Computer Labs: Lab5
VBE function 0x01: Return VBE Mode Information

2º MIEIC

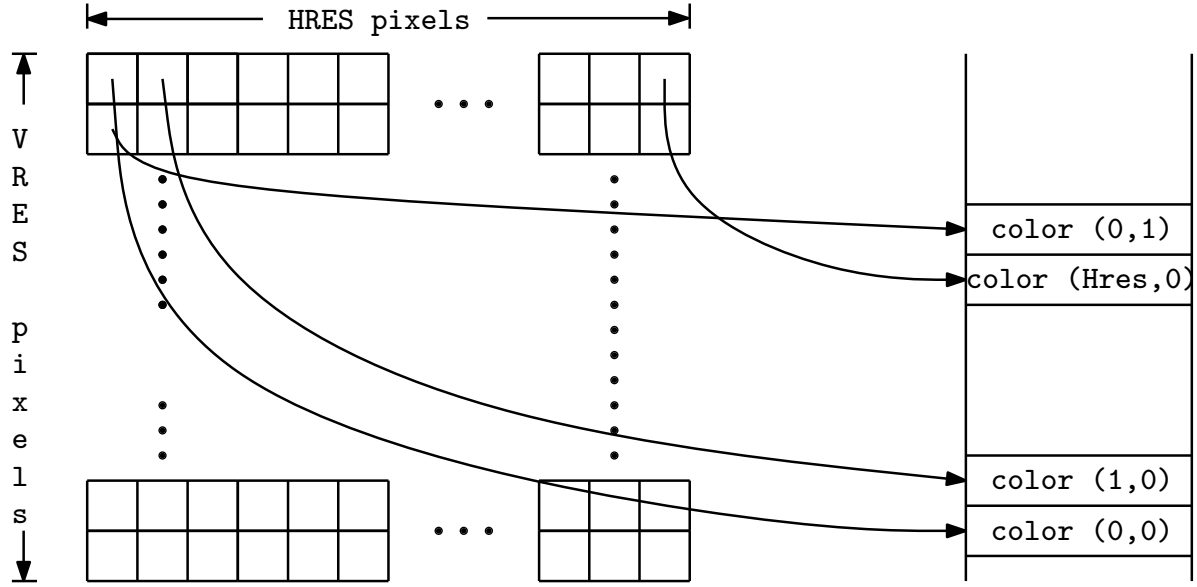
Pedro F. Souto (pfs@fe.up.pt)

November 13, 2017

Contents

Mapping the Linear Frame Buffer

- Before you can write to the frame buffer.



1. Obtain the physical memory address
   - 1.1 Using a hard-coded address (`0xE0000000`), first;
     - This address may depend on the VM used. So, I provide a program that allows you to find out this address
   - 1.2 Using Function `0x01` Return VBE Mode Information, once everything else has been completed.
2. Map the physical memory region into the process' (virtual) address space

Finding the Physical Memory Address with VBE (1/5)

Problem How do you find the physical memory address of the video frame buffer automatically (i.e. by programming)?

Answer VBE Function 01h - Return VBE Mode Information

	AX	= 4F01h	Return VBE Mode Information
Input	CX	=	Mode number
	ES:DI	=	Pointer to ModeInfoBlock structure

Ouput AX = VBE return status

- The ModeInfoBlock includes among other information:
  1. The mode attributes, which comprise a set of bits that describe some general characteristics of the mode, including whether:
     - it is supported by the adapter
     - the linear frame buffer is available
  2. The screen resolution of the mode
  3. The physical address of the linear frame buffer

Finding the Physical Memory Address with VBE (2/5)

Problem

- The ModeInfoBlock structure must be accessible both in protected mode and in real mode
  - VBE Function 01h is a real mode function
  - Real mode addresses are only 20-bit long (must be in the lower 1MiB).

Solution

- Use `liblm.a` library
  - Provides a simple interface for applications:

    `lm_init()`
    `lm_alloc()`
    `lm_free()`
  - Hides some non-documented functions provided by Minix 3
- The `mmap_t` includes both:
  - The physical address, for use by VBE
  - The virtual address, for use in Minix 3

Finding the Physical Memory Address with VBE (3/5)

```
phys_bytes buf;
struct reg86u r;

[...]                    /* use liblm.a to initialize buf */

r.u.w.ax = 0x4F01;             /* VBE get mode info */
/* translate the buffer linear address to a far pointer */
r.u.w.es = PB2BASE(buf);       /* set a segment base */
r.u.w.di = PB2OFF(buf);        /* set the offset accordingly */
r.u.w.cx = mode;
r.u.b.intno = 0x10;
if( sys_int86(&r) != OK ) { /* call BIOS  */
```

`PB2BASE` Is a macro for computing the base of a segment, a 16-bit value, given a 32-bit linear address;

`PB2OFF` Is a macro for computing the offset with respect to the base of a segment, a 16-bit value, given a 32-bit linear address;

Finding the Physical Memory Address with VBE (4/5)

Problem The parameters contained in the buffer returned by VBE function 0x01 are layed out sequentially, with no holes between them

- Simply defining a C struct with one member per parameter with an appropriate type, is not enough
- C compilers layout the members of a struct in order and place them in memory positions whose address is aligned according to their type

Solution Use GCC's `__attribute__((packed))`

- Note that this attribute must appear immediately after the `}`, otherwise it has no effect
- But, you need not do anything, as I've already defined the struct in `vbe.h`

Finding the Physical Memory Address with VBE (5/5)

```
#include <stdint.h>

typedef struct
{
   uint16_t ModeAttributes;
   [...]
   uint16_t XResolution;
   uint16_t YResolution;
   [...]
   uint8_t BitsPerPixel;
   [...]
   uint32_t  PhysBasePtr;
   [...]
} __attribute__((packed)) vbe_mode_info_t;
```