

## Video Card

**PC BIOS** → Sistema básico de input/output

### BIOS calls:

- Acesso à BIOS é feita através de software interrupt a partir de instruções int 2E → 8 bits especificam uma reunião

- Qualquer argumento necessário é passado através de registos do processador

interrupt vector

10h f0  
0ce ff  
0ce f2  
0ce 16

reunião

reunião card  
PC config  
mem config  
Mangle card

→ Para realizar as diferentes BIOS calls de LCOM temos a library libe86 emu.

int (relee\_get\_mode\_info)(uint16\_t mode,  
rebi\_mode\_info\_t \*remi\_p)

→ função: usar a função return VBE mode info para calcular a informação da moda "modo" na address do pointer "remi\_p"

(P) → estrutura map\_t

→ Recai no o nosso buffer que recai guardar as informações da moda.

② → aloca em map-map espaço igual  
ao tamanho de uma estrutura nebe-made-infant

→ usar kmalloc(size, \*buffer)

③ → chama a função VBE return made:

neg86-t neg86

- coloca o neg86 a 0x (tip → usar memset)

neg86.intme = 0x00 → reto da reidec card

neg86.al = 0x4F0F → return <sup>VBE</sup> made function

↳ Indicada de VBE function

(ou neg86.al = 0x4F neg86.al = 0x0F)

↳ phys-bytes tec segment base

neg86.es = PB2BASE(map.phys)

↳ address física da map onde reangs  
coloca a informaçao

neg86.di = PB20FF(map.phys)

↳ phys-bytes tec segment offset

neg86.cs = made

- envia o neg86 para fazer a BIOS call,  
usar sysint86(neg86).

(4) →

dar cost do mmap para uma estrutura  
vlele-malloc-ingre-t\* e coloca cálculo m\*memi-p

(5) → libera o espaço allocado no mmap (tip: mun  
lkm-free)

receid \* (reg-init)(uint16\_t mode)

função → coloca a memória em reideo modo  
no modo "modo".

(6) → O que aparece na escra depende do conteúdo  
da reideo RAM, para mudar o que aparece na  
escra temos de modificar o frame buffer.

→ A reideo RAM é uma região da memória física,  
logo, é preciso antes de mais mapear a address space da  
reideo RAM.

a) inicializa o frame-buffer Vm adddress  
com a escra real global (tip: \* receid) estética.

b) chama a função returnVmode ingre

c) a partir da escra real Vm-i-p definir  
a address física da VRAM (Vm-i-p. PhysBasePte) e  
a VRAM size (reslxresYx(bitsPerPixel/8))

↳ ter em atenção qnd elas  
m rias multiplas de 8,

d) inicializa uma escra real minix-mem-range

fimel e) definir o endereço base e a  
da VRAM meta escra real:

mr.mn\_base = (phys\_bytes)resam-

mr.mn\_limit = mr.mn\_base + resam\_size

g) Garantir o processo promissor para mapas  
a VRAM:

sys\_privctg (SELF, SYS\_PRIV\_ADD\_MEM, &mr)

g) aloca memoria para um buffer (reservado global), caso de chaves que não se alocasse para o intermediário antes de escrever para a VRAM (mudar com a renam\_rige)

h) mapa a memória do VRAM para a memória video-mem.

nem\_map\_phys (SELF, (recid\*) mr, maddr,  
renam\_rige)

meta: reescrever - o mapeamento falhou.

② → Chama a função no VBE mode:

• coloca o reg86 = 0 e memset

reg86.intma = 0 e 10 → vector da video card

reg86.al = 0 e 4F02 → função VBE no mode

↓ marcadas de função VBE

(ou reg86.alh = 0 e 4F      reg86.al = 0 e 02)

reg86.bue = P < 14 | mode

↓  
coloca em  
modo linear, facilita acesso à  
VRAM

③ → Retorna a video-mem

`int (reideo-test-init)(uint16-t mode, uint8-t delay)`

função → iniciar reideo mode na maca "mode" e após "delay" segundos sair do reideo mode.

① → Chama reg-init(mode)

② → sleep(delay)

③ → reg-exit()

`int (reideo-test-rectangle)(uint16-t mode,  
                              uint16-t x, uint16-t y, uint16-t width,  
                              uint16-t height, uint32-t color)`

função → desenhar um retângulo C/ canto superior em "(x,y)", largura "width", comprimento "height" e cor "color" na maca "mode". Térmica quando a utilizada precisa a tecla escape da teclada

① → Criar o interrupt para o Keyboard.

② → Abre de entra no ciclo do interrupt, iniciar reideo mode.

③ → Chama reg-draw-rectangle que desenha uma dentro de si reg-draw-hline para o lgf. Consegue fazer teste:

reg-draw-rectangle → desenha o retângulo  
reg-draw-hline → desenha uma linha

horizontal

→ Criar uma função para desenhar um pixel

→ Função que devemha com pixel clerce:

→ Errado para o buffer, mas posição correta a car do pixel.

Nota: inicialmente para facilitar pode ensaiar-se diretamente para a rede a ram mas é deixa a prática ensaiar só para um buffer e de seguida para a rede RAM.

→ para isto pode ser usada a função memcpq, sendo que a posição de um pixel especifica rencí:

$$\text{Ler} + \text{leRes} \times \underbrace{y}_{\substack{\text{pontos para a} \\ \text{1ª posição do} \\ \text{buffer}}} \times \underbrace{(\text{Bits per Pixel} / 8)}_{\substack{\text{posição y} \\ \text{da pixel}}} + \underbrace{\text{le}}_{\substack{\text{ponta} \\ \text{de da} \\ \text{pixel}}} \times \underbrace{(\text{Bits per Pixel} / 8)}_{\substack{\text{posição} \\ \text{de da} \\ \text{pixel}}}$$

também atençõe  
se Bits per Pixel  
não é multipla de 8

(4) → De se usar um buffer é necessária ensaiar a informaçõa da buffer para a rede-memória quando mais tiver regras de memcpq, se não este para clerce se ignorado.

(5) → depar com os interrupt's nais das rede-mede.

`int(VIDEO_TOT_PATTERN)(uint16_t mode, uint8_t  
no_rectangles, uint32_t first, uint8_t  
step)`

função → desenhar um padrão de quadradinhos  
coloridos no ecran. Sair do video mode da grande  
para tecla escape é precionada no keyboard.

- ① → cria o interrupt para o keyboard
- ② → inicia video mode.
- ③ → Separar a cor necessária nas suas cores  
primárias (RGB → red green blue), isto depende  
do modo:

0x105 → indexado, não no direcional

0x110 → direcional, 1:5:5:5 → blue

red green

0x115 → direcional, 8:8:8:8 → blue

red green

0x11A → direcional, 5:6:5:5 → blue

red green

0x19C → direcional, 8:8:8:8 → blue

red green

④ → Criar as reuniões em cada uma das fórmulas:

Para O<sub>4</sub> e 105:

$$\text{color} = (\text{first\_color} + (\text{row} \times \text{mc\_rectangles} + \text{col}) \times \text{step}) \% (\text{f} \ll \text{bitsPerPixel})$$

Para os restantes:

$$R = (\text{firstR} + \text{col} \times \text{step}) \% (1 \ll \text{RedMarkSize})$$

$$G = (\text{firstG} + \text{row} \times \text{step}) \% (1 \ll \text{GreenMarkSize})$$

$$B = (\text{firstB} + (\text{col} + \text{row}) \times \text{step}) \% (1 \ll \text{BlueMarkSize})$$

⑤ → Calcular o tamanho dos quadrados para a reunião:

4 Res  
mcRect

9 Res  
noRect

⑥ → Chamar a `reg-draw-rectangles`.

⑦ → Colocar o buffer no `noRectMode` (se não usou um buffer, nele ignorar).

⑧ → Sair de `noRectMode`.

(7) → Entrar em modo de rede com o buffer para a rede com (não está a entrar diretamente na rede-mem (ignorar))

(8) → Sair do rede modo

```
int (rede-test-conect)(lepm-map-t lepm,  
uint16-t li, uint16-t gi, uint16-t lsf, uint16-t  
gg, uint16-t speed, uint8-t fn-rate)
```

função → manda a imagem lepm com  
speed "speed" e frame rate "fr-rate", não que exibe  
é priorizada.

Atenção!! → Para passar a tela só é necessário:

→ fazer a animação em idle mode.

→ apesar com as pausas finais ser  
diferente.

Para o projeto o priority que seja necessário expandir  
esta função.

(9) → Criar o interrupt para o Keyboard e  
para timer.

(10) → Entrar em rede modo.

③) → Há duas maneiras de fazer do frame rate

1 → muda a frequência do miniz para o frame rate e altera a imagem com cada interrupt.

2 → deixa a freq do miniz a 60 e calcular quantos interrupts correspondem a uma frame.

ex.:  $fr\_rate = 120 \rightarrow 120 \text{ frames por segundo}$

$$freq = 60 \text{ Hz} \rightarrow 60 \text{ cida/s}$$

1 ciclo → 2 frames

1 frame → 1 ciclo

tem a desvantagem de que se o frame rate for menor que a frequência é impossível de ser utilizada

→ Para isso se faz pelo método do P,

④) → Temos duas opções diferentes:

a) speed é positiva:

→ aumenta speed a cada frame a posição

b) speed é negativa:

→ a posição demora -speed frames a aumentar 1 pixel.

o mecaníco calcular quantas picas avançou para frame.

(5) → é mecaníco para de mover a imagem quando ela atinge a posição final mesmo que a utilizada não tenha posicionada a leitura exata.

(6) → escrever o buffer para a rede-mem

(7) → rain da rede-mem