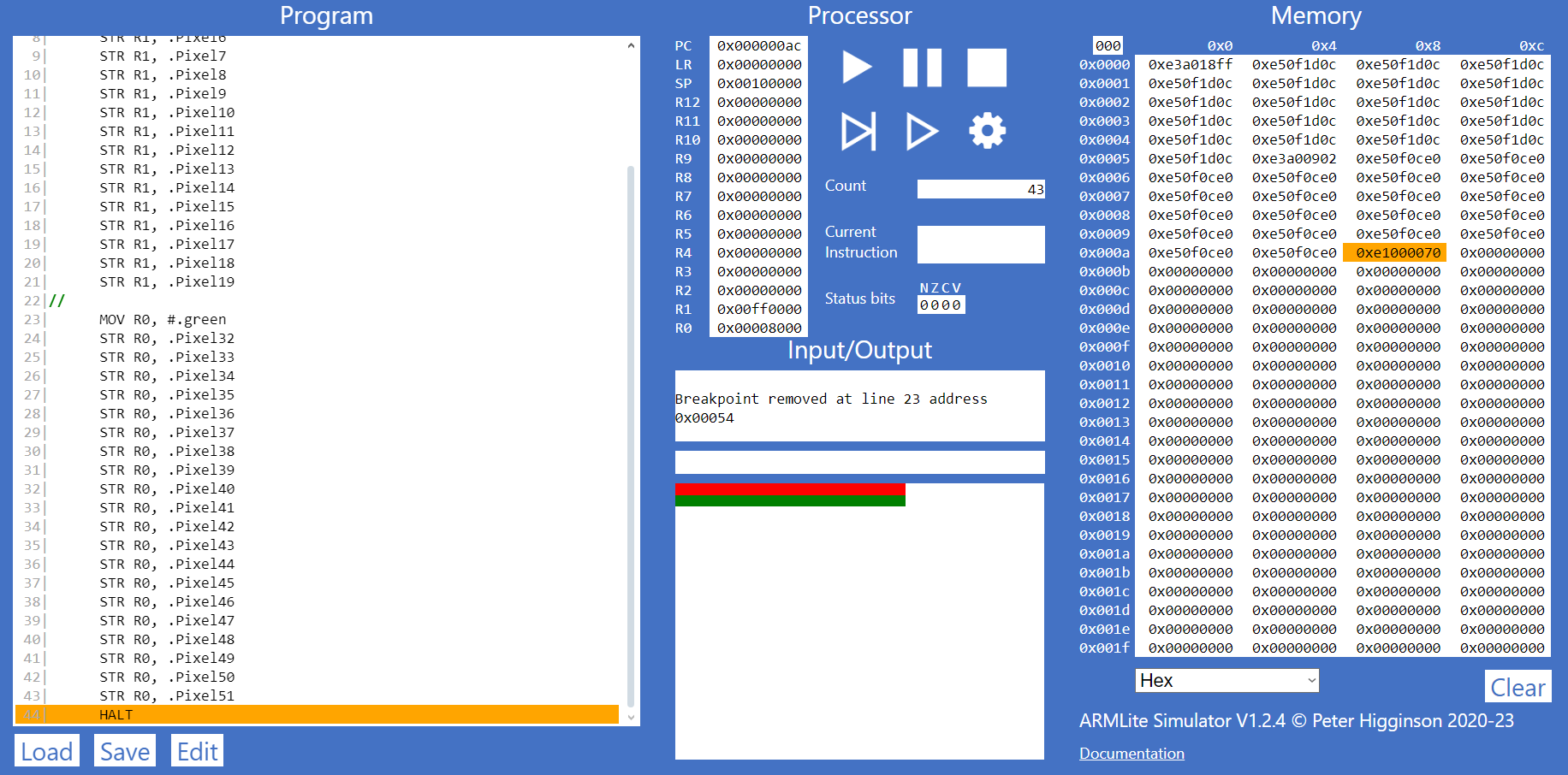
Week 9 Lab - Part 9.1

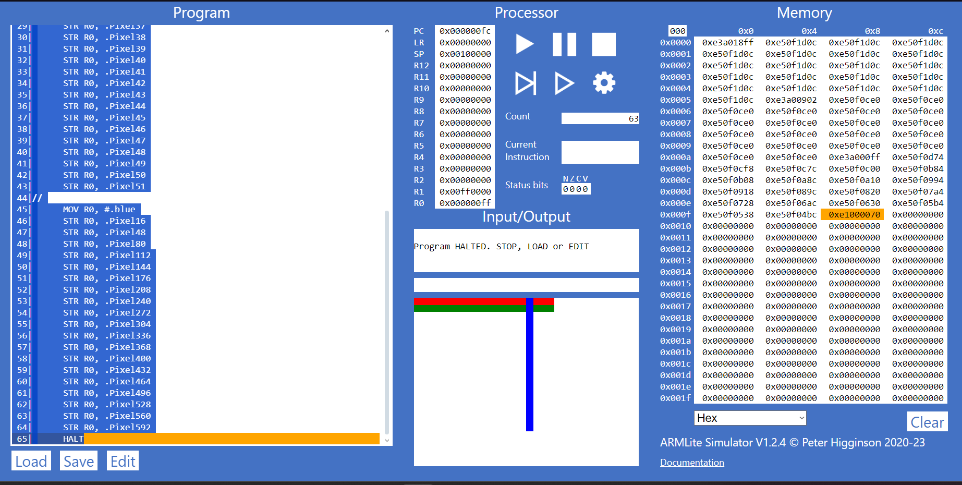
**Exercise 9.1.1**

Consider the following code, which draws a single red horizontal line of length 20 pixels in the top row of the display in Low-res display mode:

MOV R1, #.red  
STR R1, .Pixel0  
STR R1, .Pixel1  
STR R1, .Pixel2  
STR R1, .Pixel3  
STR R1, .Pixel4  
STR R1, .Pixel5  
STR R1, .Pixel6  
STR R1, .Pixel7  
STR R1, .Pixel8  
STR R1, .Pixel9  
STR R1, .Pixel10  
STR R1, .Pixel11  
STR R1, .Pixel12  
STR R1, .Pixel13  
STR R1, .Pixel14  
STR R1, .Pixel15  
STR R1, .Pixel16  
STR R1, .Pixel17  
STR R1, .Pixel18  
STR R1, .Pixel19  
  
HALT

**(a) Write a simple ARMlite assembly program that draws a single line of the same length across the second row (starting from the left-most column) in Low-res display mode.**

**(b) Add to your assembly program code that draws a single line of the same length vertically, down the middle of the display in Low-res display mode**



**Exercise 9.1.3**

**(a) Explain what specifically makes this code an example of indirect addressing ? How is it using indrect addressing to draw each pixel ?**

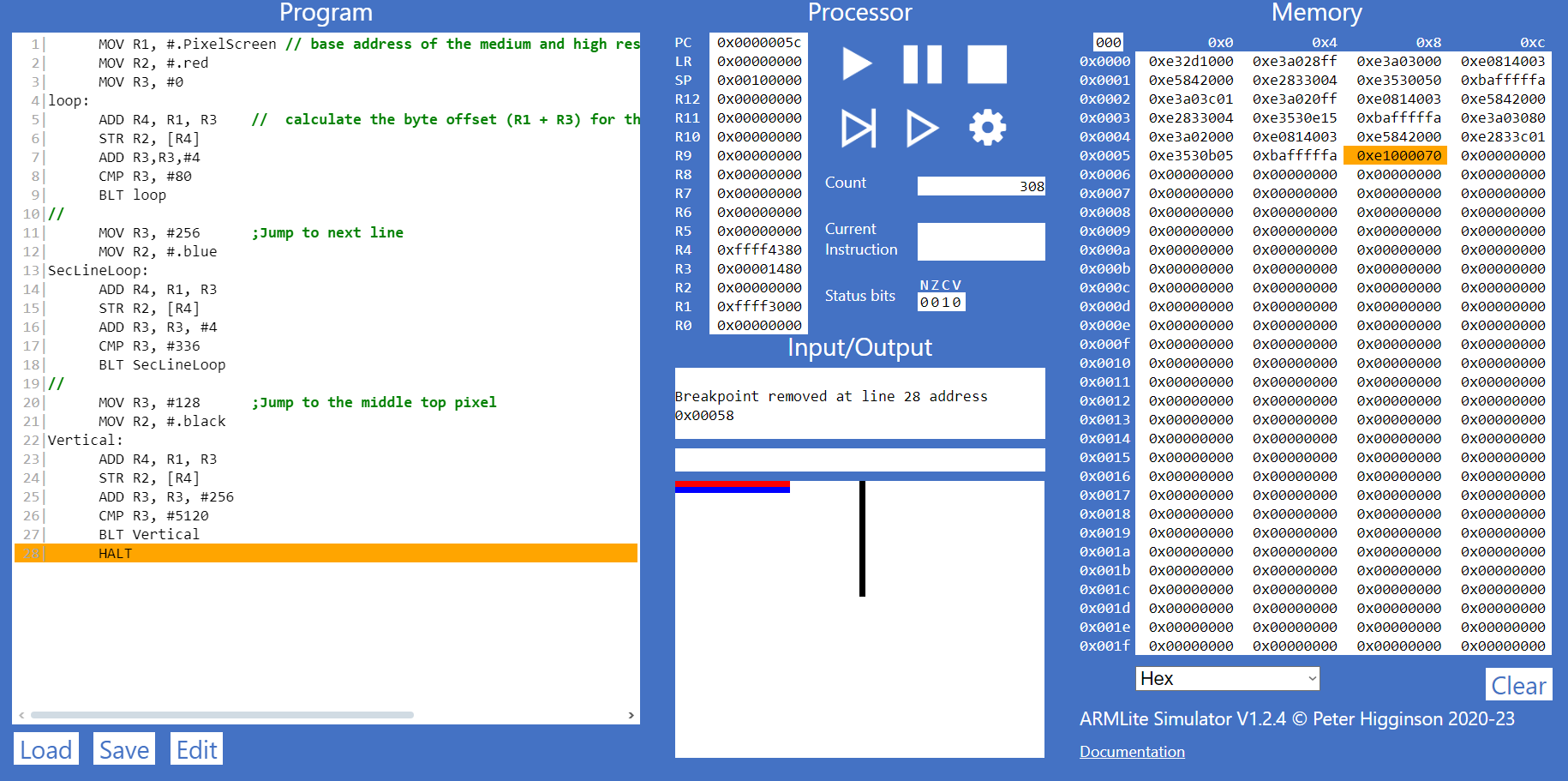
According to ARMlite documentation, the mid-resolution is mapped to memory that starts with the address .PixelScreen, therefore to “draw to the output” or store color to the addressed memory, the address memory must be loaded into a register to be able to calculate offset and also store through that register, making it indirect addressing or in this cases indirect storing.

**(b)  Once you're confident your understand the code, modify the program so that it draws a line of the same length along the second row of the Mid-res display.**

A screenshot of a computer

Description automatically generated

**(c)  Further modify your program so that it also draws a line of the same length vertically down the middle of the display.**



Week 9 Lab - Part 9.2

**Excercise 9.2.1**

Consider again the code below which we looked at in the last part:

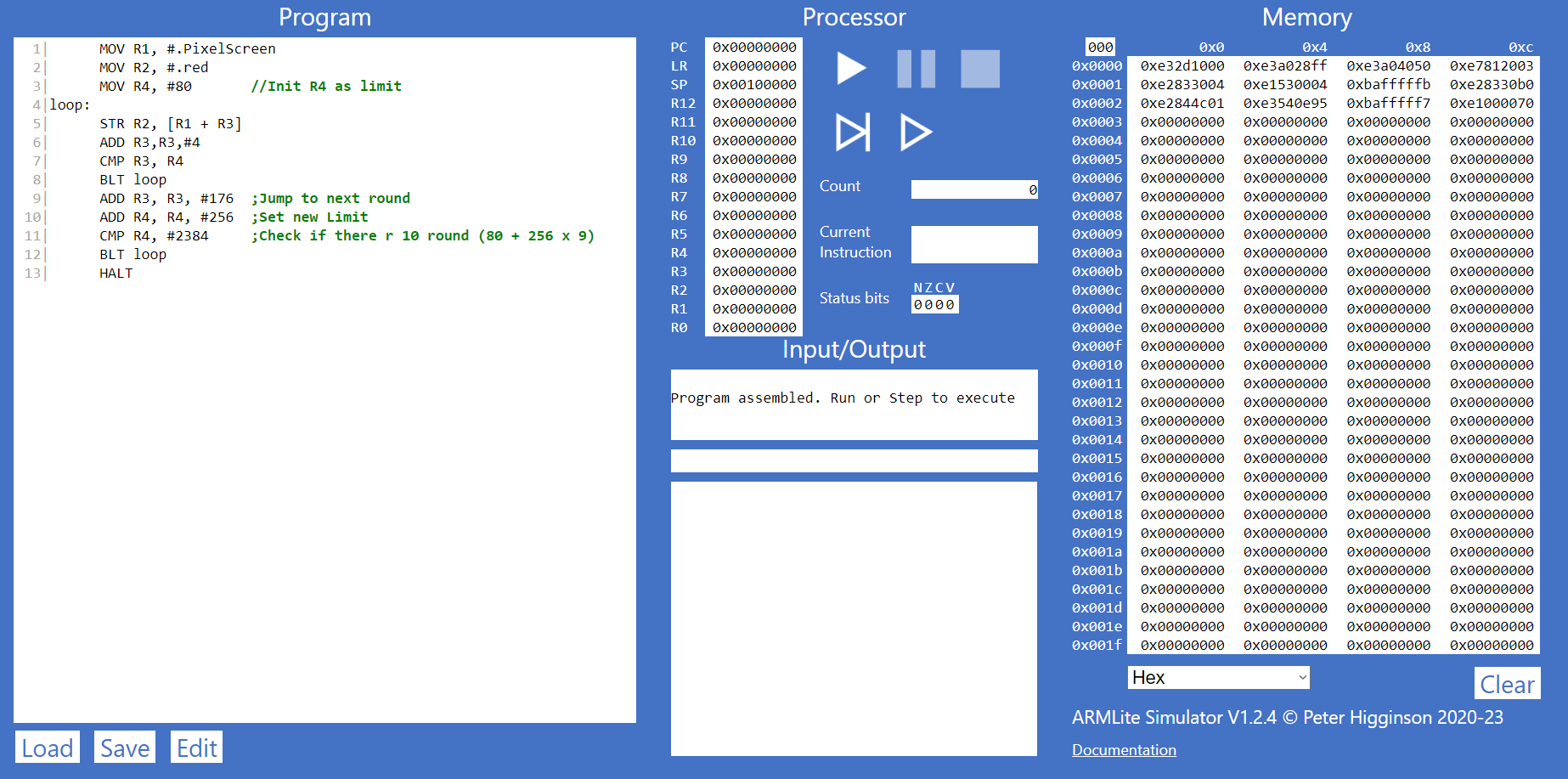
 MOV R1, #.PixelScreen   
 MOV R2, #.red  
 MOV R3, #0  
loop: ADD R4, R1, R3  
 STR R2, [R4]   
 ADD R3,R3,#4   
 CMP R3, #80  
 BLT loop  
HALT

**In ARMlite, rewrite the code above so that is uses indexed addressing to draw the line in Medium-res display mode.**

A screenshot of a computer

Description automatically generated

**Excercise 9.2.2**



Week 9 Lab - Part 9.3

Before you begin, enter the following code into ARMlite:

.ALIGN 256   
arrayLength: 10  
arrayData: 9   
 8  
 7  
 6  
 5  
 4  
 3  
 2  
 1  
 0

**Excercise 9.3.1 (a)**

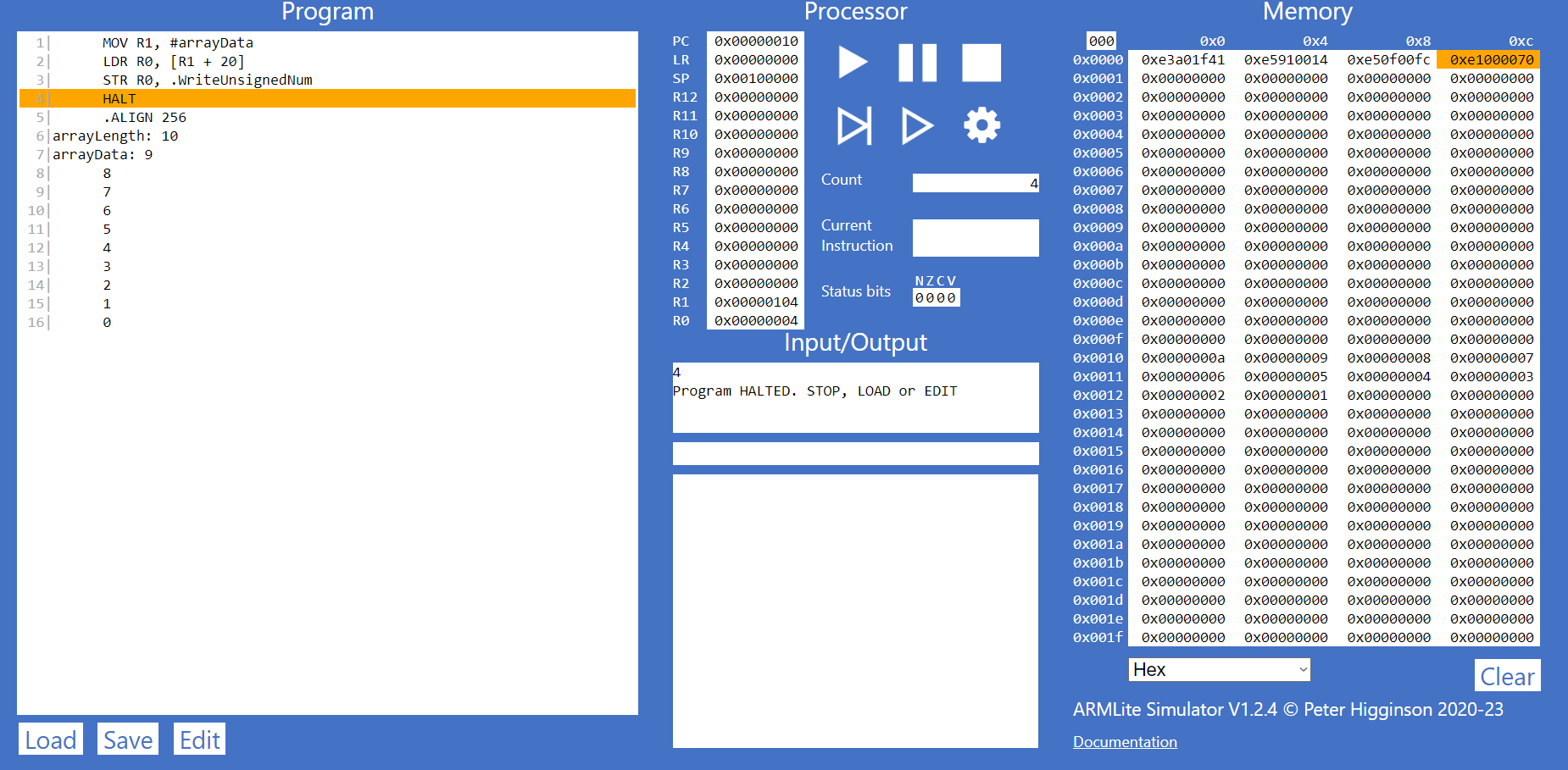
What is the purpose of the .Align 256 instruction ?

* The purpose of the .Align 256 instruction is to ensure that the next instruction with a address divisible by 256, in another words, the next instruction to be in a separate next word.

if you try and execute this program as it is above, it will cause an error on Line 2 (Uknown Instruction).  Why is this ?

* When we try and execute this program as it is above, it will execute the “instruction” on line 2 (stored at 0x00), which is basically not a instruction. It’s a value of 10 stored at 0x00 with a label.

**Excercise 9.3.1 (b)**

Add lines of code to the above to read the 5th value of the array to register R0

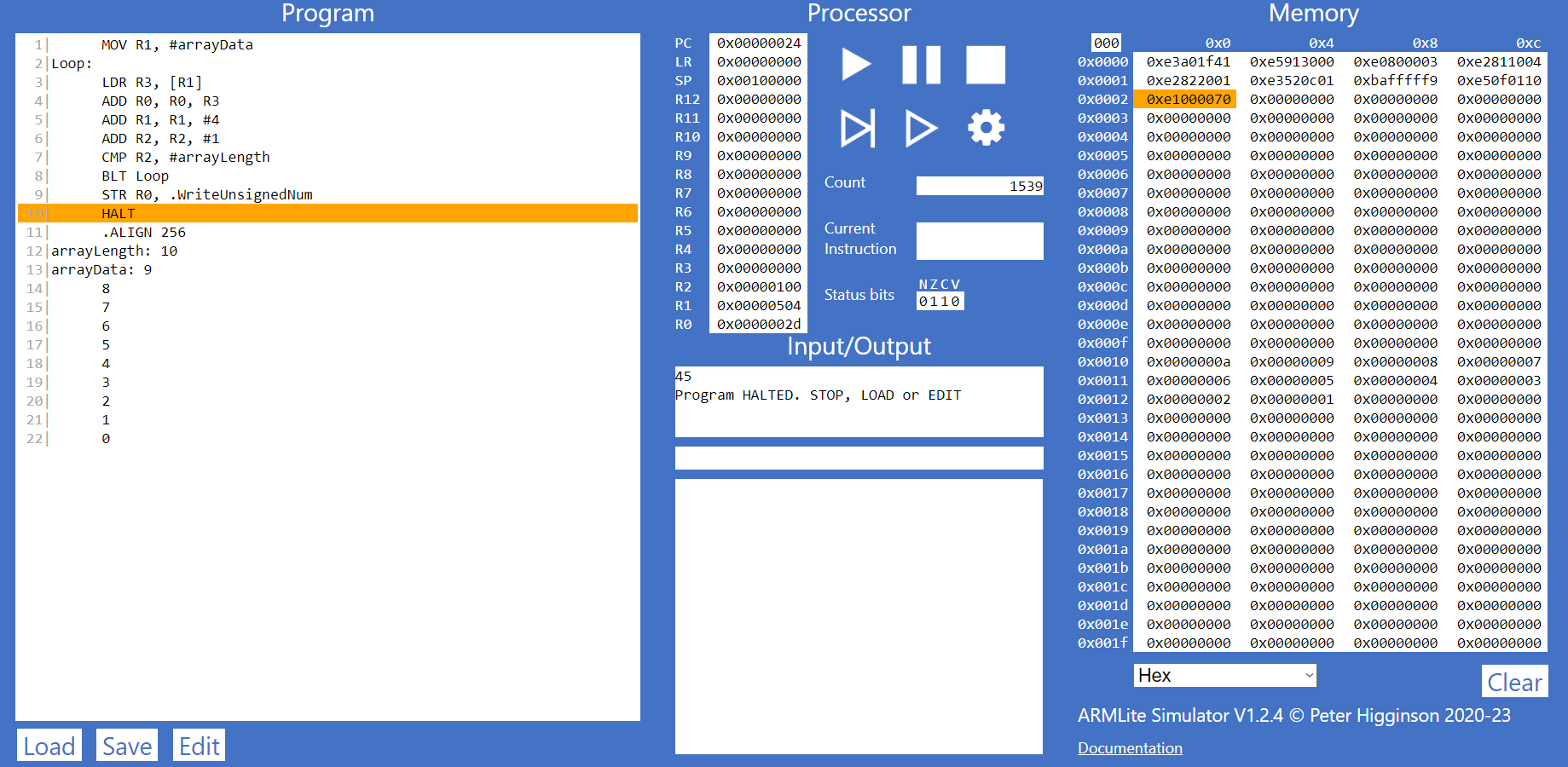
**Excercise 9.3.1 (c)**

A screenshot of a computer

Description automatically generatedNow modify your code so that the index to read from in the array is provided in R1.

**Excercise 9.3.1 (c)**

Now modify  your code so that it adds up all the values in the array.  Your program should use indexed addressing to access each value and write the result to R0.



Week 9 Lab - Part 9.4

**Excercise 9.4.1**

Using the original array defintiion give in Part 9.3, write an ARMlite program copies all the values from this array into another array of equal size (in reverse order).

A blue screen with white text and icons

Description automatically generated

**Excercise 9.4.2**

Using the original array definition, write an ARMlite program that reverses the order of the values in the array (*without using another array*)

