# Computer Labs: Lab 1

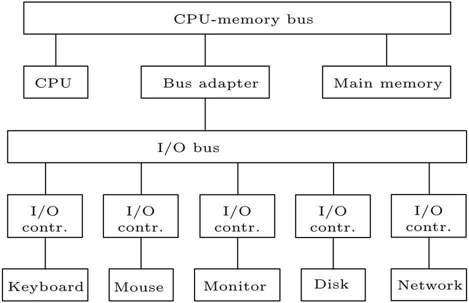
## 2º MIEIC

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

September 13, 2012

# I/O Devices

- In LCOM, we will work with the PC I/O devices.
- I/O devices provide the interface between the CPU and the outside world.



# I/O Controllers

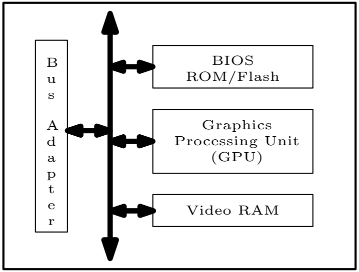
- Each I/O device is controlled by an electronic component, usually called **controller** or **adapter.**
- I/O controllers typically include three kinds of registers:

  Control: used to request I/O operations

  Status: used to get the state of the device or pending I/O operations

  Data: used to transfer data to/from the I/O devices
- Programming at the register level may require a detailed knowledge of the device's operation

# Graphics Adapter




**GPU** Earlier known as the Graphics Controller:

- Controls the display hardware (CRT vs. LCD)
- Performs 2D and 3D rendering algorithms, offloading the CPU and accelerating graphics applications

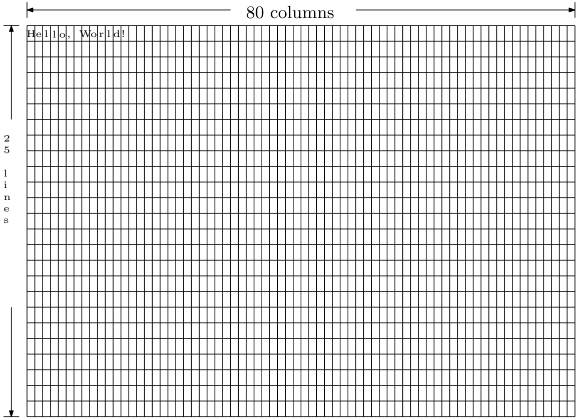
**BIOS ROM/Flash** ROM/Flash Memory with firmware. Includes code that performs some standardized basic video I/O operations, such as the Video BIOS Extension (VBE)

**Video RAM** Stores the data that is rendered on the screen.

- It is acessible also by the CPU (at least part of it)

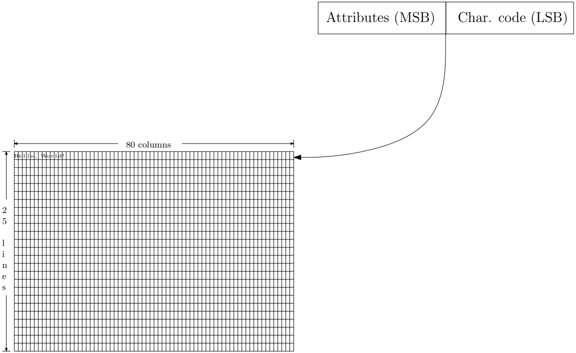
# PC's Graphics Adapter Text Modes (1)

- Used to render mostly text
- Abstracts the screen as a matrix of characters (row x cols)
  - E.g. **25x80**, 25x40, 50x80, 25x132
  - Black and white vs color (16 colors)



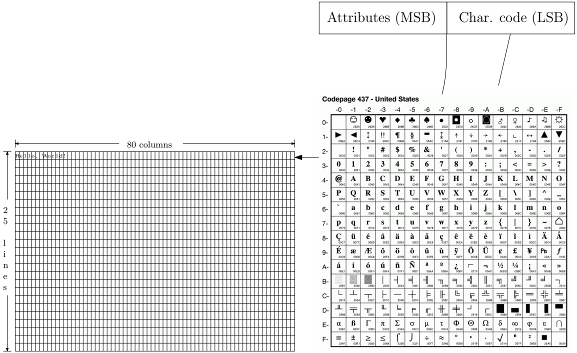
# PC's Graphics Adapter Text Modes (2)

- Each character is represented by two bytes:



# PC's Graphics Adapter Text Modes (2)

- Each character is represented by two bytes:
  - The character denoted by the code depends on the character encoding (code page ▶), which can be changed

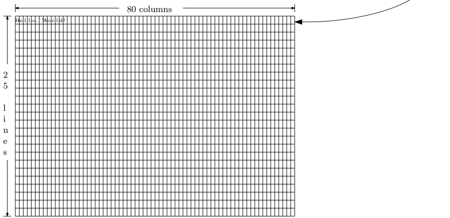


# PC's Graphics Adapter Text Modes (2)

- Each character is represented by two bytes:
  - The character denoted by the code depends on the character encoding (code page ▸), which can be changed
  - The attributes specify mostly the colors

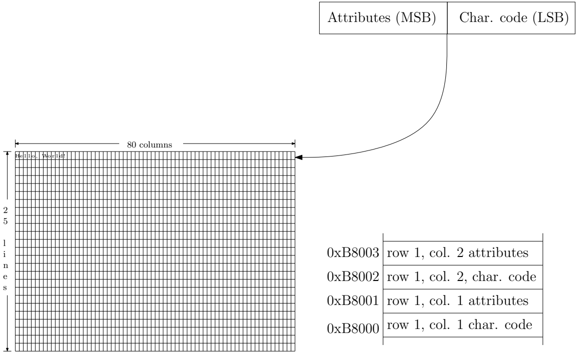
| Attributes (MSB) | Char. code (LSB) |
|---|---|

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| blink | red | green | blue | bright. | red | green | blue |
| foreg. | background color | | | foreground color | | | |



# PC's Graphics Adapter Text Modes (2)

- Video RAM contains a representation of the screen in a matrix of 25x80 16-bit words
  - In the PC, this matrix is at **physical address** 0xB8000
  - By changing the contents of this matrix an application changes what is displayed on the screen




| | |
|---|---|
| 0xB8003 | row 1, col. 2 attributes |
| 0xB8002 | row 1, col. 2, char. code |
| 0xB8001 | row 1, col. 1 attributes |
| 0xB8000 | row 1, col. 1 char. code |

# Lab 1

- Write a set of functions:

```
void vt_fill(char ch, char attr);
void vt_blank(void);
int vt_print_char(char ch, int r, int c, char attr);
int vt_print_string(char *str, int r, int c, char attr);
int vt_print_int(int n, int r, int c, char attr);
int vt_draw_frame(int width, int height, int r, int c, cha
```

  to output some characters on the screen in text mode, by writing to video RAM (VRAM)
- No need to configure the video controller/GPU:
  - You'll use the Minix 3 default configuration.
- Need "only" to write to the appropriate positions of VRAM

# Virtual and Physical Address Spaces

Issue 1 Most computer architectures support a **virtual address space** that is decoupled from the **physical address space**

- Processes can access physical memory using a **logical** address that is independent of the physical address (determined by the address bus decoding circuit)
- Most modern operating systems, including Minix, take advantage of this feature to simplify memory management.

Issue 2 In modern operating systems, **user-level processes** cannot access **directly** HW resources, including VRAM

- Minix 3 handles this by allowing to grant **privileged** user-level processes the permissions they require to perform their tasks

Nomenclature note A **program** is a sequence of instructions that can be executed by a processor. A **process** is a program in execution.

# Mapping Physical Memory to Virtual Address Space

- Each process has its own virtual address space, whose size is usually determined by the processor architecture (32-bit for IA-32)
- The operating system maps regions of the physical memory in the computer to the virtual address spaces of the different processes
  - The details of how this is done are studied in the Operating Systems course.

# Lab 1: `char *vt_init(vt_info_t vip)`

- Mainly, maps VRAM on the address space of a process
  - Returns the address of the first byte of the process' address space region onto which VRAM was mapped
  - Subsequent accesses to that region of the process' address space access VRAM
    - Usually, to change the characters displayed on the screen and/or their attributes.

**Issue** how can one access a region of a process address space in C?

# Lab 1: Preparation

- Read the material provided
  - Lab 1 script;
  - Supporting notes;
  - Class notes.
- Write the functions:

  `vt_fill()` which should fill the entire screen with the same character and attribute;

  `vt_blank()` which should blank the screen

# Lab 1: Key Programming Issue

Given a virtual address, what is the C code that allows a process to access the physical memory mapped to that virtual address?

# Character Encodings (Code Pages)

- The first 128 characters are the same for all western-language code pages.

**Codepage 437 - United States**

| | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0- | | ☺ 263A | ☻ 263B | ♥ 2665 | ♦ 2666 | ♣ 2663 | ♠ 2660 | • 2022 | ◘ 25D8 | ○ 25CB | ◙ 25D9 | ♂ 2642 | ♀ 2640 | ♪ 266A | ♫ 266B | ☼ 263C |
| 1- | ► 25BA | ◄ 25C4 | ↕ 2195 | ‼ 203C | ¶ 00B6 | § 00A7 | ▬ 25AC | ↨ 21A8 | ↑ 2191 | ↓ 2193 | → 2192 | ← 2190 | ∟ 221F | ↔ 2194 | ▲ 25B2 | ▼ 25BC |
| 2- | 0020 | ! 0021 | " 0022 | # 0023 | $ 0024 | % 0025 | & 0026 | ' 0027 | ( 0028 | ) 0029 | * 002A | + 002B | , 002C | - 002D | . 002E | / 002F |
| 3- | 0 0030 | 1 0031 | 2 0032 | 3 0033 | 4 0034 | 5 0035 | 6 0036 | 7 0037 | 8 0038 | 9 0039 | : 003A | ; 003B | < 003C | = 003D | > 003E | ? 003F |
| 4- | @ 0040 | A 0041 | B 0042 | C 0043 | D 0044 | E 0045 | F 0046 | G 0047 | H 0048 | I 0049 | J 004A | K 004B | L 004C | M 004D | N 004E | O 004F |
| 5- | P 0050 | Q 0051 | R 0052 | S 0053 | T 0054 | U 0055 | V 0056 | W 0057 | X 0058 | Y 0059 | Z 005A | [ 005B | \ 005C | ] 005D | ^ 005E | _ 005F |
| 6- | ` 0060 | a 0061 | b 0062 | c 0063 | d 0064 | e 0065 | f 0066 | g 0067 | h 0068 | i 0069 | j 006A | k 006B | l 006C | m 006D | n 006E | o 006F |
| 7- | p 0070 | q 0071 | r 0072 | s 0073 | t 0074 | u 0075 | v 0076 | w 0077 | x 0078 | y 0079 | z 007A | { 007B | \| 007C | } 007D | ~ 007E | ⌂ 2302 |
| 8- | Ç 00C7 | ü 00FC | é 00E9 | â 00E2 | ä 00E4 | à 00E0 | å 00E5 | ç 00E7 | ê 00EA | ë 00EB | è 00E8 | ï 00EF | î 00EE | ì 00EC | Ä 00C4 | Å 00C5 |
| 9- | É 00C9 | æ 00E6 | Æ 00C6 | ô 00F4 | ö 00F6 | ò 00F2 | û 00FB | ù 00F9 | ÿ 00FF | Ö 00D6 | Ü 00DC | ¢ 00A2 | £ 00A3 | ¥ 00A5 | ₧ 20A7 | ƒ 0192 |
| A- | á 00E1 | í 00ED | ó 00F3 | ú 00FA | ñ 00F1 | Ñ 00D1 | ª 00AA | º 00BA | ¿ 00BF | ⌐ 2310 | ¬ 00AC | ½ 00BD | ¼ 00BC | ¡ 00A1 | « 00AB | » 00BB |
| B- | ░ 2591 | ▒ 2592 | ▓ 2593 | │ 2502 | ┤ 2524 | ╡ 2561 | ╢ 2562 | ╖ 2556 | ╕ 2555 | ╣ 2563 | ║ 2551 | ╗ 2557 | ╝ 255D | ╜ 255C | ╛ 255B | ┐ 2510 |
| C- | └ 2514 | ┴ 2534 | ┬ 252C | ├ 251C | ─ 2500 | ┼ 253C | ╞ 255E | ╟ 255F | ╚ 255A | ╔ 2554 | ╩ 2569 | ╦ 2566 | ╠ 2560 | ═ 2550 | ╬ 256C | ╧ 2567 |
| D- | ╨ 2568 | ╤ 2564 | ╥ 2565 | ╙ 2559 | ╘ 2558 | ╒ 2552 | ╓ 2553 | ╫ 256B | ╪ 256A | ┘ 2518 | ┌ 250C | █ 2588 | ▄ 2584 | ▌ 258C | ▐ 2590 | ▀ 2580 |
| E- | α 03B1 | ß 00DF | Γ 0393 | π 03C0 | Σ 03A3 | σ 03C3 | µ 03BC | τ 03C4 | Φ 03A6 | Θ 0398 | Ω 03A9 | δ 03B4 | ∞ 221E | φ 03C6 | ε 03B5 | ∩ 2229 |
| F- | ≡ 2261 | ± 00B1 | ≥ 2265 | ≤ 2264 | ⌠ 2320 | ⌡ 2321 | ÷ 00F7 | ≈ 2248 | ° 00B0 | ∙ 2219 | · 00B7 | √ 221A | ⁿ 207F | ² 00B2 | ■ 25A0 | 00A0 |