```
;#################################################################################
#########
;

 #
;                                                                              PASSARO
BAMBOLEANTE
 #
;                                                                              Projeto
 IAC 15/16
 #
;                              Miguel Neto, 83529 | Miguel Regouga, 83530 | Ter
esa Sousa, 83565               #
;                                                        Turno pratico: sextas, 0
8h00, LAB10                                                              #
;

 #
;#################################################################################
#########


;#################################################################################
#########
;

 #
;                                                                              Tabela d
e Constantes                                                                 #
;

 #
;#################################################################################
#########

SP_INICIAL      EQU     FDFFh
INT_MASK_ADDR   EQU     FFFAh
INT_MASK        EQU     0000000000000011b               ; Interrupcoes I0 e I1 e
stao ativadas
INT_MASK_INI    EQU     0000000000000010b               ; Interrupcao I1 ativada
,

        ; para inicializar o jogo
INT_MASK_JOGO   EQU     1000000000000111b

RANDOM_MASK             EQU             8016h
; 1000000000010110b (enunciado)
IO_DISPLAY      EQU     FFF0h
DELAY_COUNT     EQU     0200h
NIBBLE_MASK     EQU     000fh
NUM_NIBBLES     EQU     4
BITS_PER_NIBBLE EQU     4
JANELA_TEXTO_E  EQU     FFFEh
JANELA_TEXTO_C  EQU     FFFCh
TRACINHO        EQU     '-'
SPACE           EQU     ' '
FIM_TEXTO               EQU     '@'
OBSTACULOS              EQU             'X'
LIMITE_TOPO     EQU     0100h
POS_OBST_MAX    EQU             15
```



```
; Posicoes necessarias para as mensagens ficarem centradas

POS_MSG_INI1    EQU             0C23h
POS_MSG_INI2    EQU             0E1Dh

POS_MSG_FIM             EQU             0C23h



TIMER_COUNT             EQU     FFF6h
TIMER_CONTROL   EQU     FFF7h


;#################################################################################
#########
;

 #
;                                                                              Tabela d
e Variaveis
 #
;

 #
;#################################################################################
#########


                ORIG    8000h
Passaro         STR     '0>', FIM_TEXTO
PassaroLimpa    STR     '  ', FIM_TEXTO
MSG_INIC1               STR     'Prepare-se', FIM_TEXTO
; Comprimento: 10 letras
MSG_INIC2               STR     'Prima o interruptor I1', FIM_TEXTO     ; Compri
mento: 22 letras
MSG_FIM                 STR     'Fim do Jogo', FIM_TEXTO
; Comprimento: 11 letras
NumeroAleatorio STR             0
FlagEspera              WORD    0
FlagSobe                WORD    0
PosPassaro              WORD    0C14h   ; Posicao inicial do passaro
ContQueda               WORD    0002h   ; Intervalo de tempo em que vai ocorrer
a proxima queda
ContMoveObs     WORD    0001h   ; Intervalo de tempo para mover os obstaculos
ContCriaObs     WORD    0006h
Obs                             TAB     14              ; So e possivel ter no m
aximo 14 obstaculos ao mesmo tempo
PonteiroObs             WORD    Obs     ; Indica onde os obstaculos serao coloca
dos na tabela
EspacoObs               WORD    0400h   ; Espaco entre os limites dos obstaculos
NumObs                  WORD    0
RandNum                 WORD    0000h   ; Numero aleatorio
ObstPassados    WORD    0000h
VelocidadeIni   WORD    0
Gravidade               WORD    0100h


;#################################################################################
#########
;
```

```
 #
;                                                                       Tabela d
e Interrupcoes                                                           #
;

 #
;#########################################################################
##########

                ORIG    FE00h
INT0            WORD    sobe_passaro
INT1            WORD    SaiEspera
                        ORIG    FE0Fh
INT15                   WORD    Timer


;#########################################################################
##########
;

 #
;                                                                       CÃ³digo
inicial
 #
;

 #
;#########################################################################
##########

                        ORIG    0000h
                        MOV     R7, SP_INICIAL
                MOV     SP, R7
                        JMP             Inicio


;#########################################################################
##########
;

 #
;                                                                       Temporiz
ador
 #
;

 #
;#########################################################################
##########

Timer:                  PUSH    R1
                        DEC     M[ContQueda]
                        DEC     M[ContMoveObs]
                        MOV     R1, 1
                        MOV     M[TIMER_COUNT], R1
                        MOV     M[TIMER_CONTROL], R1            ; Liga o
 temporizador
                        POP     R1
                        RTI


;#########################################################################
```



```
##########
;

 #
;                       Desenha_linha: Rotina que desenha uma linha dos limites
do jogo                         #
;

 #
;#########################################################################
##########

Desenha_linha:  PUSH    R1
                PUSH    R2
                MOV     R1, TRACINHO
                MOV     R2, 80                                  ; Corres
pondente as 80 colunas
Ciclo:          MOV     M[JANELA_TEXTO_E], R1
                INC     R6
; O cursor anda uma posicao para a

; frente para se desenhar 1 novo traco
                MOV     M[JANELA_TEXTO_C], R6
                DEC     R2
; O numero de tracos a

; desenhar decrementa
                BR.NZ   Ciclo                                   ; Se o n
umero de tracos nao for zero,

; o ciclo e repetido
                POP     R2
; Se for zero, nao se escreve mais
                POP     R1
                RET


;#########################################################################
##########
;

 #
;                                               SaiEspera: Rotina que sai do cic
lo espera                                               #
;

 #
;#########################################################################
##########

SaiEspera:      INC             M[FlagEspera]
                RTI


;#########################################################################
##########
;

 #
;                                       Limites: Rotina que desenha os limites d
o jogo                                          #
;
```

```
 #
;#########################################################################
##########

Limites:        DSI
                MOV     R6, R0                                          ; Desenh
a as linhas no topo da janela

; de texto
                MOV     M[JANELA_TEXTO_C], R6           ; Atribui ao cursor essa
 posicao
                CALL    Desenha_linha
                MOV     R6, 1700h                                       ; 17hex
= 23dec, correspondente ao

; fim da janela de texto
                MOV     M[JANELA_TEXTO_C], R6           ; Atribui ao cursor essa
 posicao
                CALL    Desenha_linha
                ENI
                RET


;#########################################################################
##########
;

 #
;                       Msg_BoasVindas: Rotina que desenha a mensagem de inicio
de jogo                         #
;

 #
;#########################################################################
##########

Msg_BoasVindas: PUSH    R1
                        PUSH    R2
                        MOV     R1, MSG_INIC1
; R1 tem a mensagem a ser escrita
                        MOV     R2, POS_MSG_INI1
; R2 tem a posicao de onde de escrita

                ; para o texto ficar centrado
                        CALL    EscString
                        MOV     R1, MSG_INIC2
                        MOV     R2, POS_MSG_INI2
                        CALL    EscString
                        POP     R2
                        POP     R1
                        RET


;#########################################################################
##########
;

 #
;                       PrintPassaro: Rotina que desenha o passaro na janela de
texto                           #
;
```



```
 #
;#########################################################################
##########

PrintPassaro:   PUSH    R1
                        PUSH    R2
                        MOV     R1, Passaro
; R1 tem a forma do passaro
                        MOV     R2, M[PosPassaro]
; R2 tem a posicao inicial do passaro
                        CALL    EscString
; Escreve o passaro
                        POP     R2
                        POP     R1
                        RET



;#########################################################################
##########
;

 #
;                       Esc_String: Rotina que escreve qualquer string na janela
 de texto                       #
;

 #
;#########################################################################
##########

;               Tem como entrada R1 (Endereco de memoria da string) e R2 (posica
o de escrita)

EscString:              PUSH    R3
CicloEscrita:   MOV     R3, M[R1]                                       ; M[R1]
Ã© onde vai estar

                ; a mensagem a ser escrita
                        CMP     R3, FIM_TEXTO
; Compara com o '@'
                        BR.Z    FimEscrita
; Se for igual, para de escrever
                        MOV     M[JANELA_TEXTO_C], R2           ; Caso c
ontrario, posiciona o cursor
                        MOV     M[JANELA_TEXTO_E], R3           ; Escrev
e o caracter/letra
                        INC     R1
        ; Passa para o caracter seguinte
                        INC     R2
        ; O cursor avanca uma posicao para a

                ; frente para se poder escrever a

                ; proxima letra
                        BR              CicloEscrita
FimEscrita:     POP     R3
                        RET


;#########################################################################
```

```
##########
;

 #
;               LimpaEcra: Rotina que apaga todo o conteudo presente na janela d
e texto                  #
;

 #
;################################################################################
##########


LimpaEcra:              PUSH    R1
                        PUSH    R2
                        PUSH    R3
                        MOV     R1, 0000h
; R1 corresponde a 1a coluna
                        MOV     R2, 174Fh
; R2 corresponde a ultima coluna
CicloLimpa:     MOV     M[JANELA_TEXTO_C], R1           ; Coloca o cursor na 1a
coluna
                        MOV     R3, SPACE
; Coloca o caracter correspondente

                ; ao espaco
                        MOV     M[JANELA_TEXTO_E], R3           ; Escrev
e o espaco no ecra
                        INC     R1
        ; Passa para a proxima coluna
                        CMP     R1, R2
; Compara a posicao das colunas
                        BR.NZ   CicloLimpa
; Se nao for zero, volta ao ciclo
                        POP     R3
        ; Caso contrario, nao escreve mais
                        POP     R2
                        POP     R1
                        RET


;################################################################################
##########
;

 #
;                                       SobePassaro: Rotina que faz o pa
ssaro subir                                     #
;

 #
;################################################################################
##########

sobe_passaro:   INC     M[FlagSobe]                             ; Muda a
 posicao do corpo do passaro

                ; para a linha acima
                        RTI

SobePassaro:    PUSH    R1
                        DSI
```



```
                        DEC     M[FlagSobe]
; Para o passaro so subir uma posicao
                        MOV     R1, 0004h
; R1 tem guardado o tempo que ele sobe
                        MOV     M[ContQueda], R1
                        MOV     R1, PassaroLimpa
; Limpa a posicao onde o pasaro estava
                        MOV     R2, M[PosPassaro]
; Coloca o passaro numa nova posicao
                        CALL    EscString
; Escreve um espaco na pos anterior
                        MOV     R1, 0200h

                        SUB     M[PosPassaro], R1
                        MOV     R1, Passaro
; Coloca o passaro no registo
                        MOV     R2, M[PosPassaro]
; Coloca o passaro numa nova posicao
                        CALL    EscString
                        MOV             M[VelocidadeIni], R0
                        POP     R1
                        ENI
                        RET


;################################################################################
##########
;

 #
;                               Toca_limite: Rotina que ve se o passaro toca nos
 limites                                #
;

 #
;################################################################################
##########

Toca_limite:    PUSH    R1
                        MOV     R1, M[PosPassaro]
; R1 tem a posicao do passaro
                        CMP     R1, 0014h
; Verificacao se a posicao do passaro

                ; corresponde ao limite do topo
                        CALL.Z  FimJogo
; Se sim, o jogo termina
                        MOV     R1, M[PosPassaro]
                        CMP     R1, 1714h
; Verificacao se a posicao do passaro

                ; corresponde ao limite de baixo
                        CALL.Z  FimJogo
; Se sim, o jogo termina
                        POP             R1
                        RET

;################################################################################
##########
;

 #
```

```
;                       FimJogo: Rotina que limpa o ecra e mostra a mensagem de
fim de jogo                     #
;

 #
;################################################################################
##########

FimJogo:        CALL    LimpaEcra                                       ; Chama
a funcao que limpa o fim
                CALL    Msg_fim
; Chama a funcao que escreve a

                ; mensagem de fim do jogo
                JMP     Fim


;################################################################################
##########
;

 #
;                                       Msg_fim: Rotina que escreve a mensagem d
e fim de jogo                           #
;

 #
;################################################################################
##########

Msg_fim:        PUSH    R1
                MOV             R1, MSG_FIM
        ; R1 tem a mensagem a ser escrita
                MOV     R2, POS_MSG_FIM
; R2 tem a posicao de onde de escrita

                ; para o texto ficar centrado
                CALL    EscString
                POP             R1
                RET


;################################################################################
##########
;

 #
;                       PassaroCai: Rotina que faz o passaro cair, utilizando o
temporizador             #
;

 #
;################################################################################
##########

PassaroCai:     PUSH    R1
                PUSH    R7
                DSI
                MOV     R1, 0002h
; Tempo que o passaro demora a cair
                MOV     M[ContQueda], R1
```



```
                MOV     R1, PassaroLimpa
; Limpa a posicao anterior do passaro
                MOV     R2, M[PosPassaro]
; Coloca o passaro numa nova posicao
                CALL    EscString
; Escreve um espaco na pos anterior
                MOV     R1, 0100h
; Desce uma linha
                ADD     M[PosPassaro], R1
                MOV     R1, Passaro
; Coloca o passaro no registo
                MOV     R2, M[PosPassaro]
; Coloca o passaro numa nova posicao
                CALL    EscString
                MOV             R7, Gravidade
                ADD             M[VelocidadeIni], R7
                MOV             R7, M[VelocidadeIni]
                MOV             R1, R7
                POP             R7
                POP     R1
                ENI
                RET


;################################################################################
##########
;

 #
;                                                               CriaObs: Rotina que cria
 os obstaculos                                                   #
;

 #
;################################################################################
##########

CriaObs:                PUSH    R1
                        PUSH    R2

                        MOV     R1, 0006h
; Criacao de um novo obstaculo
                        MOV     M[ContCriaObs], R1
                        MOV     R1, M[EspacoObs]
                        ADD     R1, 004Fh
; Passa para a coluna anterior
                        MOV     R2, M[PonteiroObs]
; R2 diz onde por os valores na tabela
                        MOV     M[R2], R1
; R1 e um valor que e posto na tabela

                ; M[R2] e uma entrada da tabela
                        INC     M[PonteiroObs]

                        MOV     R1, Obs
; Coloca esse novo valor na tabela
                        ADD     R1, 000Dh
; D = 13 = ultimo valor da tabela
                        CMP     M[PonteiroObs], R1
; Compara chegou ao fim da tabela
                        CALL.Z  ResetPonteiro
; Se sim, volta-se a por o ponteiro
```

```
                ; no inicio da tabela
                                CALL    AddObs
; Caso contrario, volta a escrever

                ; um novo valor na tabela

                                POP     R2
                                POP     R1
                                RET


;###############################################################################
##########
;

 #
;                       ResetPonteiro: rotina que coloca o ponteiro no inicio da
 tabela                          #
;

 #
;###############################################################################
##########

ResetPonteiro:  PUSH    R1
                                MOV     R1, Obs
; Define a nova posicao da tabela
                                MOV     M[PonteiroObs], R1
; Poe um novo valor nessa

                ; posicao da tabela
                                POP     R1
                                RET

AddObs:                 PUSH    R1

                                MOV     R1, 14
; No maximo podem estar 14 obstaculos
                                CMP     M[NumObs], R1
; Ve se o numero de obstaculos = 14
                                BR.Z    NaoAdd
; Se ja houverem 14 obstaculos, ja nao

                ; adiciona
                                INC     M[NumObs]
; Caso contrario, incremanta-se o

                ; numero de obstaculos
NaoAdd:                 POP     R1
                                RET

;###############################################################################
##########
;

 #                                      MoveObs: rotina que movimenta os obstacu
;                                                               #
los
;

 #
```



```
;###############################################################################
##########

MoveObs:                PUSH    R1
                                PUSH    R2
                                PUSH    R3
                                PUSH    R4
                                DEC     M[ContCriaObs]
                                MOV     R1, 0004h
; Tempo de movimento dos obstaculos
                                MOV     M[ContMoveObs], R1
                                MOV     R1, Obs
; R1 tem a tabela onde sao colocados

                ; os ostaculos
                                MOV     R3, R1

                                ADD     R3, M[NumObs]

cicloMove:              MOV     R2, M[R1]

                                CALL    limpaObs

                                MOV     R4, R2
                                AND     R4, 00FFh
                                CMP     R4, R0
                                BR.Z    NaoMove
                                DEC     R2

                                MOV     M[R1], R2

                                CALL    printObs
                                INC     R1
                                CMP     R1, R3
                                BR.NZ   cicloMove

NaoMove:                POP     R4
                                POP     R3
                                POP     R2
                                POP     R1
                                RET


;###############################################################################
##########
;

 #
;                                                       LimpaObs: rotina que limpa os ob
staculos                                                        #
;

 #
;###############################################################################
##########

limpaObs:               PUSH    R1
                                PUSH    R2
                                PUSH    R3
                                MOV     R3, 0100h
; Primeira linha depois do
```

```
                ; limite de cima
                                MOV     R4, 1700h
; Primeira linha antes do

                ; limite de cima
                                MOV     R1, R2

                                AND     R1, 00FFh
; 004Fh (conta os bits mais

                ; significativos)
                                ADD     R3, R1
; 014Fh (primeira linha depois dos

                ; limites e ultima coluna)
                                ADD     R4, R1
; 164Fh (ultima linha antes dos

                ; limites e ultima coluna)
                                MOV     R1, SPACE

ciclolimpaObs:  MOV     M[JANELA_TEXTO_C], R3           ; Coloca-se o cursor na
primeira

                ; linha depois do limite de cima
                                MOV     M[JANELA_TEXTO_E], R1           ; Escrev
e o 'X'
                                ADD     R3, 0100h
; Passa para a linha abaixo
                                CMP     R3, R4
; Ve se R3 ja chegou ao

                ; limite de baixo
                                BR.NZ   ciclolimpaObs
; Se nao, volta a desenhar

                ; os obstaculos
                                BR              FimlimpaObs
                                ;
FimlimpaObs:    POP     R4
                                POP     R3
                                POP     R1
                                RET
                                POP             R1
                                RET



;###############################################################################
##########
;

 #
;                                              PrintObs: rotina que escreve os
obstaculos                                              #
;

 #
;###############################################################################
##########

printObs:               PUSH    R1
                                PUSH    R3
```



```
                                PUSH    R4
                                ;CALL   Aleatorio
                                MOV     R3, 0100h
; Primeira linha depois do limite

                ; de cima
                                MOV     R4, 1700h
; Primeira linha antes do limite

                ; de cima
                                MOV     R1, R2
; Coloca-se o sitio onde o obstaculo

                ; vai ser escrito (054Fh)
                                AND     R1, 00FFh
; 004Fh (conta os bits

                ; mais significativos)
                                ADD     R3, R1
; 014Fh (primeira linha depois dos

                ; limites e ultima coluna)
                                ADD     R4, R1
; 164Fh (ultima linha antes dos

                ; limites e ultima coluna)
                                MOV     R1, OBSTACULOS
; Coloca-se o 'X'
cicloprintObs:  MOV     M[JANELA_TEXTO_C], R3           ; Poe o cursor na primei
ra linha

                ; depois do limite de cima
                                MOV     M[JANELA_TEXTO_E], R1           ; Escrev
e o 'X'
                                ADD     R3, 0100h
; Passa para a linha abaixo
                                CMP     R3, R2
; Ve se o sitio do obstaculo Ã© um dos

                ; 5 espacos que tem de haver entre

                ; os obstaculos
                                BR.Z    criaEspaco
; Se for, vai criar um espaco
                                CMP     R3, R4
; Ve se os obs estao na mesma linha
                                BR.NZ   cicloprintObs
; Se nao, volta a desenhar

                ; os obstaculos
                                BR              fimprintObs

criaEspaco:     ADD     R3, 0500h                                       ; O espa
co entre os obstaculos Ã© de

                ; 5 linhas (os primeiros dois numeros

                ; corresponde as linhas)
                                BR              cicloprintObs
; Ao acabar essas 5 linhas, ele
```

```
                ; volta a escrever os obstaculos

                ; em baixo
fimprintObs:    POP     R4
                        POP     R3
                        POP     R1
                        RET


;###############################################################################
##########
;

 #
;               Colisoes: rotina que testa se o passaro colide com os ob
staculos                        #
;

 #
;###############################################################################
##########

Colisoes:               PUSH    R1
                        PUSH    R2
                        PUSH    R3
                        PUSH    R4
                        MOV     R1, Obs

                        MOV     R3, R1
                        ADD     R3, 000Dh
; D = 13 = ultimo valor da tabela
cicloclisoes:   MOV     R2, M[R1]                                       ; Coloca
 em R2 o
                        AND     R2, 00FFh
; Conta os bits mais significativos
                        CMP     R2, 0014h

                        BR.Z    TestaLinha
                        INC     R1
                        CMP     R1, R3
                        BR.NZ   cicloclisoes
                        BR              fimColisoes
TestaLinha:     MOV     R2, M[R1]
                        MOV     R4, M[PosPassaro]
                        AND     R2, FF00h
; Seleciona os bits menos significativos
                        AND     R4, FF00h
                        CMP     R2, R4
                        CALL.P  FimJogo
                        ADD     R2, 0500h
                        CMP     R2, R4
                        CALL.N  FimJogo
fimColisoes:    POP     R4
; Se nao for zero, volta a escrever

                ; um novo valor na tabela
                        POP     R3
                        POP     R2
                        POP     R1
                        RET

;###############################################################################
```



```
##########
;

 #
;                                                                       Codigo p
rincipal
 #
;

 #
;###############################################################################
##########

;                                                       R6 tem guardada a posica
o do cursor
;                                                       R5 tem guardada a posica
o do passaro


Inicio:         MOV     R7, INT_MASK_INI
                MOV     M[INT_MASK_ADDR], R7                    ; Inicializa a i
nterrupcao I1
                MOV     R7, FFFFh
                MOV     M[JANELA_TEXTO_C], R7                   ; Inicializa o c
ursor da

        ; janela de texto
                                CALL    Msg_BoasVindas
; Escreve as mensagens de

                        ; boas vindas
                                CALL    CriaObs
                                ENI


Espera:         INC     M[RandNum]
                                CMP     M[FlagEspera], R0
; Tempo de espera ate se

                        ; carregar no interruptor
                BR.Z    Espera
; Enquanto nao se pressiona

        ; I1, volta ao ciclo
                                DSI
                                CALL    LimpaEcra
                CALL    Limites
                CALL    PrintPassaro
                                CALL    CriaObs
                                MOV     R7, INT_MASK_JOGO
                MOV     M[INT_MASK_ADDR], R7
                                MOV     R7, 1
        ; Inicializa o interruptor
                                MOV     M[TIMER_COUNT], R7
; Intervalo de tempo para a

                        ; proxima interrupcao do

                        ; temporizador
                                MOV     M[TIMER_CONTROL], R7
; Liga o temporizador
                                ENI
```

```
CicloJogo:      CMP     M[FlagSobe], R0
                        CALL.NZ SobePassaro
; Se nao for zero, chama a funcao

                        ; que sobe o passaro
                        CMP     M[ContQueda], R0
                        CALL.Z  PassaroCai
        ; Se nao for zero, chama a funcao

                        ; que faz com o pasaro caia
                        CMP     M[ContMoveObs], R0
                        CALL.Z  MoveObs
                        CMP     M[ContCriaObs], R0
                        CALL.Z  CriaObs
                        CALL    Toca_limite
                        CALL    Colisoes
                        BR      CicloJogo


Fim:            BR      Fim
```