```
;###############################################
;# Projeto de Arqutitectura de Computadores     #
;#                       LEIC-A IST 2017
#
;#
        #
;#                                MASTERMIND
#
;#
        #
;#               Francisco Figueiredo 89443                                  #
;#               Joao Marques 89473
#
;#
        #
;###############################################


;###############VARIAVEIS, CONSTANTES############

SP_INICIAL EQU FDFFh
LCD_WRITE EQU FFF5h
LCD_CONTROL EQU FFF4h
IO_DISPLAY EQU FFF0h
IO_DISPLAY1 EQU FFF1h
IO_WRITE EQU FFFEh
IO_SW EQU FFF9h
IO_LEDS EQU FFF8h
IO_CURSOR       EQU     FFFCh
TIMER_CTRL                      EQU     FFF7h
TIMER_DELAY                     EQU             FFF6h
IO EQU FFFEh
Mascara EQU 1000000000010111b

 ORG 8000h
CS TAB 1                                                ; Vareavel que vai guard
ar o Conjunto Solucao/ Sequencia Aleatoria
Jogada  TAB 1                                   ; Vareavel que vai guardar o Jog
ada Inserida pelo jogador
Semelhanca TAB 1                                ; Vareavel que vai guardar a Sem
elhanca entre as vareaveis acima
BotaoPremido TAB 1                              ; Vareavel que vai guardar o Bot
ao Botao Premido
GUARDAR_CURSOR TAB 1                    ; Vareavel que vai guardar a posicao do
cursor
Converter12Bits TAB 1                   ; Vareavel que vai ajudar na traducao pa
ra 12 bits de cada sequencia
Jogada12Bits    TAB 1                   ; Vareavel que vai guardar a Jogada inse
rida pelo jogador em 12 bits
CS12Bits TAB 1                                  ; Vareavel que vai guardar a Seq
uencia Aleatoria em 12 bits
TempoDeJogada TAB 1                             ; Vareavel que vai guardar o est
ado dos Leds
MelhorPontuacao TAB 1                   ; Vareavel que vai guardar a melhor Pont
uacao
FinalizarOJogo  TAB 1                   ; Vareavel que vai ajudar a Finalizar O
Jogo
PontuacaoAtual TAB 1                    ; Vareacel que vai guardar a jogada atua
l
NumeroDigitosInseridos WORD 4   ; Vareavel que vai guardar o numero de digitos i
nseridos
```



```
TimerTick       WORD 0
Nticks          WORD 1
VarAleatoria WORD 0

Frase   STR     'Jogada:@'
Fim_Frase       EQU     '@'
IA_Comecar      STR 'Carregue no botao IA para iniciar@'
IA_Recomecar    STR 'Carregue no botao IA para recomecar@'
FimDoJogo       STR ' Fim do Jogo.@'

;----------MASCARA DE INTERRUPCOES---------------

INT_MASK_ADDR EQU FFFAh
INT_MASK EQU 1000010001111110b

;------------TABELA DE INTERRUPCOES--------------
        ORG FE01h
INT1 WORD INT1F
INT2 WORD INT2F
INT3 WORD INT3F
INT4 WORD INT4F
INT5 WORD INT5F
INT6 WORD INT6F

        ORG FE0Ah
INTA WORD INTAF

        ORG FE0Fh
intF WORD Timer

;################### FUNCOES ###################


        ORG FE0Ah
INTA WORD INTAF

        ORG FE0Fh
intF WORD Timer

                                        ORG 0000h
                                                MOV R4, FFFFh
                                        MOV M[IO_CURSOR], R4
                                        MOV M[GUARDAR_CURSOR], R4
                                                MOV R5, 20
                                                MOV M[80f7h], R5


SegundoCicloJogada:             MOV R4, M[GUARDAR_CURSOR]
                                                MOV R5, 1
                                                MOV M[80f8h], R5
                                                MOV M[80f6h], R5
                                                MOV R5, 20
                                                CMP M[80f7h], R5
                                                BR.Z Comecar_IA

                                                MOV R5, FimDoJogo
FraseFim:                                       PUSH R4
                                                MOV R4, M[GUARDAR_CURSOR]
                                                MOV M[IO_CURSOR], R4
                                        INC R4
                                        MOV M[GUARDAR_CURSOR], R4
                                        MOV M[IO_CURSOR], R4
```

```
                                        POP R4
                                                MOV R6, M[R5]
                                                CMP R6, Fim_Frase
                                                JMP.Z Recomecar_IA
                                                MOV M[IO], R6
                                                INC R5
                                                BR FraseFim

Comecar_IA:                                     MOV R5, IA_Comecar
FraseInicio:                    PUSH R4
                                                MOV R4, M[GUARDAR_CURSOR]
                                                MOV M[IO_CURSOR], R4
                                        INC R4
                                        MOV M[GUARDAR_CURSOR], R4
                                        MOV M[IO_CURSOR], R4
                                        POP R4
                                                MOV R6, M[R5]
                                                CMP R6, Fim_Frase
                                                JMP.Z Inicio
                                                MOV M[IO], R6
                                                INC R5
                                                BR FraseInicio

Recomecar_IA:                   MOV R5, IA_Recomecar
FraseInicio2:                   PUSH R4
                                                MOV R4, M[GUARDAR_CURSOR]
                                                MOV M[IO_CURSOR], R4
                                        INC R4
                                        MOV M[GUARDAR_CURSOR], R4
                                        MOV M[IO_CURSOR], R4
                                        POP R4
                                                MOV R6, M[R5]
                                                CMP R6, Fim_Frase
                                                JMP.Z quase
                                                MOV M[IO], R6
                                                INC R5
                                                BR FraseInicio2



INTAF:  MOV R4, R0
                MOV M[IO_CURSOR], R4
                MOV M[IO_DISPLAY], R0
                MOV M[IO_DISPLAY1], R0
                MOV R1, M[VarAleatoria]
                MOV M[CS], R1
                RTI

INT1F:  PUSH R2
                MOV R2, 1
                MOV M[BotaoPremido], R2
                POP R2
                RTI

INT2F:  PUSH R2
                MOV R2, 2
                MOV M[BotaoPremido], R2
                POP R2
                RTI

INT3F:  PUSH R2
                MOV R2, 3
                MOV M[BotaoPremido], R2
```



```
                POP R2
                RTI

INT4F:  PUSH R2
                MOV R2, 4
                MOV M[BotaoPremido], R2
                POP R2
                RTI

INT5F:  PUSH R2
                MOV R2, 5
                MOV M[BotaoPremido], R2
                POP R2
                RTI

INT6F:  PUSH R2
                MOV R2, 6
                MOV M[BotaoPremido], R2
                POP R2
                RTI

Timer:  INC M[TimerTick]
                MOV  R1, 5
                MOV M[TIMER_DELAY], R1
                MOV R1, 1
                MOV M[TIMER_CTRL], R1
                RTI

Inicio1: MOV R6, R2 ; Mover para a pilha cada algarismo individual de ambas as s
equencias.
                AND R6, 000fh
                SHR R2, 4
                CMP R6, R0
                JMP.Z ReporRegistos
                MOV R4, SP
                DEC R7
                ADD R4, R7
                MOV M[R4], R6
                MOV R5, R1
                AND R5, 000fh
                SHR R1, 4
                MOV R4, SP
                INC R3
                ADD R4, R3
                MOV M[R4], R5
                Br Inicio1

GerarAleatorio: MOV R5, M[SP + 2] ; Mover para R5 o valor inicial antes guardado
 na pilha
                                        AND R5, 1h ; Isola o ultimo bit do valor anterio
r
                                        CMP R5, R0
                                        BR.NZ Salta ; Se o ultimo bit é 1, salta
                                        MOV R6, M[SP + 2]
                                        SHR R6, 1 ; gera um novo valor
                                        MOV M[SP + 2], R6
                                        BR VerificarValor


Salta:  MOV R5, M[SP + 3] ; fazer XOR enre o Valor inicial e a Mascara
                MOV R6, M[SP + 2]
                XOR R6, R5
```

```
                MOV M[SP + 2], R6
                BR VerificarValor

VerificarValor: AND R6, 000fh ; Esta retina VerificarValor serve para ver se cad
a algarismo da Sequencia Secreta está entre 1 e 6. Não estando, geraria outro
                                CMP R6, R0
                                BR.Z GerarAleatorio
                                CMP R6, 0006h
                                BR.P GerarAleatorio
                                MOV R6, M[SP + 2]
                                AND R6, 00f0h
                                CMP R6, R0
                                JMP.Z GerarAleatorio
                                CMP R6, 0060h
                                JMP.P GerarAleatorio
                                MOV R6, M[SP + 2]
                                AND R6, 0f00h
                                CMP R6, R0
                                JMP.Z GerarAleatorio
                                CMP R6, 0600h
                                JMP.P GerarAleatorio
                                MOV R6, M[SP + 2]
                                AND R6, f000h
                                CMP R6, R0
                                JMP.Z GerarAleatorio
                                CMP R6, 6000h
                                JMP.P GerarAleatorio
                                RET



ReporRegistos:  MOV R1, R0
                                MOV R2, R0
                                MOV R3, R0
                                MOV R4, R0
                                MOV R5, R0
                                MOV R6, R0
                                MOV R7, R0
                                RET

EscreverX:              MOV R1, SP ; compara o primeiro/ segundo/ terceiro/ quar
to digito de cada sequencia
                                DEC R7
                                ADD R1, R7
                                MOV R6, M[R1]
                                MOV R4, SP
                                INC R3
                                ADD R4, R3
                                MOV R5, M[R4]
                                CMP R5, R0
                                BR.Z EscreverO ; Quando todos os digitos tiverem
 sido comparados, o programa sai desta rotina
                                CMP R5, R6
                                BR.Z ValorCertoPosicaoCerta ; Caso os digitos se
jam iguais, salta para a rotina ValorCertoPosicaoCerta, onde adicionará um 2 ao
valor de R3.
                                BR EscreverX

EscreverO:              MOV R3, 3d ; Esta retina começará a comparar o primeiro
digito de cada sequencia.
                                MOV R4, R0
                                MOV R1, R0
```



```
                                MOV R7, 11d
                                MOV R5, M[SP + 3]
                                MOV R6, M[SP + 11]
                                CMP R5, R6
                                JMP.Z ValorCertoPosicaoErrada
                                JMP ReporSeqComputador

; Esta rotina anula "elimina" da pilha os valores iguais e adiciona 2 ao valor d
e R3. (Defenimos 2, como o valor certo na posição certa para representar R3)
ValorCertoPosicaoCerta:         MOV M[R1], R0 ; Poe o valor da pilha a 0
                                                        MOV M[R4], R0
                                                        PUSH R3
                                                        MOV R3, M[Semelhanca]
                                                        SHL R3, 4 ;  passar ao b
it Seguinte
                                                        ADD R3, 2h
                                                        MOV M[Semelhanca], R3
                                                        POP R3
                                                        JMP EscreverX


ValorCertoPosicaoErrada:        CMP R6, R0 ; Nesta rotina, adiciona-se 1 ao valo
r de R3. (Defenimos 2, como o valor certo na posição certa para representar R3)
                                                        MOV M[R4], R0
                                                        BR.Z Comparacao
                                                        PUSH R3
                                                        MOV R3, M[Semelhanca]
                                                        SHL R3, 4 ; passar ao bi
t Seguinte
                                                        ADD R3, 1h
                                                        MOV M[Semelhanca], R3
                                                        POP R3
                                                        BR Comparacao

ReporSeqComputador:     INC R3 ; Esta retina permitirá comparar um algarismo da
sequencia do jogador, com cada algarismo da sequencia do computador. Caso sejam
iguais, saltará para a retina ValorCertoPosicaoErrada
                                        MOV R4, SP
                                        ADD R4, R3
                                        MOV R5, M[R4]
                                        CMP R3, 7
                                        BR.Z Comparacao
                                        CMP R5, R6
                                        BR.Z ValorCertoPosicaoErrada

                                        BR ReporSeqComputador

Comparacao:             MOV R3, 2d ; Nesta retina passar-se-á para o algarimos s
eguinte da sequencia do jogador, e vai-se comparar aos algarimos da sequencia se
creta através a retina ReporSeqComputador
                                INC R3
                                MOV R4, SP
                                ADD R4, R3
                                MOV R5, M[R4]
                                DEC R7
                                MOV R1, SP
                                ADD R1, R7
                                MOV R6, M[R1]
                                CMP R7, 7 ; quando R7 atinge o valor 7, já se fi
zeram todas as comparações, pelo que o se tem de sair desta rotina
                                JMP.Z NovaJogada
                                CMP R5, R6
                                JMP.Z ValorCertoPosicaoErrada
```

```
                            JMP ReporSeqComputador

NovaJogada: RET

JogadaSeguinte: MOV R5, M[80f8h]
                            CMP R5, 12 ; Verifica se já chegamos à ultima jo
gada possivel
                            JMP.Z MelhorJogada ; Se sim, o programa chega ao
 fim
                            MOV R5, M[80f6h]
                            CMP R5, 9
                            BR.NP a
                            MOV R5, 1
                            MOV M[IO_DISPLAY1], R5
                            MOV R5, M[80f6h]
                            SUB R5, 10
                        a:  MOV M[IO_DISPLAY], R5
                            MOV R5, M[80f8h]
                            INC R5 ; se não, passamos à jogada seguinte
                            MOV M[80f6h], R5 ; ------------ GUARDA EM QUE JO
GADA VAI
                            MOV M[80f8h], R5 ; ------------ GUARDA EM Q JOGA
DA VAI E QUANDO O JOGADOR ACERTA NA RESPOSTA PASSA PARA 12
                            PUSH R4
                            MOV R4, M[GUARDAR_CURSOR]
                            AND R4, ff00h
                            ADD R4, 0100h
                            MOV M[IO_CURSOR], R4
                        MOV M[GUARDAR_CURSOR], R4
                        POP R4
                            MOV R5, Frase
                            CALL EsceverJogadaSeguinte
NovoValorR2:    MOV R2, M[Jogada] ; devolve-se a R2 o valor incialmente introduz
ido pelo jogador
                            MOV R1, 7000h
                            AND R1, R2
                            SHR R1, 9
                            MOV M[Converter12Bits], R1
                            MOV R1, 0700h
                            AND R1, R2
                            SHR R1, 8
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            SHL R2, 3
                            MOV M[Converter12Bits], R2
                            POP R2
                            MOV R1, 0070h
                            AND R1, R2
                            SHR R1, 4
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            SHL R2, 3
                            MOV M[Converter12Bits], R2
                            POP R2
                            MOV R1, 0007h
                            AND R1, R2
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            MOV M[Converter12Bits], R2
```



```
                            POP R2
                            MOV R2, M[Converter12Bits]
                            MOV M[Jogada12Bits], R2

                            MOV R2, M[CS] ; devolve-se a R2 o valor incialme
nte introduzido pelo jogador
                            MOV R1, 7000h
                            AND R1, R2
                            SHR R1, 9
                            MOV M[Converter12Bits], R1
                            MOV R1, 0700h
                            AND R1, R2
                            SHR R1, 8
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            SHL R2, 3
                            MOV M[Converter12Bits], R2
                            POP R2
                            MOV R1, 0070h
                            AND R1, R2
                            SHR R1, 4
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            SHL R2, 3
                            MOV M[Converter12Bits], R2
                            POP R2
                            MOV R1, 0007h
                            AND R1, R2
                            PUSH R2
                            MOV R2, M[Converter12Bits]
                            ADD R2, R1
                            MOV M[Converter12Bits], R2
                            POP R2
                            MOV R2, M[Converter12Bits]
                            MOV M[CS12Bits], R2
                            MOV R1, M[CS12Bits]
                            MOV R2, M[Jogada12Bits]
                            MOV R4, R0
                            JMP Ola

Inicio: MOV R7, SP_INICIAL
                MOV SP, R7
                MOV R7, INT_MASK
                MOV M[INT_MASK_ADDR], R7
                ENI
                MOV R4, 1

quase:    INC M[VarAleatoria]
                  CMP R4, 0000h
          BR.P quase
          MOV R6, 0050h

apagar:  CMP R4, R6
          BR.NN passar
          MOV R5, ' '
          MOV M[IO], R5
          INC R4
          MOV M[IO_CURSOR], R4
          BR apagar
passar:   ADD R4, 00A0h
```

```
          ADD R6, 0100h
          CMP R4, 1850h
          BR.N apagar
          MOV R4, 0000h
          MOV M[IO_CURSOR], R4
          MOV M[GUARDAR_CURSOR], R4
          MOV R5,       Frase
                CALL EsceverJogadaSeguinte

                        MOV R1, 8
                        MOV M[TIMER_DELAY], R1
                        MOV R1, 1
                        MOV M[TIMER_CTRL], R1
                        MOV R2, FFFFh
                        MOV M[IO_LEDS], R2
                        JMP Doidice

X:                      MOV R1, M[TimerTick]
                        CMP R1, M[Nticks]
                        BR.NZ X
                        SHR R2, 1
                        MOV     M[IO_LEDS], R2
                        MOV M[TempoDeJogada], R2
                        MOV     M[TimerTick], R0
                        RET


Ciclo:  PUSH R2
                MOV R2, M[Jogada]
                SHL R2, 4
                ADD R2, M[BotaoPremido] ; converte em ASCII
                MOV M[Jogada], R2
                PUSH R4
                MOV R4, M[GUARDAR_CURSOR]
                MOV M[IO_CURSOR], R4
        INC R4
        MOV M[GUARDAR_CURSOR], R4
        MOV M[IO_CURSOR], R4
        POP R4
                MOV R7, M[BotaoPremido] ; Escreve na janela de texto
                ADD R7, 30h ; Cria espaco para adicionar o novo valor
                MOV M[IO], R7
                MOV M[BotaoPremido], R0 ; Atualiza a sequencia inserida pelo jog
ador
                MOV R2, M[NumeroDigitosInseridos]
                DEC R2
                MOV M[NumeroDigitosInseridos], R2
                MOV R2, M[Jogada]
                POP R2
                RET

TerminouTempo:  MOV R5, 12
                                MOV M[80f8h], R5
                                MOV M[80f6h], R5
                                JMP JogadaSeguinte


Doidice:        CMP R0, M[TimerTick]
                        CALL.N X
                        CMP M[BotaoPremido], R0
                        CALL.NZ Ciclo
                        CMP M[TempoDeJogada], R0
```



```
                        BR.Z TerminouTempo
                        CMP M[NumeroDigitosInseridos], R0
                        BR.NZ Doidice

InicioContinuacao:      MOV R1, M[CS] ; guardar em memoria a sequencia inicial
                                MOV R6, Mascara
                                PUSH R0
                                PUSH R6
                                PUSH R1
                                CALL GerarAleatorio
                                POP R1
                                POP R0
                                POP R0
                                MOV M[CS], R1
                                JMP SequenciaSeguinte

Ola:            MOV R1, 8
                        MOV M[TIMER_DELAY], R1
                        MOV R1, 1
                        MOV M[TIMER_CTRL], R1
                        MOV R2, FFFFh
                        MOV M[IO_LEDS], R2
                        BR Doidice2

Doidice2:       MOV R1, M[CS12Bits]
                        MOV R3, M[Semelhanca]
                        CMP R0, M[TimerTick]
                        CALL.N X
                        CMP M[BotaoPremido], R0
                        CALL.NZ Ciclo
                        CMP M[TempoDeJogada], R0
                        JMP.Z TerminouTempo
                        CMP M[NumeroDigitosInseridos], R0
                        BR.NZ Doidice2

SequenciaSeguinte:      MOV R2, M[Jogada]
                                PUSH R0 ; arranjar espaço na pilha para
os algarismos de cada sequencia
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R0
                                PUSH R4
                                MOV R4, M[GUARDAR_CURSOR]
                                AND R4, ff00h
                                ADD R4, 0100h
                                MOV M[IO_CURSOR], R4
                        MOV M[GUARDAR_CURSOR], R4
                        POP R4
                                MOV R7, 12d ; contador que servirá para
por na pilha a sequencia introduzida pelo jogador
                                MOV R3, 2d ; contador que servira para p
or na pilha a sequencia secreta
                                MOV R1, M[CS] ; Move para R1 a sequencia
 Secreta
                                CALL Igual
```

```
                        CALL Inicio1
                        MOV R7, 12d ; repor o contador ao seu va
lor inicial
                        MOV R3, 2d ; repor o contador ao seu val
or inicial
                        CALL EscreverX
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        POP R0
                        MOV R1, M[CS]
                        MOV R3, M[Semelhanca]
                        MOV R5, R3
                        PUSH R4
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, f000h
                        CMP R5, 2000h
                        BR.NZ x2
                        MOV R5, 'x'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        x2:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 0f00h
                        CMP R5, 0200h
                        BR.NZ x3
                        MOV R5, 'x'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        x3:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 00f0h
                        CMP R5, 0020h
                        BR.NZ x4
                        MOV R5, 'x'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        x4:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 000fh
                        CMP R5, 0002h
                        BR.NZ o1
                        MOV R5, 'x'
```



```
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        o1:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, f000h
                        CMP R5, 1000h
                        BR.NZ o2
                        MOV R5, 'o'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        o2:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 0f00h
                        CMP R5, 0100h
                        BR.NZ o3
                        MOV R5, 'o'
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        o3:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 00f0h
                        CMP R5, 0010h
                        BR.NZ o4
                        MOV R5, 'o'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        o4:             MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, 000fh
                        CMP R5, 0001h
                        BR.NZ Tr1
                        MOV R5, 'o'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
                MOV M[IO_CURSOR], R4
                        MOV M[IO], R5
        Tr1:    MOV R5, R3
                        MOV R4, M[GUARDAR_CURSOR]
                        AND R5, f000h
                        CMP R5, 0000h
                        BR.NZ Tr2
                        MOV R5, '-'
                        MOV R4, M[GUARDAR_CURSOR]
                        MOV M[IO_CURSOR], R4
                INC R4
                MOV M[GUARDAR_CURSOR], R4
```

```
                        MOV M[IO_CURSOR], R4
                            MOV M[IO], R5
                Tr2:    MOV R5, R3
                            MOV R4, M[GUARDAR_CURSOR]
                            AND R5, 0f00h
                            CMP R5, 0000h
                            BR.NZ Tr3
                            MOV R5, '-'
                            MOV R4, M[GUARDAR_CURSOR]
                            MOV M[IO_CURSOR], R4
                        INC R4
                        MOV M[GUARDAR_CURSOR], R4
                        MOV M[IO_CURSOR], R4
                            MOV M[IO], R5
                Tr3:    MOV R5, R3
                            MOV R4, M[GUARDAR_CURSOR]
                            AND R5, 00f0h
                            CMP R5, 0000h
                            BR.NZ Tr4
                            MOV R5, '-'
                            MOV R4, M[GUARDAR_CURSOR]
                            MOV M[IO_CURSOR], R4
                        INC R4
                        MOV M[GUARDAR_CURSOR], R4
                        MOV M[IO_CURSOR], R4
                            MOV M[IO], R5
                Tr4:    MOV R5, R3
                            MOV R4, M[GUARDAR_CURSOR]
                            AND R5, 000fh
                            CMP R5, 0000h
                            BR.NZ Continuacao
                            MOV R5, '-'
                            MOV R4, M[GUARDAR_CURSOR]
                            MOV M[IO_CURSOR], R4
                        INC R4
                        MOV M[GUARDAR_CURSOR], R4
                        MOV M[IO_CURSOR], R4
                            MOV M[IO], R5

        Continuacao:    POP R4
                            MOV R5, 4
                            MOV M[NumeroDigitosInseridos], R5
                            MOV M[Semelhanca], R0
                            JMP JogadaSeguinte

EsceverJogadaSeguinte:  PUSH R4
                                MOV R4, M[GUARDAR_CURSOR]
                                MOV M[IO_CURSOR], R4
                            INC R4
                            MOV M[GUARDAR_CURSOR], R4
                            MOV M[IO_CURSOR], R4
                            POP R4
                                MOV R6, M[R5] ; escrever na jane
la de texto "Jogada Seguinte:"
                                CMP R6, Fim_Frase
                                BR.Z b
                                MOV M[IO], R6
                                INC R5
                                BR EsceverJogadaSeguinte
                            b:  RET
```



```
Igual:  CMP R2, R1 ; Se a sequencia do jogador for igual a sequencia secreta, o
jogo acaba
                BR.Z abc
                RET
abc:    MOV R5, M[80f8h]
                MOV M[80f6h], R5
                MOV R5, 12 ; O jogo acaba ou quando for acertada a sequencia sec
reta, ou quando houver 12 jogadas sem sucesso, pelo que igualando este valor a 1
1, estamos a dizer que o jogo acabou. (iguala-se a 11 porque o contador começa n
o 0)
                MOV M[80f8h], R5
                RET

MelhorJogada:   MOV R5, M[80f6h]
                                CMP R5, 9
                                BR.NP bb
                                MOV R5, 1
                                MOV M[IO_DISPLAY1], R5
                                MOV R5, M[80f6h]
                                SUB R5, 10
                            bb: MOV M[IO_DISPLAY], R5
                                MOV R5, M[80f6h]
                                MOV R4, M[80f7h]
                                CMP R4, R5
                                JMP.NP NovaSequencia
                                MOV M[80f7h], R5
                                MOV R1, FFFFh
                                MOV M[LCD_CONTROL], R1
                                MOV R1, F000h
                                MOV M[LCD_CONTROL], R1
                                CMP R5, 9
                                BR.P passa
                                ADD R5, 48
                                MOV M[LCD_WRITE], R5
                                JMP NovaSequencia
                    passa:  MOV R1, 49
                                MOV M[LCD_WRITE], R1
                                MOV R1, F001h
                                MOV M[LCD_CONTROL], R1
                                ADD R5, 38
                                MOV M[LCD_WRITE], R5
                                BR NovaSequencia


NovaSequencia: JMP SegundoCicloJogada


Fim:    Br Fim

; Projeto realizado por João Marques 89473 e Francisco Figueiredo 89443
```