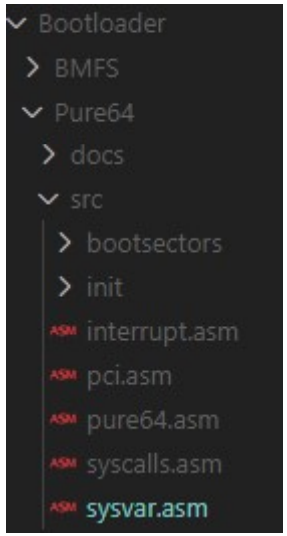
Setup modo de video

Habilitar el modo de Video



Primero, en el archivo `x64barebones/Pure64/src/sysvar.asm`

buscamos la constante `cfg_vesa` y cambiamos el valor a `1`

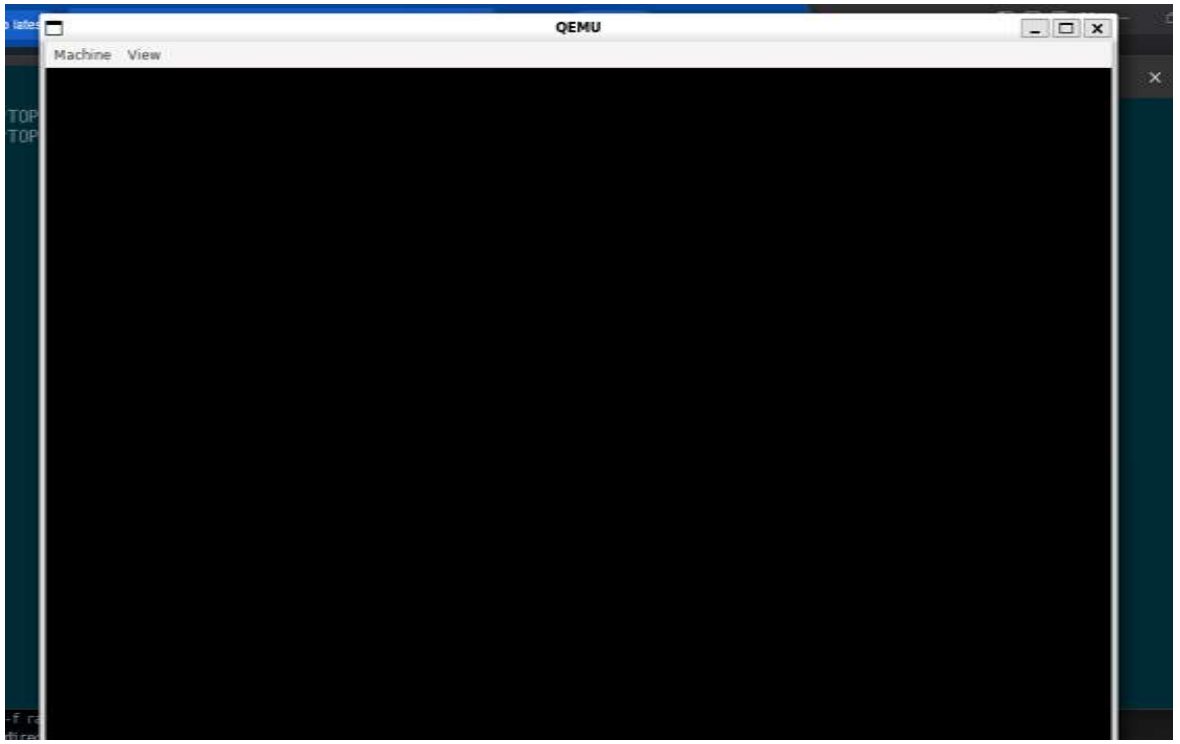
```
8
9   ;CONFIG
10  cfg_smpinit:        db 1    ; By default SMP is enabled. Set to 0 to disable.
11  cfg_vesa:       db 0    ; By default VESA is disabled. Set to 1 to enable.
12  cfg_default:        db 0    ; By default we don't need a config file so set to 0. If a con
13  cfg_e820:       db 1    ; By default E820 should be present. Pure64 will set this to 0 if
14  cfg_mbr:        db 0    ; Did we boot off of a disk with a proper MBR
15  cfg_hdd:        db 0    ; Was a bootable drive detected
16
```

Habilitar modo de video

Listo! Ya está habilitado el modo de video

1. Guardamos el archivo `sysvar.asm`
2. Recompilamos Toolchain y el proyecto con make
3. Ejecutamos ./run.sh

TROUBLESHOOTING: Un problema común



IMPORTANTE

Se dejan de lado los caracteres predefinidos y la paleta de 16 colores.

Ahora ~~pueden~~ **tienen** que dibujar la pantalla pixel por pixel

Información sobre la pantalla

Escribir en `0xA0000` ahora no va a cambiar nada, tenemos que encontrar la dirección a la cual tenemos que escribir para modificar lo que se muestra en pantalla

Información sobre la pantalla

Pure64 tiene precargado structs en direcciones específicas con sus características.

```
; Memory locations
E820Map:          equ 0x0000000000004000
InfoMap:          equ 0x0000000000005000
SystemVariables:     equ 0x0000000000005A00
VBEModeInfoBlock:    equ 0x0000000000005C00
ahci_cmdlist:        equ 0x0000000000070000
ahci_cmdtable:       equ 0x0000000000072000
```

Información sobre la pantalla

```
70  ; VESA
71  ; Mandatory information for all VBE revisions
    0 references
72  VBEModeInfoBlock.ModeAttributes      equ VBEModeInfoBlock + 0    ; DW - mode attributes
    0 references
73  VBEModeInfoBlock.WinAAttributes      equ VBEModeInfoBlock + 2    ; DB - window A attributes
    0 references
74  VBEModeInfoBlock.WinBAttributes      equ VBEModeInfoBlock + 3    ; DB - window B attributes
```

En particular, el struct `VBEModeInfoBlock` contiene toda la información necesaria sobre la pantalla

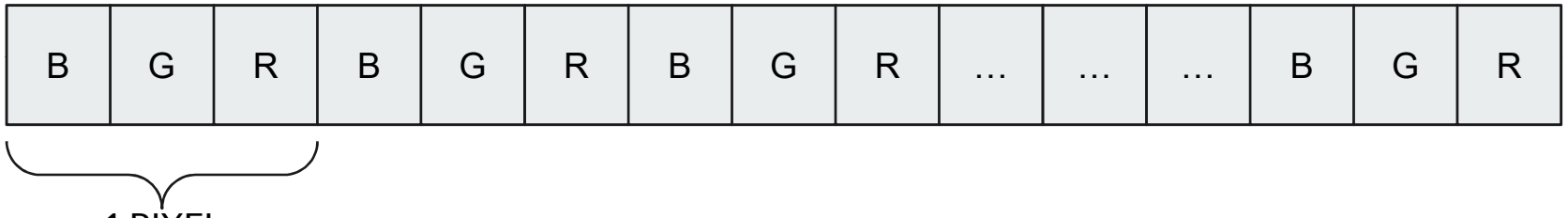
```
1 reference
VBEModeInfoBlock.PhysBasePtr
```

Estructura de la “memoria de video” en modo texto

EN UNA LÍNEA HORIZONTAL:

Estructura del array framebuffer (24 bits)

EN UNA LÍNEA HORIZONTAL:



Cada letra representa la intensidad de ese color en un byte

Creando una función para arrancar la librería gráfica

```
void putPixel(uint32_t hexColor, uint32_t x, uint32_t y)
  void putPixel(char r, char g, char b, int x, int y)
```

hexColor es el color del pixel en formato 0x00RRGGBB

x e y son las coordenadas del pixel en la pantalla

Cómo se mueve en eje x de pantalla?
Cómo se mueve en el eje y de la pantalla?

- `(VBE_mode_info->bpp)/8`
- `VBE_mode_info->pitch`

Tamaño de pantalla

- `vbe_mode_info->width // pixels`
- `vbe_mode_info->height// pixels`

Información adicional:

https://wiki.osdev.org/VESA_Video_Modes

https://wiki.osdev.org/User:Omarrx024/VESA_Tutorial

Primer uso de la función

```
putPixel(uint32_t hexColor, uint64_t x, uint64_t y)
```

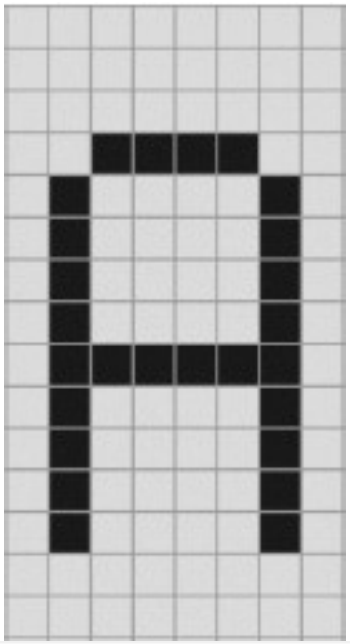
- En el Campus: videoDriver.c → /kernel/
- #include <stdint.h> →
  - uint32_t
  - uint64_t
  - etc.

Vamos a dibujar un pixel rojo al bootear el Kernel

- kernel/kernel.c → #include <videoDriver.h>
- `hexColor = 0x00RRGGBB → 0x 00 FF 00 00`
-
```
int main()
    {
    putPixel(0x00FF0000, 20, 20);
    ...
```

Y ahora... cómo dibujo los caracteres en pantalla?

- bitmap fonts
- 8 cols x 16 filas



```
uint8_t font_bitmap[4096] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7e, 0x81, 0xa5, 0x81, 0x81, 0xbd, 0x99, 0x81, 0x81, 0x7e, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x7e, 0xff, 0xdb, 0xff, 0xff, 0xc3, 0xe7, 0xff, 0xff, 0x7e, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x6c, 0xfe, 0xfe, 0xfe, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x18, 0x3c, 0x3c, 0xe7, 0xe7, 0xe7, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,
```

Más informacion:

- https://wiki.osdev.org/VGA_Fonts