

Computer Labs: Lab 1

Video Adapter in Text Mode

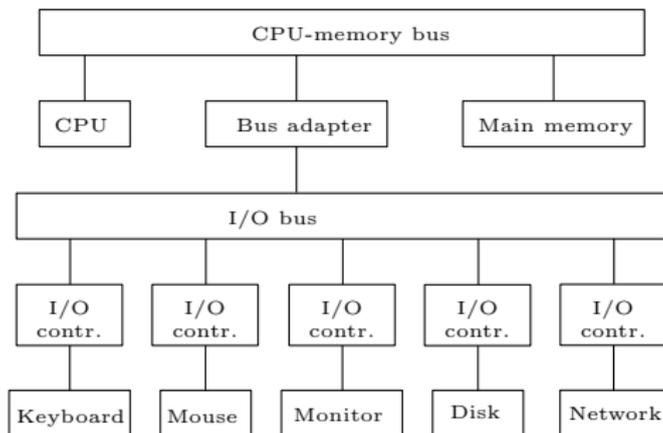
2º MIEIC

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

September 17, 2013

I/O Devices

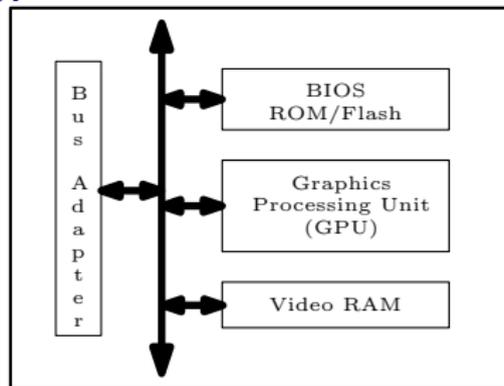
- ▶ In LCOM, we will work with the PC I/O devices.
- ▶ I/O devices provide the interface between the CPU and the outside world.



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

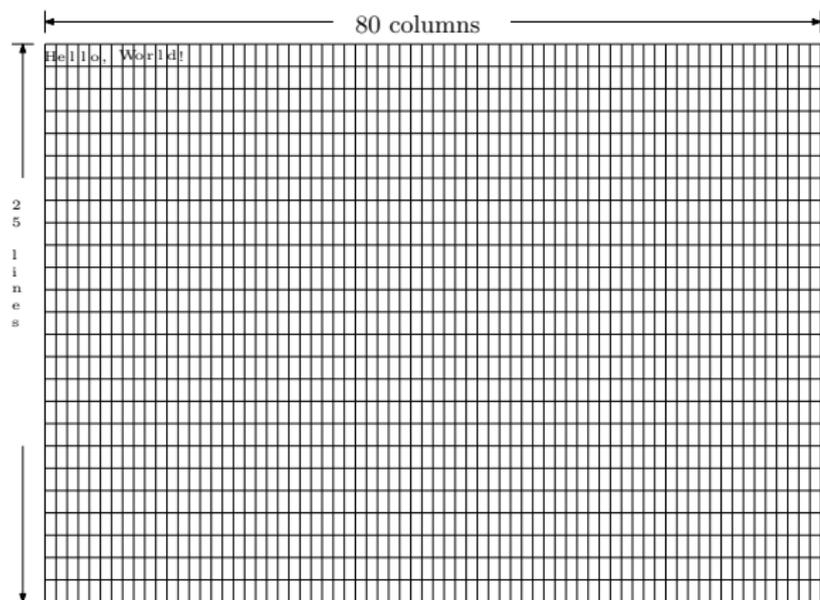
BIOS ROM/Flash ROM/Flash Memory with firmware. Includes code that performs some standardized basic video I/O operations, such as the Video BIOS Extension (VBE)

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

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

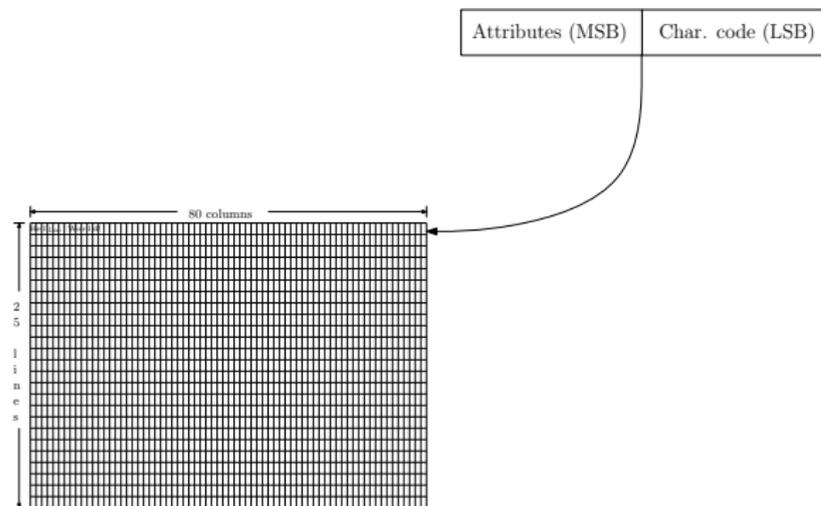
PC's Graphics Adapter Text Modes (1)

- ▶ 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)



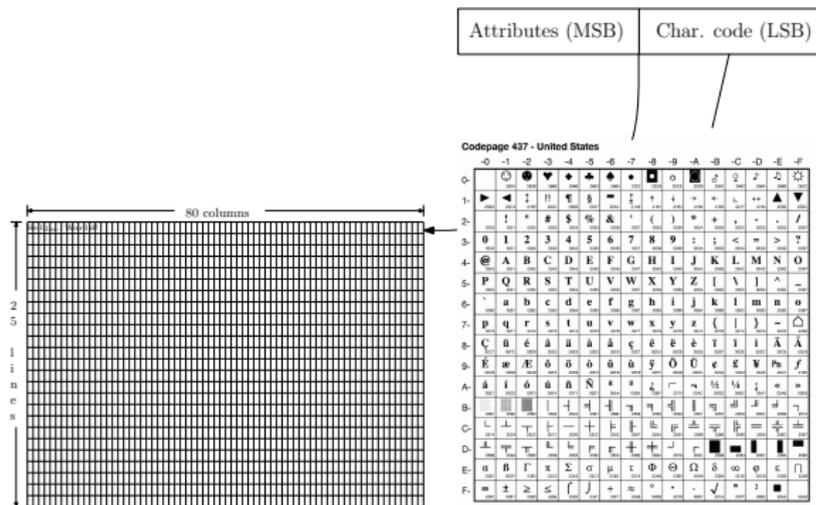
PC's Graphics Adapter Text Modes (2)

- ▶ Each character is represented by two bytes:



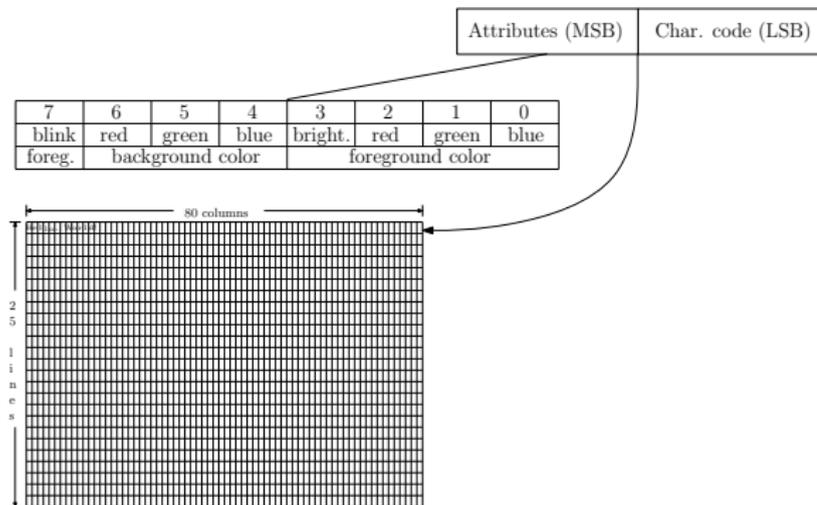
PC's Graphics Adapter Text Modes (2)

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



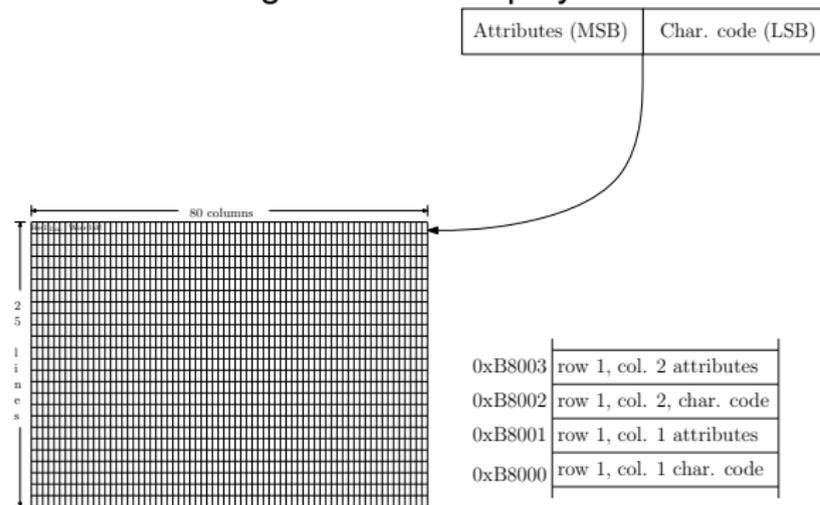
PC's Graphics Adapter Text Modes (2)

- ▶ 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



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



Lab 1

- ▶ Write a set of functions:

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

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)`

- ▶ The implementation of this function is complete
 - ▶ You should not change it.
- ▶ 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-																
1-																
2-																
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{	 	}	~	
8-	Ç	ü	ë	â	ä	à	â	ç	ê	ë	ï	î	Ï	À	Á	Â
9-	É	æ	Æ	ó	ò	ù	û	ü	ÿ	Ö	Ü	é	£	¥	Pls	f
A-	á	í	ó	ú	ñ	Ñ	ã	ç	í	¼	½	¾	;	«	»	
B-																
C-																
D-																
E-	α	β	Γ	π	Σ	σ	μ	τ	Φ	Ω	δ	∞	φ	Γ	∩	
F-	≡	±	≥	≤	∫	J	÷	≈	°	•	•	√	"	2	■	◻