Computer Organización

Iván de Jesús Deras Tábora

January 29, 2020

Period 1 2020

Graphics API

Graphics API

The MIPS32SOC graphics API include the following functions:

Function prototypes

```
void clear_screen();
void set_cursor(uint8_t row, uint8_t column);
void get_cursor(uint8_t *row, uint8_t *column);
void set_color(uint8_t fgcolor, uint8_t bgcolor);
void get_color(uint8_t *fgcolor, uint8_t *bgcolor);
void put_char(uint8_t ch);
void puts(char *str);
void put_decimal(uint32_t n);
```

You'll have to include the header file `screen.h` in order to use any of this functions.

`clear_screen` function

This function will fill the whole screen using the background color y set the cursor to position (0, 0). The easiest way to do this is to print the spaces using the current color attribute.

`set_cursor` function

This function will define the cursor position, that is the position on the screen where we are going to display the next character. Use global variables to store this position.

`get_cursor` function

This function will return the current cursor position.

`set_color` function

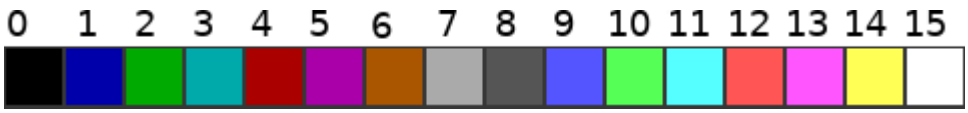
This function will define the foreground and background color to display text on the screen, use two globals variables to define this attributes.

`get_color` function

This function will return the defined attributes for the foreground and background color.

`put_char` function

To implement this function you have to understand how to work with the VGA text mode driver. The driver we'll use supports 80x30 characters resolution, that means 80 columns by 30 rows. The content that the monitor displays in this mode is mapped to a memory region (this is called memory mapped input/output). In our case the starting address is `0xb800` and the end address is `0xcabf` . Every character ocuppies 2 bytes in memory, the leftmost byte is the attribute color and the rightmost byte is the character to display. The attribute byte is further divided into two parts, the leftmost 4 bits represent the background color and the rightmost 4 bits represent the foreground color. This means we can have at most 16 colors (this is called the color palette), which are shown in the following figure:



Decimal Index	Hex index	Name
0	0	Black
1	1	Blue
2	2	Green
3	3	Cyan
4	4	Red
5	5	Magenta
6	6	Brown
7	7	White
8	8	Gray
9	9	Light blue
10	a	Light green
11	b	Light cyan
12	c	Light red
13	d	Light magenta
14	e	Yellow
15	f	Bright White

You can change the color palette (for now we'll stick to this colors) but we are limited to 16 colors, because we are using only 4 bits.

In summary to implement this function you have to get the current cursor position and mapped it to an offset in the VGA memory, then combine the foreground and the background color in one byte, the attribute color. Finally combine the attribute color and the character in one 16 bits variable and write that to the VGA memory. To clarify this let's look at the following example:

Sample code

```
uint16_t data = 0xcf00 | 'a';
uint16_t *vgaptr = (uint16_t *)0xb800;

*vgaptr = data;
```

This example will show the letter `'a'` in the first position row = 0, column = 0, using a background color Light red `0xc` and a foregound color bright white `0xf`.
Finally `put_char` should interpret the end of line character `'\n'`. When the function is called with this character it has to increment the current row, and the set the current column to zero, in this case it won't print anything on the screen.

`puts` function

The `puts` function is very simple once you have `put_char` implemented, just call it once per every character on the string.

`put_decimal` function

The `put_decimal` function will print a unsigned number on the screen, the easiest way to do it is to convert the number to string, then print the string using `puts`. Just remember in order to convert the number to string you have to divide by 10, but the simulator doesn't support division, you have to implement your own division function.