CADERNO DE EXERCÍCIOS RESOLVIDOS DE ASSEMBLY PARA 8086 (Microsoft Assembler 6.0)

2010 – 2011

ACÁCIO MANUEL RAPOSO AMARAL

PROFESSOR ADJUNTO

1. Faça um programa que contabilize o número de vezes que a letra ‘a’ surge num vector com o endereço inicial frase1, inicializado com os caracteres “acacio amaral”. O vector deve terminar com um carácter de valor zero.

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
frase1  db      'acacio amaral',0
cont    db      0
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio: mov ax,dados
        mov ds,ax

        lea     si,frase1
conta:  mov     al,[si]
        inc     si
        cmp     al,'a'
        jne     salta
        inc     cont
salta:  cmp     al,0
        jne     conta

        mov     ah,4ch
        int 21h

codigo  ends
end     inicio
```

2. Faça um programa que copie para um vector unidimensional (1×4) com endereço inicial vector1, a diagonal principal de uma matriz quadrada (4×4) cujo endereço inicial é matriz.

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
vector  db      4 dup (?)
matriz  db      'I','A','C','A'
        db      'C','S','I','0'
        db      'A','M','E','A'
        db      'R','A','L','C'
const   db      4
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio: mov ax,dados
        mov ds,ax

        LEA     SI,MATRIZ
        LEA     DI,VECTOR

        MOV     CX,-1
SALTO:  INC     CX
        MOV     BX,CX ; é necessário usar BX -> modo de endereçamento
        MOV     AX,BX ; é necessário usar AX
        MUL     CONST       ; devido à operação de MULTIPLICAÇÃO
        ADD     BX,AX ; posição da diagonal principal no vector
                      ; linha*4+coluna, para linha=coluna=CX
        MOV     DL,[SI+BX]
        MOV     [DI],DL
        INC     DI
        CMP     BX,15   ; a posição do último elemento da matriz = 15
        JB      SALTO

        mov     ah,4ch
        int 21h

codigo  ends
end     inicio
```

3. Faça um programa que obtenha a transposta de uma matriz quadrada (4×4) cujo endereço inicial é matriz.

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
matriz  db      'A','B','C','D'
        db      'E','F','G','H'
        db      'I','J','L','M'
        db      'N','O','P','Q'
aux     db      16 dup(?)
LIN     DW      0
COL     DW      0
POS     DW      ?
POSAUX      DW      ?
const   db      4
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio:  mov ax,dados
        mov ds,ax

        LEA     SI,MATRIZ
        LEA     DI,AUX

MUDA_L:     MOV   AX,LIN;posicao no vector matriz = linha*4+coluna
        MUL     CONST       ;necessário usar AX com MUL AX=AL*CONST
        ADD     AX,COL
        MOV     POS,AX

        MOV     AX,COL      ;posicao no vector aux = coluna*4+linha
        MUL     CONST       ;necessário usar AX com MUL AX=AL*CONST
        ADD     AX,LIN
        MOV     POSAUX,AX

        MOV     BX,POS      ;necessário usar BX
        MOV     AL,[SI+BX] ; modo de endereçamento (BX,SI,DI,BP)
        MOV     BX,POSAUX
        MOV     [DI+BX],AL ; modo de endereçamento (BX,SI,DI,BP)
        INC     COL
        CMP     COL,4
        JB      MUDA_L

        MOV     COL,0
        INC     LIN
        CMP     LIN,4
        JB      MUDA_L

        ;       copia para a matriz inicial
```

```
        LEA     SI,MATRIZ
        LEA     DI,AUX

        MOV     CX,15 ; copiar vector Aux para Matriz
COPIA:          MOV     BX,CX ; 16 elementos, o 1º-> pos=0
        MOV     AL,[DI+BX]
        MOV     [SI+BX],AL
        LOOP    COPIA


        mov     ah,4ch
        int 21h

codigo ends
end     inicio
```

4. Faça um programa que copie para uma tabela todos os algarismos de um número decimal cujo valor numérico se encontre compreendido entre [0,65535]. A ordem pela qual os algarismos são colocados na tabela deverá ser inversa à posição dos mesmos no número. Por exemplo: considere o número 61254, o programa deverá preencher a tabela da seguinte forma:

4 5 2 1 6

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
num     dw      61254
tab     db      5 dup (0)
const   dW      000AH
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio:  mov ax,dados
         mov ds,ax

         LEA    SI,TAB
         MOV    AX,NUM

DIVIDE:          MOV    DX,0
         DIV     CONST
         MOV     [SI],DL
         INC     SI
         CMP     AX,0
         JNE     DIVIDE

         mov     ah,4ch
         int 21h

codigo ends
end     inicio
```

5. Faça uma função que permita copiar para uma tabela todos os algarismos de um número decimal cujo valor numérico se encontre compreendido entre [0,65535]. A ordem pela qual os algarismos são colocados na tabela deverá ser inversa à posição dos mesmos no número. O valor numérico e o endereço do array devem ser recebidos pela pilha.

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
num     dw      12345
tab     db      5 dup (0)
const   dW      000AH
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio:  mov ax,dados
         mov ds,ax

         LEA    SI,TAB
         PUSH   NUM
         PUSH   SI
         CALL   PREENC

FIM:     mov    ah,4ch
         int 21h

PREENC      PROC NEAR
         POP    SI      ;IP da instrução seguinte ao CALL
         POP    DI      ;recuperar endereço de TAB atraves da pilha
         POP    AX      ;recuperar NUM atraves da pilha
DIVID:           MOV    DX,0
         DIV    CONST
         MOV    [DI],DX
         CMP    AX,0
         JE     FINAL
         INC    DI
         JMP    DIVID

FINAL:   PUSH   SI      ;colocar IP de retorno na pilha
         RET
PREENC      ENDP

codigo  ends
end     inicio
```

6. Faça uma função que permita copiar para uma tabela todos os algarismos PARES de um número decimal cujo valor numérico se encontre compreendido entre [0,65535]. A ordem pela qual os algarismos são colocados na tabela deverá ser inversa à posição dos mesmos no número. O valor numérico e o endereço do array devem ser recebidos pela pilha. Por exemplo: considere o número 41302, o programa deverá preencher a tabela da seguinte forma:

2 0 4

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
num     dw      41302
tab     db      5 dup (0)
const   dW      000AH
const1  db      02H
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio: mov ax,dados
        mov ds,ax

        LEA     SI,TAB
        PUSH    NUM
        PUSH    SI
        CALL    PREENC

FIM:    mov     ah,4ch
        int 21h


PREENC      PROC NEAR
        POP     SI      ;IP da instrução seguinte ao CALL
        POP     DI      ;recuperar endereço de TAB atraves da pilha
        POP     AX      ;recuperar NUM atraves da pilha
DIVID:      MOV     DX,0
        DIV     CONST
        PUSH    AX      ;guarda registo AX na pilha temporariamente
        MOV     AX,DX
        DIV     CONST1
        CMP     AH,0
        JNE     IMPAR
        MOV     [DI],DL
        INC     DI
IMPAR:      POP     AX      ;recuperar o registo AX
        CMP     AX,0
        JNE     DIVID

FINAL:  PUSH    SI      ;colocar IP de retorno na pilha
```

```
          RET
PREENC          ENDP
codigo ends
end    inicio
```

7. Faça uma função que troque todas as letras minúsculas por maiúsculas, e vice-versa, de um vector de caracteres com endereço inicial vector. O endereço do vector deve ser recebido pela pilha.

Solução:

```
pilha   segment     para stack 'stack'
        db      1024    dup('pilha')
pilha   ends

dados   segment     para    'data'
nome    db      'aCAaciO AmaraL',0
dif     db      0
dados   ends

codigo  segment     para    'code'
        assume      cs:codigo,ds:dados,ss:pilha

inicio: mov ax,dados
        mov ds,ax

        LEA     SI,NOME
        PUSH    SI
        CALL    TROCA
        mov     ah,4ch
        int 21h

TROCA       PROC NEAR ; 'a'>'A'
        MOV     DIF,'a'
        SUB     DIF,'A'
        POP     DX      ; endereço retorno -> DX
        POP     DI      ; offset tabela

SALTO:      MOV     AL,[DI]
        CMP     AL,'A'
        JGE     MAIUS1      ; letra<'A' -> fim
        JMP     FIM

MAIUS1:     CMP     AL,'Z'
        JG      minus1      ; letra>'Z' -> minuscula
        ADD     AL,DIF
        MOV     [DI],AL
        JMP     FIM

minus1:     CMP     AL,'a'
        JGE     minus
        JMP     FIM

minus:  CMP     AL,'z'
        JG      FIM
        SUB     AL,DIF
        MOV     [DI],AL
        JMP     FIM

FIM:    INC     DI
```

```
        CMP   AL,0
        JNE   SALTO

        PUSH DX             ;endereço retorno -> DX
        RET
TROCA   ENDP


codigo ends
end    inicio
```

8. Faça um programa que lhe permita exibir no ecrã a hora do sistema. Deverá apresentar uma opção para sair do programa.

Solução:

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'
HORA DB       2 DUP('0'),'$'
MINUTO        DB      2 DUP('0'),'$'
SEGUNDO       DB      2 DUP('0'),'$'
STRING        DB      'HORA DO SISTEMA: ','$'
SAIR   DB     'SAIR - S',0DH,0AH,'$'
DIVI   DB     10D
DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

REPETE:

        MOV   AH,06H
        MOV   AL,00H
        MOV   CX,0000H
        MOV   DX,184FH
        MOV   BH,80H
        INT     10H


        MOV   AH,02H
        MOV   BH,00H
        MOV   DX,1418H
        INT     10H
        MOV   AH,09H
        LEA     DX,SAIR
        INT     21H


        MOV   AH,02H
        MOV   BH,00H
        MOV   DX,0918H
        INT     10H

        MOV   AH,09H
        LEA     DX,STRING
        INT     21H

        CALL  TR_HOR
        LEA     SI,HORA
```

```
        PUSH  SI
        CALL  IMPRIME

        MOV   AH,02H
        MOV   DL,'-'
        INT   21H

        LEA   SI,MINUTO
        PUSH  SI
        CALL  IMPRIME

        MOV   AH,02H
        MOV   DL,'-'
        INT   21H

        LEA   SI,SEGUNDO
        PUSH  SI
        CALL  IMPRIME

        MOV   AH,01H
        INT   21H
        CMP   AL,'S'
        JE    FIM
        JMP   REPETE

FIM:
        MOV   AH,06H        ;LIMPA ECRAN
        MOV   CX,0000H
        MOV   DX,184FH
        MOV   BH,00H
        INT   10H

        MOV    Ah,4CH
        INT   21H

; PARAMETROS DE ENTRADA -> PONTEIRO PARA STRING
IMPRIME PROC NEAR
        POP   DI      ; ENDEREÇO DE RETORNO (PILHA) -> DI
        POP   DX      ; ENDEREÇO DA STRING A IMPRIMIR -> DX

        MOV   AH,09H
        INT   21H

        PUSH  DI      ; DI -> PILHA
        RET
IMPRIME      ENDP

TR_HOR       PROC NEAR

        MOV   AH,2CH
        INT   21H
        ; HORA -> CH, MINUTO -> CL, SEGUNDO -> DH

        LEA   DI,HORA
        PUSH  DI      ;PONTEIRO PARA STRING -> PILHA
```

```
        MOV   AX,00
        MOV   AL,CH
        PUSH  AX      ;REGISTO COM DADOS -> PILHA
        CALL  CON_H_D

        LEA   DI,MINUTO
        PUSH  DI      ;PONTEIRO PARA STRING -> PILHA
        MOV   AX,00
        MOV   AL,CL
        PUSH  AX      ;REGISTO COM DADOS -> PILHA
        CALL  CON_H_D

        LEA   DI,SEGUNDO
        PUSH  DI      ;PONTEIRO PARA STRING -> PILHA
        MOV   AX,00
        MOV   AL,DH
        PUSH  AX      ;REGISTO COM DADOS -> PILHA
        CALL  CON_H_D

        RET
TR_HOR  ENDP


CON_H_D PROC NEAR
        POP   DI      ; ENDEREÇO DE RETORNO IP -> DI
        POP   AX      ; REGISTO COM (HORA MINUTO SEGUNDO) -> AL
        POP   SI      ; STRING A PREENCHER (HORA, MINUTO, SEGUNDO) -> SI

SALTO1: INC   SI
        MOV   BL,[SI]
        CMP   BL,'$'          ; COLOCA PONTEIRO NO FINAL DE STRING
        JNE   SALTO1
        DEC   SI              ; posicao si no caracter $ -> necessario DEC

SALTO:  DIV   DIVI
        MOV   BH,AH
        ADD   BH,30H                ; CONVERTER ALGARISMO -> ASCII
        MOV   [SI],BH
        MOV   AH,00H
        DEC   SI
        CMP   AL,0
        JNE   SALTO

FORAS:  PUSH  DI
        RET
CON_H_D ENDP

CODIGO  ends
end     INICIO
```

9. Faça um programa que lhe permita ler ou escrever num ficheiro de texto, uma frase com um número máximo de 50 caracteres, digitados através do teclado. Este programa deverá possuir um menu inicial onde o utilizador poderá optar por criar o ficheiro, ler ou sair do menu.

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
   DB  1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT    PARA 'DATA'
HANDLE      DW     ?
CONTEU      DB     50,2,48 DUP('Y')
STR_AUX     DB     50 DUP('X')
N_CAR       DB     2 DUP(?)

FICH   DB     'C:\DADOS.TXT',00   ; NECESSÁRIO O 0 NO FINAL
;menu inicial - menus
OP     DB     'ESCOLHA A OPCAO:_','$'
OP1    DB     'CRIAR FICHEIRO - 1',0AH,0DH,'$'
OP2    DB     'LER FICHEIRO - 2',0AH,0DH,'$'
OP3    DB     'SAIR - 3',0AH,0DH,'$'
;mensagens de erro
ER1    DB     'A CRIAR FICHEIRO',0AH,0DH,'$'
ER2    DB     'ERRO - ABRIR FICHEIRO',0AH,0DH,'$'
ER3    DB     'ERRO - LER FICHEIRO - SAIR DO PROGRAMA',0AH,0DH,'$'
ER4    DB     'ERRO - ESCREVER FICHEIRO - SAIR DO PROGRAMA',0AH,0DH,'$'
ER5    DB     'ERRO - FECHAR FICHEIRO - SAIR DO PROGRAMA',0AH,0DH,'$'
;outros menus
MENU1       DB     'DIGITE A FRASE A GUARDAR: ','$'
MENU2       DB     'CONTEUDO DO FICHEIRO: ','$'
MENU3       DB     'DIGITE OUTRO CARACTER','$'
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
   ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
   MOV         AX,DADOS
   MOV         DS,AX

INI:    CALL   LIMPA
   MOV         AX,0805H      ;       MENU          INICIAL          -          INICIO
**************************************
   PUSH        AX
   CALL        CURSOR
   MOV         AH,09H
   LEADX,OP3
   INT 21H

   MOV         AX,0705H
   PUSH        AX
   CALL        CURSOR
   MOV         AH,09H
```

```
    LEADX,OP2
    INT 21H
    MOV     AX,0605H
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,OP1
    INT 21H

    MOV     AX,0A05H
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,OP
    INT 21H          ;      MENU          INICIAL          -          FIM
*****************************************

    MOV     AH,01H
    INT 21H
    CMP     AL,'3'
    JE  FIM

MENS1:      CMP   AL,'1'
    JNEMENS2
    CALL    LIMPA       ;      MENU   ESCREVER   FICHEIRO   -   INICIO
    ******************************
    MOV     AX,0505H
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,MENU1
    INT 21H
    CALL    ESC_F
    MOV     AX,1505H
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,OP3
    INT 21H
    MOV     AH,01H
    INT 21H
    CMP     AL,'3'
    JE  INI
    JMP     MENS1

MENS2:      CALL  LIMPA       ;      MENU   LE   FICHEIRO   -   INICIO
    ******************************
    MOV     AX,0505H
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,MENU2
    INT 21H
    CALL    LE_F
    MOV     AX,1505H
```

```
    PUSH    AX
    CALL    CURSOR
    MOV     AH,09H
    LEADX,OP3
    INT 21H
    MOV     AH,01H
    INT 21H
    CMP     AL,'3'
    JE  INI
    JMP     MENS2

FIM:  CALL  LIMPA
    MOV     AH,4CH
    INT 21H


;FUNÇÕES                                                                    GRÁFICAS
*****************************************************************************************

CURSOR      PROC NEAR ; POSICIONA CURSOR      -> parametro de entrada - posicao cursor
    POP     DI
    POP     DX      ; recupera a posição do cursor
    MOV     AH,02H
    MOV     BH,00H
    INT 10H
    PUSH    DI
    RET
CURSOR      ENDP

LIMPA PROC NEAR ; LIMPA ECRAN
    MOV     AH,06H
    MOV     AL,00H
    MOV     CX,0000H
    MOV     DX,184FH
    MOV     BH,80H
    INT 10H

    RET
LIMPA ENDP

ESC_STR     PROC NEAR ; retorna numero de caracteres escritos pela pilha
    POP     DI
    MOV     AH,0AH
    LEADX,CONTEU
    INT 21H

    LEASI,CONTEU
    ADD     SI,2    ; SALTA OS DOIS CARACTERES INCIAIS
ESC_S1:     MOV   AL,[SI]
    CMP     AL,0DH
    JE  ESC_S2
    INC SI
    JMP     ESC_S1
```

```
ESC_S2:     MOV   AL,'$'
    MOV     [SI],AL
    SUB     SI,2

    PUSH    SI
    PUSH    DI
    RET
ESC_STR     ENDP

;FUNÇÕES DE ABERTURA/FECHO/ESCRITA NO FICHEIRO
************************************************************


ESC_F       PROC NEAR
    CALL    ABRE_F

    CALL    ESC_STR
    POP     CX
    MOV     AH,40H
    MOV     BX,HANDLE
    LEADX,CONTEU
    INT 21H
    JNC     E_F1
    MOV     AH,09H
    LEADX,ER4
    INT 21H
    JMP     FIM

E_F1:  CALL  FECHA_F
    RET
ESC_F       ENDP


LE_F   PROC NEAR
    CALL    ABRE_F

    MOV     AH,3FH          ;LE APENAS PARA VERIFICAR O NUMERO DE
CARACTERES DIGITADOS
    MOV     BX,HANDLE
    MOV     CX,0002H    ;LE APENAS DOIS CARACTERES O SEGUNDO
ARMAZENOU O TOTAL DE CARACTERES INTRODUZIDO
    LEADX,N_CAR
    INT 21H
    JNC     L_F1
    LEA     DX,ER3
    MOV     AH,09H
    INT 21H
    JMP     FIM

L_F1:  LEA    SI,N_CAR
    INC SI
    MOV     CX,0000H
    MOV     CL,[SI]
    MOV     AH,3FH          ;LE APENAS PARA VERIFICAR O NUMERO DE
CARACTERES DIGITADOS
```

```
    MOV         BX,HANDLE
    LEADX,STR_AUX
    INT 21H
    JNC         L_F2
    LEA         DX,ER3
    MOV         AH,09H
    INT 21H
    JMP         FIM


L_F2:  LEA     SI,STR_AUX
    ADD         SI,CX
    MOV         AL,'$'
    MOV         [SI],AL

    MOV         AH,09H
    LEADX,STR_AUX
    INT 21H
    CALL        FECHA_F
    RET
LE_F   ENDP

ABRE_F        PROC NEAR

A_F2:  MOV    AH,3DH        ; PRIMEIRO TENTA ABRIR FICHEIRO
    MOV         AL,02H
    LEADX,FICH
    INT 21H
    MOV         HANDLE,AX
    JNC         A_F1
    MOV         AH,09H
    LEADX,ER1
    INT 21H     ; SE NÃO FOR POSSÍVEL PODE NAO EXISTIR - CRIAR FICHEIRO
    MOV         AH,3CH        ; TENTA CRIAR FICHEIRO
    MOV         CX,0000H
    LEADX,FICH
    INT 21H
    MOV         HANDLE,AX
    JNC         A_F2
    MOV         AH,09H
    LEADX,ER2
    INT 21H     ; SE NÃO FOR POSSÍVEL PODE NAO EXISTIR - CRIAR FICHEIRO
    JMP         FIM

A_F1:
    RET
ABRE_F        ENDP


FECHA_F       PROC NEAR
    MOV         AH,3EH
    MOV         BX,HANDLE
    INT 21H
    JNC         F_F1
    MOV         AH,09H
```

```
    LEADX,ER5
    INT 21H
    JMP         FIM
F_F1:
    RET
FECHA_F     ENDP

CODIGO    ends
end     INICIO
```

10. Faça um programa que permite posicionar o cursor em qualquer posição do ecrã (as teclas de controlo deverão ser as teclas: ←, ↑, → e ↓). O programa deverá igualmente indicar a posição da linha e coluna do cursor. A tecla 'S' deverá permitir a saída do programa.

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
POSMX       DB      00H
POSMY       DB      00H
STRING1     DB      'UTILIZE AS TECLAS COM SETAS PARA CONTROLAR A
POSICAO DO CURSOR',0AH,0DH,'$'
STRING2     DB      'PARA SAIR DIGITE S ','$'
STRING3     DB      'POSICAO DO CURSOR',0AH,0DH,'$'
POSX  DB    'POSICAO X-> ','$'
POSY  DB    '       ; POSICAO Y-> ','$'
TABX  DB    '00','$'
TABY  DB    '00','$'
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        CALL  LIMPAR
        MOV   AH,02H
        MOV   BH,00H
        MOV   DH,00H
        MOV   DL,00H
        INT   10H
        MOV   AH,09H
        LEA   DX,STRING1
        INT   21H
        MOV   AH,09H
        LEA   DX,STRING2
        INT   21H

        CALL  POSICIONA_CURSOR

FIM:    CALL  LIMPAR
        MOV   AH,4CH
        INT   21H


LIMPAR      PROC NEAR
        MOV   AH,06H
        MOV   CX,0000H
        MOV   DX,184FH
```

```
        MOV   BH,70H
        MOV   AL,00H
        INT   10H
        RET
LIMPAR  ENDP


POSICIONA_CURSOR    PROC NEAR

        MOV   AH,02H
        MOV   BH,00H
        MOV   DH,02H
        MOV   DL,00H
        INT   10H
        LEA   DX,STRING3
        MOV   AH,09H
        INT   21H
        LEA   DX,POSX
        MOV   AH,09H
        INT   21H
        LEA   DX,POSY
        MOV   AH,09H
        INT   21H

REPETE: MOV   DH,03H
        MOV   DL,0BH
        MOV   BH,00H
        MOV   AH,02H
        INT   10H
        MOV   AX,0000H
        MOV   AL,POSMX
        PUSH  AX
        LEA   SI,TABX
        PUSH  SI
        CALL  CONVERTE
        LEA   DX,TABX
        MOV   AH,09H
        INT   21H

        MOV   DH,03H
        MOV   DL,22H
        MOV   BH,00H
        MOV   AH,02H
        INT   10H
        MOV   AX,0000H
        MOV   AL,POSMY
        PUSH  AX
        LEA   SI,TABY
        PUSH  SI
        CALL  CONVERTE
        LEA   DX,TABY
        MOV   AH,09H
        INT   21H

        MOV   DH,POSMX
```

```
        MOV   DL,POSMY
        MOV   BH,00H
        MOV   AH,02H
        INT   10H


        MOV   AX,0FFFFH
        MOV   AH,00H
        INT   16H
        CMP   AL,00H
        JE    TECLA_D
        CMP   AL,'S'
        JE    FIM
        JMP   REPETE

TECLA_D:
        CMP   AH,4DH
        JNE   TECLA_E
        INC   POSMY
        JMP   TES

TECLA_E:
        CMP   AH,4BH
        JNE   TECLA_C
        DEC   POSMY
        JMP   TES

TECLA_C:
        CMP   AH,48H
        JNE   TECLA_B
        DEC   POSMX
        JMP   TES

TECLA_B:
        CMP   AH,50H
        JNE   REPETE
        INC   POSMX

TES:    MOV   AH,02H
        MOV   BH,00H
        MOV   DH,POSMX
        MOV   DL,POSMY
        CMP   POSMY,79
        JBE   TES1
        MOV   POSMY,00H
TES1:
        CMP   POSMX,24
        JBE   TES2
        MOV   POSMX,00H

TES2:   INT   10H
        JMP   REPETE

        RET
POSICIONA_CURSOR        ENDP
```

```
CONVERTE   PROC NEAR
        POP    DI
        POP    SI       ;ENDEREÇO DA TABX OU TABY
        POP    AX       ;ARGUMENTO ENTRADA POSMX OU POSMY EM AL
        MOV    DL,00H
        MOV    [SI],DL
        INC    SI
        MOV    CX,2
CONV1:  MOV    BH,10
        MOV    AH,00H
        DIV    BH
        ADD    AH,30H
        MOV    [SI],AH
        DEC    SI
        CMP    AL,00H
        JE     CONV2
        LOOP   CONV1

CONV2:
        PUSH   DI
        RET
CONVERTE   ENDP

CODIGO    ends
end     INICIO
```

11. Faça um programa que permita representar no ecrã uma tabela (3 x 3), que se encontra em memória. Deverá respeitar a seguinte formatação. Utilize a tecla 'S' para sair do programa.

El_1	El_2	El_3
El_4	El_5	El_6
El_7	El_8	El_9

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
TABELA      DB      1,2,3,4,5,6,7,8,9
MENU DB     'DIGITE QUALQUER CARACTER PARA SAIR','$'
POSX DB     05H
POSY DB     05H
TAMX DB     ?
TAMY DB     ?
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        CALL  LIMPAR
        MOV   AX,0000H
        PUSH  AX
        CALL  POS_XY
        MOV   AH,09H
        LEA   DX,MENU
        INT   21H

        CALL  IMPRIME

        MOV   AH,00H
        INT   16H

FIM:    CALL  LIMPAR
        MOV   AH,4CH
        INT   21H


IMPRIME     PROC NEAR
        MOV   AH,POSX
        MOV   AL,POSY
        PUSH  AX
        CALL  POS_XY

        MOV   AH,09H        ;CRIA LINHA
```

```
        MOV   AL,'-'
        MOV   BH,00H
        MOV   CX,13
        MOV   BL,70H
        INT   10H

        INC   POSX
        MOV   AH,POSX
        MOV   AL,POSY
        PUSH  AX
        CALL  POS_XY

        LEA   SI,TABELA
        MOV   TAMX,3
        MOV   TAMY,3

        MOV   DL,'|'
        MOV   AH,02H
        INT   21H


REPE1:  MOV   DL,' '
        MOV   AH,02H
        INT   21H
        MOV   DL,[SI]
        ADD   DL,30H
        MOV   AH,02H
        INT   21H
        MOV   DL,' '
        MOV   AH,02H
        INT   21H
        MOV   DL,'|'
        MOV   AH,02H
        INT   21H
        INC   SI
        DEC   TAMX
        MOV   CL,TAMX
        CMP   CL,00H
        JNE   REPE1
        MOV   TAMX,03H
        DEC   TAMY
        INC   POSX
        MOV   AH,POSX
        MOV   AL,POSY
        PUSH  AX
        CALL  POS_XY
        MOV   AH,09H        ;CRIA LINHA
        MOV   AL,'-'
        MOV   BH,00H
        MOV   CX,13
        MOV   BL,70H
        INT   10H
        MOV   CH,TAMY
        CMP   CH,00H
        JE    REPE2
```

```
        INC     POSX
        MOV     AH,POSX
        MOV     AL,POSY
        PUSH    AX
        CALL    POS_XY
        MOV     DL,'|'
        MOV     AH,02H
        INT     21H
        JMP     REPE1
REPE2:
        RET
IMPRIME         ENDP

POS_XY          PROC NEAR
        POP     DI
        POP     DX
        MOV     AH,02H
        MOV     BH,00H
        INT     10H
        PUSH    DI
        RET
POS_XY          ENDP



LIMPAR          PROC NEAR
        MOV     AH,06H
        MOV     CX,0000H
        MOV     DX,184FH
        MOV     BH,70H
        MOV     AL,00H
        INT     10H
        RET
LIMPAR          ENDP



CODIGO    ends
end     INICIO
```

12. Faça um programa que permita contabilizar o número de segundos que o utilizador demora a digitar uma frase com o número máximo de 100 caracteres.

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT    PARA 'DATA'
MINUTOS       DB      00H
SEGUNDOS      DB      00H
TEMPO_S1              DW      00
TEMPO_S2              DW      00
CONSTANTE DB      60
T_SEGUNDOS            DB      4 DUP(0),'$'
DECIMAL               DW      000AH
FRASE1                DB      'DEMOROU ','$'
FRASE                 DB      100,0,100 DUP(?)
FRASE2                DB      ' SEGUNDOS','$'
DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        CALL  LIMPA
        MOV   AX,0000H
        PUSH  AX
        CALL  POSXY
        MOV   AH,2CH
        INT     21H
        CALL  CONVERTE_SEGUNDOS
        MOV   TEMPO_S1,AX

        LEA     DX,FRASE
        MOV   AH,0AH
        INT     21H

        MOV   AH,2CH
        INT     21H
        CALL  CONVERTE_SEGUNDOS
        MOV   TEMPO_S2,AX

        MOV   BX,TEMPO_S1
        MOV   AX,TEMPO_S2
        SUB     AX,BX
        CALL  CONVERTER_DECIMAL_ASCII
        MOV   AX,0100H
        PUSH  AX
        CALL  POSXY
```

```
        MOV   AH,09H
        LEA   DX,FRASE1
        INT   21H
        MOV   AH,09H
        LEA   DX,T_SEGUNDOS
        INT   21H
        MOV   AH,09H
        LEA   DX,FRASE2
        INT   21H

        MOV   AH,2CH
        INT   21H

FIM:
        MOV   AH,4CH
        INT   21H

CONVERTE_SEGUNDOS   PROC NEAR
        MOV   MINUTOS,CL
        MOV   SEGUNDOS,DH
        MOV   AH,00H
        MOV   AL,CL
        MUL   CONSTANTE
        MOV   BX,0000H
        MOV   BL,SEGUNDOS
        ADD   AX,BX
        RET
CONVERTE_SEGUNDOS   ENDP

CONVERTER_DECIMAL_ASCII     PROC NEAR

              LEA   SI,T_SEGUNDOS
              ADD   SI,3
REPETE1:      MOV   DX,0000H
              DIV   DECIMAL
              MOV   [SI],DL
              DEC   SI
              CMP   AX,0000H
              JNE   REPETE1

REPETE3:      LEA   SI,T_SEGUNDOS
REPETE2:      MOV   AL,[SI]
              CMP   AL,'$'
              JE    FIM1
              ADD   AL,30H
              MOV   [SI],AL
              INC   SI
              JMP   REPETE2

FIM1:  RET
CONVERTER_DECIMAL_ASCII     ENDP

LIMPA PROC NEAR
        MOV   AH,06H
        MOV   CX,0000H
```

```
        MOV   DX,184FH
        MOV   AL,0000H
        MOV   BH,70H
        INT   10H
        RET
LIMPA ENDP

POSXY       PROC NEAR
        POP   DI
        POP   DX
        MOV   AH,02H
        MOV   BH,00H
        INT   10H
        PUSH  DI
        RET
POSXY       ENDP

CODIGO   ends
end    INICIO
```

13. Faça um programa que permita contabilizar o número de repetições do carácter mais vezes repetido numa cadeia de caracteres, composta unicamente por letras minúsculas. A cadeia de caracteres deve ser introduzida pelo teclado e o carácter mais vezes repetido assim como o número de vezes que se repete deve ser exibido no ecrã. Considere que o tamanho máximo da cadeia caracteres introduzida pelo teclado é de 18 caracteres. (caso exista mais do que um carácter mais digitado, deverá optar por escolher o último)

Exemplos:

i) "fernando pessoa" ↘ os caracteres mais digitados são: 'e', 'n', 'o', 's' e 'a' deverá escolher o 'a', pois aparece depois de 'e', 'n', 'o' e 's'.

ii) "julio dinis" ↘ o carácter mais digitados é: 'i'.

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'
        frase1 db 'Digite uma frase com um máximo de 18 caracteres : ','$'
        frase2 db       20,0,18 dup('x')
        frase3 db 'O caracter mais digitado foi -> ','$'
        frase4 db 'e repete-se  -> ','$'
        frase5 db 'vezes','$'
        N_rep_car db 18 dup(0),'$'    ; array que fica com o número de vezes que cada
caracter se repete
        conta   db      0
        pos     dw      0
        rep_max         db      0
        max     db      0
        car_max         db      0
DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV     AX,DADOS
        MOV     DS,AX

        call    limpa
        mov     dx,0105h
        call    posiciona
        call    pede_frase
        call    contar
        call    encontra

        mov     dx,0305h
        call    posiciona
        lea     dx,frase3
        mov     ah,09h
        int     21h
```

```
        mov     ah,02h
        mov     dl,car_max
        int     21h

        mov     dx,0405h
        call    posiciona
        lea     dx,frase4
        mov     ah,09h
        int     21h
        mov     ah,02h
        mov     dl,rep_max
        add     dl,30h
        int     21h
        mov     dx,0417h
        call    posiciona
        lea     dx,frase5
        mov     ah,09h
        int     21h

        mov     dx,1500h
        call    posiciona

        mov     ah,4ch
        int     21h


pede_frase      proc
        mov     ah,09h
        lea     dx,frase1
        int     21h

        mov     ah,0ah
        lea     dx,frase2
        int     21h

        ret
pede_frase      endp


le_frase        proc
        lea     si,frase2
        add     si,2

repet:  mov     dl,[si]
        inc     si
        cmp     dl,0dh
        je      sai
        mov     ah,02h
        int     21h
        jmp     repet

        mov     ah,0ah
        lea     dx,frase2
        int     21h
```

```
sai:    ret
le_frase        endp

;função contar cria um array onde são colocadas o número de repetições de cada
caracter
; exemplo:   acacio amaral
; N_rep_car: 5252111515151

contar  proc
        lea     si,frase2
        lea     bx,N_rep_car
        add     si,2
        mov     di,si
        mov     al,[si]

repet1: mov     dl,[si]
        inc     si
        cmp     dl,0dh
        je      sai1
        cmp     dl,al
        jne     segue
        inc     conta
segue:  jmp     repet1

sai1:   mov     si,di
        inc     pos
        cmp     al,0dh
        je      sai
        mov     al,conta
        mov     [bx],al  ; coloca no array o numero de vezes que cada caracter se repete
        inc     bx
        mov     conta,0
        add     si,pos   ; coloca na posicao do caracter seguinte a testar
        mov     al,[si]
        sub     si,pos   ; posiciona o ponteiro novamente no ínicio do array introduzido
pelo teclado
        jmp     repet1

sai:    ret
contar  endp

;função encontra o maior algarismo do array com o numero de repetições: N_rep_car

encontra        proc
        lea     si,N_rep_car ; array termina com $, posicoes sem algarismo
(correspondente  caracter) possuem 0
        lea     di,frase2    ;
        add     di,2          ; não contar com os dois caracteres iniciais

ciclo:  mov     al,[si]      ; numero de repeticoes do caracter
        inc     si
        mov     ah,[di]      ; caracter
        inc     di
        cmp     al,'$'
        je      sai
```

```
        cmp     rep_max,al
        ja      salta
        mov     rep_max,al
        mov     car_max,ah
salta:  jmp     ciclo

sai:    ret
encontra        endp

limpa   proc
        mov     ah,06h ; limpa ecran
        mov     al,00h
        mov     cx,0000h
        mov     dx,184fh
        mov     bh,90h
        int     10h
        ret
limpa   endp


posiciona       proc
        mov     ah,02h
        mov     bh,00h
        int     10h
        ret
posiciona       endp


CODIGO  ends
end     INICIO
```

14. Faça um programa que, recorrendo à memória de vídeo, apresente no monitor um conjunto de listas verticais de cor branco e preto, alternadamente.

Solução:

```
PILHA SEGMENT PARA STACK     'STACK'
    DB  1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'

DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
    ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
    MOV         AX,DADOS
    MOV         DS,AX

    MOV         AX,0B800H
    MOV         ES,AX

    MOV         DI,0000H
    MOV         CX,40*25
    MOV         AL,' '

REPETE:       MOV    AH,11110000B
    MOV         ES:[DI],AL
    MOV         ES:[DI+1],AH
    ADD         DI,2
    MOV         AH,00001111B
    MOV         ES:[DI],AL
    MOV         ES:[DI+1],AH
    ADD         DI,2
    LOOP        REPETE

    MOV         AH,01H
    INT 21H

FIM:
    MOV         AH,4CH
    INT 21H

CODIGO    ends
end     INICIO
```

15. Faça um programa que, recorrendo à memória de vídeo, apresente no monitor um conjunto de listas horizontais de cor branco e preto, alternadamente.

Solução:

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'

DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        MOV   AX,0B800H
        MOV   ES,AX

        MOV   DI,0000H
        MOV   CX,80*25
        MOV   BL,' '
        MOV   BH,0F0H
REPETE:       MOV   ES:[DI],BL
        MOV   ES:[DI+1],BH
        ADD   DI,2
        MOV   AX,DI
        MOV   DL,160
        DIV   DL
        CMP   AH,00H
        JNE   SALTO
        CMP   BH,0F0H
        JE    SEGU
        MOV   BH,0F0H
        JMP   SALTO
SEGU:MOV   BH,0FH
SALTO:        LOOP  REPETE

        MOV   AH,01H
        INT     21H

FIM:
        MOV   AH,4CH
        INT     21H

CODIGO    ends
end     INICIO
```

16. Faça um programa que, recorrendo à memória de vídeo, apresente no monitor um padrão em formato xadrez a preto e branco.

Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
REPETICOES          DB      24
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        MOV   AX,0B800H
        MOV   ES,AX

        MOV   DI,0000H

        CALL  LINHA_PAR

REPET:      CALL  LINHA_IMPAR
        CALL  LINHA_PAR
        DEC   REPETICOES
        MOV   AH,REPETICOES
        CMP   AH,00H
        JNE   REPET

        MOV   AH,00H
        INT   16H

FIM:
        MOV   AH,4CH
        INT   21H

LINHA_PAR   PROC NEAR

        MOV   CX,40
        MOV   BL,' '
        MOV   BH,0F0H

REP_P:      MOV   BH,0F0H
        MOV   ES:[DI],BL
        MOV   ES:[DI+1],BH
        ADD   DI,2
        MOV   BH,0FH
        MOV   ES:[DI],BL
        MOV   ES:[DI+1],BH
        ADD   DI,2
```

```
        LOOP  REP_P

RET

LINHA_PAR   ENDP


LINHA_IMPAR         PROC NEAR

        MOV   CX,40
        MOV   BL,' '

REP_I:MOV   BH,0FH
        MOV   ES:[DI],BL
        MOV   ES:[DI+1],BH
        ADD   DI,2
        MOV   BH,0F0H
        MOV   ES:[DI],BL
        MOV   ES:[DI+1],BH
        ADD   DI,2
        LOOP  REP_I

RET

LINHA_IMPAR         ENDP


CODIGO    ends
end     INICIO
```

17. Faça um programa que lhe permita representar o array bidimensional:

```
array   db   1,4,0,2,5,6,0,8,9
        db   1,2,0,0,5,0,7,8,0
        db   0,3,2,0,0,6,7,8,9
        db   1,2,3,0,0,6,7,0,9
        db   1,0,3,4,0,6,7,0,9
        db   1,2,0,4,5,0,7,0,9
        db   1,0,3,4,5,6,2,0,0
        db   1,0,3,0,5,6,7,2,9
        db   1,2,3,4,0,0,7,8,0
```

Tal que:

a) O algarismo 0 deve ser substituído por um espaço em branco.
b) O array deverá ser inserido numa tabela, cujos limitadores deverão estar igualmente definidos num array auxiliar. Exemplo:

```
Tabela  db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
        db   '| | | | | | | | | |'
        db   '---------------------'
```

o resultado final deverá assemelhar-se a:

```
---------------------
|1|4|  |2|5|6|0|8|9|
---------------------
|1|2|  |  |5|  |7|8|  |
---------------------
|  |3|2|  |  |6|7|8|9|
---------------------
|1|2|3|  |  |6|7|  |9|
---------------------
|1|  |3|4|  |6|7|  |9|
---------------------
|1|2|  |4|5|  |7|  |9|
---------------------
|1|  |3|4|5|6|2|  |  |
---------------------
|1|  |3|  |5|6|7|2|9|
---------------------
|1|2|3|4|  |  |7|8|  |
---------------------
```

Solução:

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS           SEGMENT     PARA 'DATA'
tabela  db      1,4,0,2,5,6,0,8,9
        db      1,2,0,0,5,0,7,8,0
        db      0,3,2,0,0,6,7,8,9
        db      1,2,3,0,0,6,7,0,9
        db      1,0,3,4,0,6,7,0,9
        db      1,2,0,4,5,0,7,0,9
        db      1,0,3,4,5,6,2,0,0
        db      1,0,3,0,5,6,7,2,9
        db      1,2,3,4,0,0,7,8,0

tabela1         db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
        db      '| | | | | | | | | |'
        db      '-------------------'
```

```
linha1  db      19
linha   db      9
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA


INICIO:
        MOV   AX,DADOS
        MOV   DS,AX


        call    limpa
        call    escreve_tab1
        call    escreve_tab

        mov     dx,1700h
        mov     bl,0000h
        mov     ah,02h
        int     10h

        mov     ah,01h
        int     21h

        MOV   AH,4CH
        INT     21H

escreve_tab1 proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di     ; posicao inicial linha=2 e coluna=4
        add     di,160  ; uma linha
        add     di,160  ; segunda linha
        add     di,8      ; 4 colunas
        mov     cx,19*19        ; total de caracteres
        lea     si,tabela1

ciclo2: mov     al,[si]
        inc     si
        mov     es:[di],al
        inc     di
        mov     al,0A5H
        mov     es:[di],al
        inc     di
        dec     linha1
        cmp     linha1,0
        jne     segue
        mov     linha1,19
        add     di,122  ;160-38
segue:  loop    ciclo2

        ret
escreve_tab1 endp
```

```
limpa   proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di
        mov     cx,160*25
ciclo1:  mov    al,' '
        mov     es:[di],al
        inc     di
        mov     al,0E2h
        mov     es:[di],al
        inc     di
        loop    ciclo1
        ret
limpa   endp

escreve_tab   proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di     ; posicao inicial linha=3 e coluna=5
        add     di,160  ; uma linha
        add     di,160  ; segunda linha
        add     di,160  ; terceira linha
        add     di,10   ; 5 colunas
        mov     cx,81   ; total de caracteres (tabela)
        lea     si,tabela

ciclo2: mov     al,[si]
        inc     si
        cmp     al,0
        jne     segue2
        mov     al,' '
        jmp     segue3
segue2:         add     al,30h
segue3:         mov     es:[di],al
        inc     di
        mov     al,0C0H
        mov     es:[di],al
        inc     di
        dec     linha
        cmp     linha,0
        jne     segue
        mov     linha,9
        add     di,160  ; 1 linha(160) + 160-34
        add     di,126
        jmp     segue1
segue:  add     di,2
segue1:         loop    ciclo2

        ret
escreve_tab   endp


CODIGO    ends
end     INICIO
```

18. Faça um programa que lhe permita copiar para um array numérico um dos arrays existentes num ficheiro de texto com a identificação jx: . A escolha do array a copiar deverá ser aleatória.

Por exemplo considere que o ficheiro de texto possui a seguinte informação:

j0: 123456789
j1: 234567891
j2: 345678912
j3: 456789123
j4: 567891234
j5: 678912345
j6: 789123456

Se o valor aleatório gerado fosse 5 então o array em memória deveria armazenar os seguintes algarismos:

Array → 6,7,8,9,1,2,3,4,5

Solução:

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
Jogo        dw      ?
tabela      db      9 dup(0)
ficheiro    db      'exemplo.txt',00
erro_a      db      'erro   de abertura ficheiro','$'
erro_f      db      'erro   a fechar ficheiro','$'
erro_l      db      'erro   a ler ficheiro','$'
tam_f       dw      103
tabela_aux  db      103     dup(?)
handle      dw      ?
NJ          dw      6       ;nº de jogos
tam_tabela  dw      9
tam_linha   dw      15
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        mov ah,0
        int 1Ah             ;Interupção que le o Relogio do Sistema
        mov   ax,dx             ;Dx fica com o contador de Ticks (0-18206)
        mov   dx,0000H
        div   NJ
        mov   jogo,dx
        call  abre_fich
        call  le_fich
```

```
        call    fech_fich

; como cada linha do ficheiro (tabuleiro_aux) possui 14 caracteres (contar com o 0DH)
; e o inicio do tabuleiro não começa na primeira coluna mas sim na 5 (contar com o 0AH)

        lea     si,tabela_aux
        mov     dx,0000H
        mov     ax,jogo
        mul     tam_linha
        add     ax,4
        add     si,ax           ; posicao de leitura do array = linhaxnº de colunas(14)+4(ignorar 'j1: ')

        mov     cx,tam_tabela
        lea     di,tabela

ciclo:  mov     al,[si]
        mov     [di],al
        inc     di
        inc     si
        loop    ciclo

fim:    MOV     AH,4CH
        INT     21H

abre_fich       proc
        mov     ah,3dh
        lea     dx,ficheiro
        mov     al,00
        int     21h
        jnc     segue
        lea     dx,erro_a
        mov     ah,09h
        int     21h
        jmp     fim

segue:  mov     handle,ax
        ret
abre_fich       endp

fech_fich       proc
        mov     ah,3eh
        mov     bx,handle
        int     21h
        jnc     segue
        lea     dx,erro_f
        mov     ah,09h
        int     21h
        jmp     fim

segue:

        ret
```

```
fech_fich       endp

le_fich proc
        mov     ah,3fh
        mov     bx,handle
        mov     cx,tam_f
        lea     dx,tabela_aux
        int     21h
        jnc     segue
        lea     dx,erro_l
        mov     ah,09h
        int     21h
        jmp     fim

segue:
        ret
le_fich endp

CODIGO    ends
end     INICIO
```

19. Faça um programa, em *assembly*, que a partir de uma *string* cujo último carácter é '$' e armazenada num vector com tamanho máximo de 25 bytes, apresente no ecrã, recorrendo à memória de vídeo, modo de texto 80x25, o conteúdo desse vector 4 vezes. Assim,

1. Deverá representar o vector de baixo para cima, na vertical, em que o primeiro carácter deve ser colocado na linha 25 e coluna 1.
2. Deverá representar o vector, apenas com letras maiúsculas, de cima para baixo, na vertical, em que o primeiro carácter deve ser colocado na linha 1 e coluna 25.
3. Deverá representar o vector, apenas com letras minúsculas, da esquerda para a direita, na horizontal, em que o primeiro carácter deve ser colocado na linha 1 e coluna 1.
4. Deverá representar o vector, da direita para a esquerda (em que as letras maiúsculas do vector inicial devem ser substituídos por maiúsculas e vice-versa), na horizontal, em que o primeiro carácter deve ser colocado na linha 25 e coluna 25

Exemplo: string ↘ "poRschE carRERa gt"

p	o	r	s	c	h	e		c	a	r	r	e	r	a		g	t						P
																							O
																							R
																							S
																							C
																							H
																							E
t																							
g																							C
																							A
a																							R
R																							R
E																							E
R																							R
r																							A
a																							
c																							G
																							T
E																							
h																							
c																							
s																							
R																							
o																							
p							T	G		A	r	e	r	R	A	C		e	H	C	S	r	O	P

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'
        frase1 db 'poRschE carRERa gt','$'
        diff      db       0
DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        mov     al,'a'
        mov     bl,'A'
        sub     al,bl
        mov     diff,al   ; calcular diferença entre Maiúsculas e minusculas

        call    limpa
        call    primeiro
        call    segundo
        call    terceiro
        call    quarto

        mov     dx,1900h
        mov     ah,02h
        mov     bh,00h
        int     10h

        mov     ah,01h
        int     21h

        mov     ah,4ch
        int     21h

;Deverá representar o vector de baixo para cima, na vertical,
;em que o primeiro carácter deve ser colocado na linha 25 e coluna 1.
primeiro        proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di
        lea     si,frase1

        mov     di,24*160       ; primeiro caracter linha 25 ->25x160=4000
ciclo:  mov     al,[si]
        cmp     al,'$'
        je      sai
        inc     si
        mov     es:[di],al
        inc     di
        mov     ah,0ah
        mov     es:[di],ah
```

```
        inc     di
        sub     di,162
        jmp     ciclo

sai:    ret
primeiro        endp

;Deverá representar o vector, apenas com letras maiúsculas, de cima para baixo,
;na vertical, em que o primeiro carácter deve ser colocado na linha 1 e coluna 25.
segundo         proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di
        lea     si,frase1

        mov     di,25*2
ciclo:  mov     al,[si]
        cmp     al,'$'
        je      sai
        inc     si
        cmp     al,'z'    ; caracter>z -> (mantem)
        ja      segue
        cmp     al,'a'    ; caracter<a -> (mantem)
        jb      segue
        sub     al,diff
segue:  mov     es:[di],al
        inc     di
        mov     ah,0ah
        mov     es:[di],ah
        inc     di
        add     di,158
        jmp     ciclo

sai:    ret
segundo         endp

;Deverá representar o vector, apenas com letras minúsculas, da esquerda para a
direita, na horizontal,
;em que o primeiro carácter deve ser colocado na linha 1 e coluna 1.
terceiro        proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di
        lea     si,frase1

        mov     di,0
ciclo:  mov     al,[si]
        cmp     al,'$'
        je      sai
        inc     si
        cmp     al,'Z'    ; caracter>Z -> (mantem)
        ja      segue
        cmp     al,'A'    ; caracter<A -> (mantem)
        jb      segue
        add     al,diff
```

```
segue: mov    es:[di],al
       inc    di
       mov    ah,0ah
       mov    es:[di],ah
       inc    di
       jmp    ciclo

sai:   ret
terceiro       endp

;Deverá representar o vector, da direita para a esquerda (em que as letras maiúsculas
do vector inicial devem ser substituídos por maiúsculas e vice-versa),
;na horizontal, em que o primeiro carácter deve ser colocado na linha 25 e coluna 25
quarto proc
       mov    ax,0b800h
       mov    es,ax
       xor    di,di
       lea    si,frase1

       mov    di,160*24      ;      linha 25
       add    di,25*2        ;      coluna 25

ciclo: mov    al,[si]
       cmp    al,'$'
       je     sai
       inc    si

       ;testa maiusculas
       cmp    al,'Z'
       ja     segue1
       cmp    al,'A'
       jb     segue1
       add    al,diff
       jmp    segue
segue1:       cmp    al,'z'
       ja     segue
       cmp    al,'a'
       jb     segue
       sub    al,diff
       jmp    segue
segue: mov    es:[di],al
       inc    di
       mov    ah,0ah
       mov    es:[di],ah
       inc    di
       sub    di,4
       jmp    ciclo

sai:   ret

       ret
quarto endp


limpa          proc
```

```
        mov     ax,0b800h
        mov     es,ax
        mov     di,0
        mov     cx,160*25

ciclo:  mov     al,' '
        mov     es:[di],al
        inc     di
        mov     al,0a6h
        mov     es:[di],al
        inc     di
        loop    ciclo

        ret
limpa           endp


CODIGO    ends
end     INICIO
```

(Pergunta do exame de T.A.C. de 27 de Julho de 2011)

20. Realize um programa, em *Assembly*, que permita calcular e guardar as médias semanais de quilómetros percorridos por um táxi. Assuma a existência de um vector, *VectorIni*, onde estão representados a quantidade de quilómetros percorridos diariamente referentes a várias semanas, sendo o final do vector representado pelo valor '-1'. Considere que o registo de quilómetros é feito apenas uma vez por dia, que uma semana tem 7 dias e que o vector *VectorIni* possui registos de x semanas completas (considere que o número máximo de semanas que podem ser guardadas é de 7). Assuma que que o número máximo de quilómetros percorridos diariamente é inferior a 255. Os resultados deverão ser armazenados noutro vector, *VectorFin*, em que cada elemento conterá uma média semanal. As médias também serão representadas em quilómetros e deverão ser arredondadas para o inteiro mais próximo.

Solução:

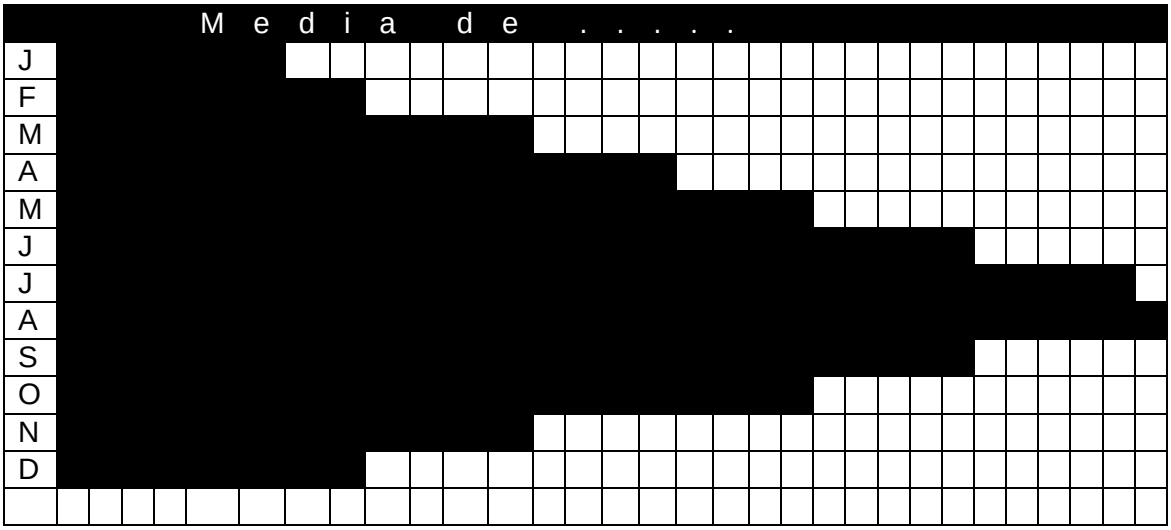
```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
vectorini   db      100,120,200,180,130,90,60
            db      110,120,209,180,130,90,167
            db      140,104,207,118,183,190,65
            db      99,110,123,186,138,95,160
            db      99,110,123,186,138,95,160
            db      99,110,123,186,138,95,160
            db      -1

vectorfim   db      7 dup(0)
cont        db      7
kmsemana    dw      0
lim         dw      0
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX


        xor     bx,bx

l1:     mov     al,vectorini[bx]
        inc     bx
        cmp     al,-1
        jne     l1

        mov     ax,bx
        div     cont
        mov     ah,00h
        mov     lim,ax  ; nº total de semanas guardadas
```

```
        xor     si,si
        xor     di,di

repete: mov     ax,0000h
        mov     al,vectorini[si]
        inc     si
        add     kmsemana,ax
        dec     cont
        cmp     cont,0
        jne     repete

        cmp     di,lim
        jae     segue

        mov     cont,7
        mov     ax,kmsemana
        mov     kmsemana,0000h
        div     cont
        cmp     ah,3
        ja      segue1
        mov     vectorfim[di],al
        inc     di
        jmp     repete

segue1:         add     al,1
        mov     vectorfim[di],al
        inc     di
        jmp     repete


segue: mov      ah,4ch
        int     21h

CODIGO    ends
end     INICIO
```

21. Faça um programa, em *assembly*, que recorrendo a memória de vídeo lhe permita representar o gráfico das médias das temperaturas mensais ao longo de 2 anos consecutivos.

- O gráfico deverá ser colocado na vertical na posição (1,1)
- Cada linha do gráfico representa a temperatura média de um mês.
- Os meses deverão ser indicados à esquerda da linha respectiva através da sua primeira letra (maiúscula), deverá criar um array com 12 caracteres para o efeito.
- Deve colocar o título do gráfico, centrado, na primeira linha: "Media de temperaturas ao longo de dois anos consecutivos - 2010 - 2011"
- As médias das temperaturas mensais são números inteiros que se encontram guardados no vector mediaTEMP.

Por exemplo: considere a título de exemplo apenas as médias ao longo de um ano, assim para um vector mediaTEMP = 6,8,12,16,20,25,30,31,25,20,12,9, o programa deveria exibir no ecrã o gráfico:



Solução:

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'
        titulo          db      '    Media de temperaturas ao longo de dois anos
consecutivos - 2010 - 2011    ',0
        mediaTEMP   db
        6,8,12,16,20,25,30,31,25,20,12,9,6,8,12,16,20,25,30,31,25,20,12,9,'x'
        meses           db      'J','F','M','A','M','J','J','A','S','O','N','D',0 ; O caracter x é
utilizado para determinar o fim do gráfico
        cont1           db      0
        cont2           db      0
        MAX             dw      160*25
DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
```

```
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        call    limpar
        call    desenha_graf

        mov     ah,01h
        int     21h

FIM:
        MOV   AH,4CH
        INT     21H

limpar  proc
        mov     ax,0b800h
        mov     es,ax

        xor     di,di

ciclo:  cmp     di,MAX
        je      sai
        mov     al,' '
        mov     es:[di],al
        inc     di
        mov     al,0f0H
        mov     es:[di],al
        inc     di
        jmp     ciclo

sai:    ret
limpar  endp

desenha_graf proc
        mov     ax,0b800h
        mov     es,ax

        xor     di,di    ; ponteiro para memória de video
        xor     si,si    ; ponteiro para array mediaTEMP
        xor     bx,bx    ; ponteiro para array meses
        xor     bp,bp    ; ponteiro para array titulo

        ;escreve legenda

ciclo:  mov     al,titulo[bp]
        cmp     al,0
        je      segue_0
        inc     bp
        mov     es:[di],al
        inc     di
        mov     al,0Fh
        mov     es:[di],al
        inc     di
```

```
        jmp     ciclo

segue_0:        add     di,2

ciclo1: mov     al,mediaTEMP[si]
        cmp     al,'x'
        je      sai     ; fim do array das temperaturas <-> fim de gráfico
        mov     cont1,al
        inc     si
        mov     cont2,0         ; reset cont2

        cmp     bx,12   ; quando chega a 12 é necessário colocar o ponteiro novamente
na posicao inicial do array
        jne     segue_1
        mov     bx,0


segue_1         :       mov     al,meses[bx]
                inc     bx
                mov     es:[di],al
                add     di,2

ciclo2: mov     al,79
        cmp     cont2,al
        jae     ciclo1          ; quando chegar ao fim de linha

        mov     al,cont1
        cmp     cont2,al
        jb      desenha                 ; enquanto n/chegar à temperatura media
        add     di,2
        jmp     segue


desenha:        mov     al,' '
                mov     es:[di],al
                inc     di
                mov     al,0FH
                mov     es:[di],al
                inc     di


segue: inc      cont2
        jmp     ciclo2

sai:    ret
desenha_graf endp


CODIGO  ends
end     INICIO
```

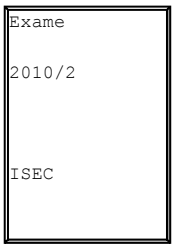
Pergunta do exame de T.A.C. de 27 de Junho de 2011)

22. Desenvolva um programa em Assembly que apresente um efeito tipo "Matrix" em todo o ecrã (apenas poderão aparecer 0s ou 1s). De forma a simular o efeito pretendido, considere que existe um vector (DADOS) com 80 valores compreendidos entre 1 e 25, que especifica a quantidade de caracteres '0' ou '1' que serão apresentados por cada coluna do ecrã. Os caracteres '0' ou '1' a apresentar, dependem do que já existir na respectiva posição do ecrã, de acordo com as seguintes regras:

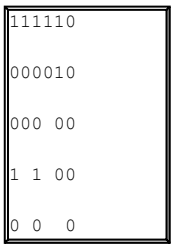
- Se existir um algarismo ou um espaço, é sobreposto o caracter '0'.
- Se existir outro carácter, é sobreposto o carácter '1'.

Tendo em consideração que o vector especifica o número de caracteres por coluna, se este for inferior a 25, os restantes elementos da coluna em questão ficarão com o carácter espaço.

A título de exemplo e considerando a parte do vector de DADOS com os valores 5,3,6,2,4,5 (apenas as primeiras 6 colunas do ecrã), o resultado será o seguinte:



Primeiras 6 colunas do ecrã
ANTES da execução do programa



Primeiras 6 colunas do ecrã
APÓS execução do programa

NOTAS:

- A memória de vídeo, no caso de sistemas policromáticos, tem início na localização B800h:0000h.
- Quem pretender um efeito Matrix mais verosímil, o atributo cor de escrita verde sobre cor de fundo preto é. Esta opção não será valorizada.
- Implementações com recurso a interrupções não serão avaliadas.

Solução:

Chama-se à atenção que na resposta a essa questão bastava implementar os procedimentos:

- *substitui_col*
- *converte*

```
PILHA SEGMENT PARA STACK    'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS       SEGMENT     PARA 'DATA'
        dados1      db      5,3,6,2,4,5,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0
                    db      0,0,0,0,0,0,0,0,0,0


        vector  db      'Exame ',0
                db      '2010/2',0
                db      '      ',0
                db      'ISEC  ',0
                db      -1
        cont    dw      0
        cont1   db      0
        cont2   db      0
        pos_x   dw      0*160  ; x=0 -> conversão para posição de ponteiro x 160
(linha)
        pos_y   dw      0*2    ; x=0 -> conversão para posição de ponteiro x 2
(coluna)
        coluna dw       0
DADOS   ENDS

CODIGO      SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV     AX,DADOS
        MOV     DS,AX

        call    limpar
        call    inicializa
        mov     ah,00h
        int     16h

        call    substitui_col


        mov     ah,01h
        int     21h

FIM:
        MOV     AH,4CH
        INT     21H

limpar  proc
        mov     ax,0b800h
        mov     es,ax
```

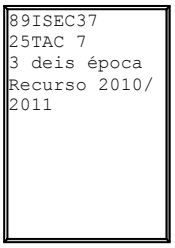
```
        mov     cx,80*25
        xor     di,di

l1:     mov     al,' '
        mov     es:[di],al
        inc     di
        mov     al,6FH
        mov     es:[di],al
        inc     di
        loop    l1

        ret
limpar  endp

inicializa      proc
        mov     ax,0b800h
        mov     es,ax
        xor     di,di
        ; posicao inicial      ->      pos_x,pos_y
        add     di,pos_x
        add     di,pos_y
        lea     bx,vector

segue1:         mov     al,[bx]
        cmp     al,-1
        je      sai

segue:  je      sai
        inc     bx
        mov     es:[di],al
        inc     di
        mov     ah,0fh
        mov     es:[di],ah
        inc     di
        add     cont,2
        cmp     al,0
        jne     segue1
        add     di,160
        sub     di,cont
        mov     cont,0
        jmp     segue1

sai:
        ret
inicializa      endp

converte        proc
        cmp     al,' '
        jne     segue
        mov     al,'0'
        jmp     sair

segue:  cmp     al,'0'
        jb      segue1
```

```
        cmp     al,'9'
        ja      segue1
        mov     al,'0'
        jmp     sair

segue1:         mov     al,'1'


sair:   ret
converte        endp


substitui_col   proc

        mov     ax,0b800h
        mov     es,ax

        lea     si,dados1
        mov     coluna,0

for1:   mov     di,coluna
        cmp     coluna,160
        je      sai
        add     coluna,2

        mov     al,[si]
        mov     cont1,al
        mov     cont2,0
        inc     si

for2:   mov     al,cont1
        cmp     cont2,al
        jb      conver
        cmp     cont2,25
        je      for1
        mov     al,' '
        mov     es:[di],al
        jmp     segue

conver:         inc     cont2
        mov     al,es:[di]
        call    converte
        mov     es:[di],al
        add     di,160
        jmp     for2

segue:  inc     cont2
        add     di,160
        jmp     for2

sai:    ret
substitui_col   endp

CODIGO  ends
end     INICIO
```

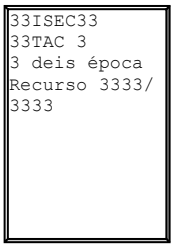
Pergunta do exame de T.A.C. de 12 de Julho de 2011)

Desenvolva um procedimento em Assembly que permita calcular a média de todos os algarismos correspondentes aos caracteres numéricos que se encontram no ecrã. O resultado da média deverá substituir todos os algarismos existentes no ecrã.

A título de exemplo, considere que o ecrã possuía o conteúdo exposto no quadrado à esquerda, o resultado da chamada do procedimento conduziria ao exposto no quadrado à direita:



Conteúdo do ecrã **ANTES** da execução do programa



Conteúdo do ecrã **APÓS** a execução do programa

NOTAS:

- A memória de vídeo, no caso de sistemas policromáticos, tem início na localização B800h:0000h.
- Implementações com recurso a interrupções não serão avaliadas.
- Deverá arredondar a média para o inteiro mais próximo (exemplo: 4.4 → 4 ou 4.51 → 5).
- No caso do exemplo exposto a média é de:

$$\frac{8+9+3+7+2+5+7+3+2+0+1+0+2+0+1+1}{16} = 3.1875 \Rightarrow \text{média} = 3$$

Solução:

Chama-se à atenção que na resposta a essa questão bastava implementar o procedimento

- pergunta_exame_R_12_7_11 proc

```
PILHA SEGMENT PARA STACK     'STACK'
        DB      1024 DUP('PILHA')
PILHA ENDS

DADOS         SEGMENT     PARA 'DATA'

soma  dw      0
total dw      0
media dw      0
resto dw      0


ecran db      '89ISEC37                                                '
      db      '25TAC 7                                                '
      db      '3 deis epoca                                            '
      db      'Recurso 2010/                                            '
      db      '2011                                                  '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      '
      db      '                                                      ',0

DADOS   ENDS

CODIGO        SEGMENT PARA 'CODE'
        ASSUME  CS:CODIGO, DS:DADOS, SS:PILHA

INICIO:
        MOV   AX,DADOS
        MOV   DS,AX

        call      limpa

        mov     ah,01h
        int 21h

        call      pergunta_exame_R_12_7_11

        mov     ah,01h
```

```
        int 21h

FIM:
        MOV   AH,4CH
        INT   21H

limpa proc
        mov   ax,0b800h
        mov   es,ax

        lea   bx,ecran
        mov   cx,80*25
        xor   di,di

repete: mov   ah,[bx]
        mov   es:[di],ah
        inc   di
        inc   bx
        mov   ah,0f2h
        mov   es:[di],ah
        inc   di
        loop  repete

        ret
limpa endp


pergunta_exame_R_12_7_11        proc

        mov   ax,0b800h
        mov   es,ax
        xor   ax,ax
        mov   cx,80*25
        xor   di,di

ciclo:  mov   al,es:[di]
        inc   di
        inc   di

        cmp   al,'0'
        jb    segue
        cmp   al,'9'
        jg    segue
        mov   ah,0
        sub   al,30h
        add   soma,ax
        inc   total
segue:  loop  ciclo

        mov   dx,0
        mov   ax,soma
        div   total
        mov   media,ax
        mov   resto,dx
        mov   dx,0
```

```
        mov     ax,resto
        mov     bx,2
        div     bx
        cmp     dx,resto
        jb      final
        inc     media

final:

        mov     ax,media
        add     al,30h
        mov     ah,0f9h
        xor     di,di

        mov     cx,80*25

repet:  mov     bl,es:[di]
        cmp     bl,'0'
        jb      seguir
        cmp     bl,'9'
        ja      seguir
        mov     es:[di],al

seguir: inc     di
        inc     di
        loop    repet

        ret
pergunta_exame_R_12_7_11        endp

CODIGO  ends
end     INICIO
```