Labirinto - Assembly 8086

Trabalho Laboratorial 2016/2017

Tecnologias e Arquiteturas de Computadores

Trabalho realizado por:

Diogo Miguel Branco Santos - 21210835

João Filipe Lopes Gaspar - 21200069

ISEC ENGENHARIA

Índice

Introdução

Este trabalho prático foi-nos proposto com o objetivo de desenvolver em linguagem Assembly um jogo que consiste em percorrer um determinado labirinto de forma a atingir uma determinada meta no menor tempo possível.

O jogo é dado por terminado, com vitória, quando a meta for atingida. Quando se tratar de uma vitória é apresentado o tempo gasto pelo jogador e a sua posição no top 10 se for o caso.

Com este trabalho pensamos que conseguimos adquirir as competências necessárias para a aprovação da disciplina e, mais tarde, poder-mos vir a desenvolver software que requira melhor *performace* coisa que o Assembly, sendo uma linguagem de baixo nível, nos presenteia.

Estrutura

Estrutura do Código

O código está estruturado como ilustra a ***Fig. 1***.

As variáveis são declaradas no início do programa. Estas são úteis para guardar conteúdos de processos futuros tais como, apresentação de conteúdos no ecrã, escrita e leitura em ficheiros, mensagens de erro, tratamento do tempo decorrido, entre outras.

De seguida temos algumas macros tais como ***GOTO_XY*** e ***MOSTRA*** que nos ajuda a posicionar o cursor no ecrã e a imprimir caracteres no ecrã respetivamente.

Em terceiro e quarto lugar da pilha de código temos os procedimentos. Através dos procedimentos é-nos possível organizar de uma forma mais elegante o imenso código que esta linguagem nos exige e este pode ser chamado entre procedimentos e finalmente pela última camada da pilha, a Main.

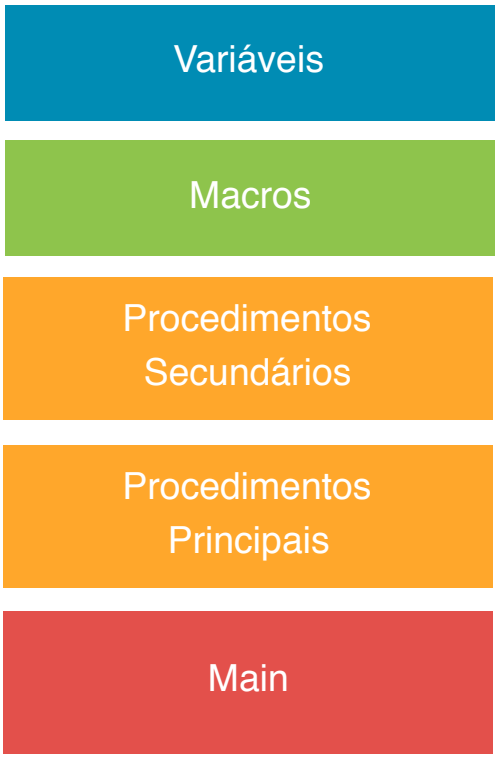

Fig. 1 - Estrutura do código

A Main apenas chama alguns procedimentos principais para a inicialização do programa.

Objetivos

- Aprender os conceitos básicos da linguagem assembly;
- Trabalhar com a memória de video;
- Manipular ficheiros;
- Trabalhar com Macros;
- Trabalhar com Procedimentos;

Estes são alguns dos objetivos que pretendemos alcançar com o desenvolvimento deste trabalho prático.

Funcionamento do Programa

Assim que o programa é executado, é apresentado ao utilizador o menu principal(***Fig. 2***).

Através deste menu é possível o utilizador aceder diretamente ao jogo, visualizar o dez melhores tempos, configurar o labirinto, ou abandonar o jogo.

Fig. 2 - Menu principal

A ***Fig. 3*** mostra as opções de configuração do labirinto ao utilizador. Todas elas permitem que o utilizador faça alterações significativas no jogo sem ter de sair ou reiniciar o programa.

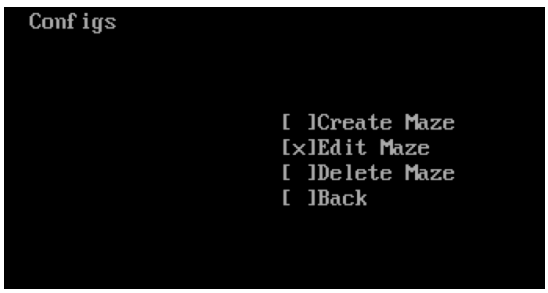

Fig. 3 - Menu de configuração

Play - O jogo é inicializado com o labirinto previamente escolhido ou por default;

Top 10 - Mostra os 10 melhores jogadores e os seus tempos respetivamente(Não implementado);

Maze Config - Salta para o Menu de configuração(***Fig. 3***);

Exit - Sai do programa;

Bonus - Carrega jogo em modo Bonus, o utilizador diz logo de inicio todas as direções que o avatar deve tomar;

Create Maze - O utilizador desenha o seu próprio labirinto;

Load Default Maze - Carrega o labirinto "MAZE.TXT";

Load User Maze - O utilizador escolhe o labirinto que quer jogar;

Edit Maze - O utilizador pode modificar um labirinto previamente criado;

Delete Maze - O utilizador pode apagar labirintos do programa.

Back - Volta ao menu principal(***Fig. 2***).

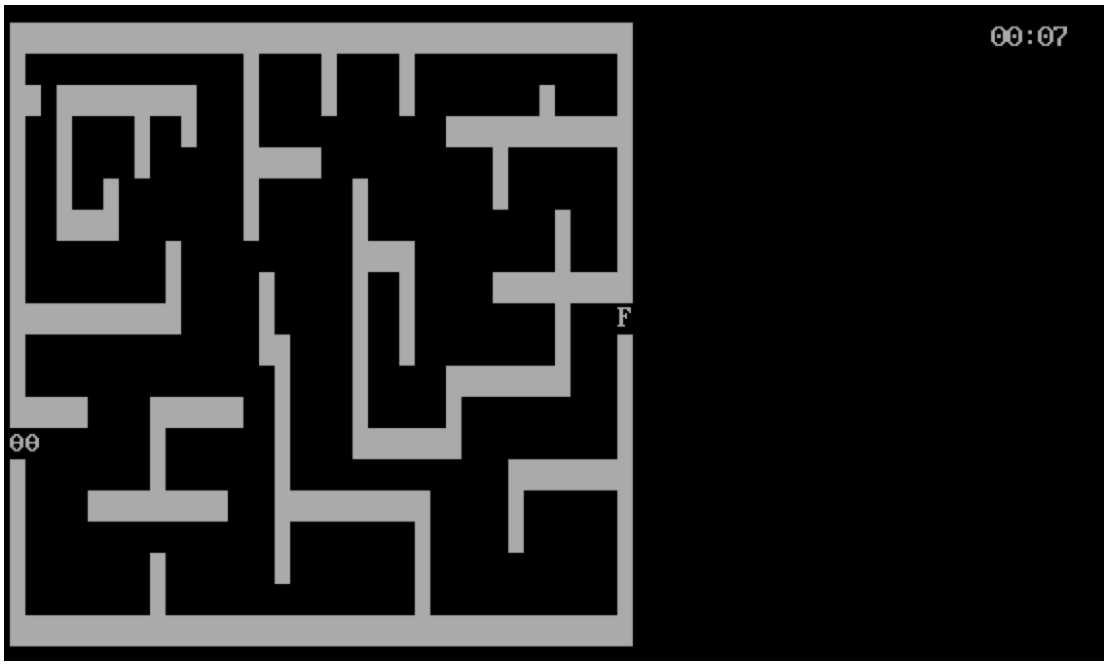

Fig. 4 - Modo Play

Algoritmos

Ler Tempo

1. Obtém a hora do sistema com a interrupção “2C”;
2. Guarda os segundos;
3. Guarda os minutos;
4. Guarda as horas.

Trata Horas

1. Chama o procedimento “Ler Tempo”;
2. Obtém tempo atual;
3. Verifica se os segundos mudaram desde a última leitura;
    1. Se Sim incrementa os segundos;
    2. Se os segundos chegarem ao limite:
        1. Incrementa a segunda variável segundos e assim sucessivamente até à última variável minutos;

Le Tecla Clock

1. Igual ao “Le Tecla”;
2. Quando nenhuma tecla é pressionada chama o procedimento “Trata Horas”

Pede Nome F

1. Apaga o ecrã;
2. Move o cursor para a posição 1,1;
3. Grava caracteres introduzidos pelo utilizador numa string;

4. Chama “Le Tecla”;
5. Processa teclas especiais(backspace, enter);
    1. Processa caracteres e imprime-os na consola;
6. Grava nome do ficheiro numa string com extensão “.txt”;

Edit Ficheiro

1. Guarda o conteúdo do ecrã;
2. Edita o conteúdo do ecrã;
3. Grava no ficheiro;

F Name Ptr To Fich

1. Passa o “fnameptr” para um ficheiro de maneira a não ser perdido da próxima vez que se abrir o programa;
2. O “fnameptr” é um ponteiro para sabermos onde colocar o próximo nome de ficheiro no vector “fname";
3. O “fname" é um vetor que contem todos os nomes dos labirintos criados pelo usuário.

F Name To Fich

1. Passa o “fname" para um ficheiro de maneira a não ser perdido da próxima vez que se abrir o programa;
2. O “fname" um vetor que contem todos os nomes dos labirintos criados pelo usuário.

F Name Nto Fich

1. Passa o “fnamen” para um ficheiro de maneira a não ser perdido da próxima vez que se abrir o programa.O “fnamen" o número de elementos do “fname”;
2. “fname" um vetor que contem todos os nomes dos labirintos criados pelo usuário.

Fich To F Name

1. O procedimento inverso do “FnameToFich”.

Fich To F Namen

1. O procedimento inverso do “FnameNtoFich”.

Fich To F Name Ptr

1. O procedimento inverso do “FnamePtrToFich”.

Menu Desenhar

1. Apaga o ecrã;
2. Limita espaço de desenho;
3. Permite troca de caracter(0, 1, 2,3);
4. Lê seta e faz desenho.

Menu Edit Maze

1. Apaga o ecrã;
2. Espera que o utilizador escreva algo no ecrã até que a tecla “Enter” seja pressionada;
3. Percorre ecrã;
4. Guarda no ficheiro.

Menu Load U Maze

1. O utilizador escolhe um labirinto disponível;
    1. Se não existir nenhum pode retroceder para o menu anterior.
2. Depois de escolhido o nome do ficheiro onde está guardado fica atribuído a uma variável para posteriormente seja possível correr o jogo a partir desta.

Menu Delete Maze

1. O utilizador escolhe um labirinto disponível;
    1. Se não existir nenhum pode retroceder para o menu anterior.
2. O Ficheiro é apagado do sistema.

Menu Config

1. Imprime o menu de configuração(carregado do ficheiro de texto);
2. O utilizador escolhe uma opção e dependendo da sua escolha é enviado para os procedimentos correspondentes;

Play

1. Carrega o jogo com o labirinto default ou com um atribuído pelo utilizador;
2. O Le Tecla Clock é responsável por ler as setas para poder mover o avatar e quando estas não são premidas o cronometro é incrementado e apresentado ao utilizador no canto superior direito;
3. Deteta parede e impede que a ultrapasse;
4. Quando chega ao fim ou desiste é apresentado o cronometro com o tempo total.

Menu

1. Apresenta menu principal(carregado do ficheiro de texto);
2. Deteta posição do ‘x’ e impede que ele sai da área pretendida;
3. Chama o procedimento respetivo.

Conclusão

Com este trabalho aprendemos como é desenvolver software a um nível mais baixo e as suas vantagens e desvantagens. Uma das desvantagens é a gestão do tempo, talvez por estarmos habituados a pensar em algoritmos de um nível superior foi, em certos casos, complicado fazer uma boa gestão do tempo.

Contudo pensamos que o balanço do trabalho foi positivo e que no futuro iremos ter um maior à-vontade tanto com a linguagem em si, como também, a forma distinta de pensar na implementação de algoritmos e resolução de problemas.

Anexos

Listagem do programa

```
;--------------------------------------------------------------------------
;
;    TRABALHO PRATICO - TECNOLOGIAS e ARQUITECTURAS de COMPUTADORES
;
;    ANO LECTIVO 2016/2017
;
; TRABALHO REALIZADO POR:
;
; DIOGO MIGUEL BRANCO SANTOS - 21210835
; JOÃO FILIPE LOPES GASPAR - 21200069


.8086
.model small
.stack 2048h

dseg    segment para public 'data'

coordenadas db 600 dup(?) ;variavel para guardar as cooredenadas do play 2
  mazedim dw 0 ;dimensao do labirinto, para imprimir
  FichDefaultMaze  db    'maze.txt',0
  fnameptrFich db 'mnamespf.txt',0 ;nome do ficheiro que contem o ponteiro para
     o ultimo elemento do array com os nomes dos labirintos
  fnamenFich db 'mnamesnf.txt',0 ;nome do ficheiro que contem o n de elementos
     do array com os nomes dos labirintos
  mazeNamesFich db 'mnamesf.txt',0 ;nome do ficheiro que contem os nomes dos
     labirintos
  fname db 390 dup(?) ;vetor de strings, 30*13  30strings de 12+1 char
  fnameptr dw 0 ;ponteiro para o fname
  fnameop db 0 ; fname option, começa a 0 que é o default, é a o labirinto a
     carregar no play
  fnameditop db 0 ; o labirinto a editar
  fnamen db 0     ; n elementos do fname
  p db 0 ; ponteiro
  pant db 0
  zeros db 0
  lastzero dw 0
  fnameopen db 13 dup(?)
  fnamedit db 13 dup(?)
  Cor                  db        7  ; Guarda os atributos de cor do caracter
  POSya                    db    5  ; Posio anterior de y
  POSxa                    db    10 ; Posio anterior de x

  POSyp db 0 ; proxima posicao y
  POSxp db 0 ; proxima posicao

  MazeX            db    1
  MazeY            db    1
```

```
  ;------------------VARIAVEIS DO MAZE1----------------------
  chIn db 65
  oitoch db 1
   quatroch db 1; limitador para inserir coordenadas
  string1 db "Type maze name:", 0
  string2 db "Choose your maze:", 0
  string3 db "Default Maze has been choosen!", 0
  string4 db "Delete Maze:", 0
  string5 db "Edit Maze:", 0
    string6 db "Insira as coordenadas:", 0

  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$'
  FichMenu                db      'menu.txt',0
  FichMazeConfig db     'mconfig.txt',0
  HandleFich      dw      0
  car_fich        db      ?
  mazenamesfichErr db 'erro na gravacao dos labirintos$'

  string    db    "Teste prtico de T.I",0
  Car       db    32
  POSy            db    18    ; a linha pode ir de [1 .. 25]
  POSx            db    33    ; POSx pode ir [1..80]

  GChar db 32 ; variavel para guardar o caracter


  fhandle dw      0
  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!$"



  Buffer db 2000 dup(0) ; Inicializa um array de 2000 posições(80*25) a 0 para
     depois serem gravados no ficheiro

;------------------VARIAVEIS DO RELOGIO----------------------

  STR12                 DB          "            "    ; String para 12 digitos
  NUMERO          DB          "                  $",      ; String destinada
     a guardar o número lido

  NUM_SP          db          "                  $"       ; PAra apagar zona
     de ecran
  DDMMAAAA        db          "                   "

  Horas                 dw          0                     ; Vai guardar a
     HORA actual
```

```
  Minutos         dw          0                       ; Vai guardar os minutos
     actuais
  Segundos        dw          0                       ; Vai guardar os segundos
     actuais
  Old_seg         dw          0                       ; Guarda os últimos
     segundos que foram lidos

  POSy2     db    10    ; a linha pode ir de [1 .. 25]
  POSx2     db    40    ; POSx pode ir [1..80]
  NUMDIG    db    0     ; controla o numero de digitos do numero lido
  MAXDIG    db    4     ; Constante que define o numero MAXIMO de digitos a ser
     aceite

  strTEMPOTOTAL db "Tempo Total:$"

  segundos1 db 47; contador de segundos
  segundos2 db 48; contador de segundos pos2

  minutos1 db 48; contador de minutos
  minutos2 db 48; contador de inutos pos2

  strsegundos1 db "      $" ;string para ser impressa
  strsegundos2 db "      $" ;string para ser impressa

  strminutos1 db "       $" ;string para ser impressa
  strminutos2 db "       $" ;string para ser impressa

  strzero db "0$"
  strdoispontos db ":$"

dseg    ends

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

;*******************************INICIO DO
     PROGRAMA*****************************

;MACROS
GOTO_XY MACRO           POSx,POSy
    mov           ah,02h
      mov         bh,0        ; numero da pgina
      mov         dl,POSx
      mov         dh,POSy
      int         10h
ENDM

MOSTRA MACRO STR
  mov ah,09H
  lea dx,STR
  int 21H
ENDM
```

```
GRAVATEMPO MACRO NOME, TEMP

; por implementar

ENDM
;MACROS

apaga_ecran proc
      xor         bx,bx
      mov         cx,25*80

apaga:
      mov         byte ptr es:[bx],' '
      mov         byte ptr es:[bx+1],7
      inc         bx
      inc   bx
      loop  apaga
      ret
apaga_ecran endp

;*************************************************************************
     **
;*************************************************************************
     **
; HORAS  - LE Hora DO SISTEMA E COLOCA em tres variaveis (Horas, Minutos,
     Segundos)
; CH - Horas, CL - Minutos, DH - Segundos
;*************************************************************************
     **

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

;********************************************************************************
     **
;********************************************************************************
     **
; Imprime o tempo e a data no monitor

Trata_Horas PROC

  pushf
  push ax
  push bx
  push cx
  push dx

  call      Ler_TEMPO                    ; Horas MINUTOS e segundos do Sistema

  mov ax, segundos
  cmp ax, Old_seg             ; Verifica se os segundos mudaram desde a ultima
     leitura
  je fim_horas                ; Se a hora não mudou desde a última leitura sai.
  mov Old_seg, ax             ; Se segundos são diferentes actualiza informação
     do tempo

contasegundos:
  inc segundos1
  cmp segundos1, 58
    je contasegundos2
  mov bl, segundos1
  mov strsegundos1, bl ; isto está bem
  goto_xy 68,1 ; canto superior direito
  mostra strsegundos1
  jmp fim_horas

contasegundos2:
  inc segundos2
  cmp segundos2, 54
    je contaminutos
  mov segundos1,47 ; repoe os segundos1
  mov bl, segundos2
  mov strsegundos2, bl ; isto está bem
  goto_xy 67,1 ; canto superior direito
  mostra strsegundos2
  jmp contasegundos

contaminutos:
```

```
  inc minutos1
  cmp minutos1, 58
    je contaminutos2
  mov segundos1, 47 ; repoe segundos1
  mov segundos2, 48 ; repoe segundos2
  mov bl, minutos1
  mov strminutos1, bl
  goto_xy 65,1
  mostra strminutos1
  goto_xy 66,1
  mostra strdoispontos
  goto_xy 67,1
  mostra segundos2
  goto_xy 68,1
  mostra segundos1
  jmp contasegundos

contaminutos2:
  inc minutos2
  mov minutos1, 47 ; repoe minutos1
  mov segundos1, 47 ; repoe segundos1
  mov segundos2, 48 ; repoe segundos2
  mov bl, minutos2
  mov strminutos2, bl
  goto_xy 64,1
  mostra strminutos2
  goto_xy 65,1
  mostra strminutos1
  goto_xy 66,1
  mostra strdoispontos
  goto_xy 67,1
  mostra segundos2
  goto_xy 68,1
  mostra segundos1
  jmp contaminutos


fim_horas:
      goto_xy     POSx,POSy               ; Volta a colocar o cursor onde
     estava antes de actualizar as horas

    popf
    pop dx
    pop cx
    pop bx
    pop ax
      ret

Trata_Horas ENDP

GUARDA_ECRA PROC   ; ------>GUARDA ECRA NO BUFFER<------
      xor bx,bx
      xor si,si
```

```
      mov cx,25*80

copia:
      mov al, byte ptr es:[bx]
      mov   ah,   byte ptr es:[bx+1]
      mov Buffer[si], al
      mov Buffer[si+1], ah
      inc bx
      inc bx
      inc si
      inc si
      loop copia
      ret

GUARDA_ECRA ENDP  ; ------>GUARDA ECRA NO BUFFER<------

LE_TECLA    PROC   ; ------>LE UMA TECLA<------

     mov          ah,08h
     int          21h
     mov          ah,0
     cmp          al,0
      jne         SAI_TECLA
     mov          ah, 08h
     int          21h
     mov          ah,1

sai_tecla:
  ret

LE_TECLA    ENDP   ; ------>LE UMA TECLA<------

LE_TECLA_CLOCK    PROC   ; ------>LE UMA TECLA CLOCK<------

sem_tecla:
     call Trata_Horas
     mov    ah,0BH
     int 21h
     cmp al,0
     je     sem_tecla

     goto_xy      POSx,POSy

     mov    ah,08H
     int    21H
     mov    ah,0
     cmp    al,0
       jne  SAI_TECLA
     mov    ah, 08H
     int    21H
     mov    ah,1

sai_tecla:
```

```
  ret

LE_TECLA_CLOCK    ENDP   ; ------>LE UMA TECLA<------

PEDENOMEF PROC  ; ------>PEDE NOME PARA SER ATRIBUIDO AO FICHEIRO<------
     mov          ax,0B800h
     mov          es,ax
     call   apaga_ecran

     mov posy, 1
     mov posx, 1
     goto_xy      POSx,POSy
     xor si, si ;si=0

puts:
     cmp string1[si], 0
      je scanf
     mov ah, 02h
     mov dl, string1[si]
     int 21h
     inc si
     jmp puts

scanf:
     mov posy, 2
     mov posx, 1
     goto_xy      POSx,POSy
     mov si, fnameptr
     mov oitoch, 1

     CICLO:
      mov chIn, 65 ;
      call  LE_TECLA
      cmp   ah, 1
            je    ciclo
      cmp al, 13        ; enter--------------------------------------
            jne backspace
      mov fname[si], 46 ;colocar a terminaçao do fich no fim do vetor
      inc si
      mov fname[si], 116 ;t
      inc si
      mov fname[si], 120 ;x
      inc si
      mov fname[si], 116 ;t
      inc si
      mov fname[si], 0
      inc si
      mov fnameptr, si
      inc fnamen ; incrementar 1 elemento
      ret

     backspace:
      cmp         AL, 8  ;BACKSPACE
```

```
 jne limitesup
 cmp oitoch, 2
 jb ciclo   ; so continua se estiver dentro do lim inferior
 mov         ah, 02h
 mov         dl, 8 ;backspace pa andar pa traz
 int         21H
 mov         ah, 02h
 mov         dl, 32 ;espaço pa limpar
 int         21H
 mov         ah, 02h
 mov         dl, 8 ;backspace pa andar pa traz
 int         21H
 dec oitoch
 dec si ;anda pa traz no nomefich
 jmp ciclo

limitesup:
 cmp oitoch, 8
 ja ciclo    ; so continua se estiver dentro do lim superior

tryagain:
 cmp al, chIn
 jne   incrementar
 mov bl, chIn
 mov   Car,  bl
 mov   ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
 mov   dl, Car
 int   21H
 inc oitoch
 mov fname[si], bl  ;coloca na variavel
 inc si
 jmp   CICLO

incrementar:
 cmp chIn, 90
       je minusculas
 inc chIn
 jmp tryagain

minusculas:
 mov   chIn, 97

tryagainmin:
 cmp   al, chIn
       jne   incrementarmin
 mov bl, chIn
 mov   Car,  bl
 mov   ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
 mov   dl, Car
 int   21H
 inc oitoch
 mov fname[si], bl  ;coloca na variavel
 inc si
```

```
      jmp   CICLO

     incrementarmin:
      cmp chin, 122
      je ciclo
      inc chIn
      jmp tryagainmin

PEDENOMEF ENDP ; ------>PEDE NOME PARA SER ATRIBUIDO AO FICHEIRO<------

pedecoordenadas proc ;------>PEDE as coordenadas para o play2<------

     mov posy, 22
     mov posx, 1
     goto_xy      POSx,POSy
     xor si, si ;si=0

puts:
     cmp string6[si], 0
      je scanf
     mov ah, 02h
     mov dl, string6[si]
     int 21h
     inc si
     jmp puts

scanf:
     mov posy, 23
     mov posx, 1
     goto_xy      POSx,POSy
     mov quatroch, 1
     xor si, si

     CICLO:
      call  LE_TECLA
     ;      cmp   ah, 1
     ;            je    ciclo
      cmp quatroch, 5
            jne backspace
      cmp al, 13        ; enter-------------------------------------
            jne espaco
      mov coordenadas[si], 0
      ret

     espaco:
      cmp al, 32 ; espaco
            jne backspace
      mov   ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
      mov   dl, 32 ;espaco
      int   21H
      mov quatroch, 1
      jmp   CICLO
```

```
    backspace:
     cmp         AL, 8  ;BACKSPACE
           jne limitesup
     cmp quatroch, 2
           jb ciclo   ; so continua se estiver dentro do lim inferior
     mov         ah, 02h
     mov         dl, 8 ;backspace pa andar pa traz
     int         21H
     mov         ah, 02h
     mov         dl, 32 ;espaço pa limpar
     int         21H
     mov         ah, 02h
     mov         dl, 8 ;backspace pa andar pa traz
     int         21H
     dec quatroch
     dec si ;anda pa traz no nomefich
     jmp ciclo

    limitesup:
     cmp quatroch, 4
           ja ciclo    ; so continua se estiver dentro do lim superior

     cmp al, 48 ; 0
           jne um
     mov   ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
     mov   dl, 48 ;0
     int   21H
     inc quatroch
     mov coordenadas[si], 48  ;coloca na variavel
     inc si
     jmp   CICLO

    um:
     cmp al, 49 ; 1
           jne ciclo
     mov   ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
     mov   dl, 49 ;1
     int   21H
     inc quatroch
     mov coordenadas[si], 49  ;coloca na variavel
     inc si
     jmp   CICLO


pedecoordenadas endp ;------>PEDE as coordenadas para o play2<------
```

```
edit_ficheiro proc  ;; ------>edita o maze<------
     mov          Car, 32           ;ESPAO  ;para começar com ' '
     mov          ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
     mov          dl, car  ; espaco
     int          21H

     mov posx, 3
     mov posy, 3
     goto_xy posx, posy

CICLO:  ;************************************
     cmp posy, 21
      jne notdown
     mov posy, 1

notdown:                           ;manter o y dentro dos limites
     cmp posy, 0
      jne notup
     mov posy, 20

notup:
     cmp posx, 0              ;manter o x dentro dos limites
      jne notleft
     mov posx, 40
notleft:
     cmp posx, 41
      jne notright
     mov posx, 1
notright:

;************************************
     goto_xy      POSx,POSy

IMPRIME:
     mov          ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
     mov          dl, Car
     int          21H
     mov          GChar, al   ; Guarda o Caracter que est na posio do Cursor
     goto_xy      POSx,POSy

     call GUARDA_ECRA

     call CRIA_FICHEIRO2

     call         LE_TECLA
     cmp          ah, 1
     je           ESTEND

     cmp al, 13  ;enter ---------------------------------
      jne zero
     ret
```

```
ZERO:
  cmp             AL, 48            ; Tecla 0
     jne          UM
     mov          Car, 32           ;ESPAO
     jmp          CICLO

UM:
  cmp             AL, 49            ; Tecla 1
     jne          DOIS
     mov          Car, 219          ;Caracter CHEIO
     jmp          CICLO

DOIS:
  cmp AL, 50             ; Tecla 2
     jne    TRES
     mov Car, 73 ; Define Inicio CHAR 'I'
     jmp    CICLO

TRES:
  cmp AL, 51             ; Tecla 3
     jne    QUATRO
     mov Car, 70; Define Fim CHAR 'F'
     jmp    CICLO

QUATRO:
  cmp AL, 52             ; Tecla 4
     jne    NOVE
     mov    Car, 32           ;epaço, serve para apagar
     jmp    CICLO

NOVE:       jmp         CICLO

ESTEND:
  cmp             al,48h
     jne          BAIXO
     dec          POSy        ;cima
     jmp          CICLO

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

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

DIREITA:
     cmp          al,4Dh
```

```
     jne          CICLO
     inc          POSx        ;Direita
     jmp          CICLO
edit_ficheiro endp ; ------>edita UM FICHEIRO<------

cria_ficheiro2 proc ; ------>CRIA UM FICHEIRO para o edit<------
     mov    ah, 3ch                 ; abrir ficheiro para escrita
     mov    cx, 00H                 ; tipo de ficheiro

     lea    dx, fnamedit            ; 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 vamos escrever

     mov    ah, 09h                 ; Aconteceu erro na leitura
     lea    dx, msgErrorCreate
     int    21h
     ret

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, 3520                ; 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
     ret                       ; se não acontecer erro termina

     mov    ah, 09h
     lea    dx, msgErrorClose
     int    21h
cria_ficheiro2 endp ; ------>CRIA UM FICHEIRO para o edit<------


CRIA_FICHEIRO PROC  ; ------>CRIA UM FICHEIRO<------

  mov ah, 3ch                  ; abrir ficheiro para escrita
     mov    cx, 00H                 ; tipo de ficheiro
```

```
     mov si, fnameptr ; vamos descrobrir onde começa o ultimo nome inserido
     -------------------
     dec si
     ciclo:
      dec si
      cmp fname[si], 0
            jne ciclo
     inc si

     lea    dx, fname[si]           ; 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 vamos escrever

     mov    ah, 09h                 ; Aconteceu erro na leitura
     lea    dx, msgErrorCreate
     int    21h
     ret

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, 4000                ; 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
     ret                       ; se não acontecer erro termina

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

CRIA_FICHEIRO ENDP  ; ----------->cria um ficheiro<----------------------

fnameptrToFich proc  ; ----------->passa o ptr para o ultimo elemento do array
     com os labirintos para um ficheiro<---------------------
     mov    ah, 3ch                 ; abrir ficheiro para escrita
     mov    cx, 00H                 ; tipo de ficheiro
     lea    dx, fnameptrfich        ; 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, mazenamesfichErr
     int    21h

     jmp    fim

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

     lea    dx, fnameptr                ; Vamos escrever o que estiver no
     endereço DX
     mov    cx, 2            ; 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

fnameptrToFich endp  ; ----------->passa o ptr para o ultimo elemento do array
     com os labirintos para um ficheiro<---------------------



fnamenToFich proc   ; ----------->passa o nr de elementos do array com os
     labirintos para um ficheiro<---------------------
     mov    ah, 3ch                ; abrir ficheiro para escrita
     mov    cx, 00H                ; tipo de ficheiro
     lea    dx, fnamenfich         ; 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, mazenamesfichErr
     int    21h

     jmp    fim
```

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

     lea    dx, fnamen              ; Vamos escrever o que estiver no endereço
     DX
     mov    cx, 1             ; 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

fnamenToFich endp  ; ----------->passa o nr de elementos do array com os
     labirintos para um ficheiro<----------------------


fnameToFich proc   ; ----------->passa o array com os labirintos para um
     ficheiro<---------------------
     mov    ah, 3ch                 ; abrir ficheiro para escrita
     mov    cx, 00H                 ; tipo de ficheiro
     lea    dx, mazenamesfich       ; 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, mazenamesfichErr
     int    21h

     jmp    fim

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

     lea    dx, fname               ; Vamos escrever o que estiver no endereço
     DX
     mov    cx, 390                 ; 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

fnameToFich endp  ; ----------->passa o array com os labirintos para um
     ficheiro<----------------------



fichTofname proc ; ----------->passa o array com os labirintos do ficheiro para
     o array<----------------------
;abre ficheiro

  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  lea     dx, mazenamesfich         ; 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     ler           ; depois de abero vamos ler o ficheiro

erro_abrir:
  mov     ah,09h
  lea     dx,Erro_Open
  int     21h
  jmp     sai

ler:
  mov     ah,3fh              ; indica que vai ser lido um ficheiro
  mov     bx,HandleFich       ; bx deve conter o Handle do ficheiro previamente
     aberto
  mov     cx, 390             ; numero de bytes a ler
  lea     dx, fname           ; 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
  mov     ah,3eh  ; vamos fechar o ficheiro
  mov     bx,HandleFich
  int     21h
  jnc     sai
mov     ah,09h                ; o ficheiro pode não fechar correctamente
```

```
  lea     dx,Erro_Close
  Int     21h

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

sai:
  ret
fichTofname endp ; ----------->passa o array com os labirintos do ficheiro para
     o array<---------------------



fichTofnamen proc ; ----------->passa o nr de elementos do array com os nomes
     dos mazes para o fnamen<---------------------
;abre ficheiro

  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  lea     dx, fnamenfich            ; 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     ler           ; depois de abero vamos ler o ficheiro

erro_abrir:
  mov     ah,09h
  lea     dx,Erro_Open
  int     21h
  jmp     sai

ler:
  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, fnamen          ; 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
  mov     ah,3eh  ; vamos fechar o ficheiro
  mov     bx,HandleFich
  int     21h
    jnc     sai
  mov     ah,09h              ; o ficheiro pode não fechar correctamente
  lea     dx,Erro_Close
  int     21h

erro_ler:
  mov     ah,09h
  lea     dx,Erro_Ler_Msg
```

```
  int     21h

sai:
  ret

fichTofnamen endp ; ----------->passa o nr de elementos do array com os nomes
     dos mazes para o fnamen<---------------------


fichTofnameptr proc ; ----------->passa o ptr para o ultimo elemento do array
     com os nomes dos mazes para o fnamen<----------------------
;abre ficheiro

  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  lea     dx, fnameptrfich          ; 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     ler           ; depois de abero vamos ler o ficheiro

erro_abrir:
  mov     ah,09h
  lea     dx,Erro_Open
  int     21h
  jmp     sai

ler:
  mov     ah,3fh              ; indica que vai ser lido um ficheiro
  mov     bx,HandleFich       ; bx deve conter o Handle do ficheiro previamente
     aberto
  mov     cx, 2               ; numero de bytes a ler
  lea     dx, fnameptr        ; 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
  mov     ah,3eh  ; vamos fechar o ficheiro
  mov     bx,HandleFich
  int     21h
    jnc     sai
  mov     ah,09h              ; o ficheiro pode não fechar correctamente
  lea     dx,Erro_Close
  int     21h

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

sai:
  ret
```

```
fichTofnameptr endp ; ----------->passa o ptr para o ultimo elemento do array
     com os nomes dos mazes para o fnamen<---------------------

MENUDESENHAR PROC  ;##########################----CRIACAO DE LABIRINTO---
     ####################################

  call pedeNomeF
  mov       ax,0B800h
  mov       es,ax
  call            apaga_ecran
  mov       Car, 32           ;ESPAO
  mov       ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
  mov       dl, car  ; espaco
  int       21H
  mov posx, 3
  mov posy, 3
  goto_xy posx, posy

CICLO:  ;************************************
     cmp posy, 21
      jne notdown
     mov posy, 1

notdown:                            ;manter o y dentro dos limites
     cmp posy, 0
      jne notup
     mov posy, 20

notup:
     cmp posx, 0              ;manter o x dentro dos limites
      jne notleft
     mov posx, 40
notleft:
     cmp posx, 41
      jne notright
     mov posx, 1
notright:

;************************************
     goto_xy      POSx,POSy

IMPRIME:
     mov          ah, 02h      ;WRITE CHARACTER TO STANDARD OUTPUT
     mov          dl, Car
     int          21H
     mov          GChar, al   ; Guarda o Caracter que est na posio do Cursor
     goto_xy      POSx,POSy

     call GUARDA_ECRA

     call CRIA_FICHEIRO

     call         LE_TECLA ;**************************
```

```
     cmp          ah, 1
     je           ESTEND

     cmp al, 13  ;enter
      jne zero
     mov posx, 25 ;colocar o x no sitio onde vai aparece os menus
     mov posy, 6
     ret


ZERO:
  cmp             AL, 48            ; Tecla 0
     jne          UM
     mov          Car, 32           ;ESPAO
     jmp          CICLO

UM:
  cmp             AL, 49            ; Tecla 1
     jne          DOIS
     mov          Car, 219          ;Caracter CHEIO
     jmp          CICLO

DOIS:
  cmp AL, 50             ; Tecla 2
     jne    TRES
     mov Car, 73 ; Define Inicio CHAR 'I'
     jmp    CICLO

TRES:
  cmp AL, 51             ; Tecla 3
     jne    QUATRO
     mov Car, 70; Define Fim CHAR 'F'
     jmp    CICLO

QUATRO:
  cmp AL, 52             ; Tecla 4
     jne    NOVE
     mov    Car, 32           ;epaço, serve para apagar
     jmp    CICLO

NOVE:       jmp         CICLO

ESTEND:
  cmp             al,48h
     jne          BAIXO
     dec          POSy        ;cima
     jmp          CICLO

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

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

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

MENUDESENHAR ENDP ;##########################----CRIACAO DE LABIRINTO---
     ####################################

menuEditMaze proc ;##########################----Editar LABIRINTO---
     ####################################
     mov          ax,0B800h
     mov          es,ax
     call apaga_ecran
     mov posy, 2
     mov posx, 10
     goto_xy      POSx,POSy
     xor si, si ;si=0
     mov ax, fnameptr
     mov lastzero, ax
     dec lastzero

titulo:
cmp string5[si], 0
     je checkelements
mov ah, 02h
mov dl, string5[si]
int 21h
inc si
jmp titulo

checkelements:
cmp fnamen, 0
     jne mazesinic
mov posy, 2
mov posx, 20
goto_xy     POSx,POSy
jmp lerop

mazesinic:
mov posy, 3
mov posx, 20
goto_xy     POSx,POSy
mov ah, 02h
mov dl, 91 ;[
int 21h
```

```
mov ah, 02h
mov dl, 32 ; espaco
int 21h
mov ah, 02h
mov dl, 93 ;]
int 21h
xor si, si ;si=0
mazes:

cmp fname[si], 0
     jne notnline
cmp lastzero, si
     je lerOp
inc posy
goto_xy     POSx,POSy
inc si
mov ah, 02h
mov dl, 91 ;[
int 21h
mov ah, 02h
mov dl, 32 ; espaco
int 21h
mov ah, 02h
mov dl, 93 ;]
int 21h
jmp mazes
notnline:
     mov ah, 02h
     mov dl, fname[si]
     int 21h
     inc si
     jmp mazes

lerOp:
  inc posy
  goto_xy   POSx,POSy
  mov ah, 02h
  mov dl, 91 ;[
  int 21h
  mov ah, 02h
  mov dl, 32 ; espaco
  int 21h
  mov ah, 02h
  mov dl, 93 ;]
  int 21h
  mov ah, 02h
  mov dl, 66 ; b
  int 21h
  mov ah, 02h
  mov dl, 65 ; a
  int 21h
  mov ah, 02h
  mov dl, 67 ; c
```

```
  int 21h
  mov ah, 02h
  mov dl, 75 ; k
  int 21h
  mov posy, 3
  mov posx, 21  ;meter o x dentro do []
  goto_xy   POSx,POSy

CICLO:
     mov al, fnamen
     add al, 4  ;
     cmp posy, al
      jne snake
     mov posy, 3 ;posicao inicial

snake:                                ;manter o 'x' dentro dos limites
  cmp posy, 2
      jne fimsnake
     dec al
     mov posy, al

fimsnake:
;*************************************

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

     goto_xy      POSx,POSy   ; Vai para nova possio
     mov    ah, 08h
     mov          bh,0        ; numero da pgina
     int          10h
     mov          Car, al     ; Guarda o Caracter que est na posio do Cursor
     mov          Cor, ah     ; Guarda a cor que est na posio do Cursor

     goto_xy      78,0        ; Mostra o caractr que estava na posio do AVATAR
     mov          ah, 02h     ; IMPRIME caracter da posio no canto
     mov          dl, Car
     int          21H

     goto_xy      POSx,POSy   ; Vai para posio do cursor

IMPRIME:
     mov          ah, 02h
     mov          dl, 120     ; Coloca AVATAR x -> 120
     int          21H
     goto_xy      POSx,POSy   ; Vai para posio do cursor

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

```
LER_SETA:
     call   le_tecla
     cmp          ah, 1
      je          ESTEND
     cmp al, 13          ;enter--------------------------------------
      jne estend
     mov al, fnamen
     add al, 3    ; +3 onde começa
     cmp posy, al   ; se o 'x' estiver no back
      jne qualmaze
     ret

qualmaze:
     mov al, posy
     mov fnameditop, al
     sub fnameditop, 2  ;-2 porque o 0 é o default
     jmp guessMaze

ESTEND:
     cmp    al,48h
     jne          BAIXO
     dec          POSy        ;cima
     jmp          CICLO

BAIXO:
     cmp          al,50h
     jne          ciclo
     inc    POSy        ;Baixo
     jmp          CICLO


guessMaze:
  ;descobrir fnamedit -------------------
  mov cx, 13
  xor si, si
  xor di, di

inicializa:  ;colocar fnamedit toda a zeros
  mov fnamedit[si], 0
  inc si
     loop inicializa

  mov p, 0 ;p e um ponteiro
  xor si, si

pset:  ;vamos descobrir onde esta o nome do ficheiro correspondente á opçao
  mov al, fnameditop
  dec al ;pq 0 é o mazedefault
  cmp p, al ;p é o n de zeros
     je strcpy

srchzero:
```

```
     cmp fname[si], 0
      je incp
     inc si
      jmp srchzero

incp:
     inc p
     inc si
     jmp pset

strcpy:
  cmp fname[si], 0
     je openmaze
  mov al, fname[si]
  mov fnamedit[di], al
  inc si
  inc di
  jmp strcpy


openmaze:
  mov mazedim, 0
  goto_xy 0,0
  ; imprime ficheiro-----------
      ;abre ficheiro

  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  lea     dx, fnamedit              ; 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     ler_ciclo           ; depois de abero vamos ler o ficheiro

erro_abrir:
  mov     ah,09h
  lea     dx,Erro_Open
  int     21h
  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
```

```
  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 mazedim
  cmp mazedim, 3520 ; 22*80*2   22linhas *80colunas *2bytes
     jae fecha_ficheiro
  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

 ;-------------------------------------------------------------------------------
     -------

call edit_ficheiro
ret

menuEditMaze endp ;############################----Editar LABIRINTO---
     ###################################

menuLoadUmaze proc   ;############################----carregar labirinto---
     ###################################

     call apaga_ecran
     mov posy, 2
     mov posx, 3
     goto_xy      POSx,POSy
     xor si, si ;si=0
     mov ax, fnameptr
     mov lastzero, ax
     dec lastzero

titulo:
     cmp string2[si], 0
      je checkelements
     mov ah, 02h
     mov dl, string2[si]
     int 21h
     inc si
  jmp titulo

checkelements:
     cmp fnamen, 0
      jne mazesinic
     mov posy, 2
     mov posx, 20
```

```
     goto_xy      POSx,POSy
     jmp lerop

mazesinic:
     mov posy, 3
     mov posx, 20
     goto_xy      POSx,POSy
     mov ah, 02h
     mov dl, 91 ;[
     int 21h
     mov ah, 02h
     mov dl, 32 ; espaco
     int 21h
     mov ah, 02h
     mov dl, 93 ;]
     int 21h
     xor si, si ;si=0

mazes:
     cmp fname[si], 0
      jne notnline
     cmp lastzero, si
      je lerOp
     inc posy
     goto_xy      POSx,POSy
     inc si
     mov ah, 02h
     mov dl, 91 ;[
     int 21h
     mov ah, 02h
     mov dl, 32 ; espaco
     int 21h
     mov ah, 02h
     mov dl, 93 ;]
     int 21h
     jmp mazes

notnline:
     mov ah, 02h
     mov dl, fname[si]
     int 21h
     inc si
     jmp mazes

lerOp:
     inc posy
     goto_xy      POSx,POSy
     mov ah, 02h
     mov dl, 91 ;[
     int 21h
     mov ah, 02h
     mov dl, 32 ; espaco
     int 21h
```

```
     mov ah, 02h
     mov dl, 93 ;]
     int 21h
     mov ah, 02h
     mov dl, 66 ; b
     int 21h
     mov ah, 02h
     mov dl, 65 ; a
     int 21h
     mov ah, 02h
     mov dl, 67 ; c
     int 21h
     mov ah, 02h
     mov dl, 75 ; k
     int 21h
     mov posy, 3
     mov posx, 21  ;meter o x dentro do []
     goto_xy      POSx,POSy

CICLO:
     mov al, fnamen
     add al, 4  ;
     cmp posy, al
      jne snake
     mov posy, 3 ;posicao inicial

snake:                              ;manter o 'x' dentro dos limites
     cmp posy, 2
      jne fimsnake
     dec al
     mov posy, al

fimsnake:
;*************************************

     goto_xy      POSxa,POSya ; Vai para a posio anterior do cursor

     mov          ah, 02h
     mov          dl, Car     ; Repoe Caracter guardado
     int          21H

     goto_xy      POSx,POSy   ; Vai para nova possio
     mov    ah, 08h
     mov          bh,0        ; numero da pgina
     int          10h
     mov          Car, al     ; Guarda o Caracter que est na posio do Cursor
     mov          Cor, ah     ; Guarda a cor que est na posio do Cursor

     goto_xy      78,0        ; Mostra o caractr que estava na posio do AVATAR
     mov          ah, 02h     ; IMPRIME caracter da posio no canto
     mov          dl, Car
     int          21H
```

```
     goto_xy      POSx,POSy   ; Vai para posio do cursor

IMPRIME:
     mov          ah, 02h
     mov          dl, 120     ; Coloca AVATAR x -> 120
     int          21H
     goto_xy      POSx,POSy   ; Vai para posio do cursor

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

LER_SETA:
     call   le_tecla
     cmp          ah, 1
      je          ESTEND
     cmp al, 13          ;enter---------------------------------------
      jne estend
     mov al, fnamen
     add al, 3    ; +3 onde começa
     cmp posy, al   ; se o 'x' estiver no back
      jne qualmaze
     ret
     qualmaze:
            mov al, posy
            mov fnameop, al
            sub fnameop, 2  ;-2 porque o 0 é o default
            ret

ESTEND:
     cmp    al,48h
     jne          BAIXO
     dec          POSy        ;cima
     jmp          CICLO

BAIXO:
     cmp          al,50h
     jne          ciclo
     inc    POSy        ;Baixo
     jmp          CICLO

menuLoadUmaze endp   ;############################----carregar labirinto---
     ####################################

menuDeleteMaze proc   ;############################----apagar labirintos---
     ####################################

  inicio:
     call apaga_ecran
     mov posy, 2
     mov posx, 8
     goto_xy      POSx,POSy
```

```
xor si, si ;si=0
mov ax, fnameptr
mov lastzero, ax
dec lastzero

titulo:
 cmp string4[si], 0
       je checkelements
 mov ah, 02h
 mov dl, string4[si]
 int 21h
 inc si
 jmp titulo

checkelements:
cmp fnamen, 0
 jne mazesinic
mov posy, 2
mov posx, 20
goto_xy      POSx,POSy
jmp lerop

mazesinic:
 mov posy, 3
 mov posx, 20
 goto_xy     POSx,POSy
 mov ah, 02h
 mov dl, 91 ;[
 int 21h
 mov ah, 02h
 mov dl, 32 ; espaco
 int 21h
 mov ah, 02h
 mov dl, 93 ;]
 int 21h
 xor si, si ;si=0
mazes:

 cmp fname[si], 0
       jne notnline
 cmp lastzero, si
       je lerOp
 inc posy
 goto_xy     POSx,POSy
 inc si
 mov ah, 02h
 mov dl, 91 ;[
 int 21h
 mov ah, 02h
 mov dl, 32 ; espaco
 int 21h
 mov ah, 02h
 mov dl, 93 ;]
```

```
 int 21h
 jmp mazes
 notnline:
       mov ah, 02h
       mov dl, fname[si]
       int 21h
       inc si
       jmp mazes

lerOp:
 inc posy
 goto_xy     POSx,POSy
 mov ah, 02h
 mov dl, 91 ;[
 int 21h
 mov ah, 02h
 mov dl, 32 ; espaco
 int 21h
 mov ah, 02h
 mov dl, 93 ;]
 int 21h
 mov ah, 02h
 mov dl, 66 ; b
 int 21h
 mov ah, 02h
 mov dl, 65 ; a
 int 21h
 mov ah, 02h
 mov dl, 67 ; c
 int 21h
 mov ah, 02h
 mov dl, 75 ; k
 int 21h
 mov posy, 3
 mov posx, 21  ;meter o x dentro do []
 goto_xy     POSx,POSy


 CICLO:
       mov al, fnamen
       add al, 4  ;
       cmp posy, al
             jne snake
       mov posy, 3 ;posicao inicial

       snake:                              ;manter o 'x' dentro dos
limites
             cmp posy, 2
                   jne fimsnake
             dec al
             mov posy, al

       fimsnake:
```

```
        ;**************************************

              goto_xy     POSxa,POSya ; Vai para a posio anterior do cursor

              mov         ah, 02h
              mov         dl, Car     ; Repoe Caracter guardado
              int         21H

              goto_xy     POSx,POSy   ; Vai para nova possio
              mov   ah, 08h
              mov         bh,0        ; numero da pgina
              int         10h
              mov         Car, al     ; Guarda o Caracter que est na posio
do Cursor
              mov         Cor, ah     ; Guarda a cor que est na posio do
Cursor

              goto_xy     78,0        ; Mostra o caractr que estava na
posio do AVATAR
              mov         ah, 02h     ; IMPRIME caracter da posio no canto
              mov         dl, Car
              int         21H

              goto_xy     POSx,POSy   ; Vai para posio do cursor

        IMPRIME:
              mov         ah, 02h
              mov         dl, 120     ; Coloca AVATAR x -> 120
              int         21H
              goto_xy     POSx,POSy   ; Vai para posio do cursor

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

        LER_SETA:
              call  le_tecla
              cmp         ah, 1
                    je          ESTEND
              cmp al,
13             ;enter---------------------------------------
                    jne estend
              mov al, fnamen
              add al, 3   ; +3 onde começa
              cmp posy, al   ; se o 'x' estiver no back
                    jne qualmaze
              ret

              qualmaze:

                    mov zeros, 0
                    mov al, posy
```

```
                  sub al, 3
                  xor si, si
                  xor di, di

                  incsidi:
                        cmp zeros, al
                              je incsi
                        inc si
                        inc di
                        cmp fname[si], 0
                              jne notzero
                        inc zeros
                        notzero:
                        jmp incsidi

                  incsi:
                        cmp al, 0  ;caso especial, se for o primeiro
elemento do vetor
                              je first
                        inc di
                        mov ah, fnamen
                        dec ah
                        cmp al, ah  ;caso especial, se for o ultimo
elemento do vetor
                              je last
                        first:
                              cmp fnamen, 1  ;caso especial, se for o
unico elemento do vetor
                                    je last
                              inc si
                              cmp fname[si], 0
                                    jne first
                        inc si
                  strcpy:
                        mov ah, fname[si]
                        mov fname[di], ah
                        inc si
                        inc di
                        cmp si, fnameptr
                              jne strcpy
                  last:
                        mov fnameptr, di
                        dec fnamen
                        jmp inicio


      ESTEND:
            cmp   al,48h
            jne         BAIXO
            dec         POSy        ;cima
            jmp         CICLO

      BAIXO:
```

```
                cmp         al,50h
                jne         ciclo
                inc   POSy        ;Baixo
                jmp         CICLO


menuDeleteMaze endp   ;############################----apagar labirintos---
     ###################################

MENUCONFIG PROC  ;############################----MENU DE CONFIGURACAO---
     ###################################
inicio:
     mov posx, 25 ;colocar o x no sitio onde vai aparece os menus
     mov posy, 6
     mov          ax,0B800h       ; Apaga
     mov          es,ax                       ; o
     call   apaga_ecran         ; Ecran

;abre ficheiro
     mov     ah,3dh                    ; vamos abrir ficheiro para leitura
     mov     al,0                    ; tipo de ficheiro
     lea     dx,FichMazeConfig                      ; 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     ler_ciclo            ; depois de aberto vamos ler o ficheiro

erro_abrir:
     mov     ah,09h
     lea     dx,Erro_Open
     int     21h
    jmp     fim

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

goto_xy     POSx,POSy   ; Vai para nova possio
     mov    ah, 08h     ; Guarda o Caracter que est na posio do Cursor
     mov          bh,0        ; numero da pgina
     int          10h
     mov          Car, al     ; Guarda o Caracter que est na posio do Cursor
     mov          Cor, ah     ; Guarda a cor que est na posio do Cursor

;*************************************
CICLO:
     cmp posy, 12
     jne snake
     mov posy, 6

  snake:                            ;manter o 'x' dentro dos limites
     cmp posy, 5
      jne fimsnake
     mov posy, 11

  fimsnake:
;*************************************

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

     goto_xy      POSx,POSy   ; Vai para nova possio
      mov   ah, 08h
      mov         bh,0        ; numero da pgina
      int         10h
      mov         Car, al     ; Guarda o Caracter que est na posio do Cursor
      mov         Cor, ah     ; Guarda a cor que est na posio do Cursor

     goto_xy      78,0        ; Mostra o caractr que estava na posio do AVATAR
      mov         ah, 02h     ; IMPRIME caracter da posio no canto
      mov         dl, Car
      int         21H

     goto_xy      POSx,POSy   ; Vai para posio do cursor

IMPRIME:
     mov          ah, 02h
```

```
     mov          dl, 120     ; Coloca AVATAR x -> 120
     int          21H
     goto_xy      POSx,POSy   ; Vai para posio do cursor

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

LER_SETA:
     call   le_tecla
     cmp          ah, 1
     cmp al, 13          ;enter
     jne estend
     cmp posy, 11   ; se o 'x' estiver no back
     jne createmaze
     mov posx, 33
     mov posy, 18
     ret

createmaze:
  cmp posy, 8  ; se o 'x' estiver no create maze
     jne loadUserMaze
  call menuDesenhar ; Call MenuDesenhar PROC
  jmp inicio

loadUserMaze:
     cmp posy, 7 ;se o x estiver no load user maze
      jne loadDefaultMaze
     call menuLoadUmaze
     jmp inicio

loadDefaultMaze:
     cmp posy, 6 ;se o x estiver no load default maze
      jne deletemaze
     mov fnameop, 0
     call apaga_ecran
     mov posy, 2
     mov posx, 4
     goto_xy      POSx,POSy
     xor si, si
     puts:
      cmp string3[si], 0
            je getchar
      mov ah, 02h
      mov dl, string3[si]
      int 21h
      inc si
      jmp puts
     getchar:
      call  le_tecla
      cmp   al, 13       ; enter
            je          inicio
```

```
     jmp getchar

deletemaze:
     cmp posy, 10 ;se o x estiver no delete maze
      jne editmaze
     call menudeletemaze
     jmp inicio

editmaze:
     call menueditmaze
     jmp inicio

ESTEND:
     cmp    al,48h
     jne          BAIXO
     dec          POSy        ;cima
     jmp          CICLO

BAIXO:
     cmp          al,50h
     jne          ciclo
     inc    POSy        ;Baixo
     jmp          CICLO

fim:
     mov          ah,4CH
     int          21H

MENUCONFIG ENDP ;############################----MENU DE CONFIGURACAO---
     ###################################

PLAY PROC ;############################----INICIA O JOGO---
     ###################################

  mov       ax,0B800h           ; Apaga
  mov       es,ax                   ;     o
  call apaga_ecran ; ecra

  mov MazeX, 1
  mov MazeY, 1
  goto_xy   MazeX,MazeY

  mov segundos1,47
  mov segundos2,48

  mov minutos1,48
  mov minutos2,48

  ; ate aqui esta igual ao MENU

     ;mov cx, 13
     ;xor si, si
     ;inicializa:  ;colocar fnameopen toda a zeros
```

```
      ;mov fnameopen[si], 0
      ;inc si
      ;loop inicializa

     ;mov p, 0
     ;xor si, si
     ;cmp fnameop, 0
      ;jne notdefault  ;vamos descobrir onde esta o nome do ficheiro
     correspondente á opçao
     ;strcpy:
      ;cmp fichdefaultmaze, 0
            ;je openmaze
      ;mov al, fichdefaultmaze[si]
      ;mov fnameopen[si], al
      ;inc si
      ;jmp strcpy
     ;notdefault:
      ;mov al, fnameop
      ;dec al ;pq 0 é o mazedefault
      ;cmp p, al ;pe é o n de zeros
            ;je prepstrcpy2
      ;srchzero:
            ;cmp fname[si], 0
                  ;je incp
            ;inc si
            ;jmp srchzero
      ;incp:
            ;inc p
            ;inc si
            ;jmp notdefault
     ;prepstrcpy2:
      ;xor di, di
     ;strcpy2:
      ;cmp fname[si], 0
            ;je openmaze
      ;mov al, fname[si]
      ;mov fnameopen[di], al
      ;inc si
      ;inc di
      ;jmp strcpy2

openmaze:
  mov mazedim, 0
  goto_xy 0,0
  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  ;lea     dx, fnameopen                  ; nome do ficheiro
  lea     dx, FichDefaultMaze
  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
  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
  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 mazedim
  cmp mazedim, 3520 ; 22*80*2   22linhas *80colunas *2bytes
      jae fecha_ficheiro
  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       ax, dseg
  mov       ds,ax
  mov       ax,0B800h
  mov       es,ax

  goto_xy   MazeX,MazeY ; Vai para nova posio
  mov             ah, 08h     ; Guarda o Caracter que est na posio do Cursor
  mov       bh,0        ; numero da pgina
  int       10h
  mov       Car, al     ; Guarda o Caracter que est na posio do Cursor
  mov       Cor, ah     ; Guarda a cor que est na posio do Cursor

;procura o I e o F no maze e substitui por
     espaço-----------------------------------------------
     mov ax, 0B800h  ;memoria de video
     mov es, ax

  mov cx, 20*80   ; 20 altura * 80 largura
```

```
  mov si, 80*2    ; comea na linha 1

procuraF:
     mov bx, es:[si] ;memoria de video
  add si, 2

  cmp bl, 70 ; F
    loopne procuraF
  sub si, 2
  mov bl, 20h ;substitui por espaço em branco
  mov es:[si], bx

  mov ax, si
  mov bl, 160 ;imediat operand not allowed
  div bl
  mov mazey, al  ; o quociente  o y
  mov mazex, ah  ; o resto  o x

procuraI:
     mov bx, es:[si]
  add si, 2
  cmp bl, 73 ; I
    loopne procuraI
  sub si, 2
  mov bl, 20h ;substitui por espaço em branco
  mov es:[si], bx

  mov ax, si
  mov bl, 160 ;imediat operand not allowed
  div bl
  mov mazey, al  ; o quociente  o y
  mov mazex, ah  ; o resto  o x


;-------------------------------------------------------------------------

  ; monta o cornometro
  goto_xy 68,1
  mostra strzero

  goto_xy 67,1
  mostra strzero

  goto_xy 66,1
  mostra strdoispontos

  goto_xy 65,1
  mostra strzero

  goto_xy 64,1
  mostra strzero

CICLO:
```

```
;*********************************************************
     cmp mazey, 21
      jne notdown
     mov mazey, 1

notdown:                              ;manter o y dentro dos limites
     cmp mazey, 0
      jne notup
     mov mazey, 20

notup:
     cmp mazex, 0             ;manter o x dentro dos limites
      jne notleft
     mov mazex, 40
notleft:
     cmp mazex, 42
      jne notright
     mov mazex, 1
notright:
  goto_xy   mazex,mazey

  cmp mazex, 41 ; deteta o fim do labirinto
    je TEMPOTOTAL

;*****************************************************

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

  goto_xy   MazeX,MazeY ; Vai para nova posição
  mov       ah, 08h
  mov       bh, 0       ; numero da pgina
  int       10h

  mov       Car, al     ; Guarda o Caracter que est na posio do Cursor
  mov       Cor, ah     ; Guarda a cor que est na posio do Cursor

  cmp Car,20h  ;compara com espaço
      jne parede  ; se não for espaço é parede
     jmp imprime

parede:

     goto_xy POSxa,POSya ; vai para a posição anterior
  mov Car, 20h
     mov ch,POSxa
     mov MazeX,ch
     mov ch,POSya
     mov MazeY,ch

IMPRIME:
```

```
  mov       ah, 02h
  mov       dl, 233     ; Coloca AVATAR
  int       21H
  goto_xy   MazeX,MazeY ; Vai para posio do cursor

  mov       al, MazeX   ; Guarda a posio do cursor
  mov       POSxa, al
  mov       al, MazeY   ; Guarda a posio do cursor
  mov       POSya, al

LER_SETA:
  call      LE_TECLA_CLOCK
  cmp       ah, 1
    je            CIMA
  cmp       al, 27      ; ESCAPE
    je            TEMPOTOTAL
  jmp       LER_SETA

CIMA:
  cmp       al,48h
    jne           BAIXO
  dec       MazeY       ;cima
  jmp       CICLO

BAIXO:
  cmp       al,50h
    jne           ESQUERDA
  inc       MazeY       ;Baixo
  jmp       CICLO

ESQUERDA:
  cmp       al,4Bh
    jne           DIREITA
  dec       MazeX       ;Esquerda
  jmp       CICLO

DIREITA:
  cmp       al,4Dh
    jne           LER_SETA
  inc       MazeX       ;Direita
  jmp       CICLO

TEMPOTOTAL:
  ; Mostra no ecrã
  call apaga_ecran
  goto_xy 32,11
  mostra strTEMPOTOTAL ; String fixa TEMPOTOTAL

  goto_xy 35,12
  mov ah, ' '
  cmp strminutos2, ah
    je mostrazero3
```

```
  mostra strminutos2

mostrazero3:
  mostra strzero

  goto_xy 36,12
  mov ah, ' '
  cmp strminutos1, ah
    je mostrazero2
  mostra strminutos1

mostrazero2:
  mostra strzero

  goto_xy 37,12
  mostra strdoispontos      ; mostra separador ':'

  goto_xy 38,12             ; segunda posicao pode estar vazia
  mov ah, ' '
  cmp strsegundos2, ah      ; se estiver vazia tem de ser comparada para
     mostrar o caracter '0'
    je mostrazero1
  mostra strsegundos2

mostrazero1:
    mostra strzero

  ; monta o cornometro
  goto_xy 39,12
  mostra strsegundos1       ; primeira posicao tem sempre algum caracter

  call      LE_TECLA
  cmp       al, 27      ; ESCAPE
    ret
  jmp       TEMPOTOTAL

fim:
     mov          ah,4CH
     int          21H

PLAY ENDP ;##########################----INICIA O JOGO---
     ###################################

MENU PROC ;##########################----MENU PRINCIPAL---
     ###################################
  inicio:
     mov          ax,0B800h           ; Apaga
     mov          es,ax                         ;   o
     call         apaga_ecran           ; Ecran

     mov POSy,               18
     mov POSx,               33
     goto_xy posx, posy
```

```
   ;abre ficheiro
   mov     ah,3dh                    ; vamos abrir ficheiro para leitura
   mov     al,0                    ; tipo de ficheiro
   lea     dx,FichMenu                       ; 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     ler_ciclo             ; depois de aberto vamos ler o ficheiro

erro_abrir:
   mov     ah,09h
   lea     dx,Erro_Open
   int     21h
  jmp     fim

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

goto_xy   POSx,POSy     ; Vai para nova possio
   mov    ah, 08h         ; Guarda o Caracter que est na posio do Cursor
   mov          bh,0            ; numero da pgina
   int          10h
   mov          Car, al         ; Guarda o Caracter que est na posio do
   Cursor
```

```
   mov          Cor, ah         ; Guarda a cor que est na posio do Cursor

CICLO:
;*************************************
   cmp posy, 23
    jne snake
   mov posy, 18
   snake:                          ;manter o 'x' dentro dos limites

   cmp posy, 17
    jne fimsnake
   mov posy, 22
   fimsnake:
;*************************************

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

   goto_xy      POSx,POSy   ; Vai para nova possio
    mov         ah, 08h
    mov         bh,0        ; numero da pgina
    int         10h
    mov         Car, al     ; Guarda o Caracter que est na posio do Cursor
    mov         Cor, ah     ; Guarda a cor que est na posio do Cursor

   goto_xy      78,0        ; Mostra o caractr que estava na posio do AVATAR
    mov         ah, 02h     ; IMPRIME caracter da posio no canto
    mov         dl, Car
    int         21H

    goto_xy     POSx,POSy   ; Vai para posio do cursor

IMPRIME:
   mov          ah, 02h
    mov         dl, 120     ; Coloca AVATAR x -> 120
    int         21H
    goto_xy     POSx,POSy   ; Vai para posio do cursor

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

LER_SETA:
   call   LE_TECLA
   cmp          ah, 1
   je           ESTEND
   cmp    al, 27      ; ESCAPE
    jne lerop
   call fnameToFich
   call fnamenToFich
```

```
   call fnameptrToFich
   jmp fim
   lerop:
;************************************
;                            ler opcao

   cmp al, 13                              ;enter
    jne estend
   cmp posy, 18 ; se o 'x' estiver no play
    jne exit
   call play
   jmp inicio
   exit:
    cmp posy, 21  ; se o 'x' estiver no exit
          jne mazeconfig
    call fnameToFich
    call fnamenToFich
    call fnameptrToFich
    jmp fim
    jmp fim
   mazeconfig:
   cmp posy, 20 ; se o 'x' estiver no maze config
    jne bonus
   call menuconfig
   jmp inicio


   bonus:
    cmp posy, 22 ; se o 'x' estiver no maze config
          jne estend
    call menubonus
    jmp inicio

;************************************
   jmp          LER_SETA

ESTEND:
   cmp          al,48h
    jne         BAIXO
   dec          POSy        ;cima
   jmp          CICLO

BAIXO:
   cmp          al,50h
   jne          ciclo
   inc          POSy        ;Baixo
   jmp          CICLO

fim:
    mov         ah,4CH
    int         21H
```

```
MENU ENDP ;############################----MENU PRINCIPAL---
     ####################################




menubonus proc ;############################----MENU PRINCIPAL---
     ####################################
 mov  ax,0B800h           ; Apaga
  mov       es,ax                    ;     o
  call apaga_ecran ; ecra


  ; ate aqui esta igual ao MENU

     ;mov cx, 13
     ;xor si, si
     ;inicializa:  ;colocar fnameopen toda a zeros
      ;mov fnameopen[si], 0
      ;inc si
      ;loop inicializa

     ;mov p, 0
     ;xor si, si
     ;cmp fnameop, 0
      ;jne notdefault  ;vamos descobrir onde esta o nome do ficheiro
     correspondente á opçao
     ;strcpy:
      ;cmp fichdefaultmaze, 0
            ;je openmaze
      ;mov al, fichdefaultmaze[si]
      ;mov fnameopen[si], al
      ;inc si
      ;jmp strcpy
     ;notdefault:
      ;mov al, fnameop
      ;dec al ;pq 0 é o mazedefault
      ;cmp p, al ;pe é o n de zeros
            ;je prepstrcpy2
      ;srchzero:
            ;cmp fname[si], 0
                  ;je incp
            ;inc si
            ;jmp srchzero
      ;incp:
            ;inc p
            ;inc si
            ;jmp notdefault
     ;prepstrcpy2:
      ;xor di, di
     ;strcpy2:
      ;cmp fname[si], 0
```

```
            ;je openmaze
      ;mov al, fname[si]
      ;mov fnameopen[di], al
      ;inc si
      ;inc di
      ;jmp strcpy2


openmaze:
  mov mazedim, 0
  goto_xy 0,0
  mov     ah,3dh              ; vamos abrir ficheiro para leitura
  mov     al,0                ; tipo de ficheiro
  ;lea     dx, fnameopen                  ; nome do ficheiro
  lea     dx, FichDefaultMaze
  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
  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
  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 mazedim
   cmp mazedim, 3520 ; 22*80*2   22linhas *80colunas *2bytes
      jae fecha_ficheiro
  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       ax, dseg
mov       ds,ax
mov       ax,0B800h
mov       es,ax

goto_xy   MazeX,MazeY ; Vai para nova posio
mov             ah, 08h     ; Guarda o Caracter que est na posio do Cursor
mov       bh,0        ; numero da pgina
int       10h
mov       Car, al     ; Guarda o Caracter que est na posio do Cursor
mov       Cor, ah     ; Guarda a cor que est na posio do Cursor

;procura o I e o F no maze e substitui por
   espaço-----------------------------------------------
   mov ax, 0B800h  ;memoria de video
   mov es, ax

  mov cx, 20*80   ; 20 altura * 80 largura
  mov si, 80*2    ; comea na linha 1

procuraF:
   mov bx, es:[si] ;memoria de video
  add si, 2

  cmp bl, 70 ; F
    loopne procuraF
  sub si, 2
  mov bl, 20h ;substitui por espaço em branco
  mov es:[si], bx

  mov ax, si
  mov bl, 160 ;imediat operand not allowed
  div bl
  mov mazey, al  ; o quociente  o y
  mov mazex, ah  ; o resto  o x

procuraI:
   mov bx, es:[si]
  add si, 2
  cmp bl, 73 ; I
    loopne procuraI
  sub si, 2
  mov bl, 20h ;substitui por espaço em branco
  mov es:[si], bx

  mov ax, si
  mov bl, 160 ;imediat operand not allowed
  div bl
  mov mazey, al  ; o quociente  o y
  mov mazex, ah  ; o resto  o x
```

```
;--------------------------------------------------------------------------

    dec mazex


;imput das
   coordenadas-----------------------------------------------------------------
   ------------------------------------------------------
   call pedecoordenadas


   xor si, si
   verpassos:
          mov di, si
          add di, 2
          cmp coordenadas[di], 48 ;0
                jne   tres
          inc di
          cmp coordenadas[di], 48 ;0
                jne dois
          mov cx, 1         ;cx vai conter o nr de passos a dar
          mov di, si        ;repor o di antes de ver a direcao
          jmp verdirecao
    dois:
          mov cx, 2  ;cx vai conter o nr de passos a dar
          mov di, si  ;repor o di antes de ver a direcao
          jmp verdirecao
    tres:
          inc di
          cmp coordenadas[di], 48 ;0
                jne quatro
          mov cx, 3         ;cx vai conter o nr de passos a dar
          mov di, si     ;repor o di antes de ver a direcao
          jmp verdirecao
    quatro:
          mov cx, 4         ;cx vai conter o nr de passos a dar
          mov di, si       ;repor o di antes de ver a direcao

    verdirecao:
          cmp coordenadas[di], 48 ;0
                jne este
          inc di
          cmp coordenadas[di], 48 ;0
                jne moversul
          movernorte:
                dec mazey
                goto_xy     MazeX, MazeY

                getchar1:
                      call  le_tecla
                      cmp   al, 27       ; escape
                            je fim
```

```
                  cmp   al, 13       ; enter
                        jne         getchar1
      loop movernorte
      jmp incrementar
 moversul:
      inc mazey
      goto_xy     MazeX, MazeY
      getchar2:
                  call  le_tecla
                  cmp   al, 27       ; escape
                        je fim
                  cmp   al, 13       ; enter
                        jne         getchar2
      loop moversul
      jmp incrementar
 este:
      inc di
      cmp coordenadas[di], 48 ;0
            jne moveroeste
      movereste:
            inc mazex
            goto_xy     mazex,mazey


      getchar3:
                        call  le_tecla
                        cmp   al, 27       ; escape
                              je fim
                        cmp   al, 13       ; enter
                              jne         getchar3

      loop movereste
      jmp incrementar
 moveroeste:
      dec mazex
      goto_xy     MazeX, MazeY
      getchar4:
                  call  le_tecla
                  cmp   al, 27       ; escape
                        je fim
                  cmp   al, 13       ; enter
                        jne         getchar4
      loop moveroeste

incrementar:


 add si, 4
 cmp coordenadas[si], 0
      jne verpassos
```

```
fim:
     ret
menubonus endp ;############################----MENU PRINCIPAL---
     ###################################
```

```
MAIN PROC ;################################----MAIN---
     ########################################
  mov     ax,DSEG
  mov     ds,ax
     call fichTofname
     call fichTofnamen
     call fichTofnameptr
     call MENU

MAIN  ENDP ;################################----MAIN---
     ########################################
CSEG ENDS
END MAIN
```