Test Cases

1. Routine that prints “Hello World” (done)
   1. Getting basic stub working, loading an assembly program into memory and running it, basic understanding of loops and 6502 mnemonics and instructions
2. Routine that makes use of the vic timer to introduce delays in execution (done – extract form other routines)
   1. Understanding the vic timer
3. Routine that clears the screen of the Vic20(done – extract from other routines)
   1. Understanding screen codes and screen memory
4. Routine that draws a custom sprite to screen (done)
   1. Screen codes for custom character graphics (testing the use of custom character graphics and having the vic identify those graphics with built in screen codes)
5. Routine that plays sounds (done)
   1. Testing the speaker and volume registers and their capabilities to gain an understanding as to how they can be used
6. Routine that allows sprite movement with user input (done)
   1. Exploring the input buffer and the use of screen memory locations
7. Routine that, upon collision, decrements health, plays a sound, and displays the scene.
   1. Testing the interaction between rendering of graphics and outputting sound. Attempting to understand how sounds can be played while execution of unrelated code takes place simultaneously
8. Routine that changes background color, border color, and character color (done)
   1. Testing the ability of the Vic to display different colors and how those can be used in our implementation.
9. Routine that tests animation of large custom sprites (done)
   1. Testing the ability of the Vic to animate larger characters (are there visual artifacts when the size and number of animations increases?)
10. Routine that enables and handles interrupts (done)
    1. Explore interrupts triggered by timer 2
11. Implement Multi-Color mode and test it on sprites (done)
    1. Testing the ability of the Vic to render sprites with varying colors within each 8x8 pixel cell/exploring color options available to us

TODO – optimize all working test programs so the code is efficient

Note that we have to design some sort of background data

Ideally two binary maps

1 represents the bottom half of the screen, the floor

1 represents the top half, the scenery