CSCI 260 – Project 1

Due: Mar 24 11^{59}

The goal of this homework is to write a non-trivial MIPS program, and execute it using a MIPS simulator (MARS). It is an **individual** assignment and no collaboration on coding is allowed (other than general discussions).

1 Assignment

For this assignment, you will draw a shape on a 2D bitmap display. The shape is a capped flask partially filled with a liquid.

Inputs

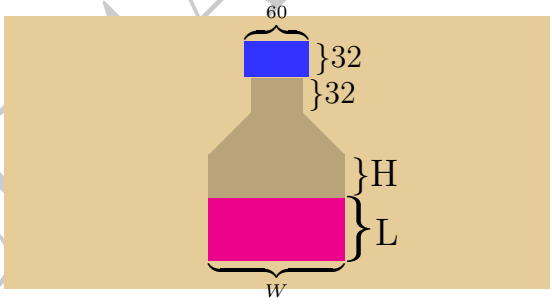
The input will be the assembly language version of the following C structure (details in Section 3):

```
struct projInput {
  unsigned w, h, l;   // size parameters
  unsigned bgcol;     // background color
}
```

You may assume that your implementation of C uses 32-bit ints (so there are 4 words in the above definition).

Output

Your output will be the following figure:



The dimensions/braces shown on the diagram are **not** to be drawn. Additional details of the shape are as follows:

- The flask (including cap) is centered both vertically and horizontally in the bitmap.
- The tapered part of the flask is at a 45 degree angle (*i.e.*,slope of +/- 1).
- The cap overhangs the flask horizontally by 6 pixels on each side.
- As the flask is made of glass, each RGB component of its color will be half the corresponding component of the background color (this is not a particularly good capture of lighting/shading/refraction properties, but you can learn that in a graphics course). For example, if the background color is 0x00406080, the flask will have color 0x00203040.

- The liquid color is constructed by swapping the red and blue components of the background color. For example, if the background color is 0x00ABCDEF, the liquid color would be 0x00EFCDAB.
- The cap is pure blue.

Please note that you should follow the above specs, and the colors given in the previous diagram will not look exactly the same as what you draw due to differences in tools/media.

Task

Write MIPS code to draw the given diagram on the bitmap display supplied as part of MARS. I suggest that you first figure out all relevant dimensions and colors after reading the specification in detail.

Your program should first check the values given in `projInput`. If the supplied input dimensions are not realizable, your program should treat it as an error and draw **only** the background. Some error cases are:

- The body of the flask is not wider than the cap.
- The flask is not *exactly* centerable in the bitmap, even if it is only off by $\frac{1}{2}$ pixel.

There are many other error conditions that you will need to identify.

The file should follow normal MIPS conventions including comments. Your program will be tested on multiple values of the inputs, so you should test your program adequately. You may use any of the instructions and pseudo-instructions we covered in class *except* functions, and multiplication/division (you're probably thinking the wrong way if you think you need these). Note that MARS may treat some seeming instructions as pseudoinstructions (for example something with a 32-bit immediate) and this is allowed; however, you may regret using these when debugging.

2 Submission Instructions

Please rename your file to `surnameName.txt` (*e.g.*,doeJohn.txt) and submit it using blackboard (under the Homework menu). Note that a text file is requested since blackboard does not support `.asm` files. Make sure to follow the formatting rules stated in Section 3. Do **not** include your alias in the file.

Your program will be graded on both correctness (on all test cases) and style (meaningful comments on every line, register conventions, etc.). Although you don't need to have the most efficient implementation, it should be reasonable. See syllabus for late policy.

3 Writing The Program

Accessing the Bitmap Display

In your program, your data segment should start with the following (replace `--` with appropriate values):

```
.data
# DONOTMODIFYTHISLINE
frameBuffer:           .space  0x80000  # 512 wide X 256 high pixels
w:                     .word   --
h:                     .word   --
l:                     .word   --
bgcol:                 .word   --
# DONOTMODIFYTHISLINE
# Your variables go BELOW here only (and above .text)
```

You **must** have the exact names/format given above as your program will not pass our tests otherwise. It also needs to include the two DONOTMODIFY lines exactly as given (1 space after #, no other spaces), and that section should contain only those variables.

The framebuffer is essentially memory-mapped I/O storing a 512-pixel×256-pixel×32-bit image in row-major order. For example, MEM[frameBuffer+4] would contain the pixel at row 0, column 1.

Each **pixel** is a 32-bit value consisting of 8-bits each for the red, green, and blue components in bits 23:16, 15:8, and 7:0 respectively (the upper 8 bits are ignored). For example, the value 0x0000FF00 would correspond to bright green.

Using the MARS Bitmap Display

Please see the other guide posted on bb (Course Materials) for how to use MARS. After reading that, you will need to do a few additional things to use the bitmap display as follows:

1. Select `Tools→Bitmap Display`
2. Click the `Connect to MIPS` button
3. Follow the instructions on the MARS guide to assemble and run your program.

Sample code: The following snippet (at the beginning of the text segment) draws a 10-pixel green line segment somewhere near the top center of the display (assuming you have the data segment from above):

```
.text
drawLine:  la       $t1,frameBuffer
           li       $t3,0x0000FF00   # $t3 ← green
           sw       $t3,15340($t1)
           sw       $t3,15344($t1)
           sw       $t3,15348($t1)
           sw       $t3,15352($t1)
           sw       $t3,15356($t1)
           sw       $t3,15360($t1)
           sw       $t3,15364($t1)
           sw       $t3,15368($t1)
           sw       $t3,15372($t1)
           sw       $t3,15376($t1)
           li       $v0,10           # exit code
           syscall                   # exit to OS
```

4 Clarifications

I will update this section based on student questions.

- In the figure I gave, there is a background line between the bottle and the cap. This line is an unfortunate artifact of the style file I use for my drawing and **not** meant to be drawn – *i.e.,*your drawing should have the bottle touching the cap.
- When dividing a color component by 2, you can either truncate or round. Also, having an odd color component should **not** be considered an error case.