# Preparation

# Story 1: Draw the starting snake

## Requirement

Create a two-segment snake (head + tail) near the middle of the screen. The snake should be a green colour

## Techniques

The simulation provides ‘addressable video memory’, that runs from (word) location 256 (0x100 in hex) for the top-left corner, to 1023 (0x3ff in hex) for the bottom-right.

Exercise

What does the following code produce?

mov r0,#0

str r0,256

str r0,1023

**[Peter: assembler does not accept direct addressing to the screen memory, I think it should]**

[Paste in a partial screenshot showing only the output window]

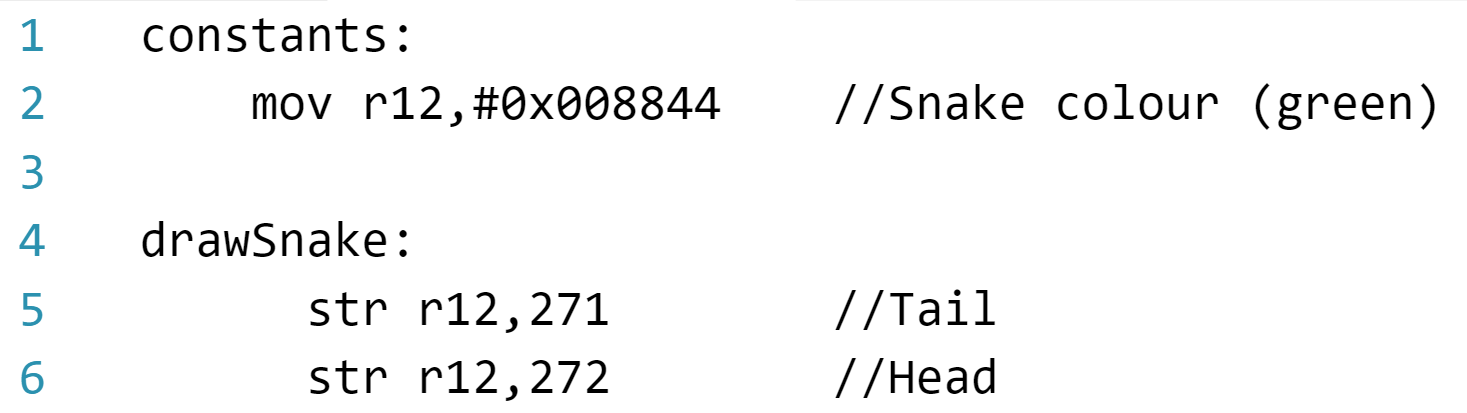
mov r0,#1, means ‘move into register 0, the immediate value 0’. This is known as ‘immediate address mode. In this case, the value 0 represents the colour black (no colour).

When the program starts register 0 should default to the value 0, but it is not safe to assume this, so we set it to 0 explicitly. This is equivalent to the practice of initialising all variables in a higher level language.

We can specify other colours using the same RGB (Red Green Blue) format as used when creating a web page. This is best specified in hex, so for example, 0x008844 results in a suitable hue of green for the snake.

## Implementation

Create and save a new file called Snake, and add the following code:



Load the file into the simulator, assemble and run.

Paste in a partial screenshot showing the assembled code and the Output after running.

Notice that we have added two labels: ‘constants:’ and ‘drawSnake:’. These aren’t actually used by the program at this point, but they make the code more readable. [Paste in a partial screenshot showing the assembled code and the Output after running].

Why have we switched from using r0 to r12 for the green colour? We are simply adopting a common convention to use the lower-number registers for handling variable pieces of data and the higher-number ones for constants.

# Story 2: Move the snake continuously forward

To move the snake one pixel to the right we can draw a new head in the next screen memory location (273) and then reset the tail (271) to the background colour (white), for example: