Computer Labs: BIOS and VBIOS Access

2º MIEIC

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

September 22, 2011

PC BIOS

- Basic Input-Output System is:
  - A firmware interface for accessing PC HW resources
  - The implementation of this interface
  - The non-volatile memory (ROM, more recently flash-RAM) containing that implementation
- It is used mostly when a PC when it starts up
  - It is 16-bits: even IA-32 processors start in real-mode
  - It is used essentially to load the OS (or part of it)
  - Once the OS is loaded, it usually uses its own code to access the HW not the BIOS

BIOS Calls

- Access to BIOS services is via the SW interrupt instruction `INT xx`
  - The `xx` is 8 bit and specifies the service.
  - Any arguments required are passed via the processor registers
- Standard BIOS services:

Interrupt vector (`xx`)	Service
`10h`	video card
`11h`	PC configuration
`12h`	memory configuration
`16h`	keyboard

BIOS Call: Example

- Set Video Mode: `INT 10h`, function `00h`

```
; set video mode

MOV AH, 0        ; function
MOV AL, 3        ; text, 25 lines X 80 columns, 16 colors
INT 10h
```

BIOS Call: From Minix 3

Problem

- The previous example is in real address mode
- Minix 3 uses protected mode with 32-bit

Solution

- Use Minix 3 kernel call `SYS_INT86`

  > "Make a real-mode BIOS on behalf of a user-space device driver. This temporarily switches from 32-bit protected mode to 16-bit real-mode to access the BIOS calls."

BIOS Call in Minix 3: Example

```
#include <machine/int86.h>
int vg_exit() {
  struct reg86u reg86;

  reg86.u.b.intno = 0x10;
  reg86.u.b.ah = 0x00;
  reg86.u.b.al = 0x03;

  if( sys_int86(&reg86) != OK ) {
    printf("vg_exit(): sys_int86() failed \n");
    return 1;
  }
  return 0;
}
```

- `struct reg86u` is a struct with a union of structs
  - `b` is the member to access 8-bit registers
  - `w` is the member to access 16-bit registers
  - `l` is the member to access 32-bit registers
- The names of the members of the structs are the standard names of IA-32 registers.

Video BIOS Extension (VBE)

- The BIOS specification supports only VGA graphics modes
  - VGA stands for Video Graphics Adapter
  - Specifies very low resolution: 640x480 @ 16 colors and 320x240 @ 256 colors
- The Video Electronics Standards Association (VESA) developed the Video BIOS Extension (VBE) standards in order to make programming with higher resolutions portable
- Early VBE versions specify only a real-mode interface
- Later versions added a protected-mode interface, but:
  - In version 2, only for some time-critical functions;
  - In version 3, supports more functions, but they are optional.

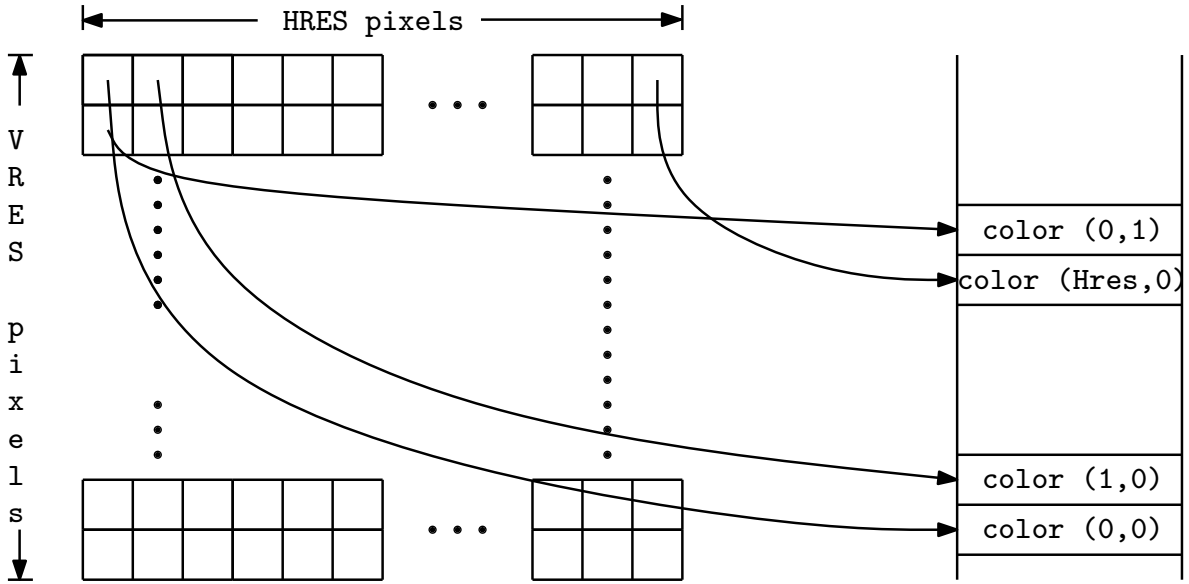
VBE `INT 0x10` Interface

- VBE still uses `INT 0x10`, but to distinguish it from basic video BIOS services
  - `AH = 4Fh` - BIOS uses `AH` for the function
  - `AL = function`
- VBE graphics mode 105h, 1024x768@256, **linear** mode:

```
struct reg86u r;
r.w.ax = 0x4F02; // VBE call, function 02 -- set VBE mode
r.w.bx = 1<<14|0x105; // set bit 14: linear framebuffer
r.b.intno = 0x10;
if( sys_int(&r) != OK ) {
    printf("set_vbe_mode: sys_int86() failed \n");
    return 1;
}
```

Video Card in Graphics Mode

- Like in text mode, the screen can be abstracted as a matrix
  - Now, a matrix of points, or **pixels**, instead of characters
  - For each pixel, the VRAM holds its color



- With a linear framebuffer, addressing of a pixel is very easy. Need only know:
  - The base address of the frame buffer
  - The coordinates of the pixel
  - The number of bytes required to encode the color

Accessing the Linear Frame Buffer

1. Obtain the physical memory address
   1.1 Using a hard-coded address (`0xD0000000`), first;
   1.2 Using Function `0x01` Return VBE Mode Information, once everything else has been completed.
2. Map the physical memory region into the process' address space

- Steps 2 was already described in the Lab 1 slides

Obtaining the Physical Memory Address with VBE

- VBE Function 01h - Return VBE Mode Information:

  Input

AX	= 4F01h	Return VBE Mode Information
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

Obtaining the Physical Memory Address with VBE

Problem

- The ModeInfoBlock structure must be accessible both in protected mode and in real mode
  - VBE Function 01h is a real mode function

Solution

- Use the `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` (already used in Lab 1) includes both:
  - The physical address, for use by VBE
  - The virtual address, for use in Minix 3

Obtaining the Physical Memory Address with VBE

```
int vbe_get_mode_info(unsigned short mode, phys_bytes buf) {
  struct reg86u r;

  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;

Obtaining the Physical Memory Address with VBE

Problem (Last) The ACK-based C compiler does not support packed structs.

- GCC supports this via the `__attribute__ ((packed))` extension;
- In principle, this should be handled by the `#pragma` directive

Solution Use the function provided `vbe_unpack_mode_info()`

- Copies the data in the VBE `ModeInfoBlock` struct, to a C struct with the same fields.
- Use of the C struct is less error prone than using the unpacked buffer.

Obtaining the Physical Memory Address with VBE

```
typedef struct
{
   unsigned short ModeAttributes;
   [...]
   unsigned short XResolution;
   unsigned short YResolution;
   [...]
   unsigned char  BitsPerPixel;
   [...]
   unsigned long  PhysBasePtr;
   [...]
} vbe_mode_info_t;
```