Trabalho Prático Nº 7

Instruções 8086: MOVSB,MOVSW,STOSB,STOSW (utilização do prefixo REP)

Acesso à memória vídeo

Chamadas à BIOS - Consulte os anexos para a descrição de funções da BIOS

Introdução

Com este trabalho procura-se exemplificar a programação de um periférico que partilha uma zona de memória com o microprocessador, neste caso concreto, o sistema vídeo do PC (Memory Mapped I/O).

O sistema vídeo de um IBM PC permite a utilização de vários modos gráficos e alfanuméricos (texto). A memória de vídeo é um bloco de RAM onde a placa de vídeo consulta a informação a visualizar no monitor. Esta memória encontra-se mapeada no espaço de endereçamento do CPU (640K-728K), por forma a que os programas possam ler e escrever nela como se se tratasse de outra qualquer zona de memória.

De acordo com o modo gráfico/texto seleccionado, a memória é tratada de forma diferente pela placa gráfica. Este mapeamento pode ser mais ou menos complexo, consoante o modo gráfico/texto seleccionado.

Chamadas à BIOS para programação do modo gráfico/texto

Para programar a placa gráfica no modo desejado e depois voltar a colocá-la em modo texto, devem ser utilizadas as rotinas existentes em ROM, mais propriamente na BIOS (Basic Input Output System). Estas rotinas proporcionam um interface standart para acesso aos periféricos, tais como o relógio, controladores de disco duros, disquetes, impressoras e, é claro, o sistema de vídeo. As rotinas de vídeo da BIOS implementam um conjunto de tarefas simples que realizam algumas operações básicas como sejam limpar o ecrã, escrever caracteres, mudar a forma do cursor, etc, permitindo também a selecção do modo gráfico.

O acesso a estas rotinas é feito através da interrupção por software 10h. A transferência de parâmetros é feita por registos, estando associado a cada um dos registos do CPU uma função específica.

Para o presente trabalho, o modo gráfico 320x200,256 cores (modo 19=13h) é seleccionado pela chamada à BIOS:

```
MOV AH,00h
MOV AL,13h
INT 10H
```

A selecção do modo texto nº 3, deve ser efectuada pela chamada à BIOS:

```
MOV AH,00h
MOV AL,03h
INT 10H
```

Parte Nº 1

Implementação de um conjunto de subprogramas para o modo texto

Neste trabalho, considera-se o modo texto nº 3 (25 linhas, 80 colunas, 16 cores). O ecrã é organizado numa matriz com 25 linhas e 80 colunas, sendo cada caracter representado por dois bytes em memória. O primeiro representa o código ASCII do caracter a visualizar e o segundo representa os seus atributos. Cada matriz ocupa 4000 bytes de memória (25x80x2), existindo 4 blocos com estas dimensões designados por páginas. Em cada instante, apenas uma das páginas se encontra activa.

Esta memória encontra-se mapeada no segmento de memória B800h, conforme se descreve na figura seguinte:

Página 0	Endereço (Hex.)	Coluna 0		Coluna 1		Coluna 2		....	Coluna 79	
Linha 0	B800:0000	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
Linha 1	B800:00A0	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
:	:	:	:	:	:	:	:	....	:	:
Página 1										
Linha 0	B800:1000	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
Linha 1	B800:10A0	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
:	:	:	:	:	:	:	:	....	:	:
Página 3										
Linha 0	B800:3000	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
Linha 1	B800:30A0	Ch	Atr	Ch	Atr	Ch	Atr	....	Ch	Atr
:	:	:	:	:	:	:	:	....	:	:

Para uma determinada posição no display (página=0..3, linha=0..24, coluna=0..79) a informação relativa a um caracter ocupará as posições:

Segmento:B800h

Deslocamento (Código ASCII): 4096 x página + 160 x linha + 2 x coluna

Deslocamento (Atributo): 4096 x página + 160 x linha + 2 x coluna +1

O byte correspondente aos atributos codifica o conjunto de características da representação dos caracteres, conforme se descreve na figura seguinte:

Bit 7 Bit 0

BLNK	BAK2	BAK1	BAK0	INT	FOR2	FOR1	FOR0

BLNK
1 - Blinking
0 - Not blinking
BAK2, BAK1, BAK0
000 = black 001=blue 010=green 011=cyan
100 = red 101=magenta 110=brown 111=white
INT, FOR2, FOR1, FOR0
0000 = black 0001=blue 0010=green 0011=cyan
0100 = red 0101=magenta 0110=brown 0111=white
1000 = grey 1001=light blue 1010=light green 1011=light cyan
1100 = light red 1101= light magenta 1110=yellow 1111=bright white

Subprogramas a implementar

Implemente os seguintes subprogramas, considerando que se utiliza apenas a página 0 da memória vídeo.

```
Procedure ClearTextScreen (ATR:Byte); Far; Assembler;
(*Limpa o ecrã colocando-o com os atributos ATR*)

Procedure CursorXY (X,Y:Byte); Far; Assembler;
(*Posiciona o cursor na coluna X, linha Y*)

Procedure WriteStr (X,Y:Byte; S:String; ATR:Byte);Far;Assembler;
(*Escreve a string S na coluna X, linha Y com atributos ATR *)

Procedure SaveScreen(P:Pointer);Far;Assembler;
(*Salvaguarda o conteúdo do display para a posição de memória
endereçada por P*)

Procedure RestoreScreen(P:Pointer);Far;Assembler;
(*Repõe o conteúdo do display a partir da posição de memória
endereçada por P*)
```

Parte Nº 2
Desenvolvimento de um conjunto de primitivas gráficas
Descrição do trabalho:
Neste trabalho, considera-se o modo gráfico VGA 320x200, 256 cores (modo 19=13h). Neste modo gráfico, o ecrã é organizado numa matriz de pixels (0..319,0..199) sendo cada pixel representado por um byte em memória (256 cores). São, portanto, necessários 320x200 bytes (64000 bytes) para armazenar toda a informação necessária para a definição da imagem vídeo. Esta memória encontra-se mapeada no segmento de memória A000h, conforme se descreve na figura seguinte:

Endereço
(segmento:deslocamento/offset)

A000:0000	0	1	2	....	318	319
A000:0140	320	321	322	....	638	639
	:	:	:		:	:

Um determinado pixel de coordenada (x,y) ocupará o byte com offset 320*y+x. Por exemplo, ao pixel de coordenadas (2,0) está associado a célula de memória A000h:0002h.
Para colocar um determinado pixel no ecrã com uma determinada cor, basta armazenar na célula de memória correspondente o valor da cor desejada.
Note-se que o sistema de coordenadas utilizado é diferente do habitual. O ponto de coordenadas (0,0) coincide com o canto superior esquerdo do ecrã e o ponto de coordenadas (319,199) coincide com o canto inferior direito.

Primitivas gráficas a implementar
Pretende-se desenvolver as primitivas gráficas em Assembly, para o modo 13h (19), 320x200, 256 cores, de acordo com os seguintes cabeçalhos:

```
Procedure ClearDisplay (Cor:Byte); Far; Assembler;
(*limpa o ecrã colocando-o a uma determinada cor*)

Procedure Plot (X,Y:Word;Cor:Byte); Far; Assembler;
(*desenha um pixel com uma determinada cor*)

Procedure DrawVertLine (X,Y1,Y2:Word;Cor:Byte);Far;Assembler;
(*desenha linha vertical com uma determinada cor*)

Procedure DrawHorizLine (X1,X2,Y:Word;Cor:Byte);Far;Assembler;
(*desenha linha horizonal com uma determinada cor*)

Procedure DrawLine (X1,Y1,X2,Y2:Word;Cor:Byte);Far;Assembler;
(*desenha linha com uma determinada cor*)

Procedure DrawText (X,Y:Word;S:String;FCor,BCor:Byte);Far;Assembler;
(* desenha, a partir da posição X,Y, uma cadeia de caracteres com uma
determinada cor de texto (FCor) e cor de fundo (BCor)*)
```

Algoritmos:
Apenas se apresenta informação adicional sobre o desenho de uma linha e a escrita de caracteres alfanuméricos em modo gráfico, já que os restantes procedimentos são de fácil implementação.

Desenho de uma linha

Aparentemente, a forma mais imediata de desenhar uma linha entre dois pontos é o recurso à equação da recta. No entanto, isso envolve uma relativa complexidade em termos de cálculo. Em alternativa, deverá ser usado o algoritmo de Bresenham, de menor complexidade e maior rapidez.

Dadas as coordenadas X1 e Y1 do primeiro ponto da linha, pode-se calcular a localização do próximo pixel através do incremento das coordenadas X e Y em proporção ao declive da recta.

Este algoritmo assume que X1 é menor que X2 e que o declive da recta está compreendido entre 0 e 1. No caso de X1 ser igual a X2 ou no caso de Y1 ser igual a Y2, deverão ser utilizadas rotinas adequadas para o efeito (DrawVertLine e DrawHorizLine). No caso do declive da recta estar compreendido entre -1 e 0, pode-se usar o mesmo algoritmo, bastando apenas decrementar o valor da coordenada Y em vez de a incrementar.

O algoritmo resultante será o seguinte:

```
Se X1=X2  então desenha linha vertical
Se Y1=Y2  então desenha linha horizontal
Se X1>X2  então troca coordenadas (X1 ⇔ X2, Y1⇔Y2)
Se Y2>Y1  então IncY=1
          senão IncY=-1
Calcular
     DDx  = X2-X1
     DDy  = | Y2-Y1 |
     D    = 2 DDy-DDx
     IncA = 2 (DDy-DDx)
     IncB = 2 DDy
Incializar X=X1; Y=Y1
Até X=X2 efectuar
     Desenha ponto na posição (X,Y)
     Se D>=0   então
                    Y=Y+IncY
                    D=D+IncA
               senão
                    D=D+IncB
     Incrementar X
```

Para declives superiores a 1 e inferiores a -1, o mesmo algoritmo pode ser utilizado, desde que se considere uma rotação no sistema de eixos.

Escrita de caracteres alfanuméricos (texto) em modo gráfico

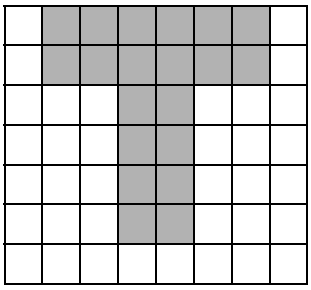
Para escrever qualquer caracter alfanumérico em modo gráfico, é necessário desenhá-lo ponto a ponto. Para isso, existe uma tabela que define a forma como os caracteres devem ser desenhados. Nesta tabela, cada caracter é representado à custa da utilização de uma tabela de bytes, onde cada bit representa um ponto do caracter.

No modo gráfico utilizado (320x200), cada caracter é representado à custa de uma matriz de 8x8 pontos, ou seja 8 bytes (cada byte tem a definição de 8 pontos). Se um determinado bit for 1, o ponto correspondente pertence ao caracter e se o mesmo for 0 o ponto faz parte do background.

Por exemplo, a definição da letra 'T' é apresentada na figura seguinte:

Endereço	Memória (8 bytes)							
F000:FA76	7E	7E	18	18	18	18	18	0

Hex.	Binário	
7E	01111110	
7E	01111110	
18	00011000	
18	00011000	
18	00011000	
00	00000000	



Existem armazenadas em ROM, várias tabelas com a definição de todos os 256 caracteres para todos os modo gráficos suportados. Quando se altera o modo gráfico corrente, a BIOS, automaticamente, selecciona qual a tabela de definição de caracteres adequada para o modo gráfico em questão. O endereço (segment, offset) do local onde se encontra armazenada essa tabela pode ser obtido no endereço 0000:010Ch, conforme se indica na figura:

Endereço (h)	RAM
0000:010Fh	<segmento>
0000:010Eh	
0000:010Dh	
0000:010Ch	<offset>