DEPARTAMENTO DE ENGENHARIA
INFORMÁTICA E DE SISTEMAS
INSTITUTO SUPERIOR DE ENGENHARIA DE
COIMBRA
2016/2017

Tecnologias e Arquiteturas de Computadores

Trabalho Prático
-Relatório-
Engenharia informática

Eduardo Fernandes nº 21250192

Daniel Couto nº 21250164

Coimbra, 8 de junho de 2017

Objetivos

Para este trabalho prático no âmbito da disciplina de Tecnologias e Arquiteturas de Computadores, da Licenciatura em Engenharia Informática no ISEC, pretende-se que se desenvolva um jogo em linguagem *Assembly*.

Este, consiste em percorrer um determinado labirinto, podendo este ser criado ou mesmo editado pelo utilizador, de forma a atingir uma determinada meta com o designado Avatar. O jogo é dado por terminado, como vitória, quando a meta é atingida pelo Avatar, após de se tratar de uma vitória é apresentado o tempo gasto pelo jogador e a sua posição no top 10 de pontuação do jogo.

Neste mesmo trabalho foi ainda implementada uma funcionalidade extra, que consiste numa forma alternativa de percorrer o mesmo labirinto recorrendo à indicação de todo o percurso através de uma sequência de dígitos em hexadecimal.

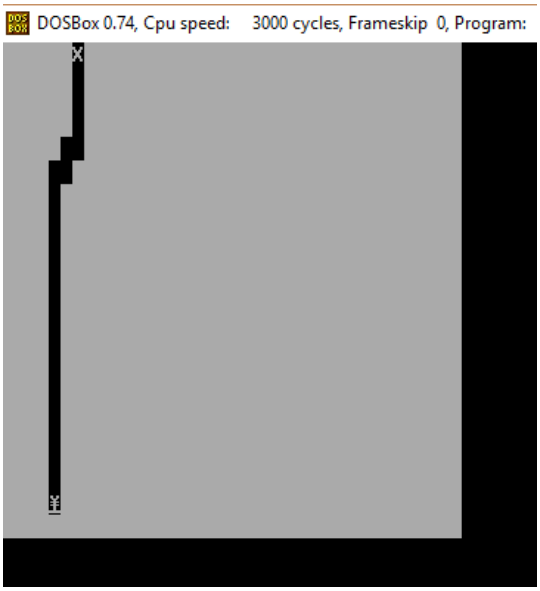
Funcionamento do Programa

Ao iniciar o programa é apresentado um Menu com as seguintes opções:

1. **Jogar**
2. **Top 10**
3. **Configuração do Labirinto**
4. **Bónus**
5. **Sair**

Jogar

Ao escolher esta opção é apresentado um labirinto com a dimensão **40 por 20**, recorrendo a uma variável no programa(defeito) ou a um ficheiro de texto, previamente escolhida na opção "Configuração do Labirinto".



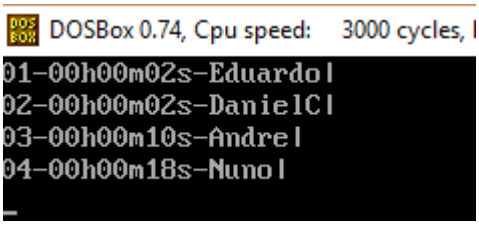
No centro do labirinto na primeira e última linha são mostrados o avatar e o fim do labirinto, respetivamente.

O Avatar é controlado pelas setas do teclado **(←, ↑, ↓, →)**, ou através de uma de uma sequência de dígitos em hexadecimal, previamente escolhida na opção 4 - Bónus.

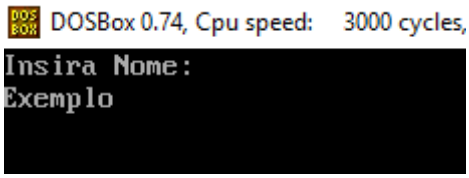
Para acabar o jogo é necessário chegar com o avatar ao fim, representado com um "**X**", evitando as paredes do labirinto. Ou clicando no botão ESC do teclado.

Top 10

Este simples jogo do labirinto dispõe um "Top 10" sempre que um jogador vencer, em que também é verificado se o seu tempo de jogo lhe permite a entrada no Top.



Caso o jogador se classifique para o Top 10, deverá ser introduzido o seu nome e respetivo tempo de jogo, tal como se pode verificar no seguinte exemplo:



Configuração do Labirinto

Ao escolher esta opção será apresentado um novo Menu com as opções:

1. **Carregar Labirinto Por Omissão**
2. **Carregar do ficheiro MAZE.TXT**
3. **Criar/Editar**

Carregar Labirinto Por Omissão

Ao selecionar esta opção, ao começar o jogo será apresentado um labirinto pré-definido no código através de uma variável do programa.

Carregar do Ficheiro MAZE.TXT

Ao ser selecionada esta opção, começando o jogo (Opção ***1 - Menu Principal***) será apresentado um labirinto lido do ficheiro "*Maze.txt*", que se encontra na mesma diretoria que se encontra o jogo.

Criar/Editar

Ao selecionar esta opção, será apresentado o labirinto correspondente ao ficheiro "*Maze2.txt*", em que podemos proceder à edição do mesmo.

Na edição recorremos às **setas (←; ↑; ↓; →)** do teclado para controlar o cursor e as teclas **1** e **0**, correspondendo o **1 - Criar Parede** e o **0 - Apagar Parede**.

Funcionalidade Extra (Bónus)

Tal como foi referido anteriormente nos “objetivos”, a funcionalidade extra consiste numa forma alternativa de percorrer o labirinto.

Esta forma consiste na indicação de todo o percurso através de uma sequência de dígitos em hexadecimal. Cada dígito hexadecimal corresponderá a um movimento.

Numa representação hexadecimal de 4 bits, os dois primeiros bits codificam a direção, como por exemplo:

00 - Norte | 01 -Sul | 10 - Este | 11 - Oeste.

Os restantes dois bits indicam o número de passos a avançar, como por exemplo:

00 - Um | 01 - Dois | 10 - Três | 11- Quatro.

Como exemplo final:

Norte, 2 Passos - (*0001*) | Este, 4 Passos - (*1011*).

Principais Funções

```
Main  proc

    mov     ax, dseg
    mov     ds,ax
    mov     ax,0B800h
    mov     es,ax

    CICLO:
        call    MostraMenu
        mov     ah,01h
        int     21h

        cmp     al,'0'
        je FIM
        cmp     al,'1'
        je JOGAR
        cmp     al,'2'
        je TOP
        cmp     al,'3'
        je CONF_MAZE
        cmp     al,'4'
        je ativarBonus
    jmp CICLO

    JOGAR:
        call    opJogar
        jmp CICLO
    TOP:
        call    opTop
        jmp CICLO
    CONF_MAZE:
        call    opMazeConfig
        jmp CICLO
    ativarBonus:
        not FlagBonus
        jmp CICLO
    FIM:
        mov     ah,4CH
        INT     21H
Main    endp
```

Na função *Main* é iniciado o programa, começando por armazenar as referencias do *Data Segment* e *Extra Segment* nos registos adequados.

De seguida é executada uma função que apresenta o menu no ecrã. É pedido ao utilizador um carater e interpretado executando a função correspondente.

```
opJogar proc
    call APAGA_ECRAN

    cmp auxConfigMaze, 1
    je Maze
    cmp auxConfigMaze, 2
    je MazeFich
    cmp auxConfigMaze, 3
    je CriaMaze
Maze:
    call MostraMaze
    jmp JOGAR

MazeFich:
    call MostraMazeFich
    jmp JOGAR

CriaMaze:

    call MostraMazeFich
    jmp JOGAR


JOGAR:
    MOV POSy, 19
    MOV POSya, 19
    call Ler_TEMPO

    cmp FlagBonus, 0
    je j1
    jmp j2
    j1:
        call JOGO
        jmp done
    j2:
        call JOGO2
    done:
    cmp flagAcabouJogo, 0
    je FimOpJogar

    call FimJogo
FimOpJogar: ret
opJogar endp
```

A função apresentada é executada na escolha da opção jogar no Menu principal.

É verificado o valor da variável *auxConfigMaze*, cujo é alterado na opção de *"Configuração do Maze"*, sendo assim executado a função correspondente.

Por fim é chamada a função principal, ***"JOGO"***, no entanto, antes é lido o tempo do sistema para que no final do jogo seja possível comparar o mesmo, obtendo assim o tempo gasto para o termino do labirinto.

```
JOGO proc

    goto_xy POSx,POSy   ; Vai para nova possição
    mov     ah, 08h ; Guarda o Caracter que está na posição do Cursor
    mov     bh,0        ; numero da página
    int     10h
    mov     Car, al ; Guarda o Caracter que está na posição do Cursor
    mov     Cor, ah ; Guarda a cor que está na posição do Cursor


    CICLO:
        goto_xy POSxa,POSya ; Vai para a posição anterior do cursor
        mov     ah, 02h
        mov     dl, Car ; Repoe Caracter guardado
        int     21H

        goto_xy POSx,POSy   ; Vai para nova possição
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h
        mov     Car, al ; Guarda o Caracter que está na posição do Cursor
        mov     Cor, ah ; Guarda a cor que está na posição do Cursor

        cmp al, 'X' ;se esta no fim sai
        je FIM

        goto_xy POSx,POSy   ; Vai para posição do cursor
    IMPRIME:
        mov     ah, 02h
        mov     dl, 190 ; Coloca AVATAR
        int     21H
        goto_xy POSx,POSy   ; Vai para posição do cursor

        mov     al, POSx    ; Guarda a posição do cursor
        mov     POSxa, al
        mov     al, POSy    ; Guarda a posição do cursor
        mov     POSya, al

    LER_SETA:
        call        LE_TECLA

        jmp     FRENTE


    FRENTE:
        cmp     al,48h
        jne     BAIXO

        dec     POSy        ;cima

        goto_xy POSx, POSy
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h

        cmp al, 219 ;se for parede

        jne CICLO

        inc POSy
        jmp     CICLO

    BAIXO:
        cmp     al,50h
        jne     ESQUERDA
        inc         POSy        ;Baixo

        goto_xy POSx, POSy
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h

        cmp al, 219
        jne CICLO

        dec POSy

        jmp     CICLO

    ESQUERDA:
        cmp     al,4Bh
        jne     DIREITA
        dec     POSx        ;Esquerda

        goto_xy POSx, POSy
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h

        cmp al, 219
        jne CICLO

        inc POSx
        jmp     CICLO

    DIREITA:
        cmp     al,4Dh
        jne     ESCAPE
        inc     POSx        ;Direita
        goto_xy POSx, POSy
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h

        cmp al, 219
        jne CICLO

        dec POSx
        jmp     CICLO

    ESCAPE:
        cmp al, 27
        jne LER_SETA
        ret
    FIM:
        mov flagAcabouJogo, 1
        ret
JOGO endp
```

A função **JOGO** é responsável pelo o movimento do *Avatar*. É começado por colocar o cursor na posição inicial do *Avatar*, depois é começado um ciclo que é repetido sempre que é lido e interpretada uma tecla, neste caso uma seta ou o botão ***ESC***.

Se for premida uma seta, é decrementado ou incrementado as variáveis da posição do Avatar, mas se a nova posição for uma parede desfaz a operação feita, ou seja, a tecla premida perde o efeito, deixando assim o Avatar no mesmo local.

```
JOGO2 proc
    goto_xy POSx,POSy   ; Vai para nova possição
    mov     ah, 08h ; Guarda o Caracter que está na posição do Cursor
    mov     bh,0        ; numero da página
    int     10h
    mov     Car, al ; Guarda o Caracter que está na posição do Cursor
    mov     Cor, ah ; Guarda a cor que está na posição do Cursor

    xor cx, cx

    CICLO:
        goto_xy POSxa,POSya ; Vai para a posição anterior do cursor
        mov     ah, 02h
        mov     dl, Car ; Repoe Caracter guardado
        int     21H

        goto_xy POSx,POSy   ; Vai para nova possição
        mov     ah, 08h
        mov     bh,0        ; numero da página
        int     10h
        mov     Car, al ; Guarda o Caracter que está na posição do Cursor
        mov     Cor, ah ; Guarda a cor que está na posição do Cursor

        cmp al, 'X' ;se esta no fim sai
        je FIM

        goto_xy POSx,POSy   ; Vai para posição do cursor
    IMPRIME:
        mov     ah, 02h
        mov     dl, 190 ; Coloca AVATAR
        int     21H
        goto_xy POSx,POSy   ; Vai para posição do cursor

        mov     al, POSx    ; Guarda a posição do cursor
        mov     POSxa, al
        mov     al, POSy    ; Guarda a posição do cursor
        mov     POSya, al

    LER_SETA:
        call        LE_TECLA2
        jmp     FRENTE
    FRENTE:
        cmp     al,48h
        jne     BAIXO
        passos1:
            dec cl
            dec     POSy        ;cima

            goto_xy POSx, POSy
            mov     ah, 08h
            mov     bh,0        ; numero da página
            int     10h

            cmp al, 219 ;se for parede
            je NFrente
        cmp cl, 0
        jne passos1
        jmp CICLO

            NFrente:
                inc POSy
        cmp cl, 0
        jne passos1
        jmp     CICLO
    BAIXO:
        cmp     al,50h
        jne     ESQUERDA
        passos2:
            dec cl
            inc         POSy        ;Baixo

            goto_xy POSx, POSy
            mov     ah, 08h
            mov     bh,0        ; numero da página
            int     10h

            cmp al, 219
            je NBaixo
        cmp cl, 0
        jne passos2
        jmp     CICLO
            NBaixo:
                dec POSy
        cmp cl, 0
        jne passos2
        jmp     CICLO

    ESQUERDA:
        cmp     al,4Bh
        jne     DIREITA
        passos3:
            dec cl
            dec     POSx        ;Esquerda

            goto_xy POSx, POSy
            mov     ah, 08h
            mov     bh,0        ; numero da página
            int     10h

            cmp al, 219
            je NEsquerda
        cmp cl, 0
        jne passos3
        jmp     CICLO
            NEsquerda:
                inc POSx
        cmp cl, 0
        jne passos3
        jmp     CICLO

    DIREITA:
        cmp     al,4Dh
        jne     ESCAPE
        passos4:
            dec cl
            inc     POSx        ;Direita
            goto_xy POSx, POSy
            mov     ah, 08h
            mov     bh,0        ; numero da página
            int     10h

            cmp al, 219
            je NDireita
        cmp cl, 0
        jne passos4
        jmp     CICLO
            NDireita:
                dec POSx
          cmp cl, 0
          jne passos4
          jmp       CICLO
          jmp       CICLO

     ESCAPE:
          cmp al, 27
          jne LER_SETA
          ret

     FIM:
          mov flagAcabouJogo, 1
          ret
JOGO2 endp
```

O Seguinte excerto de código do programa tem semelhanças com o anterior (JOGO), pois tem o mesmo propósito, mas adaptado para a funcionalidade extra.

A função de leitura da seta é diferente pois a mesma irá decifrar o número de passos a dar pelo o Avatar e a direção do mesmo.

O número de passos ficará no registo ***CL***, enquanto a direção é guardada no registo ***AH*** com os mesmos valores das setas para o aproveitamento do código da antiga função.

```
LE_TECLA2   PROC
    xor cx, cx
    mov     ah,08h
    int     21h

    cmp al, '0'
    jae M0
    jmp prox1

    M0:
        cmp al, '9'
        jbe m9
        jmp prox1

    m9:
        mov bh, 30h
        jmp prox3

    prox1:
        cmp al, 'A'
        jae MA
        jmp prox2

        MA:
            cmp al, 'F'
            jbe mF
            jmp prox2

        mF:
            mov bh, 41h
            sub bh, 0AH
            jmp prox3
    prox2:
        cmp al, 'a'
        jae MA1
        jmp prox3

        MA1:
            cmp al, 'f'
            jbe mF1
            jmp SAI_TECLA

        mF1:
            mov bh, 51h
            sub bh, 0AH
            jmp prox3

    prox3:
        sub al, bh
        jc SAI_TECLA
        cmp al, 0
        jae maior0
        jmp SAI_TECLA

        maior0:
            cmp al, 3
            jbe menor3
            jmp prox4

            menor3:
                inc al
                mov cl, al
                mov al, 48h;
                jmp SAI TECLA
     prox4:
         cmp al, 4
         jae maior4
         jmp prox5

         maior4:
             cmp al, 7
             jbe menor7
             jmp prox5

             menor7:
                 inc al
                 sub al, 4h
                 mov cl, al
                 mov al, 50h;
                 jmp SAI_TECLA
     prox5:
         cmp al, 8
         jae maior8
         jmp prox6

         maior8:
             cmp al, 11
             jbe menorb
             jmp prox6

             menorb:
                 inc al
                 sub al, 08h
                 mov cl, al
                 mov al, 4Dh;
                 jmp SAI_TECLA
     prox6:
         cmp al, 12
         jae maiorc
         jmp SAI_TECLA

         maiorc:
             cmp al, 15
             jbe menorf
             jmp SAI_TECLA

             menorf:

                 inc al
                 sub al, 0Ch
                 mov cl, al
                 ; mov ch, al
                 mov al, 4Bh;
                 jmp SAI_TECLA


SAI_TECLA:
    RET
LE_TECLA2   endp
```

A função mostrada é responsável por pedir o valor Hexadecimal ao utilizador e decifrar o mesmo a direção e o número de passos a dar pelo o *Avatar*.

Podemos dividir a função em 2 partes, a primeira regista no ***BH***, o valor a retirar ao valor do carater lido para obter o valor em hexadecimal do carater. A segunda parte é verificado e em que intervalo se encontra o tal valor, como é apresentada na tabela seguinte:

	Binary	Hex	
N	0000	0	0
	0001	1	
	0010	2	
	0011	3	
S	0100	4	4
	0101	5	
	0110	6	
	0111	7	
E	1000	8	8
	1001	9	
	1010	A	
	1011	B	
O	1100	C	12
	1101	D	
	1110	E	
	1111	F	

Assim podemos concluir a direção do movimento.

Quanto ao numero de passos foi criada e usada a seguinte formula para calcular:

(Valor Hexadecimal + 1) – número correspondente em azul.

Anexo:

```
.8086
.model small
.stack 2048

dseg        segment para public 'data'
            MenuPrincipal           db              10,'MENU:',10,' 1 - Jogar ',10,' 2 - Top 10 ',10,' 3 - Configuracao do labirinto ',10,' 4 -
Bonus ',10,' 0 - Sair',10,' Opcao(0 - 3): $'
            MenuConfigMaze          db              10,'MENU Configuracao do Maze:',10,' 1 - Carregar Labirinto Por Omissao ',10,' 2 -
Carregar do ficheiro MAZE.TXT',10,' 3 - Criar/Editar ',10,' 0 - Sair',10,' Opcao(0 - 3): $'

            Erro_Open       db      'Erro ao tentar abrir o ficheiro$'
            Erro_Ler_Msg    db      'Erro ao tentar ler do ficheiro$'
            Erro_Close      db      'Erro ao tentar fechar o ficheiro$'

            Fich            db      'MAZE.TXT',0
            Fich2           db      'MAZE2.TXT',0
            FichTOP                 db              'TOP.TXT',0

            HandleFich      dw      0
            car_fich        db      ?
            car_fich2       dw      ?
            flagFich                db              1

            contaColuna             db              0
            contaLinha              db              0
            POSx                    db              ?
            POSy                    db              20
            POSxa                   db              ?
            POSya                   db              20
            Car                             db              32      ; Guarda um caracter do Ecran
            CarN                            db              32      ; Guarda um caracter do Ecran
            Cor                             db              7       ; Guarda os atributos de cor do caracter
            flagAcabouJogo          db              0

            Horas                   dw              0                                       ; Vai guardar a HORA
actual
            Minutos                 dw              0                                       ; Vai guardar os minutos
actuais
            Segundos                dw              0                                       ; Vai guardar os
segundos actuais

            msgErrorCreate          db      "Ocorreu um erro na criacao do ficheiro!$"
            msgErrorWrite           db      "Ocorreu um erro na escrita para ficheiro!$"
            msgErrorClose           db      "Ocorreu um erro no fecho do ficheiro!$"
            stringTempo                     db                      '       ';9 digitos para tempo
            stringSI                db              '  ';3 digitos para num TOP

            FlagCompararTempo       dw              0                                       ; Vai guardar a HORA actual

            FlagBonus       db              0

            auxConfigMaze           db              1

            defaultMaze             db              '000000X00000000000000000000000000000000|'
                                    db              '00000010000000000000000000000000000000|'
                                    db              '00000010000000000000000000000000000000|'
                                    db              '00000010000000000000000000000000000000|'
                                    db              '00000110000000000000000000000000000000|'
                                    db              '00001100000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '00001000000000000000000000000000000000|'
                                    db              '0000I000000000000000000000000000000000|'
                                    db              '00000000000000000000000000000000000000|',0
```

```
        NomeUser        db              10, 0, 10 dup (' ')


        buffer      db      22 dup (41 dup (' '));CRIAR MAZE

        bufferTOP       db      25 dup (10 dup (' '))


        stringTestes1       db      10 dup(' '), '$'
        msgNome                 db      'Insira Nome: ', 10,'$'

        numLinhasTOP        db          ?



dseg    ends

PILHA   SEGMENT PARA STACK 'STACK'
                db 2048 dup(?)
PILHA   ENDS

cseg    segment para public 'code'
assume          cs:cseg, ds:dseg

goto_xy macro           POSx,POSy
        mov             ah,02h
        mov             bh,0                ; numero da página
        mov             dl,POSx
        mov             dh,POSy
        int             10h
endm

; TESTES PROC
                ; PUSH AX
        ; PUSH BX
        ; PUSH CX
        ; PUSH DX

        ; PUSHF

                ; mov stringTestes1[0], 10
                ; mov bl, stringTempo[7]
                ; mov stringTestes1[1], bl
                ; mov stringTestes1[2], '-'
                ; mov stringTestes1[3], al
                ; mov stringTestes1[4], '-'

                ; mov     ax,si
                ; MOV             bl, 10
                ; div     bl
                ; add     al, 30h                                 ; Caracter Correspondente às dezenas
                ; add             ah,     30h                                 ; Caracter Correspondente às
unidades

                ; mov stringTestes1[5], al
                ; mov stringTestes1[6], ah
                ; mov stringTestes1[7], 10
                ; lea dx, stringTestes1
                ; mov ah, 09h
                ; int 21h

                ; mov ah, 01
                        ; int 21h

                        ; POPF
        ; POP DX
        ; POP CX
        ; POP BX
        ; POP AX

; TESTES endp



APAGA_ECRAN     PROC
                PUSH BX
                PUSH AX
                PUSH CX
                PUSH SI
                XOR     BX,BX
```

```
                        MOV         CX,25*80
                        mov bx,0
                        ; MOV SI,BX
APAGA:
                        MOV         AL,' '
                        MOV         BYTE PTR ES:[BX],AL
                        MOV         BYTE PTR ES:[BX+1],7
                        INC         BX
                        INC BX
                        ; INC SI
                        LOOP        APAGA
                        POP SI
                        POP CX
                        POP AX
                        POP BX
                        goto_xy 0, 0
                        RET
APAGA_ECRAN             ENDP

MostraMenu              proc
            CALL APAGA_ECRAN

            mov ah, 09h
            lea dx, MenuPrincipal
            int 21h
            ret
MostraMenu              endp

Ler_TEMPO PROC

            PUSH AX
            PUSH BX
            PUSH CX
            PUSH DX

            PUSHF

            MOV AH, 2CH         ; Buscar a hORAS
            INT 21H

            XOR AX,AX
            MOV AL, DH          ; segundos para al
            mov Segundos, AX                    ; guarda segundos na variavel correspondente

            XOR AX,AX
            MOV AL, CL          ; Minutos para al
            mov Minutos, AX     ; guarda MINUTOS na variavel correspondente

            XOR AX,AX
            MOV AL, CH          ; Horas para al
            mov Horas, AX                               ; guarda HORAS na variavel correspondente

            POPF
            POP DX
            POP CX
            POP BX
            POP AX
            RET
Ler_TEMPO   ENDP

CALCULA_TEMPO PROC
            PUSH AX
            PUSH BX
            PUSH CX
            PUSH DX

            PUSHF

            MOV AH, 2CH         ; Buscar a hORAS
            INT 21H

            XOR AX,AX
            MOV AL, DH          ; segundos para al
            sub ax, Segundos                    ; guarda segundos na variavel correspondente
            mov Segundos,ax

            XOR AX,AX
            MOV AL, CL          ; Minutos para al
            sub AX, Minutos             ; guarda MINUTOS na variavel correspondente
            mov Minutos,ax
```

```
		XOR AX,AX
		MOV AL, CH		; Horas para al
		sub AX, Horas					; guarda HORAS na variavel correspondente
		mov Horas,ax


		; ---------TEMPO PARA STRING----------------

		mov		ax,Horas
		MOV			bl, 10
		div		bl
		add		al, 30h					; Caracter Correspondente às dezenas
		add			ah,		30h				; Caracter Correspondente às unidades
		MOV		stringTempo[0],al			;
		MOV		stringTempo[1],ah
		MOV		stringTempo[2],'h'


		mov		ax,Minutos
		MOV			bl, 10
		div		bl
		add		al, 30h					; Caracter Correspondente às dezenas
		add			ah,		30h				; Caracter Correspondente às unidades
		MOV		stringTempo[3],al			;
		MOV		stringTempo[4],ah
		MOV		stringTempo[5],'m'

		mov		ax,Segundos
		MOV			bl, 10
		div		bl
		add		al, 30h					; Caracter Correspondente às dezenas
		add			ah,		30h				; Caracter Correspondente às unidades
		MOV		stringTempo[6],al			;
		MOV		stringTempo[7],ah
		MOV		stringTempo[8],'s'
		MOV		stringTempo[9],'$'

		POPF
		POP DX
		POP CX
		POP BX
		POP AX

		RET
CALCULA_TEMPO	ENDP

MostraMazeFich proc
		mov	ah,3dh				; vamos abrir ficheiro para leitura
		mov	al,0			; tipo de ficheiro

		cmp auxConfigMaze, 2
		je Ficheiro1
		jmp Ficheiro2

		Ficheiro1:
			lea	dx,Fich
			jmp		done
		Ficheiro2:
			lea	dx,Fich2
			jmp		done
		done:
			int	21h				; abre para leitura
		jc	erro_abrir			; pode aconter erro a abrir o ficheiro
		mov	HandleFich,ax		; ax devolve o Handle para o ficheiro
		jmp	ler_ciclo			; depois de abero vamos ler o ficheiro

		erro_abrir:
			mov	ah,09h
			lea	dx,Erro_Open
			int	21h
			mov		flagFich, 0
			ret
		mov contaColuna, 0
		ler_ciclo:
			mov	ah,3fh				; indica que vai ser lido um ficheiro
			mov	bx,HandleFich		; bx deve conter o Handle do ficheiro previamente aberto
			mov	cx,1				; numero de bytes a ler
			lea	dx,car_fich			; vai ler para o local de memoria apontado por dx (car_fich)
			int	21h					; faz efectivamente a leitura
			jc		erro_ler		; se carry é porque aconteceu um erro
			cmp		ax,0			;EOF?	verifica se já estamos no fim do ficheiro
```

```
                je              fecha_ficheiro          ; se EOF fecha o ficheiro

                ;alterar os 0 e 1...

                cmp car_fich, '0'
                je PAREDE
                cmp car_fich, '1'
                JE NPAREDE
                cmp car_fich, '|'
                JE NOVALINHA
                cmp car_fich, 'X'
                JE FIM_MAZE
                cmp car_fich, 'I'
                JE AVATAR
        PAREDE:
                mov car_fich, 219
                JMP MOSTRA
        NPAREDE:
                mov car_fich, 32
                JMP MOSTRA
        NOVALINHA:
                mov car_fich, 10
                mov contaColuna, 0
                inc contaLinha
                JMP MOSTRA
        FIM_MAZE:
                mov car_fich, 'X'
                JMP MOSTRA
        AVATAR:
                mov car_fich, 32
                mov ah, contaColuna
                sub ah, 1
                mov POSx, ah
                mov POSxA, ah

                JMP MOSTRA
        MOSTRA:
                mov     ah,02h                          ; coloca o caracter no ecran
                mov             dl,car_fich             ; este é o caracter a enviar para o ecran
                int             21h                     ; imprime no ecran
                inc           contaColuna
                jmp             ler_ciclo               ; continua a ler o ficheiro

        erro_ler:
                mov     ah,09h
                lea     dx,Erro_Ler_Msg
                int     21h

        fecha_ficheiro:                                 ; vamos fechar o ficheiro
                mov     ah,3eh
                mov     bx,HandleFich
                int     21h
                ; mov          flagFich, 0
                ret

                mov     ah,09h                          ; o ficheiro pode não fechar correctamente
                lea     dx,Erro_Close
                Int     21h
        ret
MostraMazeFich endp

MostraMaze proc

        ; mov dl,10
        ; mov ah,2h
        ; int 21h

        mov contaColuna, 0
        xor si, si
        ler_ciclo:
                mov ah, defaultMaze[si]
                mov car_fich, ah
                inc si

                cmp car_fich, 0
                je FIM
                cmp car_fich, '0'
                je PAREDE
                cmp car_fich, '1'
                JE NPAREDE
                cmp car_fich, '|'
```

```
                JE NOVALINHA
                cmp car_fich, 'X'
                JE FIM_MAZE
                cmp car_fich, 'I'
                JE AVATAR
        PAREDE:
                mov car_fich, 219
                JMP MOSTRA
        NPAREDE:
                mov car_fich, 32
                JMP MOSTRA
        NOVALINHA:
                mov car_fich, 10
                mov contaColuna, 0
                inc contaLinha
                JMP MOSTRA
        FIM_MAZE:
                mov car_fich, 'X'
                JMP MOSTRA
        AVATAR:
                mov car_fich, 32
                mov ah, contaColuna
                sub ah, 1
                mov POSx, ah
                mov POSxA, ah

                JMP MOSTRA
        MOSTRA:
                mov     ah,02h                          ; coloca o caracter no ecran
                mov             dl,car_fich             ; este é o caracter a enviar para o ecran
                int             21h                     ; imprime no ecran
                inc             contaColuna
                jmp             ler_ciclo       ; continua a ler o ficheiro
FIM:
        ret
MostraMaze endp

LE_TECLA    PROC
        xor cx, cx
        mov             ah,08h
        int             21h
SAI_TECLA:
        RET
LE_TECLA    endp
LE_TECLA2   PROC
        xor cx, cx
        mov             ah,08h
        int             21h

        cmp al, '0'
        jae M0
        jmp prox1

        M0:
                cmp al, '9'
                jbe m9
                jmp prox1

        m9:
                mov bh, 30h
                jmp prox3

        prox1:
                cmp al, 'A'
                jae MA
                jmp prox2

                MA:
                        cmp al, 'F'
                        jbe mF
                        jmp prox2

                mF:
                        mov bh, 41h
                        sub bh, 0AH
                        jmp prox3
        prox2:
                cmp al, 'a'
                jae MA1
                jmp prox3
```

```
        MA1:
                cmp al, 'f'
                jbe mF1
                jmp SAI_TECLA

        mF1:
                mov bh, 51h
                sub bh, 0AH
                jmp prox3

prox3:
        sub al, bh
        jc SAI_TECLA


        ; mov ah, 02h
        ; mov dl, 30h
        ; int 21h
        ; mov ah, 02h
        ; mov dl, bh
        ; int 21h
        ; mov ah, 01h
        ; int 21h


        cmp al, 0
        jae maior0
        jmp SAI_TECLA

        maior0:
                cmp al, 3
                jbe menor3
                jmp prox4

                menor3:
                        ; mov ah, al;


                        inc al
                        ; mov ch, al
                        mov cl, al
                        mov al, 48h;
                        jmp SAI_TECLA
prox4:
        cmp al, 4
        jae maior4
        jmp prox5

        maior4:
                cmp al, 7
                jbe menor7
                jmp prox5

                menor7:
                        ; mov ah, 02h
                        ; mov dl, '*'
                        ; int 21h

                        inc al
                        sub al, 4h
                        ; mov ch, al
                        mov cl, al
                        mov al, 50h;
                        jmp SAI_TECLA
prox5:
        cmp al, 8
        jae maior8
        jmp prox6

        maior8:
                cmp al, 11
                jbe menorb
                jmp prox6

                menorb:
                        ; mov ah, 02h
                        ; mov dl, '*'
                        ; int 21h
                        ; mov ah, 01h
                        ; int 21h
```

```
                                        inc al
                                        sub al, 08h
                                        ; mov ch, al
                                        mov cl, al
                                        mov al, 4Dh;
                                        jmp SAI_TECLA
        prox6:
                cmp al, 12
                jae maiorc
                jmp SAI_TECLA

                maiorc:
                        cmp al, 15
                        jbe menorf
                        jmp SAI_TECLA

                        menorf:

                                inc al
                                sub al, 0Ch
                                mov cl, al
                                ; mov ch, al
                                mov al, 4Bh;
                                jmp SAI_TECLA



SAI_TECLA:

        ; mov ah, 02h
        ; mov dl, cl
        ; int 21h
        ; mov ah, 01h
        ; int 21h
        RET
LE_TECLA2  endp


JOGO2 proc

        goto_xy     POSx,POSy  ; Vai para nova possição
        mov         ah, 08h    ; Guarda o Caracter que está na posição do Cursor
        mov                    bh,0                    ; numero da página
        int                    10h
        mov                    Car, al     ; Guarda o Caracter que está na posição do Cursor
        mov                    Cor, ah     ; Guarda a cor que está na posição do Cursor

        xor cx, cx

        CICLO:
                goto_xy     POSxa,POSya            ; Vai para a posição anterior do cursor
                mov                    ah, 02h
                mov                    dl, Car     ; Repoe Caracter guardado
                int                    21H

                goto_xy     POSx,POSy  ; Vai para nova possição
                mov         ah, 08h
                mov                    bh,0                    ; numero da página
                int                    10h
                mov                    Car, al     ; Guarda o Caracter que está na posição do Cursor
                mov                    Cor, ah     ; Guarda a cor que está na posição do Cursor

                cmp al, 'X' ;se esta no fim sai
                je FIM

                goto_xy     POSx,POSy  ; Vai para posição do cursor
        IMPRIME:
                mov                    ah, 02h
                mov                    dl, 190     ; Coloca AVATAR
                int                    21H
                goto_xy     POSx,POSy  ; Vai para posição do cursor

                mov                    al, POSx    ; Guarda a posição do cursor
                mov                    POSxa, al
                mov                    al, POSy    ; Guarda a posição do cursor
                mov         POSya, al

        LER_SETA:
                ;se cx == 0 faz prox linha
                ; cmp cx, 0
                ; jne         NLeTecla
```

```
        ; mov ah, 01h
        ; int 21h
        ; cmp FlagBonus, 0
        ; je l1
        ; jmp l2
        ; l1:
                ; call          LE_TECLA
                ; jmp frente
        ; l2:
        call            LE_TECLA2
        jmp             FRENTE

FRENTE:


        cmp     al,48h
        jne             BAIXO
        ; xor cx, cx
        ; mov cx, 2
        passos1:
                dec cl
                ; mov ah, 02h
                ; mov dl, cl
                ; int 21h
                ; mov ah, 01h
                ; int 21h
                dec             POSy            ;cima

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0            ; numero da página
                int             10h

                cmp al, 219 ;se for parede
                je NFrente
        cmp cl, 0
        jne passos1
        jmp CICLO

                NFrente:
                        inc POSy
        cmp cl, 0
        jne passos1
        jmp             CICLO

BAIXO:
        cmp             al,50h
        jne             ESQUERDA
        passos2:
                dec cl
                inc             POSy            ;Baixo

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0            ; numero da página
                int             10h

                cmp al, 219
                je NBaixo
        cmp cl, 0
        jne passos2
        jmp             CICLO
                NBaixo:
                        dec POSy
        cmp cl, 0
        jne passos2
        jmp             CICLO

ESQUERDA:
        cmp             al,4Bh
        jne             DIREITA
        passos3:
                dec cl
                dec             POSx            ;Esquerda

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0            ; numero da página
```

```
                int                     10h

                cmp al, 219
                je NEsquerda
        cmp cl, 0
        jne passos3
        jmp             CICLO
                NEsquerda:
                        inc POSx
        cmp cl, 0
        jne passos3
        jmp             CICLO
        ; jmp           CICLO

DIREITA:
        cmp             al,4Dh
        jne             ESCAPE
        passos4:
                dec cl
                inc                     POSx            ;Direita
                goto_xy POSx, POSy
                mov     ah, 08h
                mov                     bh,0            ; numero da página
                int                     10h

                cmp al, 219
                je NDireita
        cmp cl, 0
        jne passos4
        jmp             CICLO
                NDireita:
                        dec POSx
        cmp cl, 0
        jne passos4
        jmp             CICLO
        jmp             CICLO

ESCAPE:
        cmp al, 27
        jne LER_SETA
        ret

FIM:
        mov flagAcabouJogo, 1
        ret
JOGO2 endp


JOGO proc

        goto_xy     POSx,POSy  ; Vai para nova possição
        mov         ah, 08h    ; Guarda o Caracter que está na posição do Cursor
        mov                    bh,0            ; numero da página
        int                    10h
        mov                    Car, al     ; Guarda o Caracter que está na posição do Cursor
        mov                    Cor, ah     ; Guarda a cor que está na posição do Cursor


        CICLO:
                goto_xy     POSxa,POSya        ; Vai para a posição anterior do cursor
                mov                    ah, 02h
                mov                    dl, Car     ; Repoe Caracter guardado
                int                    21H

                goto_xy     POSx,POSy  ; Vai para nova possição
                mov         ah, 08h
                mov                    bh,0                ; numero da página
                int                    10h
                mov                    Car, al     ; Guarda o Caracter que está na posição do Cursor
                mov                    Cor, ah     ; Guarda a cor que está na posição do Cursor

                cmp al, 'X' ;se esta no fim sai
                je FIM

                goto_xy     POSx,POSy  ; Vai para posição do cursor
        IMPRIME:
                mov                    ah, 02h
                mov                    dl, 190     ; Coloca AVATAR
                int                    21H
                goto_xy     POSx,POSy  ; Vai para posição do cursor
```

```
                mov             al, POSx        ; Guarda a posição do cursor
                mov             POSxa, al
                mov             al, POSy        ; Guarda a posição do cursor
                mov     POSya, al

LER_SETA:
                call            LE_TECLA

                jmp             FRENTE


FRENTE:
                cmp     al,48h
                jne             BAIXO

                dec             POSy                    ;cima

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0                    ; numero da página
                int             10h

                cmp al, 219 ;se for parede

                jne CICLO

                inc POSy
                jmp             CICLO

BAIXO:
                cmp             al,50h
                jne             ESQUERDA
                inc             POSy                    ;Baixo

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0                    ; numero da página
                int             10h

                cmp al, 219
                jne CICLO

                dec POSy

                jmp             CICLO

ESQUERDA:
                cmp             al,4Bh
                jne             DIREITA
                dec             POSx                    ;Esquerda

                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0                    ; numero da página
                int             10h

                cmp al, 219
                jne CICLO

                inc POSx
                jmp             CICLO

DIREITA:
                cmp             al,4Dh
                jne             ESCAPE
                inc             POSx                    ;Direita
                goto_xy POSx, POSy
                mov     ah, 08h
                mov             bh,0                    ; numero da página
                int             10h

                cmp al, 219
                jne CICLO

                dec POSx
                jmp             CICLO

ESCAPE:
                cmp al, 27
                jne LER_SETA
                ret
```

```
        FIM:
                mov flagAcabouJogo, 1
                ret
JOGO endp



LeProxCaraterTOP proc
        mov     ah,3fh                                  ; indica que vai ser lido um ficheiro
        mov     bx,HandleFich           ; bx deve conter o Handle do ficheiro previamente aberto
        mov     cx,1                                    ; numero de bytes a ler
        lea     dx,car_fich             ; vai ler para o local de memoria apontado por dx (car_fich)
        int     21h                                     ; faz efectivamente a leitura

        ; mov           bx, ax;aux

        ; mov     ah,02h                                ; coloca o caracter no ecran
        ; mov             dl,car_fich                   ; este é o caracter a enviar para o ecran
        ; int             21h                           ; imprime no ecran

        ; mov ax, bx
        RET

LeProxCaraterTOP endp
LeProxCaraterTOP2 proc
        mov     ah,3fh                                  ; indica que vai ser lido um ficheiro
        mov     bx,HandleFich           ; bx deve conter o Handle do ficheiro previamente aberto
        mov     cx,1                                    ; numero de bytes a ler
        lea     dx,car_fich             ; vai ler para o local de memoria apontado por dx (car_fich)
        int     21h                                     ; faz efectivamente a leitura

        mov           bx, ax;aux

        mov     ah,02h                                  ; coloca o caracter no ecran
        mov             dl,car_fich                     ; este é o caracter a enviar para o ecran
        int             21h                             ; imprime no ecran

        mov ax, bx
        RET

LeProxCaraterTOP2 endp


CompararTempo proc
        ; mov dl,10
        ; mov ah,2h
        ; int 21h

        mov     ah,3dh                                  ; vamos abrir ficheiro para leitura
        mov     al,0                            ; tipo de ficheiro
        lea     dx,FichTOP                              ; nome do ficheiro
        int     21h                             ; abre para leitura
        jc      erro_abrir                      ; pode aconter erro a abrir o ficheiro
        mov     HandleFich,ax                   ; ax devolve o Handle para o ficheiro
        jmp     Start                   ; depois de abero vamos ler o ficheiro

        erro_abrir:
                mov     ah,09h
                lea     dx,Erro_Open
                int     21h
                mov             flagFich, 0
                ret

        Start:
                xor si, si
                inc si

        Ciclo:
                CMP SI, 11
                je fecha_ficheiro
                ;------------------ LE NUMERO DO TOP ------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                                    ;EOF?           verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro
                ;-------------------------------------------------------

                ;------------------ DESPREZA '-' ------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                                    ;EOF?           verifica se já estamos no fim do ficheiro
```

```
je              fecha_ficheiro          ; se EOF fecha o ficheiro
;--------------------------------------------------------
;----------------- DESPREZA '-' -----------------
call LeProxCaraterTOP
jc              erro_ler                ; se carry é porque aconteceu um erro
cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
je              fecha_ficheiro          ; se EOF fecha o ficheiro
;--------------------------------------------------------

call LeProxCaraterTOP
jc              erro_ler                ; se carry é porque aconteceu um erro
cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
je              fecha_ficheiro          ; se EOF fecha o ficheiro

mov al, stringTempo[0]
mov ah, car_fich
cmp al, ah
je HorasIgual
cmp al, car_fich
jb MelhorTempo
cmp al, car_fich
ja proxLinha

HorasIgual:

                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro



                mov al, stringTempo[1]
                cmp al, car_fich
                je HorasIgual2
                cmp al, car_fich
                jb MelhorTempo
                cmp al, car_fich
                ja proxLinha
HorasIgual2:




                ;--------DESPREZA O 'h'  ------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro
                ;-----------------------------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro

                mov al, stringTempo[3]

                cmp al, car_fich
                je MinutosIgual
                cmp al, car_fich
                jb MelhorTempo
                cmp al, car_fich
                ja proxLinha

MinutosIgual:

                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                            ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro
; mov ah, 02h
; mov dl, car_fich
; int           21h

                mov al, stringTempo[4]

                cmp al, car_fich
                je MinutosIgual2
                cmp al, car_fich
                jb MelhorTempo
```

```
                cmp al, car_fich
                ja proxLinha
MinutosIgual2:


                ;--------DESPREZA O 'm'  ------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                                    ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro
                ;-----------------------------------------
                call LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                                    ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro

                mov al, stringTempo[6]

                cmp al, car_fich
                je SegundosIgual
                cmp al, car_fich
                jb MelhorTempo
                cmp al, car_fich
                ja proxLinha

SegundosIgual:
                call            LeProxCaraterTOP
                jc              erro_ler                ; se carry é porque aconteceu um erro
                cmp             ax,0                                    ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro          ; se EOF fecha o ficheiro

                mov al, stringTempo[7]
                ; cmp al, car_fich
                ; je SegundosIgual2
                cmp al, car_fich
                jb MelhorTempo
                cmp al, car_fich
                ja proxLinha

; SegundosIgual2:
                ; call          LeProxCaraterTOP
                ; jc            erro_ler                ; se carry é porque aconteceu um erro
                ; cmp           ax,0                                    ;EOF?       verifica se já estamos no fim do ficheiro
                ; je            fecha_ficheiro          ; se EOF fecha o ficheiro

                ; mov al, stringTempo[8]

                ; call TESTES

                ; cmp al, car_fich
                ; jb MelhorTempo
                ; cmp al, car_fich
                ; ja proxLinha

MelhorTempo:

                mov FlagCompararTempo, si



                ; mov ah, 09h
                ; lea dx, stringTempo
                ; int 21h

                ; mov ah, 02h
                ; MOV DL, '-'
                ; INT 21H

                ; mov ah, 0Ah
                ; lea dx, NomeUser
                ; xor si, si
                ; mov NomeUser[si], 12
                ; int           21h

                ; mov ah, 01h
                ; int           21h
                ; mov ah, 01h
                ; int           21h

                jmp FIM
```

```
proxLinha:
        ; mov ah, 01h
        ; int     21h
        call LeProxCaraterTOP
        jc      erro_ler        ; se carry é porque aconteceu um erro
        cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        je      fecha_ficheiro  ; se EOF fecha o ficheiro

        cmp car_fich, '|'
        jne proxLinha
        inc si

        call LeProxCaraterTOP
        jc      erro_ler        ; se carry é porque aconteceu um erro
        cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        je      fecha_ficheiro  ; se EOF fecha o ficheiro

        ; mov ah, 02h
        ; mov dl, car_fich
        ; int 21h
        ; mov ah, 01h
        ; int 21h

        call LeProxCaraterTOP
        jc      erro_ler        ; se carry é porque aconteceu um erro
        cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        je      fecha_ficheiro  ; se EOF fecha o ficheiro

        ; mov ah, 02h
        ; mov dl, car_fich
        ; int 21h
        ; mov ah, 01h
        ; int 21h

        jmp Ciclo

; MostraResto:
        ; call LeProxCaraterTOP
        ; jc      erro_ler        ; se carry é porque aconteceu um erro
        ; cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        ; je      fecha_ficheiro  ; se EOF fecha o ficheiro
        ; jmp MostraResto
erro_ler:
; mov ah, 01h
; int     21h
mov     ah,09h
lea     dx,Erro_Ler_Msg
int     21h
jmp                     fecha_ficheiro

; fim_ficheiro:
; cmp si, 10
; jae fecha_ficheiro

;senao

; mov ah, 09h
        ; lea dx, stringTempo
        ; int 21h

        ; mov ah, 02h
        ; MOV DL, '-'
        ; INT 21H

        ; mov ah, 0Ah
        ; lea dx, NomeUser
        ; xor si, si
        ; mov NomeUser[si], 12
        ; int     21h

        ; mov ah, 01h
        ; int     21h

; mov ah, 01h
; int     21h
fecha_ficheiro:
; mov al, 01h
; int 21h
cmp si, 10
jnbe FIM
mov FlagCompararTempo, si
```

```
FIM:
                    mov    ah,3eh
                    mov    bx,HandleFich
                    int    21h
                    ret

CompararTempo endp

MostraNovoTOP proc
        mov ah, 09h
        lea dx, msgNome
        int 21h

        cmp FlagCompararTempo, 0
        je NPedeNome
        mov ah, 0Ah
        lea dx, NomeUser
        ; xor di, di
        ; mov NomeUser[di], 12
        int         21h
        NPedeNome:
                    call APAGA_ECRAN
                    xor si, si
                    inc si

                    mov ah, 02h
                    add dL, 10
                    INT 21H

                    mov    ah,3dh                       ; vamos abrir ficheiro para leitura
                    mov    al,0                 ; tipo de ficheiro
                    lea    dx,FichTOP                   ; nome do ficheiro
                    int    21h                  ; abre para leitura
                    jc     erro_abrir           ; pode aconter erro a abrir o ficheiro
                    mov    HandleFich,ax        ; ax devolve o Handle para o ficheiro
                    jmp    Ciclo          ; depois de abero vamos ler o ficheiro

        erro_abrir:
                    mov    ah,09h
                    lea    dx,Erro_Open
                    int    21h
                    mov         flagFich, 0
                    ret

        Ciclo:

                    CMP SI, 11
                    je FIM


                    cmp si, FlagCompararTempo
                    je InserirDados
                    ; cmp dx, FlagCompararTempo
                    ; jne Continua
        Continua:
                    call LeProxCaraterTOP2
                    jc          erro_ler        ; se carry é porque aconteceu um erro
                    cmp         ax,0                    ;EOF?       verifica se já estamos no fim do ficheiro
                    je          FIM       ; se EOF fecha o ficheiro

                    CMP dl, '|'
                    je NOVALINHA

                    jmp Ciclo

        InserirDados:
                    cmp si, 1
                    jne NMostraNum

                    ; ----NUM TOP-----
                    mov         ax,si
                    MOV                 bl, 10
                    div         bl
                    add         al, 30h                                 ; Caracter Correspondente às dezenas
                    add                 ah,         30h                                     ; Caracter Correspondente às
unidades
                    MOV         stringSI[0],al                          ;
                    MOV         stringSI[1],ah
                    MOV         stringSI[2],'$'
```

```
        lea dx, stringSI
        mov ah, 09h
        int 21h

        call LeProxCaraterTOP
        jc      erro_ler        ; se carry é porque aconteceu um erro
        cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        je      FIM     ; se EOF fecha o ficheiro
        call LeProxCaraterTOP
        jc      erro_ler        ; se carry é porque aconteceu um erro
        cmp     ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        je      FIM     ; se EOF fecha o ficheiro
        ; call LeProxCaraterTOP
        ; jc    erro_ler        ; se carry é porque aconteceu um erro
        ; cmp   ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        ; je    FIM     ; se EOF fecha o ficheiro
        ; call LeProxCaraterTOP
        ; jc    erro_ler        ; se carry é porque aconteceu um erro
        ; cmp   ax,0                    ;EOF?   verifica se já estamos no fim do ficheiro
        ; je    FIM     ; se EOF fecha o ficheiro


NMostraNum:
        ;----TRAÇO-----
        mov ah, 02h
        MOV DL, '-'
        INT 21H

        ;----TEMPO DEMORADO-----
        mov ah, 09h
        lea dx, stringTempo
        int 21h

        ;----TRAÇO-----
        mov ah, 02h
        MOV DL, '-'
        INT 21H

        ;----MOSTRA NOME-----
        xor di, di
        inc di
        inc di
        mov ch, 0
        mov cl, NomeUser[1]
        teste:
                mov ah, 02h

                MOV DL, NomeUser[di]
                INT 21H
                inc di
        loop teste

        mov ah, 02h
        MOV DL, '|'
        INT 21H



        ; mov dl,10
        ; mov ah,2h
        ; int 21h
        mov dl,10
        mov ah,2h
        int 21h

        inc si

        mov     ax,si
        MOV             bl, 10
        div     bl
        add     al, 30h
        add             ah,     30h
        MOV     stringSI[0],al
        MOV     stringSI[1],ah
        MOV     stringSI[2],'$'

        lea dx, stringSI
        mov ah, 09h
        int 21h
```

```
		jmp Ciclo


	NOVALINHA:
		inc numLinhasTOP

		mov dl,10
		mov ah,2h
		int 21h
		inc si
		;------despreza num TOP-----------
		; call LeProxCaraterTOP
		; jc		erro_ler		; se carry é porque aconteceu um erro
		; cmp		ax,0			;EOF?	verifica se já estamos no fim do ficheiro
		; je		FIM	; se EOF fecha o ficheiro
		call LeProxCaraterTOP
		jc		erro_ler		; se carry é porque aconteceu um erro
		cmp		ax,0			;EOF?	verifica se já estamos no fim do ficheiro
		je		FIM	; se EOF fecha o ficheiro
		call LeProxCaraterTOP
		jc		erro_ler		; se carry é porque aconteceu um erro
		cmp		ax,0			;EOF?	verifica se já estamos no fim do ficheiro
		je		FIM	; se EOF fecha o ficheiro
		call LeProxCaraterTOP
		jc		erro_ler		; se carry é porque aconteceu um erro
		cmp		ax,0			;EOF?	verifica se já estamos no fim do ficheiro
		je		FIM	; se EOF fecha o ficheiro
		;---------------------------------
		mov		ax,si
		MOV			bl, 10
		div		bl
		add		al, 30h				; Caracter Correspondente às dezenas
		add			ah,		30h				; Caracter Correspondente às
unidades
		MOV		stringSI[0],al				;
		MOV		stringSI[1],ah
		MOV		stringSI[2],'$'

		lea dx, stringSI
		mov ah, 09h
		int 21h

		JMP Ciclo


	erro_ler:

		mov	ah,09h
		lea	dx,Erro_Ler_Msg
		int	21h
		; mov ah, 01h
		; int		21h
		jmp			FIM

	FIM:

	cmp FlagCompararTempo, si
	jne NMostraNome
	inc numLinhasTOP
	;----NUM TOP-----
		mov		ax,si
		MOV			bl, 10
		div		bl
		add		al, 30h				; Caracter Correspondente às dezenas
		add			ah,		30h				; Caracter Correspondente às
unidades
		MOV		stringSI[0],al				;
		MOV		stringSI[1],ah
		MOV		stringSI[2],'$'

		lea dx, stringSI
		mov ah, 09h
		int 21h

		;----TRAÇO-----
		mov ah, 02h
		MOV DL, '-'
		INT 21H

		;----TEMPO DEMORADO-----
		mov ah, 09h
```

```
                lea dx, stringTempo
                int 21h

                ;----TRAÇO-----
                mov ah, 02h
                MOV DL, '-'
                INT 21H

                ;----MOSTRA NOME-----
                xor di, di
                inc di
                inc di
                mov ch, 0
                mov cl, NomeUser[1]
                teste1:
                        mov ah, 02h

                        MOV DL, NomeUser[di]
                        INT 21H
                        inc di
                loop teste1

                mov ah, 02h
                MOV DL, '|'
                INT 21H

NMostraNome:


                mov ah, 01h
                int             21h
                mov     ah,3eh
                mov     bx,HandleFich
                int     21h
                ret

MostraNovoTOP endp

GuardaTOP proc
        xor si, si
        xor bl, bl
        xor cl, cl
        inc bl
        inc cl
        inc numLinhasTOP
        inc numLinhasTOP

        i:

                cmp bl, numLinhasTOP
                je sai

                xor cl, cl
                j:
                        cmp cl, 25
                        je avancaI

                        goto_xy cl, bl
                        mov             ah, 08h
                        mov                             bh,0
                        int                             10h



                        cmp al, '|'
                        je avancaI


                        moveBuffer:

                                        mov bufferTOP[si], al
                                        inc si
                                        inc cl
                jmp j

        avancaI:
                inc bl

                mov al, '|'
                mov bufferTOP[si], al
                INC SI
```

```
                mov al, 10
                mov bufferTOP[si], al
                INC SI

                jmp i


        sai:
                ; mov al, '|'
                ; mov bufferTOP[si], al
                ; INC SI

                ; mov al, 10



        mov     ah, 3ch                 ; abrir ficheiro para escrita
        mov     cx, 00H                 ; tipo de ficheiro
        lea     dx, FichTOP             ; dx contem endereco do nome do ficheiro
        int     21h                             ; abre efectivamente e AX vai ficar com o Handle do ficheiro
        jnc     escreve                 ; se não acontecer erro vai vamos escrever

        mov     ah, 09h                 ; Aconteceu erro na leitura
        lea     dx, msgErrorCreate ;//mete endereco da msg em dx
        int     21h ; //carater lido em AL

        jmp     fim

escreve:
        mov     bx, ax                  ; para escrever BX deve conter o Handle
        mov     ah, 40h                 ; indica que vamos escrever

        lea     dx, bufferTOP                   ; Vamos escrever o que estiver no endereço DX
        mov     cx, si          ; vamos escrever multiplos bytes duma vez só
        int     21h                             ; faz a escrita
        jnc     close                   ; se não acontecer erro fecha o ficheiro

        mov     ah, 09h
        lea     dx, msgErrorWrite
        int     21h
close:
        mov     ah,3eh                  ; indica que vamos fechar
        int     21h                             ; fecha mesmo
        jnc     fim                             ; se não acontecer erro termina

        mov     ah, 09h
        lea     dx, msgErrorClose
        int     21h


fim:
        ret


GuardaTOP endp

FimJogo proc
        call APAGA_ECRAN
        call CALCULA_TEMPO
        call CompararTempo
        call MostraNovoTOP
        call GuardaTOP
        ;call MostraTOP

        ;call GUARDA_TOP
        ;calcular tempo
        ;ler ficheiro e comparar com o tempo
        ;se horas <= + se min <= + s <= + se linha a ler <= 10
        ret
FimJogo endp


opJogar proc
        call APAGA_ECRAN

        cmp auxConfigMaze, 1
        je Maze
        cmp auxConfigMaze, 2
        je MazeFich
        cmp auxConfigMaze, 3
```

```
            je CriaMaze
Maze:
            call MostraMaze
            jmp JOGAR

MazeFich:
            call MostraMazeFich
            ; cmp flagFich, 0
            ; je FimOpJogar
            jmp JOGAR


CriaMaze:

            call MostraMazeFich
            ; cmp flagFich, 0
            ; je FimOpJogar
            jmp JOGAR


JOGAR:
            MOV POSy, 19
            MOV POSya, 19
            call Ler_TEMPO

            cmp FlagBonus, 0
            je j1
            jmp j2
            j1:
                        call JOGO
                        jmp done
            j2:
                        call JOGO2
            done:
            cmp flagAcabouJogo, 0 ;se acabou jogo
            je FimOpJogar

            call FimJogo
            ;call CALCULA_TEMPO
            ;call GUARDA_TOP

FimOpJogar:         ret
opJogar     endp


criarMaze proc
            ; mov al, 01h
            ; int 21h

            call        apaga_ecran
            call MostraMazeFich

            mov POSx, 5
            mov POSy, 5



                        ; mov dl, 219
                        ; mov ah, 02h
                        ; mov cx, 40
                        ; preencheCima:
                                    ; goto_xy     cl,1
                                    ; mov dl, 219
                                    ; mov ah, 02h
                                    ; int 21h
                                    ; goto_xy     cl,19
                                    ; mov dl, 219
                                    ; mov ah, 02h
                                    ; int 21h
                                    ; goto_xy     cl,20
                                    ; mov dl, 219
                                    ; mov ah, 02h
                                    ; int 21h
                        ; loop preencheCima

                        ; mov cx, 20
                        ; preencheLados:
                                    ; goto_xy     0,cl
                                    ; mov dl, 219
                                    ; mov ah, 02h
```

```
                    ; int 21h
                    ; goto_xy     40,cl
                    ; mov dl, 219
                    ; mov ah, 02h
                    ; int 21h
                ; loop preencheLados

                goto_xy     21,1
                mov ah, 02h
                mov dl, 'X'
                int 21h

                goto_xy     21,19
                mov ah, 02h
                mov dl, 'I'
                int 21h

                ;Obter a posição
                ; dec                POSy
                ; dec                POSx

                MOV Car, 219

CICLO:  goto_xy     POSx,POSy
IMPRIME: mov                ah, 02h
                mov                 dl, Car
                int                 21H
                goto_xy     POSx,POSy

                call                LE_TECLA
                cmp                 ah, 1
                je                  CIMA
                CMP                 AL, 27              ; ESCAPE
                JE                  Guarda

ZERO:   CMP                 AL, 48              ; Tecla 0
                JNE                 UM
                mov                 Car, 32             ;ESPAÇO
                jmp                 CICLO

UM:             CMP                 AL, 49              ; Tecla 1
                JNE                 CIMA
                mov                 Car, 219            ;Caracter CHEIO
                jmp                 CICLO

CIMA:   cmp                 al,48h
                jne                 BAIXO
                cmp POSy, 3
                jnge CICLO
                dec                 POSy                ;cima

                jmp                 CICLO

BAIXO:  cmp                 al,50h
                jne                 ESQUERDA
                cmp POSy, 17
                jnle CICLO
                inc                 POSy                ;Baixo
                jmp                 CICLO

ESQUERDA:
                cmp                 al,4Bh
                jne                 DIREITA
                cmp POSx, 2
                jnge CICLO
                dec                 POSx                ;Esquerda
                jmp                 CICLO

DIREITA:
                cmp                 al,4Dh
                jne                 CICLO
                cmp POSx, 38
                jnle CICLO
                inc                 POSx                ;Direita
                jmp                 CICLO

Guarda:

        xor si, si
        xor bl, bl
        xor cl, cl
```

```
                inc cl
                i:
                        cmp bl, 21
                        je sai

                        xor cl, cl
                        inc cl
                        j:
                                cmp cl, 42
                                je avancaI

                                goto_xy cl, bl
                                mov     ah, 08h
                                mov                     bh,0                    ; numero da página
                                int                     10h


                                cmp al, 219
                                je parede

                                cmp al, ' '
                                je vazio

                                cmp al, 'I'
                                je moveBuffer

                                cmp al, 'X'
                                je moveBuffer

                                parede:
                                        mov al, 30h
                                        jmp moveBuffer

                                vazio:
                                        mov al, 31h
                                        jmp moveBuffer



                                moveBuffer:

                                        mov buffer[si], al
                                        inc si
                                        inc cl
                        jmp j

                avancaI:
                        inc bl
                        mov al, '|'
                        mov buffer[si], al
                        INC SI

                        mov al, 10
                        mov buffer[si], al
                        INC SI

                        jmp i


                sai:
                        mov al, '|'
                        mov buffer[si], al
                        INC SI

                        mov al, 10



                mov     ah, 3ch                         ; abrir ficheiro para escrita
                mov     cx, 00H                         ; tipo de ficheiro
                lea     dx, Fich2                       ; dx contem endereco do nome do ficheiro
                int     21h                                     ; abre efectivamente e AX vai ficar com o Handle do ficheiro
                jnc     escreve                         ; se não acontecer erro vai vamos escrever

                mov     ah, 09h                         ; Aconteceu erro na leitura
                lea     dx, msgErrorCreate ;//mete endereco da msg em dx
                int     21h ; //carater lido em AL

                jmp     fim

escreve:
```

```
        mov     bx, ax                  ; para escrever BX deve conter o Handle
        mov     ah, 40h                 ; indica que vamos escrever

        lea     dx, buffer              ; Vamos escrever o que estiver no endereço DX
        mov     cx, 902                 ; vamos escrever multiplos bytes duma vez só
        int     21h                             ; faz a escrita
        jnc     close                           ; se não acontecer erro fecha o ficheiro

        mov     ah, 09h
        lea     dx, msgErrorWrite
        int     21h
close:
        mov     ah,3eh                  ; indica que vamos fechar
        int     21h                             ; fecha mesmo
        jnc     fim                             ; se não acontecer erro termina

        mov     ah, 09h
        lea     dx, msgErrorClose
        int     21h


fim:
        ret
criarMaze endp


opTop   proc
        call apaga_ecran
        mov   ah,3dh                            ; vamos abrir ficheiro para leitura
        mov   al,0                      ; tipo de ficheiro
        lea   dx,FichTOP
        int   21h                       ; abre para leitura
        jc    erro_abrir                ; pode aconter erro a abrir o ficheiro
        mov   HandleFich,ax             ; ax devolve o Handle para o ficheiro
        jmp   ler_ciclo                 ; depois de abero vamos ler o ficheiro

        erro_abrir:
                mov   ah,09h
                lea   dx,Erro_Open
                int   21h
                mov         flagFich, 0
                ret
        ler_ciclo:
                mov   ah,3fh                            ; indica que vai ser lido um ficheiro
                mov   bx,HandleFich             ; bx deve conter o Handle do ficheiro previamente aberto
                mov   cx,1                      ; numero de bytes a ler
                lea   dx,car_fich               ; vai ler para o local de memoria apontado por dx (car_fich)
                int   21h                               ; faz efectivamente a leitura
                jc              erro_ler        ; se carry é porque aconteceu um erro
                cmp             ax,0                    ;EOF?       verifica se já estamos no fim do ficheiro
                je              fecha_ficheiro  ; se EOF fecha o ficheiro

        MOSTRA:
                mov   ah,02h                            ; coloca o caracter no ecran
                mov             dl,car_fich             ; este é o caracter a enviar para o ecran
                int             21h                     ; imprime no ecran
                jmp             ler_ciclo       ; continua a ler o ficheiro

        erro_ler:
                mov   ah,09h
                lea   dx,Erro_Ler_Msg
                int   21h

        fecha_ficheiro:                                                 ; vamos fechar o ficheiro
                mov   ah,3eh
                mov   bx,HandleFich
                int   21h

                mov ah, 01h
                int 21h

                ; mov         flagFich, 0
                ret

                mov   ah,09h                            ; o ficheiro pode não fechar correctamente
                lea   dx,Erro_Close
                Int   21h


        ret
opTop   endp
```

```
MostraMenuConfig        proc
                call APAGA_ECRAN

                mov ah, 09h
                lea dx, MenuConfigMaze
                int 21h

                ret
MostraMenuConfig        endp

opMazeConfig            proc
                call            MostraMenuConfig

                mov             ah,1h
                int                             21h

                cmp                             al,'0'
                je FIM
                cmp                             al,'1'
                je OP1
                cmp                             al,'2'
                je OP2
                cmp                             al,'3'
                je OP3

                OP1:
                                mov auxConfigMaze, 1
                                jmp FIM
                OP2:
                                mov auxConfigMaze, 2
                                jmp FIM
                OP3:
                                mov auxConfigMaze, 3
                                call criarMaze
                                jmp FIM


FIM:            ret
opMazeConfig            endp

; opAtivarBonus proc
                ; not FlagBonus
; opAtivarBonus endp



Main  proc

                mov                             ax, dseg
                mov                             ds,ax
                mov                             ax,0B800h
                mov                             es,ax

                CICLO:
                                call            MostraMenu
                                mov             ah,01h
                                int                             21h

                                cmp                             al,'0'
                                je FIM
                                cmp                             al,'1'
                                je JOGAR
                                cmp                             al,'2'
                                je TOP
                                cmp                             al,'3'
                                je CONF_MAZE
                                cmp                             al,'4'
                                je ativarBonus
                jmp CICLO

                JOGAR:
                                call            opJogar
                                jmp CICLO
                TOP:
                                call            opTop
                                jmp CICLO
                CONF_MAZE:
                                call            opMazeConfig
```

```
                jmp CICLO
        ativarBonus:
                not FlagBonus
                jmp CICLO
        FIM:
                mov             ah,4CH
                INT             21H
Main    endp
Cseg    ends
end     Main
```