_PER   WORD    0000h
OBS_ULTRAPASSA  WORD    0000h
MUDA_NIVEL      WORD    0002h
ESPACO_ATUAL    WORD    000Ah
SCORE           TAB     5
TUBOS           TAB     8

;#########################################
;#                STRINGS                #
;#########################################
TextoM1         STR     '========================================@'
TextoM1_2       STR     '||          PASSARO BAMBOLEANTE         ||@'
TextoM1_3       STR     '|| Dificuldade          <<I1 1/3 I2>> ||@'
TextoM1_4       STR     '||             <<PLAY I3>>              ||@'
TextoM1_5       STR     '||                                      ||@'

TextoInicial    STR     'Prepare-se!@'
TextoInicial2   STR     'Prima o interruptor I1 para comecar.@'
TextoFinal      STR     'Fim do Jogo@'
TextoFinal2     STR     'Pontuacao: @'
TextoFinal3     STR     'Prima o interruptor I1 para recomecar.@'
NUM_1           STR     '1'
NUM_2           STR     '2'
NUM_3           STR     '3'
TextoLCD        STR     'Distancia: 00000@'


;#########################################
;#         TABELA DE INTERRUPÇOES        #
;#########################################
                ORIG    FE00h
INT0            WORD    SOBE
INT1            WORD    COMECO_D_NIVEL
INT2            WORD    AUMENTA_NIVEL
INT3            WORD    STARTGAME

                ORIG    FE0Fh
INT15           WORD    TEMPORIZADOR
```



```
;Documentação:
;       R1 - Registo fixo com o valor da posição do pássaro
;       R2 - Registo fixo com o valor da posição do pássaro
;       I0 - Saltar
;       I1 - Diminuir nível/Continuar
;       I2 - Aumentar nível
;       I3 - Continuar

; *=====================================================================================*
; *                                    Código                                           *
; *=====================================================================================*
                ORIG    0000h
                JMP     INICIO


;#########################################
;#              INTERRUPÇOES             #
;#=======================================#
;# SOBE - I0                             #
;# COMECO_D_NIVEL - I1                   #
;# AUMENTA_NIVEL - I2                    #
;# STARTGAME - I3                        #
;# TEMPORIZADOR - I15                    #
;#########################################
; =====================================================================================*
; TEMPORIZADOR: Rotina que trata da interrupção I15                                    *
; =====================================================================================*
TEMPORIZADOR:   DEC     M[ASS_NIVEL]                    ;Decrementa sempre o con
tador ASS_NIVEL
                CMP     M[SALTOU],R0                    ;SALTOU=0 ?, salto foi r
ecentemente ativado?
                BR.NZ   SALTO1                          ;Se sim -> SALTO1
                INC     M[TEMPO_GRAVIDADE]              ;Gravidade ativa, tempo_
gravidade informa que ja pode ser atualizado o passaro
                INC     M[CONTADOR_TEMPO]
                BR      FIMINT15                        ;Vai para o fim da inter
rupção
SALTO1:         MOV     M[TEMPO_GRAVIDADE],R0           ;Reinicia os contadores
da gravidade.
                MOV     M[CONTADOR_TEMPO],R0
                INC     M[TEMPO_SALTO]                  ;Passou tempo suficiente
 para subir mais uma posição
FIMINT15:       MOV     R7,DIF_TEMPO                    ;Configuração para a pró
xima interrupção do temporizador
                MOV     M[DIF_TEMPO_M],R7
                MOV     R7,EST_RELOG
                MOV     M[EST_RELOG_M],R7
                RTI

; =====================================================================================*
; SOBE: Rotina que trata da interrupção I0, gere o salto do pássaro                    *
; =====================================================================================*
SOBE:           PUSH    R3
                CMP     M[SALTOU],R0                    ;SALTOU recentemente ?
                BR.NZ   FIMSOBE                         ;Se sim não incrementa m
ais -> FIM SOBE
                MOV     R3,ALTURA_SALTO                 ;Caso contrário coloca a
ltura do salto em SALTOU
                ADD     M[SALTOU],R3
FIMSOBE:        POP     R3
                RTI
```

```
; ==============================================================================*
; COMECO_D_NIVEL:Rotina que trata da interrupção I1, gere a mudança de nível   *
;                (diminui) e o inicio do jogo*                                 *
; ==============================================================================*
COMECO_D_NIVEL: PUSH    R7
                MOV     R7,2
                CMP     R7,M[MUDA_NIVEL]              ;Incrementa nível até ao
 limite 2. Ficando o nivel mais lento(necessita de mais interrupçoes para avanca
r)
                BR.Z    FIM_INT1
                INC     M[MUDA_NIVEL]
FIM_INT1:       MOV     R7, 1                         ;Cada vez que I1 é gerad
a, M[I1]=1
                MOV     M[I1],R7
                POP     R7
                RTI

; ==============================================================================*
; AUMENTA_NIVEL: Rotina que trata da interrupção I2, gere a mudança de nível   *
;                (aumenta)                                                     *
; ==============================================================================*
AUMENTA_NIVEL:  PUSH    R7
                CMP     M[MUDA_NIVEL],R0              ;Decrementa nível até li
mite 0. Ficando o nivel mais rapido(necessita de menos interrupçoes para avancar
)
                BR.Z    FIM_INT2
                DEC     M[MUDA_NIVEL]
FIM_INT2:       POP     R7
                RTI

; ==============================================================================*
; STARTGAME: Rotina que trata da interrupção I3, gere a continuação do menu1   *
; ==============================================================================*
STARTGAME:      PUSH    R7
                MOV     R7, 1
                MOV     M[I3],R7                      ;Cada vez que I3 é gerad
a, M[I3]=1
                POP     R7
                RTI

;########################################
;#           ROTINAS SECUNDARIAS        #
;#======================================#
;# RESET_VARS                           #
;# LINHA                                #
;# ESCREVESTR                           #
;# ESCREVESTR_LCD                       #
;# ESCREVESTRNLCD                       #
;# AtualizaD7S                          #
;# MENU1                                #
;# MENU2                                #
;# PASSARO                              #
;# SALTO                                #
;# GRAVIDADE                            #
;# OBSTACULOS                           #
;# DISTANCIA_TUBOS                      #
;# ATUALIZA_DISPLAYS                    #
;# DESENHACOL                           #
;# APAGA_COL                            #
;# LIMPA_JT                             #
;# RANDOMIZER                           #
;# COLISION_V                           #
```



```
;########################################
; ==============================================================================*
; RESET_VARS: Rotina que reinicializa as variáveis.                             *
; ==============================================================================*
RESET_VARS:     PUSH    R3
                PUSH    R4
                MOV     R3, ZONA_VARS
                MOV     R4, NUM_VARS
CICLO:          MOV     M[R3],R0                      ;Ciclo que coloca 'NUM_V
ARS' espaços de memória a 0
                INC     R3
                DEC     R4
                BR.NZ   CICLO
                MOV     R3, 0002h                     ;Outras variáveis que nã
o começam a 0
                MOV     M[MUDA_NIVEL], R3
                MOV     R3, 000Ah
                MOV     M[ESPACO_ATUAL],R3
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; LINHA: Rotina que cria uma linha de 79 traços horizontalmente. O argumento é *
;        recebido pelo porto COORD_TRACO                                        *
; ==============================================================================*
LINHA:          PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     R5,COUNT_TRACO                ;R4=COUNT_TRACO=79
LINHA_C:        MOV     R3,M[COORD_TRACO]             ;R2=Coordenada onde o pr
oximo traco sera escrito
                MOV     M[CURSOR_JT],R3               ;Envia coordenada para o
 porto de posicionamento
                MOV     R4,TRACO                      ;R3 tem o  em ASCI
                MOV     M[ESCRITA_JT],R4              ;Envia  para o porto de
escrita
                INC     M[COORD_TRACO]                ;Proxima coordenada, ape
nas aumenta a coluna
                DEC     R5                            ;Decrementa o COUNT_TRAC
O
                BR.NZ   LINHA_C                       ;Se nao for 0 e porque a
inda nao escreveu as 79 tracos
                POP     R5
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; ESCREVESTR: Rotina escreve uma string na Janela de Texto. Recebe os argumen- *
;             tos pela pilha (Posicao e String)                                 *
; ==============================================================================*
ESCREVESTR:     PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    R6
                MOV     R3,M[SP+7]                    ;Em R3 fica a string
                MOV     R4,M[SP+6]                    ;Em R4 a posicao do prim
eiro caracter
                MOV     R5,FIM_STR                    ;Em R5 o caracter de fim
 de texto
CICLO_ESC_STR:  CMP     M[R3],R5                      ;O carater é o @?
```

```
                BR.Z    FIM_ESC_STR
                MOV     R6,M[R3]
                MOV     M[CURSOR_JT],R4
                MOV     M[ESCRITA_JT],R6
                INC     R4
                INC     R3
                BR      CICLO_ESC_STR
FIM_ESC_STR:    POP     R6
                POP     R5
                POP     R4
                POP     R3
                RETN    2

; ==============================================================================*
; ESCREVESTR_LCD: Rotina escreve uma string no LCD. Recebe os argumentos pela  *
;                 pilha (Posicao e String)                                      *
; ==============================================================================*
ESCREVESTR_LCD: PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    R6
                MOV     R3,M[SP+7]                      ;Em R3 fica a string (pe
la pilha)
                MOV     R4,M[SP+6]                      ;Em R4 a posicao do prim
eiro caracter (pela pilha)
                MOV     R5,FIM_STR                      ;Em R5 o caracter de fim
 de texto
CICLOESCSTR_LCD:CMP     M[R3],R5                        ;O carater é o @?
                BR.Z    FIMESC_STR_LCD
                MOV     R6,M[R3]
                MOV     M[CURSOR_LCD],R4
                MOV     M[ESCRITA_LCD],R6
                INC     R4
                INC     R3
                BR      CICLOESCSTR_LCD
FIMESC_STR_LCD: POP     R6
                POP     R5
                POP     R4
                POP     R3
                RETN    2

; ==============================================================================*
; EscStringNLCD: Processa um numero de 16 bits e escreve no LCD, em cada posi- *
;                ção consecutiva, um digito que corresponde a cada 4 bits do   *
;                numero.                                                       *
;                Recebe os argumentos pela pilha (Posicao e String)            *
; ==============================================================================*
EscStringNLCD:  PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     R3,M[SP+6]                      ;Recebe o numero pela pi
lha
                MOV     R5,M[SP+5]                      ;Recebe a posiçao pela p
ilha
                ADD     R5,5                            ;Escrevemos o numero de
tras para a frente (direita para a esquerda) logo o R5 tem a ultima posiçao
CICLONLCD:      CMP     R5,M[SP+5]
                BR.Z    FIMNLCD
                MOV     R4,10                           ;Dividir o numero por 10
 para tirar o algarismo da direita
                DIV     R3,R4                           ;Que fica em R2
                OR      R4,CONV_ASCII                   ;Fazemos um OR com 0030h
```



```
 para obter o codigo ASCII
                MOV     M[CURSOR_LCD],R5
                MOV     M[ESCRITA_LCD],R4
                DEC     R5
                BR      CICLONLCD
FIMNLCD:        POP     R5
                POP     R4
                POP     R3
                RETN    2

; ==============================================================================*
; AtualizaD7S: Atualiza o display de 7 segmentos com o numero de obstaculos    *
;              ultrapassados,                                                   *
;              Entradas adicionais: OBS_ULTRAPASSA                              *
; ==============================================================================*
AtualizaD7S:    PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     R5,SCORE
                MOV     R4,M[OBS_ULTRAPASSA]            ;R4=Nr obstaculos ultrap
asados (em hexa)
                MOV     R3,10                           ;Tira o algarismo mais a
 direita do numero
                DIV     R4,R3                           ;Que fica em R3
                MOV     M[IO_DISPLAY_1],R3              ;Coloca no display
                OR      R3,CONV_ASCII
                MOV     M[R5+3],R3                      ;Coloca na 4ª posição do
 TAB SCORE (ser usado como string)
                MOV     R3,10
                DIV     R4,R3
                MOV     M[IO_DISPLAY_2], R3
                OR      R3,CONV_ASCII
                MOV     M[R5+2],R3                      ;Coloca na 3ª posição do
 TAB SCORE (ser usado como string)
                MOV     R3,10
                DIV     R4,R3
                MOV     M[IO_DISPLAY_3], R3
                OR      R3,CONV_ASCII
                MOV     M[R5+1],R3                      ;Coloca na 2ª posição do
 TAB SCORE (ser usado como string)
                MOV     M[IO_DISPLAY_4], R4
                OR      R4,CONV_ASCII
                MOV     M[R5],R4                        ;Coloca na 1ª posição do
 TAB SCORE (ser usado como string)
                POP     R5
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; MENU1: Rotina que desenha a interface do primeiro menu. Neste menu é possivel*
;        mudar os níveis do jogo                                               *
; ==============================================================================*
MENU1:          PUSH    R3                              ;Escreve as strings por
ordem na Janela de Texto
                PUSH    R4
                PUSH    R5
                PUSH    R7
                MOV     R3,TextoM1
                MOV     R5,POS_TEXTO_M1_L1
                MOV     R4,R5
                PUSH    R3
```

```
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_5
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_2
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_5
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                ADD     R5,INC_LINHA
                ADD     R5,INC_LINHA
                MOV     R3,TextoM1
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_5
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_3
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_5
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1_4
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoM1
                ADD     R5,INC_LINHA
                MOV     R4,R5
                PUSH    R3
```



```
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoLCD
                MOV     R4,POS_LCD_L1_1
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR_LCD
CICLO_MENU1:    MOV     R7,0001h
                CMP     R7,M[MUDA_NIVEL]
                BR.N    LEVEL_1                 ;Se MUDA_NIVEL=2  1-2<0
-> Nivel 1
                BR.Z    LEVEL_2                 ;Se MUDA_NIVEL=1  1-1=0
-> Nivel 2
                JMP.P   LEVEL_3                 ;Se MUDA_NIVEL=0  1-0>0
-> Nivel 3
LEVEL_1:        MOV     R7,NIVEL_1
                MOV     M[NIVEL],R7             ;Coloca a velocidade niv
el 1 na variavel NIVEL
                MOV     R3,POS_STR_NIVEL
                MOV     R4,M[NUM_1]             ;Escreve o número 1 na J
anela de Texto [1/3]
                MOV     M[CURSOR_JT],R3
                MOV     M[ESCRITA_JT],R4
                MOV     R3,IO_LED
                MOV     R4,DISPLAY_N1           ;Atualiza os LED para o
nível 1
                MOV     M[R3],R4
                JMP     TEST_I3                 ;Testa a interrupção I3
LEVEL_2:        MOV     R7,NIVEL_2
                MOV     M[NIVEL],R7
                MOV     R3,POS_STR_NIVEL
                MOV     R4,M[NUM_2]             ;Escreve o número 1 na J
anela de Texto [2/3]
                MOV     M[CURSOR_JT],R3
                MOV     M[ESCRITA_JT],R4
                MOV     R3,IO_LED
                MOV     R4,DISPLAY_N2           ;Atualiza os LED para o
nível 2
                MOV     M[R3],R4
                JMP     TEST_I3                 ;Testa a interrupção I3
LEVEL_3:        MOV     R7,NIVEL_3
                MOV     M[NIVEL],R7
                MOV     R3,POS_STR_NIVEL
                MOV     R4,M[NUM_3]             ;Escreve o número 3 na J
anela de Texto [3/3]
                MOV     M[CURSOR_JT],R3
                MOV     M[ESCRITA_JT],R4
                MOV     R3,IO_LED
                MOV     R4,DISPLAY_N3           ;Atualiza os LED para o
nível 3
                MOV     M[R3],R4
TEST_I3:        CMP     M[I3],R0
                JMP.Z   CICLO_MENU1             ;Se I3 for ativa, Limpa
Janela de Texto e retorna
                MOV     M[I3],R0
                CALL    LIMPA_JT
                POP     R7
                POP     R5
                POP     R4
                POP     R3
                RET
```

```
; ==============================================================================*
; MENU2: Rotina que desenha a interface do segundo menu. Este é o menu de      *
;        preparação.                                                           *
; ==============================================================================*
MENU2:          PUSH    R3                      ;Escreve as mensagens do
 menu2 na Janela de Texto
                PUSH    R4
                MOV     R3,TextoInicial
                MOV     R4,POS_TEXTO_L1
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3, TextoInicial2
                MOV     R4, POS_TEXTO_L2
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
CICLO_MENU2:    INC     M[NUMRANDOM]            ;Enquanto I1 não for ati
va, incrementa o contador para a rotina RANDOMIZER (Gerar número aleatório)
                CMP     M[I1],R0
                BR.Z    CICLO_MENU2             ;Se I1 for ativa, Limpa
Janela de Texto e retorna
                CALL    LIMPA_JT
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; PASSARO: Rotina que apaga o pássaro na posicao antiga e escreve o pássaro na *

;          nova posição                                                        *
;          Rotina só é chamada se e só se R1!=R2                               *
; ==============================================================================*
PASSARO:        PUSH    R3
                MOV     M[CURSOR_JT],R1         ;Cursor vai para o valor
 R1 que contem o h_antigo
                MOV     R3,ESPACO               ;R3 e o codigo ASCI espa
co
                MOV     M[ESCRITA_JT],R3        ;Escreve o espaco onde e
sta o cursor (para apagar)
                INC     R1                      ;Incrementa uma coluna
                MOV     M[CURSOR_JT],R1         ;Cursor vai para o valor
 R1 que contem o h_antigo
                MOV     M[ESCRITA_JT],R3        ;Escreve o espaco onde e
sta o cursor (para apagar)
                MOV     M[CURSOR_JT],R2         ;Cursor vai para o valor
 R1 que contem o h_novo
                MOV     R3,CABECA               ;R3=CABECA
                MOV     M[ESCRITA_JT],R3        ;Escreve a cabeca
                INC     R2                      ;Coordenada para escreve
r o bico
                MOV     M[CURSOR_JT],R2         ;Coloca a coordenada no
porto
                MOV     R3,BICO                 ;R3=BICO
                MOV     M[ESCRITA_JT],R3        ;Escreve o bico
                DEC     R2                      ;Coordenada da cabeca (o
riginal)
                MOV     R1,R2                   ;R1=R2
                POP     R3
                RET

; ==============================================================================*
```



```
; SALTO: Rotina que, se houve salto decrementa a linha da posição do pássaro   *
;        (R2). Guarda a posição final do salto em Y0                           *
;        R2 é alterado                                                         *
; ==============================================================================*
SALTO:          CMP     M[TEMPO_SALTO],R0       ;TEMPO_SALTO=0 => ainda
não passou tempo suficiente
                BR.Z    FIMSALTO                ;Retorna
                MOV     M[TEMPO_SALTO],R0       ;Caso contrário, reinici
a e subtrai uma linha à posição do
                SUB     R2,INC_LINHA            ;pássaro (o que implica
que sobe)
                DEC     M[SALTOU]
                MOV     M[Y0],R2                ;Guarda sempre a posição
 do passaro em Y0 (ser usado na gravidade)
FIMSALTO:       RET

; ==============================================================================*
; GRAVIDADE: Rotina que após um tempo sem saltar, é calculada a nova posição do*

;            pássaro (de tick em tick) de acordo com a fórmula y=y0+v0t+gt^2,  *

;            v0=0. R2 é alterado                                               *
; ==============================================================================*
GRAVIDADE:      PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    R6
                PUSH    R7
                MOV     R3,DURACAOTICK
                CMP     M[TEMPO_GRAVIDADE],R3   ;Se já passou o tempo (d
uraçãotick) desde a última vez
                BR.NP   FIMGRAVIDADE            ;Caso contrário retorna
                MOV     R3,ACELERACAO
                MOV     R4,M[CONTADOR_TEMPO]    ;R4 TEM TEMPO
                MOV     R5,R4                   ;R5=R4=T//10

                MOV     R6,M[Y0]                ;R6 é a posição inicial
(Y0)
                AND     R6,GET_LINHA            ;R6 é a linha inicial (l
inha de Y0)
                MUL     R4,R5                   ;R5=T^2
                MUL     R3,R5                   ;R5=ACELERACAO X T^2
                ENI
                ADD     R6,R5                   ;Soma à linha inicial o
valor calculado em R5
                AND     R6,GET_LINHA            ;Virgula fixa, parte int
eira é o octeto de maior peso
                ADD     R6,COLPASSARO           ;Soma a coluna do pássar
o para perfazer a coordenada da
                MOV     R2,R6                   ;posição do pássaro e é
passado para R2
                MOV     M[TEMPO_GRAVIDADE],R0   ;Reinicializa variável
FIMGRAVIDADE:   POP     R7
                POP     R6
                POP     R5
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; OBSTACULOS: Rotina que move e atualiza os obstáculos. É atualizado também as *
;             váriáveis: DISTANCIA_PER e OBS_ULTRAPASSA                         *
```

```
; ==============================================================================*
OBSTACULOS:     PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    R6
                PUSH    R7
                MOV     R3,M[NIVEL]
                MOV     M[ASS_NIVEL],R3             ;Rinicializa o contador
ASS_NIVEL
                MOV     R4,NUM_MAX_TUBOS            ;R4 numero máximos de tu
bos no vetor TUBOS
                MOV     R6,TUBOS                    ;R6 é o endereço da prim
eira posição do vetor TUBOS
                MOV     R3,M[ESPACO_ATUAL]          ;Se o espaço entre o últ
imo obstaculo não for ESPACO_TUBOS
                CMP     R3,ESPACO_TUBOS             ;=> não cria novo obstác
ulo => INIC_OBS
                BR.NZ   INIC_OBS                    ;Caso contrário
                MOV     M[ESPACO_ATUAL],R0          ;Reinicia o contador
                CALL    RANDOMIZER
                MOV     R3,M[NEW_COL]               ;R3 tem a coordenada com
 a linha random
                MOV     R5,M[POS_TUBOS]             ;R5 é o indice do vetor
TUBOS para ser overwrited
                MOV     R7,R6                       ;R7 passa a ser o endere
ço da primeira posição do vetor TUBOS
                ADD     R7,R5                       ;Adiciona o índice que e
stava em R5
                MOV     M[R7],R3                    ;E coloca a nova coorden
ada que estava em R3
                CMP     R5,R4                       ;Se o índice dos vetores
 for igual ao número máximo de tubos do vetor
                BR.Z    COND_R0_POS                 ;=> COND_R0_POS
                INC     M[POS_TUBOS]                ;Caso contrário o índice
 é incrementado
                BR      CICLO_OBS
COND_R0_POS:    MOV     M[POS_TUBOS],R0             ;Onde o índice retorna a
 0
                BR      CICLO_OBS
INIC_OBS:       INC     M[ESPACO_ATUAL]             ;O contador do nr de col
unas que passaram desde que o último obstáculo foi criado
CICLO_OBS:      CMP     R4,R0                       ;R4 agora é visto como u
m contador do CICLO_OBS
                BR.N    FIM_OBS                     ;Se chegou a menor que 0
 acaba o ciclo
                MOV     R7,M[R6]                    ;R7=Conteúdo de uma posi
ção do vetor (começando no primeiro)
                MOV     R5,R7
                AND     R5,GET_COL                  ;R5 é a coluna dessa pos
ição do vetor
                BR.Z    CONDITION                   ;Se for 0 então não é at
ualizada e passa ao próximo
                MOV     M[COL_DESE],R7              ;Caso contrário, apaga a
 coluna
                CALL    APAGA_COL
                DEC     M[R6]                       ;Escreve na nova posição
 (=decremento de uma coluna)
                DEC     M[COL_DESE]
                CALL    DESENHACOL
CONDITION:      DEC     R4
                INC     R6
                BR      CICLO_OBS
```



```
FIM_OBS:        MOV     M[COL_DESE],R0              ;Apaga, incondicionalmen
te a coluna 0 da Janela de Texto
                CALL    APAGA_COL
                CALL    DISTANCIA_TUBOS             ;É atualizada as variáve
is da distância percorrida e do nº de obstáculos
                POP     R7
                POP     R6
                POP     R5
                POP     R4
                POP     R3
                RET

; ==============================================================================*
; DISTANCIA_TUBOS: Rotina que atualiza as variáveis da distância percorrida e o*

;                 nº obstáculos ultrapassados.                                 *
; ==============================================================================*
DISTANCIA_TUBOS:PUSH    R3
                PUSH    R4
                PUSH    R5
                INC     M[DISTANCIA_PER]            ;Sempre que a rotina é c
hamada, é porque percorreu 1 coluna
                MOV     R3,M[DISTANCIA_PER]         ;R3 é a distancia percor
rida
                MOV     R4,DIST_SEM_OBS             ;R4 é a distancia que o
passaro percorre ate atingir o primeiro obstáculo
                MOV     R5,ESPACO_TUBOS             ;R5=espaço entre tubos
                INC     R5                          ;R5=R5+1
                SUB     R3,R4                       ;O número de obstáculos
é calculado pela diferença da distância
                BR.N    FIM_DIST_TUB                ;pela distância inicialm
ente percorrida sem obstáculos.
                DIV     R3,R5                       ;O resultado é dado pelo
 resto da divisão da distância percorrida
                INC     R3                          ;pelo espaço entre tubos
+1, incrementando mais 1
                MOV     M[OBS_ULTRAPASSA],R3        ;Só é atualizado caso a
diferença inicial não seja negativa
FIM_DIST_TUB:   POP     R5
                POP     R3
                POP     R4
                RET

; ==============================================================================*
; ATUALIZA_DISPLAYS: Rotina que atualiza a distância e o nr obstáculos ultra-  *
;                    passados nos displays.                                    *
; ==============================================================================*

ATUALIZA_DISPLAYS:      PUSH    R3
                PUSH    R4
                MOV     R3,M[DISTANCIA_PER]         ;Escreve a distância no
LCD
                MOV     R4,POS_LCD_L1_2
                PUSH    R3
                PUSH    R4
                CALL    EscStringNLCD
                CALL    AtualizaD7S
                POP     R4
                POP     R3
                RET

; ==============================================================================*
```

```
; DESENHACOL: Rotina que cria uma coluna de X com o espaço                    *
;             Recebe o argumento pelo espaço de memória COL_DESE              *
; ============================================================================*
DESENHACOL:     PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     R4,OBS                      ;R4='X'
                MOV     R3,M[COL_DESE]              ;Recebe o argumento (coo
rdenada)
                AND     R3,GET_COL                  ;R3=00|Coluna
                MOV     R5,R3
                ADD     R5,LINHA_22                 ;R5=linha22|Coluna
CICLODES:       ADD     R3,INC_LINHA                ;Escreve linha a linha
                CMP     R3,M[COL_DESE]
                BR.NZ   DESENHA                     ;Quando chega à linha qu
e endereça o espaço
                ADD     R3,ESPACAMENTO              ;Faz skip de 'ESPACAMENT
O' linhas
DESENHA:        MOV     M[CURSOR_JT],R3             ;Desenha as restantes at
é linha 22
                MOV     M[ESCRITA_JT],R4
                CMP     R3,R5
                BR.NZ   CICLODES
                POP     R5
                POP     R4
                POP     R3
                RET

; ============================================================================*

; APAGA_COL: Rotina que apaga uma coluna.                                     *
;            Recebe o argumento pelo espaço de memória COL_DESE               *
; ============================================================================*
APAGA_COL:      PUSH    R3
                PUSH    R4
                PUSH    R5
                MOV     R4,ESPACO                   ;R4=' '
                MOV     R3,M[COL_DESE]              ;Recebe o argumento (coo
rdenada
                AND     R3,GET_COL                  ;R3=00|Coluna
                MOV     R5,R3
                ADD     R5,LINHA_22                 ;R5=linha22|Coluna
CICLOAPA:       ADD     R3,INC_LINHA                ;Apaga linha a linha
                CMP     R3,M[COL_DESE]
                BR.NZ   APAGA                       ;Quando chega à linha qu
e endereça o espaço
                ADD     R3,ESPACAMENTO              ;Faz skip de 'ESPACAMENT
O' linhas
APAGA:          MOV     M[CURSOR_JT],R3             ;Apaga as restantes até
linha 22
                MOV     M[ESCRITA_JT],R4
                CMP     R3,R5
                BR.NZ   CICLOAPA
                POP     R5
                POP     R4
                POP     R3
                RET

; ============================================================================*
; LIMPA_JT: Rotina que limpa todo o conteúdo da Janela de Texto exceto a linha *
;           0 e 23                                                            *
; ============================================================================*
```



```
LIMPA_JT:       PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    R6
                MOV     R4,ESPACO                   ;R4=' '
                MOV     R3,R0                       ;R3=0
                MOV     R5,R0
                ADD     R5, LINHA_22                ;R5=linha22|00
                MOV     R6, COUNT_TRACO             ;R6=79
CICLOAPA_JT:    ADD     R3,INC_LINHA                ;Apaga coluna a coluna
                MOV     M[CURSOR_JT],R3
                MOV     M[ESCRITA_JT],R4
                CMP     R3,R5
                BR.NZ   CICLOAPA_JT
                INC     R5
                INC     R3
                AND     R3,GET_COL
                DEC     R6                          ;Até perfazer as 79 colu
nas
                BR.NZ   CICLOAPA_JT
                POP     R6
                POP     R5
                POP     R4
                POP     R3
                RET

; ============================================================================*
; RANDOMIZER: Rotina que gera uma coordenada sendo gerado um numero aleatorio  *
;             entre 1 e 17 para a linha, a coluna é sempre constante (78)     *
;             Recebe o argumento pelo espaço de memória NUMRANDOM             *
;             Devolve o resultado pelo espaço de memória NEW_COL              *
; ============================================================================*
RANDOMIZER:     PUSH    R3
                PUSH    R4
                PUSH    R6
                MOV     R6,DIVISOR                  ;R6=Nr em que os valores
 aleatórios variam
                MOV     R3,M[NUMRANDOM]             ;R3=Ni
                MOV     R4,R3                       ;R4=Ni
                AND     R3,BIT_MENOR                ;Se o bit de menor peso
for 1 => ELSE
                BR.NZ   ELSE                        ;Se for 0
                ROR     R4,1                        ;Rotate Right
                MOV     M[NUMRANDOM],R4             ;Guarda
                DIV     R4,R6                       ;Divide e guarda o resto
 em R6 => FimRandom
                BR      FIMRANDOM
ELSE:           XOR     R4,RANDOM_MASK              ;Faz um XOR com a máscar
a
                ROR     R4,1                        ;Rotate right
                MOV     M[NUMRANDOM],R4             ;Guarda Valor
                DIV     R4,R6                       ;Divide e guarda o resto
 em R6
FIMRANDOM:      INC     R6                          ;R6=R6+1, não pode ser 0
                SHL     R6,8                        ;Shifta 8 bits para esqu
erda para ficar no octeto de maior peso (linha)
                ADD     R6,COL_78                   ;Soma a coluna 78
                MOV     M[NEW_COL],R6               ;E guarda o valor no esp
aço de memória NEWCOL
                POP     R6
                POP     R4
                POP     R3
```

```
                RET

; ==============================================================================*
; COLISION_V: Rotina que verifica se o pássaro colidiu. Ora com os limites, ora*
;             com os obstáculos                                                *
; COLISÕES: Rotina auxiliar do COLISION_V                                      *
; ==============================================================================*
COLISION_V:     CMP     R2,0114h                        ;Verifica se o passaro b
ateu nos limites
                JMP.N   FIM
                CMP     R2,1614h
                JMP.NN  FIM1
                CALL    COLISOES
                RET

COLISOES:       PUSH    R4
                PUSH    R5
                PUSH    R6
                PUSH    R7
                PUSH    R3
                MOV     R4,M[POINTER_TUBO]              ;Contem o indice do obst
aculo a ser avaliado
                MOV     R6, TUBOS
                ADD     R6,M[POINTER_TUBO]
                MOV     R6,M[R6]
                CMP     R6,R0                           ;Nao faz nada com as col
unas vazias (Primeiras colunas comecam em 0000h)
                JMP.Z   NEXT
                MOV     R7,R6                           ;R7 SOFRE AS ATUALIZACOE
S DO TUBOS
                AND     R7,00FFh
                CMP     R7,NO_NEED                      ;NONEED = 2 COLUNAS DEPO
IS DE PASSAR O PASSARO.
                BR.NZ   CARRYON
                INC     M[POINTER_TUBO]                 ;Passa para o proximo ob
staculo caso ja tenha passado o passaro
                MOV     R3,NUM_MAX_TUBOS                ;Certifica-se que recome
ça na primeira posicao do vetor depois de chegar a ultima
                CMP     M[POINTER_TUBO],R3
                BR.NZ   CARRYON
                MOV     M[POINTER_TUBO],R0
CARRYON:        MOV     R3,R2
                CALL    TRIAL                           ;COLUNAS MATCHING COM O
CORPO
                MOV     R3,R2
                INC     R3
                CALL    TRIAL                           ;COLUNAS MATCHING COM O
BICO
NEXT:           POP     R3
                POP     R7
                POP     R6
                POP     R5
                POP     R4
                RET
TRIAL:          PUSH    R4
                MOV     R4,R3
                MOV     R5,R6
                AND     R5,00FFh
                AND     R4,00FFh
                CMP     R5,R4                           ;Verifica se posição do
bico/corpo tem a mesma coluna do obstaculo a ser avaliado
                BR.NZ   TERMINA
```



```
                MOV     R5,R6                           ;Coloca em R5 as coorden
adas do obstaculo
                CMP     R5,R3                           ;Posição do inicio do es
paço no obstaculo (menos) posição do bico/corpo se for positivo, bateu
                JMP.P   FIM
                ADD     R5,ESPACAMENTO
                CMP     R5,R3                           ;Posição do fim do espaç
o no obstaculo (menos) posição do bico/corpo se nao for positivo, bateu
                JMP.NP  FIM
TERMINA:        POP     R4
                RET


;########################################
;#           PROGRAMA PRINCIPAL         #
;#----------------------------------------#
;# INICIO                               #
;# INICIO_JOGO                          #
;# JOGO                                 #
;# FIM                                  #
;########################################
; ==============================================================================*
; INICIO: Execução principal do programa, inicializa interrupções e portos de  *

;         controlo                                                             *
; ==============================================================================*
INICIO:         MOV     R7,SP_INICIAL
                MOV     SP,R7
                MOV     R7,INT_MASK
                MOV     M[INT_MASK_ADDR],R7     ;Inicializa as interrupções
                MOV     R7,CONTROLO
                MOV     M[CURSOR_JT],R7         ;Inicializa o porto de posiciona
mento
                MOV     M[TEMPO_GRAVIDADE],R0
                MOV     R7,DIF_TEMPO
                MOV     M[DIF_TEMPO_M],R7
                MOV     R7,EST_RELOG            ;Inicializa o temporizador
                MOV     M[EST_RELOG_M],R7
                ENI

; ==============================================================================*
; INICIO_JOGO: Rotina que cria o espaco de jogo e chama os menus.              *
; JOGO: Rotina onde chama as rotinas essencias. Joga o jogo propriamente       *
; FIM: Rotina no qual se escreve o Score e as mensagens de fim de jogo         *
; ==============================================================================*
INICIO_JOGO:    CALL    RESET_VARS
                MOV     M[COORD_TRACO],R0
                CALL    LINHA
                MOV     R3,1700h                ;1700h corresponde a coordenada
linha 23, coluna 0
                MOV     M[COORD_TRACO],R3       ;Atualiza COORD_TRACO com a coor
denada da linha 23, coluna 0
                CALL    LINHA
                MOV     R3,FIM_STR              ;Coloca @ no último espaço de me
mória de SCORE (ser usado como string)
                MOV     R4,SCORE
                MOV     M[R4+4],R3
                CALL    ATUALIZA_DISPLAYS       ;É colocada a zeros a distância
no LCD e o nº de Obstáculos nos Displays
                CALL    MENU1
                MOV     M[I1],R0
                CALL    MENU2
```

```
                MOV     M[I1],R0
                MOV     R1,POS_PASSARO          ;R1=primeira posição do pássaro
                MOV     R2,POS_PASSARO          ;R2=primeira posição do pássaro
                CALL    PASSARO                 ;Escreve o passaro
                MOV     M[Y0],R2                ;Y0 é iniciado com a primeira po
sição do pássaro
                MOV     M[CONTADOR_TEMPO],R0    ;Reiniciar algumas variáveis par
a evitar erros
                MOV     M[SALTOU],R0
                MOV     R3,M[NIVEL]
                MOV     M[ASS_NIVEL],R3         ;Coloca o nivel no contador assi
stente de nível

JOGO:           CALL    ATUALIZA_DISPLAYS
                CMP     M[ASS_NIVEL],R0
                CALL.Z  OBSTACULOS              ;OBSTÁCULOS é chamada se M[ASS_N
IVEL]=0
                CMP     M[SALTOU],R0
                CALL.NZ SALTO                   ;SALTO é chamada se M[SALTOU]!=0
                CMP     M[SALTOU],R0
                CALL.Z  GRAVIDADE               ;GRAVIDADE é chamada se M[SALTOU
]=0
                CALL    COLISION_V
                CMP     R1,R2
                BR.Z    JOGO                    ;Se R1=R2 não chama o PASSARO
                CALL    PASSARO
                BR      JOGO
FIM1:           MOV     R2,1614h                ;Vem do COLISION_V
                CALL    PASSARO
FIM:            CALL    LIMPA_JT                ;Limpa a Janela de Texto
                MOV     R3,TextoFinal           ;Escreve a pontuação e as mensag
ens finais.
                MOV     R4,POS_TEXTO_L1
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoFinal2
                MOV     R4,POS_PONTUACAO
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,SCORE
                MOV     R4,POS_SCORE
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
                MOV     R3,TextoFinal3
                MOV     R4,POS_TEXTO_L2
                PUSH    R3
                PUSH    R4
                CALL    ESCREVESTR
CICLO_FIM:      CMP     M[I1],R0                ;Se I1 não for ativo, continua n
o ciclo
                BR.Z    CICLO_FIM
                CALL    LIMPA_JT                ;Caso contrário Limpa Janela de
Texto e volta para o inicio
                JMP     INICIO
```