+--------------+
¦ ASSEMBLY XXI ¦
+--------------+

Olá!!... Acho que você concorda comigo que essa série de textos não estaria completa se eu não falasse alguma coisa a respeito de progrmação da placa de vídeo VGA, né?! Acho que nós temos razão em pensar assim! :)

Inicialmente começarei a descrever a placa VGA, depois vem as descriçöes da SVGA e VESA. Não pretendo gastar "trocentas" horas de digitação e depuração de código na descrição desses padröes.. quero apenas dar uma idéia geral do funcionamento desses dispositivos para que você possa caminhar com as próprias pernas mais tarde...

¦ Video Graphics Array

O padräo VGA é o sucessor dos padröes EGA e CGA, todos criados pela IBM... A diferença básica do VGA para os outros dois é o aumento da resolução e de cores. Eis uma comparação dos modos de maior resolução e cores desses três padröes (aqui estäo listados apenas os modos gráficos!):

	CGA	EGA	VGA
Maior resolução	640x200	640x350	640x480
Maior número de cores	4 (320x200)	16 (640x350)	16 (640x480) 256 (320x200)

O padräo VGA suporta até 256 cores simultanemente no modo de vídeo 13h (320x200x256). E no modo de mais alta resolução suporta o mesmo número de cores que a EGA, que säo apenas 16.

Quanto ao número de cores, as placas EGA e VGA säo mais flexíveis que a irmä mais velha (a CGA). As cores säo "reprogramáveis", isto é, de uma palette de 256k cores (256 * 1024 = 262144 cores), na VGA, podemos escolher 256... Duma palette de 64 cores podemos usar 16, na EGA... A VGA é, sem sombra de dúvidas, superior!

A forma como podemos selecionar essas cores todas será mostrada mais abaixo (Como sempre as coisas boas säo sempre deixadas pra depois, né?! hehe).

Em tempo: O modo 640x480 (16 cores) será usado como exemplo nas próximas listagens dos textos daqui pra frente... O modo gráfico de 320x200 com 256 cores será discutido em outra oportunidade, bem como o famoso MODE X (modo de vídeo não documentado da VGA - e largamente descrito por Michael Abrash em seus artigos para a revista Dr. Dobb's).

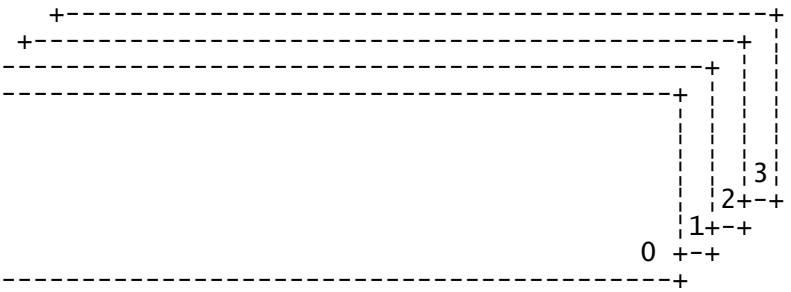
¦ Memória de vídeo

Existe um grande obstáculo com relação a modos gráficos de resoluçöes altas: A segmentação de memória! Lembre-se que os processadores Intel enxergam a memória como blocos de 64k não

sequenciados (na verdade, sobrepostos!)... No modo gráfico de resolução 640x480 da VGA (que suporta 16 cores no máximo), suponha que cada byte da memória de vídeo armazenasse 2 pixeis (16 cores poderia equivaler a 4 bits, näo poderia?!)... Well isso nos dá 320 bytes por linha (meio byte por pixel -> 640 / 2 = 320!).

Com os 320 bytes por linha e 480 linhas teriamos 153600 bytes numa tela cheia! Ocupando 3 segmentos da memória de vídeo (2 segmentos contíguos completos e mais 22528 bytes do terceiro!)... Puts... Imagine a complexidade do algoritmo que escreve apenas um ponto no vídeo! Seria necessário selecionarmos o segmento do pixel e o offset... isso pra aplicativos gráficos de alta performance seria um desastre!

A IBM resolveu esse tipo de problema criando "planos" de memória... Cada plano equivale a um bit de um pixel. Dessa forma, se em um byte temos oito bits e cada plano armazena 1 bit de 1 pixel... em um byte de cada plano teremos os 8 bits de 8 pixeis. Algo como: O byte no plano 0 tem os oito bits 0 de oito pixeis... no plano 1 temos os oito bits 1 de oito pixeis... e assim por diante. De forma que o circuito da VGA possa "sobrepor" os planos para formar os quatro bits de um único pixel... A representaçäo gráfica abaixo mostra a sobreposição dos planos:



Esses säo os quatro planos da memória de vídeo. O plano da frente é o plano 0, incrementando nos planos mais interiores. Suponha que na posição inicial de cada plano tenhamos os sequintes bytes:

```
Plano 0: 00101001b
Plano 1: 10101101b
Plano 2: 11010111b
Plano 3: 01010100b
```

Os bits mais significativos de cada plano formam um pixel: (0110b), os bits seguintes o segundo pixel (0011b), o terceiro (1100b), e assim por diante até o oitavo pixel (1110b). Como temos 16 cores no modo 640x480, cada pixel tem 4 bits de tamanho.

Com esse esquema biruta temos um espaço de apenas 38400 bytes sendo usados para cada plano de vídeo... Se cada byte suporta um bit de cada pixel entäo temos que uma linha tem 80 bytes de tamanho (640 / 8). Se temos 480 linhas, teremos 38400 bytes por plano.

Tome nota de duas coisas... estamos usando um modo de 16 cores como exemplo para facilitar o entendimento (os modos de 256 cores säo mais complexos!) e esses 38400 bytes em cada plano de bits é um espaço de memória que pertence à placa de vídeo e é INACESSíVEL a CPU!!!! Apenas a placa de vídeo pode ler e gravar nessa memória. A placa VGA (e também a EGA) usam a memória RAM do sistema para

saberem quais posiçöes de um (ou mais) planos de bits serão afetados. Isso é assunto para o próximo tópico:

¦ A memória do sistema:

Os adaptadores VGA usam o espaço de "memória linear" entre 0A0000h e 0BFFFFh (todo o segmento 0A000h e todo o segmento 0B000h)... Essa memória é apenas uma área de rascunho, já que a placa VGA tem memória própria... A CPU precisa de uma memória fisicamente presente para que possa escrever/ler dados... daí a existencia desses dois segmentos contíguos de memória, mas a VGA não os usa da mesma forma que a CPU!

Citei dois segmentos contíguos... mas näo existe a limitação de apenas um segmento?! Well... existe... o segmento 0B000h é usado apenas nos modos-texto (onde o segmento 0B800h é usado... 0B000h é para o adaptador monocromático - MDA)... os modos-gráficos utilizam o segmento 0A000h (a não ser aqueles modos gráficos compatíveis com a CGA!).

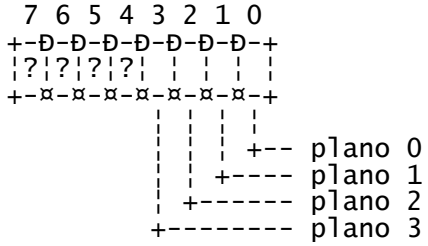
A memória do sistema é usada como rascunho pela VGA (e pela EGA também!!)... A VGA colhe as modificaçöes feitas na memória do sistema e transfere para a memória de vídeo. A forma com que isso é feito depende do modo com que programamos a placa de vídeo para fazê-lo... podemos modificar um plano de bits por vez ou vários planos, um bit por vez, vários bits de uma vez, etc. Na realidade, dependendo do modo com que os dados são enviados para a placa VGA não precisamos nem ao menos saber O QUE estamos escrevendo na memória do sistema, a VGA toma conta de ajustar a memória de vídeo por si só, usando apenas o endereço fornecido pela CPU para saber ONDE deve fazer a modificação!

¦ Selecionando os planos de bits...

Em todos os modos de escrita precisamos selecionar os planos de bits que serão afetados... Isso é feito através de um registrador da placa VGA: MapMask... Porém, antes de sairmos futucando tudo quanto é endereço de I/O da placa VGA precisamos saber COMO devemos usá-los!

A maioria dos registradores da placa VGA estão disponíveis da seguinte maneira: Primeiro informamos à placa qual é o registrador que queremos acessar e depois informamos o dado a ser escrito ou lido... A técnica é a seguinte: escrevemos num endereço de I/O o número do registrador... no endereço seguinte o dado pode ser lido ou escrito...

No caso de MapMask, este registrador é o número 2 do CIRCUITO SEQUENCIADOR da placa VGA. O circuito sequenciador pode ser acessado pelos endereços de I/O 3C4h e 3C5h (3C4h conterá o número do registro e 3C5h o dado!). Eis a estrutura do registro MapMask:



De acordo com o desenho acima... os quatro bits inferiores informam a placa VGA qual dos planos será modificado. Lembre-se que cada plano tem um bit de um pixel (sendo o plano 0 o proprietário do bit menos significativo). Vamos a nossa primeira rotina:

```
; VGA1.ASM
; Compile com:
;
;   TASM vga1
;   TLINK /x/t vga1
;
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    ; Aponta para o registro
    mov     al,2        ; "MapMask"
    out     dx,al

    inc     dx          ; Incrementa endereço de I/O

    mov     al,0001b    ; Ajusta para o plano 0
    out     dx,al

    mov     [byte es:bx],0FFh   ; Escreve 0FFh

    mov     al,0100b    ; Ajusta para o plano 2
    out     dx,al

    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
```

Depois de compilar e rodar o VGA1.COM você vai ver uma pequena linha magenta no canto superior esquerdo do vídeo... Se você quiser que apenas o pixel em (0,0) seja aceso, entäo mude o valor 0FFh nas instruçöes "mov [byte es:bx],0FFh" para 80h. O motivo para isso é que cada byte tem apenas um bit de um pixel, isto é, cada bit do byte equivale a um bit do pixel... necessitamos alterar os quatro planos de bits para setarmos os quatro bits de cada pixel (quatro bits nos däo 16 combinaçöes)... assim, se um byte tem oito bits, o primeiro byte dos quatro planos de bits tem os oito pixeis iniciais,

sendo o bit mais significativo do primeiro byte de cada plano o primeiro pixel.

Deu pra notar que apenas modificamos os planos 0 e 2, né?! Notamos também que desta maneira näo temos como alterarar um único pixel... sempre alteraremos os oito pixels!! Mas, näo se preocupe... existem outros recursos na placa VGA... Entendendo o esquema de "planos de bits" já está bom por enquando...

Até a próxima...