+---------------+
¦ ASSEMBLY XXII ¦
+---------------+

Alguma vez aconteceu de você ter aquela rotina quase concluída e quando foi testá-la viu que estava faltando alguma coisa?! Bem... se não aconteceu você é um sortudo... Quando eu estava começando a entender o funcionamento da placa VGA me dispus a construir rotinas básicas de traçagem de linhas horizontais e verticais... porém, quando tinha algum bitmap atrás da linha acontecia uma desgraça!!! Parte do bitmap sumia ou era substituído por uma sujeirinha chata!

Obviamente eu ainda näo tinha dominado o funcionamento da placa... por isso, vamos continuar com os nossos estudos...

¦ A mascara de bits e os LATCHES da VGA.

Existe uma maneira de näo alterarmos bits indesejáveis em um byte de cada plano... Suponha que queiramos modificar apenas o bit mais significativo de um byte nos planos de bits, deixando o restante exatamente como estavam antes!

Well... Isso pode ser feito de duas formas: Primeiro lemos o byte de um plano, realizamos um OR ou um AND com esse byte e o byte com o bit a ser alterado (zerando-o ou setando-o de acordo com a modificação que faremos... veja as instruções AND e OR num dos textos iniciais do curso de ASM para ter um exemplo de como isso pode ser feito!)... depois da operação lógica, escrevemos o byte na mesma posição... Essa é a maneira mais dispendiosa!

A placa VGA permite que criemos uma mascara de bits para podermos alterar apenas aqueles bits desejados... Isso é feito pelo registrador BitMask. Mas, antes temos que ler o byte inteiro... hummm... acontece que existe um registrador intermediário, interno, que retém o último byte lido de um plano de bits... esse registrador é conhecido como LATCH.

Basta ler um byte da memória do sistema que os bytes dos quatro planos de bits vão para seus LATCHES... Depois precisamos mascarar os bits que näo queremos modificar no registrador BitMask para só então escrever na memória do sistema (no plano de bits!)... Näo esquecendo de setar os planos de bits que queremos alterar via MapMask, como visto no último texto!

O funcionamento dos latches em conjunto com BitMask é o seguinte: Uma vez carregados os latches, apenas os bits ZERADOS de BitMask serão copiados de volta para os planos de bits selecionados por MapMask. Em contrapartida, os bits SETADOS em BitMask correspondem aos bits vindos da memória do sistema, que säo fornecidos pela CPU. Dessa maneira a nossa rotina näo tem que propriamente ler o conteúdo de um plano de bits (aliás, o que for lido pela CPU pode muito bem ser ignorado!)... näo necessitamos nem ao menos efetuar operações lógicas para setar ou resetar um determinado bit do byte que será escrito num plano de bits!

Vímos no último texto que o registro MapMask faz parte do circuito SEQUENCIADOR da VGA. O registro BitMask está localizado em outro circuito. Mais exatamente no controlador gráfico (Graphics Controller - que chamaremos de GC)... O funcionamento é o mesmo do que o circuito sequenciador, em termos de endereços de I/O, citado no último texto: Primeiro devemos informar o número do registro e depois o valor. O GC pode ser acessado a partir do endereço de I/O 03CEh e o número do registro BitMask é 8.

Eis nosso segundo exemplo:

```
; VGA2.ASM
; Compile com:
;
;   TASM vga2
;   TLINK /x/t vga2
;
ideal
model tiny
locals
jumps

codeseg

org 100h
start:
    mov     ax,12h      ; Poe no modo 640x480
    int     10h

    mov     ax,0A000h   ; Faz ES = 0A000h
    mov     es,ax
    sub     bx,bx       ; BX será o offset!

    mov     dx,03C4h    ; Seleciona planos 0 e 2...

    mov     ax,0502h    ; ídem a fazer: mov al,2
                        ;               mov ah,0101b

    out     dx,ax

    mov     dx,03CEh    ; Mascara todos os bits,
    mov     ax,8008h    ;  exceto o bit 7
    out     dx,ax

    mov     al,[byte es:bx]     ; carrega os latches da VGA
                                ;  note que AL näo nos
                                ;  interessa!!!
    mov     [byte es:bx],0FFh   ; Escreve 0FFh

    sub     ah,ah       ; Espera uma tecla!
    int     16h         ; ... senäo näo tem graça!!! :)

    mov     ax,3        ; Volta p/ modo texto 80x25
    int     10h

    int     20h         ; Fim do prog

end start
```

Temos algumas novidades aqui... Primeiro: é possível escrever o número de um registro e o dado quase que ao mesmo tempo... basta usar a instrunção OUT DX,AX - recorra a textos anteriores para ver o funcionamento dessa instruçäo!. Segundo: mesmo escrevendo 0FFh (todos os bits setados) na memória do sistema, apenas o bit que näo está mascarado será modificado, graças ao BitMask!! Terceiro: Mais de um plano de bits pode ser alterado ao mesmo tempo! Note que nesse código escrevemos na memória de vídeo apenas uma vez e os planos 0 e 2 foram alterados (continua a cor MAGENTA, näo?!).

¦ Problemas à vista!

Ok... aparentemente a coisa funciona bem... dai eu faço uma simples pergunta: O que aconteceria se o ponto em (0,0) estivesse inicialmente "branco" e usassemos a rotina acima?!

Hummmm... Se o ponto é branco, a cor é 15... 15 é 1111b em binário, ou seja, todos os planos de bits teriam o bit 7 do primeiro byte setados... A rotina acima "seta" os bits 7 do primeiro byte dos planos 0 e 2... assim a cor CONTINUARIA branca!! MAS COMO SOU TEIMOSO, EU QUERO MAGENTA!!!

A solução seria colocar as seguintes linhas antes da instrução "sub ah,ah" na listagem acima:

```
+-------------------------------------------------------------------+
¦  mov     dx,03C4h    ; Seleciona os planos 1 e 3                  ¦
¦  mov     ax,0A02h                                                 ¦
¦  out     dx,ax                                                    ¦
¦                                                                   ¦
¦  mov     [byte es:bx],0 ; escreve 0 nos planos 1 e 3              ¦
+-------------------------------------------------------------------+
```

Precisamos zerar os bits 7 dos planos 1 e 3... Note que nas linhas acima näo carreguei os latches da VGA através de leitura... aliás... näo carreguei de forma alguma. Não preciso fazer isso os latches dos planos 1 e 3 não foram alterados desde a sua última leitura... repare que näo "desmascarei" os bits no registro BitMask... dai näo ter a necessidade de mascará-los de novo... só preciso escrever 0 nos planos 1 e 3 para que o bit 7 seja alterado.

Puts... que mäo-de-obra!!... Felizmente existem meios mais simples de fazer isso tudo... Ahhhhhh, mas é claro que isso fica pra um próximo texto! :))