+--------------+
¦ ASSEMBLY XXV ¦
+--------------+

O modo de escrita 1 näo é täo útil, como vimos no último texto... A plca VGA possui algumas redundancias que podem parecer desnessesárias à primeira vista, como por exemplo o modo de escrita 3. Nesse modo podemos despresar o registrador "Enable Set/Reset" e usar "Set/Reset" para ajustar os bits dos quatro planos de vídeo.

¦ Modo de escrita 3

Well... No modo 0 vimos como atualizar os quatro planos de bits de uma só vez... Isso é feito setando o registrador "Enable Set/Reset" e "Set/Reset"... usando também MapMask e BitMask para habilitarmos os planos e os bits desejados, respectivamente. Acontece que no modo 0 podemos ter uma mistura de dados vindos da CPU, dos latches e do registro Set/Reset... a mistura pode ser täo confusa que podemos ter a CPU atualizando um plano e Set/Reset outro. É, sem sombra de dúvida, um recurso interessante e bastante útil... mas se näo tomarmos cuidado pode ser uma catastrofe, em termos visuais!

O modo de escrita 3 trabalha da mesma forma que o modo 0 só que "seta" automaticamente os quatro bits de "Enable Set/Reset". Isto é, a CPU näo escreve nada nos planos de bits... isso fica sob responsabilidade do registrador "Set/Reset". O que a CPU escreve na memória so sistema sofre uma operaçäo lógica AND com o conteúdo atual de BitMask... O resultado é usado como se fosse o BitMask! (Para facilitar as coisas... se BitMask for 11111111b e a CPU escrever 01100011b, entäo o "novo" BitMask será 01100011b, sem que o registrador BitMask seja afetado!!)

Com esse modo de escrita descartamos a necessidade de ajustar "Enable Set/Reset", eliminando a confusäo que pode ser causada no modo 0... descartamos a atualizaçäo de BitMask, que pode feita indiretamente pela CPU... Mas, infelizmente näo descartamos a necessidade de leitura da memória do sistema para carga dos latches e nem mesmo a necessidade de habilitarmos os planos de bits em MapMask! Se MapMask estiver zerado nenhum plano de bit será atualizado, lembre-se sempre disso!!! Isso é válido para TODOS os modos de escrita!

Eis um exemplo prático do uso do modo de escrita 3... Uma rotina que traça uma linha horizontal:

```
¦  ideal                                                                ¦
¦  model small,c                                                        ¦
¦  locals                                                              ¦
¦  jumps                                                               ¦
¦  p386                                                                ¦
¦                                                                      ¦
¦  ; inclui os macros definidos no último texto!                       ¦
¦  include "VGA.INC"                                                   ¦
¦                                                                      ¦
¦  SCREEN_SEGMENT  equ 0A000h                                          ¦
¦                                                                      ¦
¦  ; Tamanho de uma linha... (modo 640x480)                            ¦
¦  LINE_SIZE       equ 80                                              ¦
¦                                                                      ¦
¦  ; Coordenadas máximas...                                            ¦
¦  MAX_X_POS       equ 639                                             ¦
```

```
MAX_Y_POS       equ 479

global  grHorizLine:proc
global  grVertLine:proc
global  setGraphMode:proc
global  setTextMode:proc

codeseg

;*** DESENHA LINHA HORIZONTAL ***
proc    grHorizLine
arg     left:word, right:word, y:word, color:word
local   bitmask1:byte, bitmask2:byte
uses    si, di

        ; Verifica se a coordenada Y é válida...
        mov     ax,[y]
        or      ax,ax
        js      @@grHorizLineExit

        cmp     ax,MAX_Y_POS
        ja      @@grHorizLineExit

        ; Verifica validade das coordenadas "left" e "right"...
        mov     ax,[left]
        cmp     ax,[right]
        jb      @@noSwap

        ; Troca "left" por "right"
        ;  se "right" for menor que "left".
        xchg    ax,[left]
        mov     [right],ax

@@noSwap:
        ; Verifica a validade das coordenadas "left" e "right"
        cmp     ax,MAX_X_POS    ; "left" é valido?
        ja      @@grHorizLineExit

        or      [right],0       ; "right" é valido?
        js      @@grHorizLineExit

        WriteMode   3     ; Ajusta no modo de escrita 3.
        BitMask     0FFh  ; BitMask totalmente setado!
        MapMask     1111b ; Habilita todos os quatro planos
                          ;  de bits.
        SetReset    <[byte color]> ; Ajusta a cor desejada...

        mov     ax,SCREEN_SEGMENT
        mov     es,ax   ; ES = segmento de vídeo.

        ; Calcula os offsets das colunas...
        mov     si,[left]
        mov     di,[right]
        shr     si,3        ; si = offset da coluna 'left'
        shr     di,3        ; di = offset da coluna 'right'

        ; Calcula o offset da linha 'y'
        mov     bx,[y]
        mov     ax,LINE_SIZE
        mul     bx
        mov     bx,ax   ; BX contém o offset da linha.

        ; Pré-calcula a mascara da coluna 'left'
        mov     cx,[left]
```

```
        mov     ch,cl
        and     ch,111b
        mov     cl,8
        sub     cl,ch
        mov     ah,0FFh
        shl     ah,cl
        not     ah
        mov     [bitmask1],ah

        ; pré-calcula a mascara da coluna 'right'
        mov     cx,[right]
        and     cl,111b
        inc     cl
        mov     ah,0FFh
        shr     ah,cl
        not     ah
        mov     [bitmask2],ah

        ; Verifica se apenas um byte será atualizado.
        cmp     si,di
        jz      @@OneByte

        mov     ah,[bitmask1]
        xchg    [es:bx+si],ah  ; Escreve na memória da video...
                               ; ... XCHG primeiro lê o que
                               ;  está no operando destino,
                               ;  depois efetua a troca.
                               ;  Com isso economizamos um MOV!
        inc     si
        cmp     si,di
        je      @@doMask2

@@MiddleDraw:
        mov     [byte es:bx+si],0ffh    ; Linha cheia...
                                        ; Não precisamos
                                        ;  carregar os latches
                                        ;  pq todos os bits
                                        ;  serão atualizados!
        inc     si
        cmp     si,di
        jne     @@MiddleDraw

@@doMask2:
        mov     ah,[bitmask2]
        xchg    [es:bx+si],ah   ; Escreve na memória de vídeo
        jmp     @@HorizLineEnd

@@OneByte:
        and     ah,[bitmask1]
        xchg    [es:bx+si],ah

@@HorizLineEnd:
        WriteMode 0         ; Poe no modo 0 de novo...
                            ;  Necessário somente se essa
                            ;  rotina for usada em conjunto
                            ;  com as rotinas da BIOS ou de
                            ;  seu compilados (p.ex: BGIs!).
@@grHorizLineExit:
        ret
endp

;;*** DESENHA LINHA VERTICAL ***
proc    grVertLine
arg     x:word, top:word, bottom:word, color:byte
```

```
uses    si, di

        ; Verifica se X está na faixa
        mov     ax,[x]
        or      ax,ax               ; x < 0?
        js      @@grVertLineExit

        cmp     ax,MAX_X_POS        ; x > 639?
        ja      @@grVertLineExit

        ; Verifica se precisa fazer swap
        mov     ax,[top]
        cmp     ax,[bottom]
        jb      @@noSwap

        xchg    ax,[bottom]
        mov     [top],ax

@@noSwap:
        ; Verifica se as coordenadas "Y" estäo dentro da faixa.
        cmp     ax,MAX_Y_POS
        ja      @@grVertLineExit

        cmp     [bottom],0
        js      @@grVertLineExit

        mov     ax,SCREEN_SEGMENT
        mov     es,ax

        WriteMode 3
        BitMask 0FFh
        MapMask 0Fh
        SetReset <[byte color]>

        mov     si,[top]

        mov     ax,LINE_SIZE
        mul     si
        mov     bx,ax       ; BX contém o offset da linha

        mov     di,[x]
        mov     cx,di
        shr     di,3        ; DI contém o offset da coluna

        and     cl,111b
        mov     ah,10000000b
        shr     ah,cl

@@SetPixelLoop:
        mov     cl,ah
        xchg    [es:bx+di],cl
        add     bx,LINE_SIZE
        inc     si
        cmp     si,[bottom]
        jbe     @@SetPixelLoop

        WriteMode 0

@@grVertLineExit:
        ret
endp

proc    setGraphMode
    mov     ax,12h
```

```
    int     10h
    ret
endp

proc    setTextMode
    mov     ax,3
    int     10h
    ret
endp

end
```

Näo sei se percebeu a engenhosidade dessa pequena rotina... Ela pré-calcula os bitmasks do inicio e do fim da linha... Se a linha está contida somente em um byte entäo fazemos um AND com os dois bitmasks pré-calculados pra obter o bitmask necessário para atualizar um único byte... Suponha que queiramos traçar uma linha de (2,0) até (6,0). Eis os bitmasks:

```
BitMask1    =   00111111b   ; BitMask do inicio da linha
BitMask2    =   11111110b   ; BitMask do fim da linha
----------------------------
BitMask3    =   00111110b   ; BitMask1 AND BitMask2
```

Ok... E se a linha ocupar 2 bytes?! Por exemplo, de (2,0) até (11,0)... O ponto (2,0) está, com certeza, no primeiro byte... mas o ponto (11,0) näo (já que um byte suporta apenas 8 pixeis!). Entäo calculados os dois bitmasks:

```
BitMask1    =   00111111b   ; BitMask do inicio da linha
BitMask2    =   11110000b   ; BitMask do fim da linha
```

Dai escrevemos o primeiro byte com o bitmask1 e o segundo com o bitmask2. Se a linha ocupar mais de 2 bytes o processo é o mesmo, só que os bytes intermediários teräo bitmasks totalmente setados (näo necessitando, neste caso, carregar os latches!).

Na mesma listagem temos a rotina de traçagem de linhas verticais... dê uma olhada nela. É bem mais simples que grHorizLine!

No próximo texto: O modo de escrita 2! E depois, os modos de 256 cores! (finalmente, né?!)