LAB 01:
Hello World!

Provided Files

- `Makefile`

Files to Edit/Add

- `.vscode`
  - `tasks.json`
- `main.c`

Instructions

Make sure to copy over your .vscode folder (including your tasks.json file) from Lab00! In this lab, you will write your very first GBA program, which will allow you to set the colors of individual pixels and draw horizontal and vertical lines.

TODO 1 – Mode 3 Setup & setPixel

In this TODO, you will be implementing macros to search for pixels in the video buffer and change their colors. Then, you will use these macros to implement the `setPixel` method.

TODO 1.0

- Set up the display control register to use mode 3 and enable background 2.
  - The `REG_DISPCTL` macro can be used to access the display control register.
  - If done correctly, the screen should turn black.
  - Hint: This step can be completed using a bitwise operation!

TODO 1.1

- Complete the `RGB` macro. This macro should take in three color values [0, 31] and output a 16-bit color value with its red, green, and blue values bit shifted to the correct positions.

TODO 1.2

- Complete the `OFFSET` macro. This should take in an x value, a y value, and a width value, and it should output a single number indicating the distance of a cell in a 2D grid from the top-left corner when traversed row by row from left to right.

- The x and y values will denote x and y position in a 2D grid.
- The width value will denote the width of the grid.
- NOTE: Don't use the `PIXEL00` macro in this step! The `OFFSET` macro is a general purpose tool that can be used for more than just locating pixels in the video buffer. The setPixel function will handle the logic for that!

TODO 1.3

- Complete the setPixel function. This should use the OFFSET macro you previously completed.
  - The `PIXEL00` macro represents the position of the top-left pixel on the screen.
  - When calling the `OFFSET` macro, set its x and y input values to those of the pixel you want to access, and set its width input value to the width of the GBA screen.
  - Remember that each pixel in the video buffer is 2 bytes long!
  - NOTE: You will need to use the `volatile` keyword to access the video buffer without any issues. Look at the `REG_DISPCTL` macro if you're unsure how to use it!

UNCOMMENT TODO 1.4

- Uncomment the lines of code between TODO 1.4 and END TODO 1.4.

Build and run. You should see white, red, blue and green pixels stacked vertically in the top left corner of the screen. If you do not, something is wrong that needs to be fixed before moving on to the next sections.

TODO 2 – drawing lines

TODO 2.0

- Complete drawHorizontalLine. This function should take in an int for y, and then an x1 and x2 value. The line should start at x1 and stop at x2.

TODO 2.1

- Complete drawVerticalLine. This is the same as the previous TODO except the line is vertical, so it is taking in an x, and then the line starts at y1 and stops at y2.

UNCOMMENT TODO 2.2

- Uncomment the lines of code between TODO 2.2 and END TODO 2.2.

Build and run. You should see the word 'WORLD' spelled out on the screen along with a horizontal line below it.

TODO 3 – HELLO

TODO 3.0

- Write a series of drawVerticalLine, drawHorizontalLine, and setPixel to write the word ‘HELLO’ above the provided ‘WORLD’.

Build and run. You should see ‘HELLO WORLD’ spelled on the screen! If so, congratulations! You have completed the ‘Hello World’ of the GBA.

You will know your lab runs correctly if:

- You see the text ‘Hello World’ printed on the screen

Submission Instructions:

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of previous assignments for instructions on how to perform a "clean" command.**

Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab01_LastnameFirstname, for example:

“Lab01_KirschRussell.zip”

It is your responsibility to ensure that all the appropriate files have been submitted, and that your submitted zip can be opened and everything cleans, builds, and runs as expected.