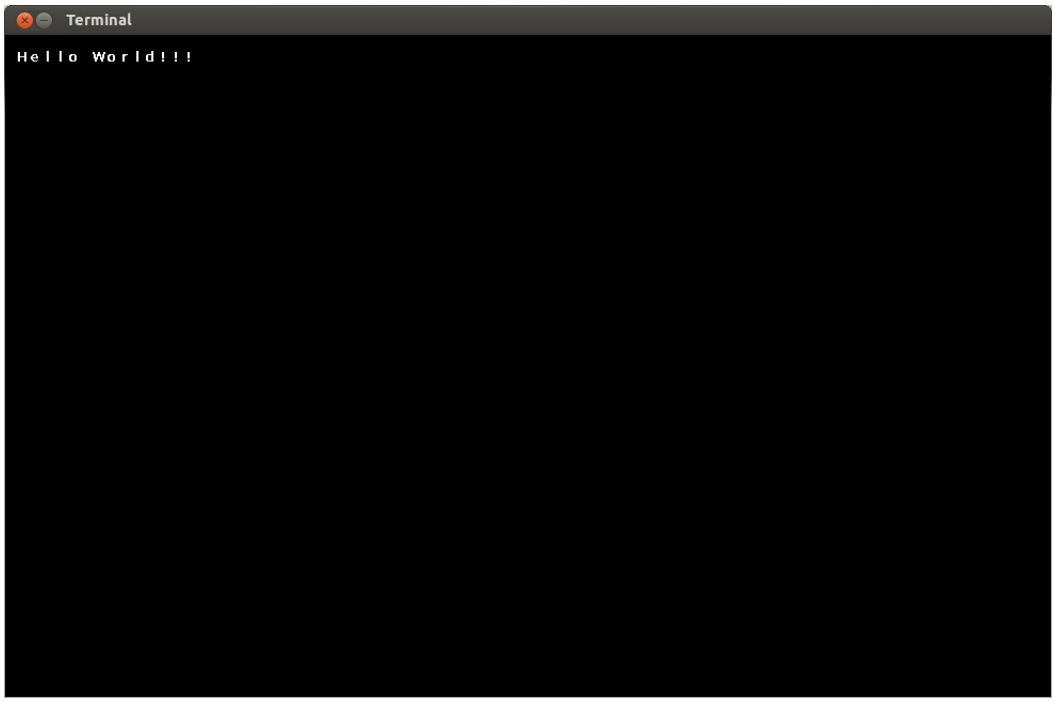
# Project 3 - The Matrix

CS 0447 — Computer Organization & Assembly Language

Check the Due Date on the CourseWeb

The purpose of this project is for you to practice writing assembly language to a terminal (console) screen. The hardware for this project is a terminal that allows you to put a printable character with a specific color on the screen. This terminal consists of 40 rows and 80 columns of characters. An example, of the terminal for this project is shown below:

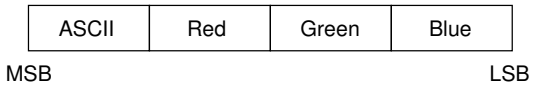



The Terminal (Mars Tool) can be found in `Terminal.zip` located in the CourseWeb under this project. Extract all files to your `[..]/mars4_5/mars/tools` directory. If you extract all files to the right directory, when you run the MARS program, you should see "Terminal (Memory) V0.1" under the "Tools" menu.

## Introduction to the Terminal (Mars Tool)

This terminal is a simple hardware that maps a memory region (from `0xFFFF8000` to `0xFFFFB199`) into characters on the screen. This is a one-to-one mapping where 4 bytes in the memory is mapped into one character on the console screen. The mapping starts from left to right and from top to bottom. For example, a **word** at the memory location `0xFFFF8000` is the character at row 0 column 0 (top-left corner). A **word** at the memory location `0xFFFF8004` is the character at row 0 column

1, and so on. To put a character on this terminal, you have to set a word into a specific value as shown below:




The word for a character is organized as follows:

- Bit 31 to Bit 24: The ASCII value of the character to be displayed (32 - 126)
- Bit 16 to Bit 23: The value of the red color (0 - 255)
- Bit 8 to Bit 15: The value of the green color (0 - 255)
- Bit 0 to Bit 7: The value of the blue color (0 - 255)

For example, if you want to put the character `'A'` (ASCII value = 0x41) with the white color (red = 255, green = 255, and blue = 255) at the top-left corner (row 0 column 0) on the terminal screen, the word value should be `0x41FFFFFF` and this word must be stored at the memory location `0xFFFF8000`. For another example, if you want to put the character `'M'` (ASCII value = 0x4D) with the green color (red = 0, green = 255, and blue = 0) at the row 5 column 12, the word value should be `0x4D00FF00` and this word must be stored at the memory location `0xFFFF8000` + (5 * 80 * 4) + (12 * 4). Note that (5 * 80 * 4) + (12 * 4) is the offset (in bytes in decimal) from the address `0xFFFF8000`. A simple example of the program that print `Hello World!!!` in white color as shown in previous page is shown below:

```
.data
        hello:  .asciiz  "Hello World!!!"
.text
        li   $s0, 0xffff8000         # $s0 - Base address of the terminal
        li   $s1, 0x00ffffff         # $s1 - Color of character (white)
        la   $s2, hello              # $s2 - Address of the string hello
loop:   lb   $s3, 0($s2)             # Load a character
        beq  $s3, $zero, done        # Encounter the null-character, done
        sll  $s3, $s3, 24            # Move ascii value to the top 8-bit
        or   $s3, $s3, $s1           # Put color to the bottom 24-bit
        sw   $s3, 0($s0)             # Put character on the terminal
        addi $s2, $s2, 1             # Go to the next character (string hello)
        addi $s0, $s0, 4             # Go to the next word (terminal)
        j    loop
done:
        addi $v0, $zero, 10          # Syscall 10: terminate program
        syscall                      # Terminate program
```

This terminal also has the ability to receive keyboard input. If the terminal window is highlight (active), when you press a key on your keyboard, the ASCII value of the key that you press will be stored in a **word** at the memory location `0xFFFFB204`. **Note** that you must load the whole word not just a byte and the ASCII value will be at the lower 8-bit. For example:

```
        li   $s0, 0xffffb204          # $s0 - Address of keyboard input
        lw   $s1, 0($s0)              # $s1 - Content (in word) at 0xffffb204
        andi $s1, $s1, 0x000000ff     # Get rid of the top 24 bits
        beq  $s1, $zero, noInput      # If the content is 0, no input
        # handle the input
        sw   $zero, 0($s0)            # Clear the content to 0 for the next input
noInput:
        ...
```

Note that the keyboard input will be disabled if the content in the memory at `0xFFFFB204` is not 0. So, if you see that there is a keyboard input, after you handle the input, you **must** store 0 back to the address `0xFFFFb204`.

## The Matrix

Some of you may feel like we live in the Matrix. We are not who we are. If that is the case, we are simply slaves who produce electricity for machines. We are living in the Matrix which is simply a code that no one could understand except Neo. Too bad he was kidnapped by the source. So, we cannot ask him to explain the Matrix for us. The screen below show what the Matrix looks like from a human (who is not in the Matrix) point-of-view:


## What to Do?

For this project, you have to create the **animated** Matrix screen on the terminal. Note that the animation effect can be obtain by updating the terminal repeatedly. The following are requirements that you must follow:

1. Fill up the terminal screen with random printable characters (33 - 126) with dark-green color (red = 0, green = 0x22, and blue = 0).

2. Create the animation to make the screen looks like characters are falling. **Note** that characters are not actually falling. Their color are changed to make them look like they are falling. Look at a second of the matrix below:

The table below show values of colors (`0xRRGGBB`) of characters (from bottom to top) at this iteration and next iteration:

| This Iteration | | Next Iteration | |
|---|---|---|---|
| Character | Color | Character | Color |
| 5 | 0x002200 | 5 | 0x002200 |
| H | 0x002200 | H | 0x00FF00 |
| Q | 0x00FF00 | Q | 0x00EE00 |
| } | 0x00EE00 | } | 0x00DD00 |
| ^ | 0x00DD00 | ^ | 0x00CC00 |
| E | 0x00CC00 | E | 0x00BB00 |
| ; | 0x00BB00 | ; | 0x00AA00 |
| x | 0x00AA00 | x | 0x009900 |
| & | 0x009900 | & | 0x008800 |
| A | 0x008800 | A | 0x007700 |
| R | 0x007700 | R | 0x007700 |
| ⋮ | ⋮ | ⋮ | ⋮ |

**Note** that once the color of a characters goes down to `0x002200`, it should not be changed any more. Otherwise, the color may be 0 (disappear) or negative values.

## Grading Rubric

We will use this grading rubric when we grade this project

| **Total Points** | Objective |
|---|---|
| 10 | If we can compile your program but nothing works |
| 50 | If you can fill up the terminal screen with random characters with dark-green color (red = 0, green = 0x22, and blue = 0). |
| 70 | If you can animate **one** randomly pick column at a time |
| 80 | If you can animate **multiple** randomly picked columns but they start at the same time and fall at the same speed |
| 90 | If you can animate **multiple** randomly picked columns but they start at different time and fall at the same speed |
| 100 | If you can animate **multiple** randomly picked columns and they start at different time with variable falling speed |

## Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. You should submit the file `calculator.asm` via CourseWeb.