UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
Laboratório de Microprocessadores

09_10

LINGUAGEM DE MONTAGEM (8086) – PARTE IV

Objetivos

- Aprimorar/desenvolver programas básicos em linguagem "*Assembly* / 8086";
- Acessar o modo gráfico através dos serviços da BIOS/DOS;
- Estudar novas instruções do 8086.

Roteiro

1. O programa mostrado na listagem I desenha na tela gráfica os dois lados de um retângulo. Implemente, rode, entenda e comente este programa. Em seguida:

   1.1 Adicione um código para desenhar os lados que estão faltando. ☐

   1.2 Desenhe de um quadrado SÓLIDO (30x30 pixels); ☐

2. O programa mostrado na listagem II faz o uso de um aplicativo externo ("thermometer.exe") que simula virtualmente um conjunto queimador-termômetro. A interface entre o programa em assembler e este aplicativo é feita através dos endereços de I/O 125 (leitura da temperatura) e 127 (saída para acionamento do queimador). 1.Implemente, execute, entenda e comente este programa.

3. Modifique o programa fornecido na listagem I para que o mesmo, em tempo real, exiba na tela de saída o valor da temperatura em graus Celsius e Fahrenheit. Exemplo de formato: "25°F (77°F)". ☐

4. A partir do programa desenvolvido no item anterior, inclua um código adicional para receber um valor de temperatura mínima e outro de temperatura máxima, os quais devem ser utilizados para definir os limites do controle de temperatura. ☐

5. Modifique o programa da listagem III para configurar "corretamente" os estados (não pode, por exemplo, ocorrer "verdes e/ou amarelos conflitantes"). Em seguida, ajuste "adequadamente" os tempos de amarelo e verde. (por exemplo, 4 segundos para amarelo e 10 segundos para verde). ☐

Listagens de programas

Listagem I

```
org  100h
w equ 10
h equ 7
mov ah, 0
mov al, 13h
int 10h
mov cx, 100+w
mov dx, 20
mov al, 2
u1:
mov ah, 0ch
int 10h
dec cx
cmp cx, 100
jae u1
mov cx, 100+w
mov dx, 20+h
mov al, 15
u2:
mov ah, 0ch
int 10h
dec dx
cmp dx, 20
ja u2
mov ah,00
int 16h
mov ah,00
mov al,03
int 10h
ret
```

Listagem II

```
#start=thermometer.exe#
org 100h
start:
        in AL,125
        cmp AL,22
        jl low
        cmp AL,55
        jle ok
        jg high
low:
        mov AL,1
        out 127,AL
        jmp ok
high:
        mov AL,0
        out 127,AL
ok:
        jmp start
```

Listagem III

```
#start=Traffic_Lights.exe#
org 100h
jmp start
semaforo DW     0000_0011_0000_1100b
s1       DW     0000_0110_1001_1010b
s2       DW     0000_1000_0110_0001b
s3       DW     0000_1000_0110_0001b
s4       DW     0000_0100_1101_0010b
final    DW     ?
start:
loop0:
  MOV   SI , OFFSET semaforo
loop1:
  MOV   AX, [SI]
  OUT   04,  AX
  MOV   CX, 4Ch
  MOV   DX, 4B40h
  MOV   AH, 86h
  INT   15h
  ADD   SI, 2
  CMP   SI, OFFSET final
  JB    loop1
  JMP   loop0
.EXIT
```

Questões adicionais

1) Considere o programa desenvolvido no item 1.2 (quadrado sólido) e o modifique de forma que durante o processo de desenho, ao pressionar a tecla X, a cor dos pontos subseqüentes deverá ser alterada (uma mudança de cor para cada toque). Atenção: o desenho deve ocorrer continuamente, sem ficar esperando toques de tecla. (DICA: verifique o teclado através da INT 21h / AH=6).

2) Programe um código para desenhar um triângulo eqüilátero.

3) O programa mostrado na listagem IV exibe o conteúdo dos 10 primeiros bytes da memória de programa no formato hexadecimal. Adicione um código para listar 16 bytes, em 4 linhas, conforme o formato indicado a seguir:

   0100: 41 42 43 44 ==> "ABCD"

   0104: 30 31 32 33 ==> "0123"

   0108: 33 33 33 33 ==> "3333"

   010C:41 42 43 44 ==> "ABCD"

4) Considere o código fonte (incompleto) de um jogo de memória fornecido na listagem V. A dinâmica deste jogo consiste em repetir os dígitos anteriormente gerados pelo programa. No nível 1, o programa exibe um digito e o jogador deve então digitar o digito apresentado. No próximo nível, o programa exibe um `*` e um novo digito, e o jogador deve entrar com o digito anterior e repetir o digito atual. Da forma como foi fornecido, o programa segue avançando os níveis indefinidamente, e não detecta a ocorrência de erros. Pede-se:

   a) Retire os erros grosseiros e rode o programa.

   b) Modifique este código para que o programa seja encerrado quando o jogador cometer um erro.

   c) Inclua uma limitação no numero máximo de níveis. Ao atingir um numero máximo (por exemplo 7), o programa deve exibir uma "mensagem de elogio", e em seguida deve ser encerrado.

5) Altere o código do jogo de memória e implemente as seguintes funcionalidades:

   a) No início do jogo, solicitar ao jogador que informe o número máximo de níveis do jogo.

   b) Informar ao final do jogo, juntamente com a mensagem de encerramento, o número de níveis atingidos.

Listagens adicionais

Listagem IV

```
org 100h
ini:
 mov cx,10
 lea si,ini
l1:
 mov  dl,[si]
 call hexb
 inc  si
loop l1
fim:
.EXIT

hexb proc near
mov  al,dl
shr  al, 4
call hexd
mov  al,dl
call hexd
mov ah, 2
mov dl, ' '
int 21h
ret

hexd proc near
push dx
and  al, 0fh
cmp  al, 9
jg  f1
add al, 48
mov ah, 2
mov dl, al
int 21h
pop dx
ret
f1:
add al, ('A'-10)
mov ah, 2
mov dl, al
int 21h
pop dx
ret
```

Listagem V – jogo de memória

```
ORG 100H
JMP INI
MEMO DB 20 DUP(0)
MSG_ERRO DB "!!      ERROU      !!$"
MSG_END DB "!!    MEMORIA BOA    !!$"
INI:
  MOVV BL,1
  LEA DI,MEMO
L0:
  CALL ALEO
  MOV  [DI],DL
  CALL SHOWDL
  CALL CR
  MOV  CL,BL
  MOV  CH,0
  LEA SI,MEMO
L1:
  CALL INUM
  MOV BH,[SI]
  INC SI
  LOOP L1
  CALLAA APAGA
  INC  BL
  INC  DI
  JMP L0
ERRO:
  CALL CR
  mov dx, offset MSG_ERRO
  mov ah, 9
  int 21h
 .EXIT

GAME_OVER:
  CALL CR
  mov dx, offset MSG_END
  mov ah, 9
  int 21h
  .EXIT

INUM:
  MOV AH,1
  INT 21H
RET
CR:
  PUSH AX
  PUSH DX
  MOV AH,2
  MOV DL,0X0D
  INT 21H
  POP DX
  POP AX
RET
APAGA:
  PUSH AX
  PUSH DX
  PUSH CX
  MOV CL,BL
  MOV CH,0
  MOV AH,2
  MOV DL,0X0D
  INT 21H

A1:
      MOV DL,"*"
      INT 21H
      LOOP A1:
  POP CX
  POP DX
  POP AX
RET
SHOWDL:
  PUSH AX
  MOV  AH,2
  INT  21H
  POP  AX
  RET
ALEO:
   PUSH AX
   PUSH CX
   MOV AH,00H
   INT 1AH
   MOV AL,DL
   MOV CL,26
   DIV CL
   MOV DL,AL
   ADD DL,48

   POP CX
   POP AX
RET
```