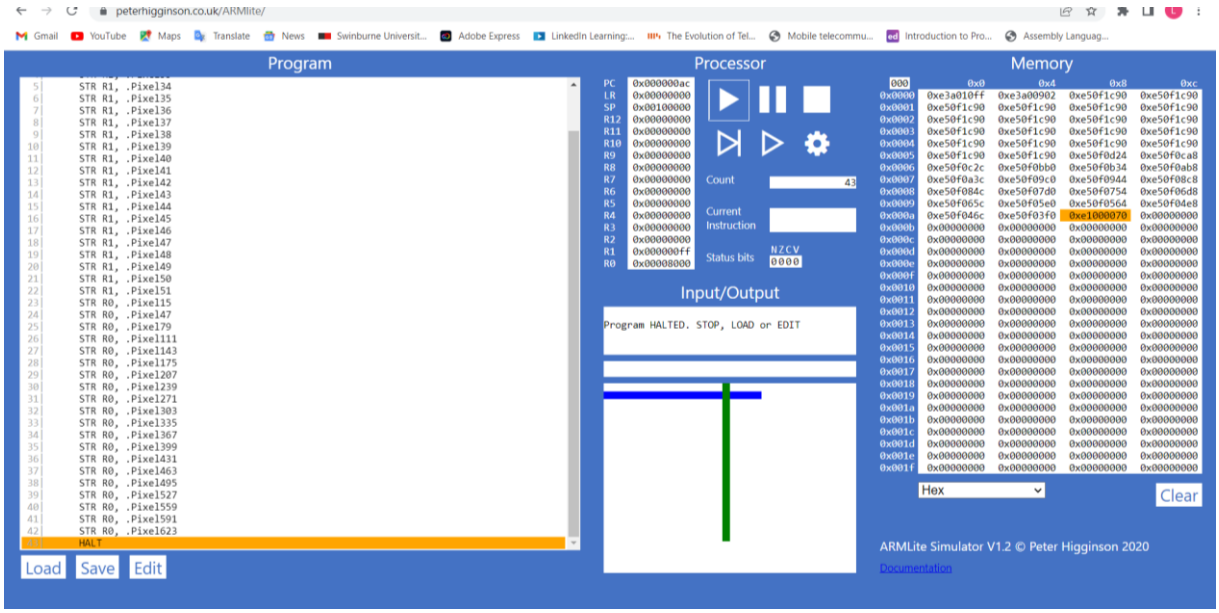
Week 9 Lab - Part 9.1

Indirect and Indexed Addressing

Exercise 9.1.1

(a) Write a simple ARM lite 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

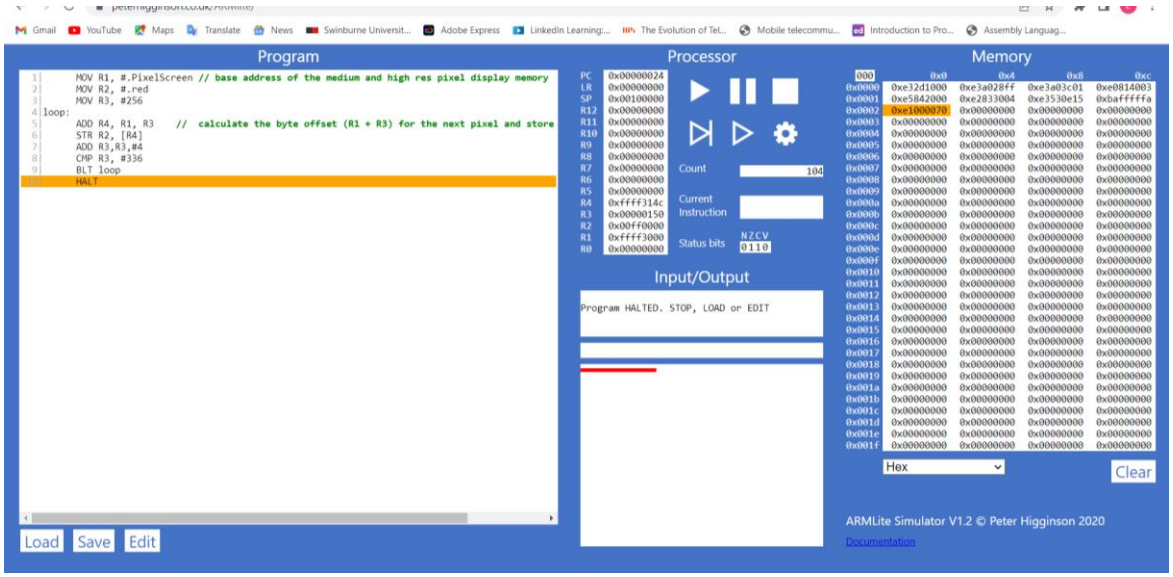


Exercise 9.1.3

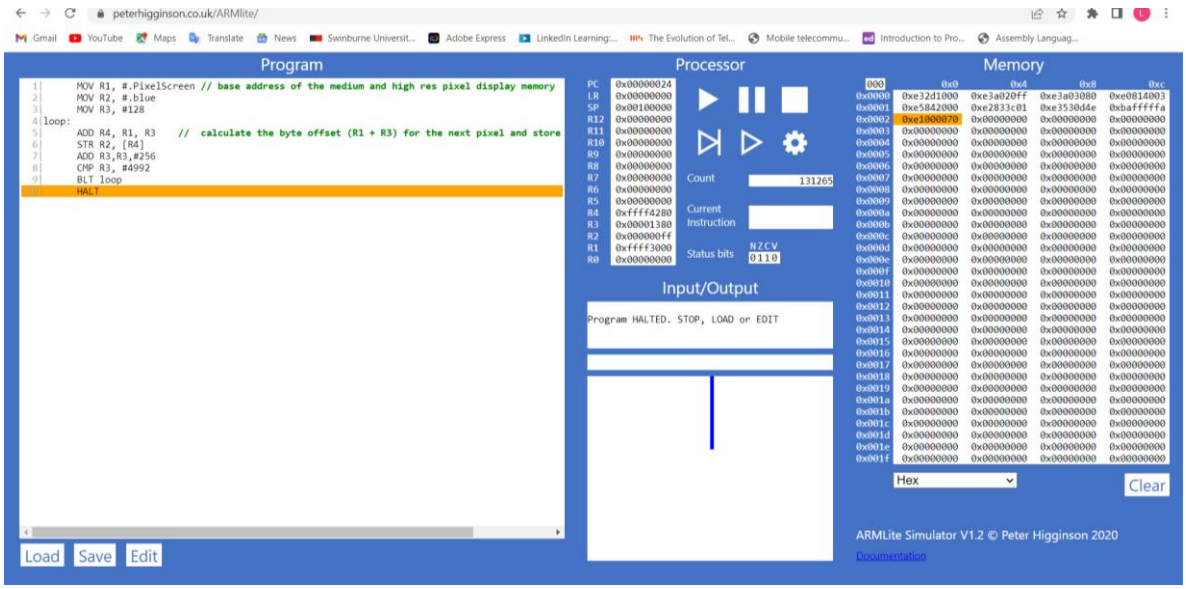
(a) Explain what specifically makes this code an example of indirect addressing? How is it using indirect addressing to draw each pixel?

An indirect Addressing is an absolute address that contains another address. Therefore, `STR R2, [R4]` will store the memory address of R4 into R2.

(b) Once you're confident to 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.



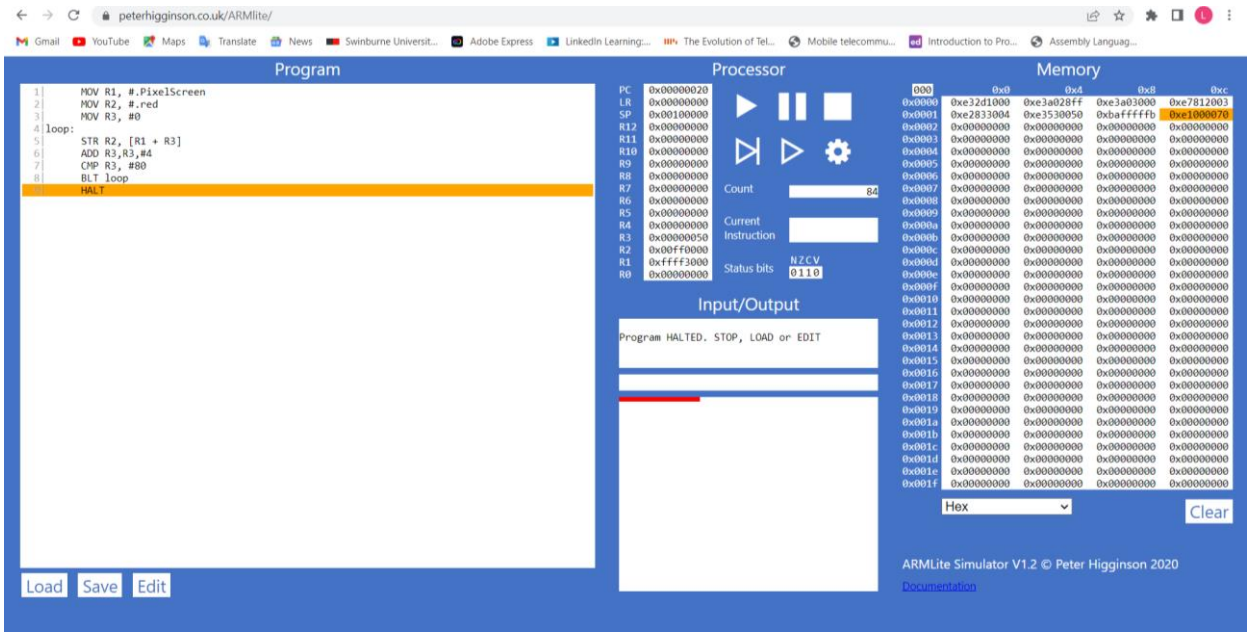
(c) Further modify your program so that it also draws a line of the same length vertically down the middle of the display.



Week 9 Lab - Part 9.2

Indexed Addressing

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



Week 9 Lab - Part 9.3

Arrays

Exercise 9.3.1 (a)

The above code defines an array of 10 32-bit integers. What is the purpose of the `Align 256` instruction?

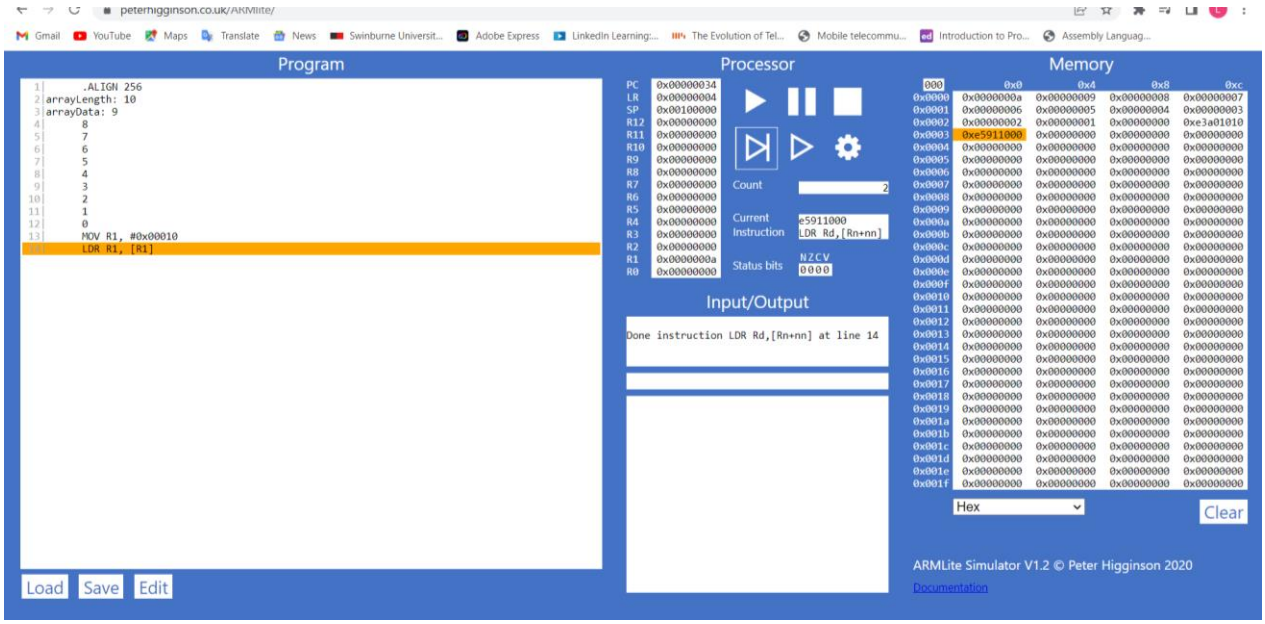
Ensure the next instruction is aligned with a word address divisible by `256`

Exercise 9.3.1 (b)

Add a line of code to the above to read the 5th value of the array to register R0 (i.e.,. it should use indirect addressing to access the 5th cell in the array)

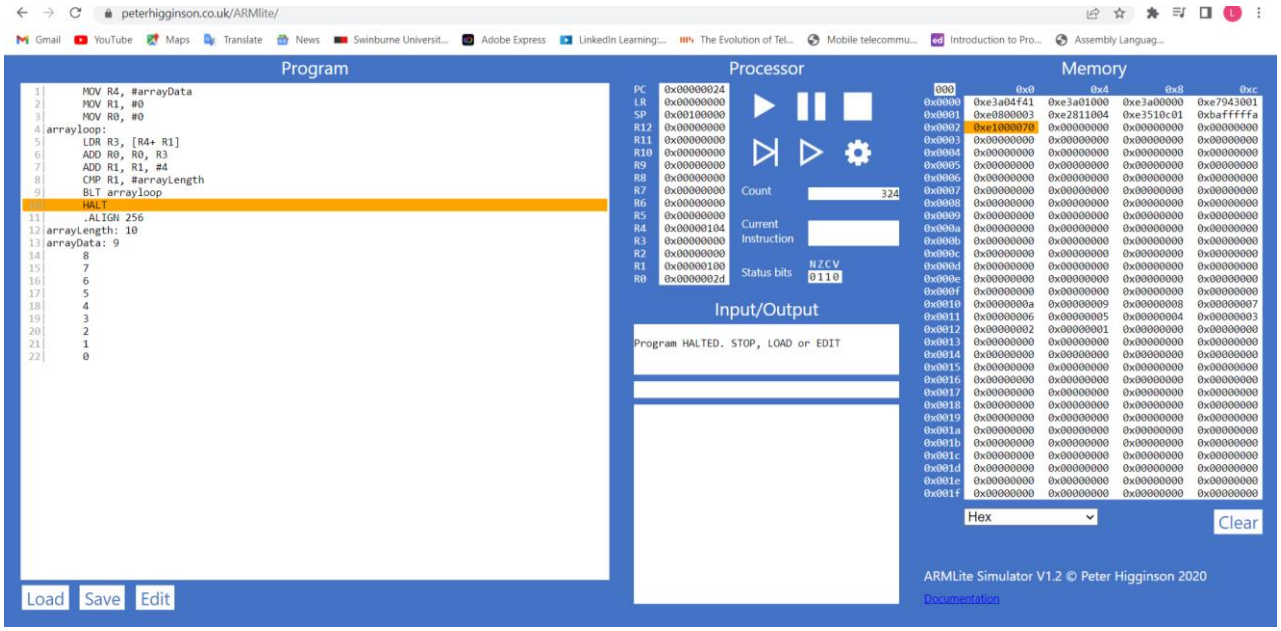
Exercise 9.3.1 (c)

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



Exercise 9.3.3

Using the original array definition, 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.


peterhigginson.co.uk/ARMlite/
Program
MOV R4, #arrayData
MOV R1, #0
MOV R0, #0
arrayloop:
LDR R3, [R4+ R1]
ADD R0, R0, R3
ADD R1, R1, #4
CMP R1, #arrayLength
BLT arrayloop
HALT
.ALIGN 256
arrayLength: 10
arrayData: 9
8
7
6
5
4
3
2
1
0
Load
Save
Edit
Processor
PC 0x00000024
LR 0x00000000
SP 0x00100000
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x00000000
R7 0x00000000
R6 0x00000000
R5 0x00000000
R4 0x00000104
R3 0x00000000
R2 0x00000000
R1 0x00000100
R0 0x0000002d
Count 324
Current Instruction
Status bits NZCV 0110
Input/Output
Program HALTED. STOP, LOAD or EDIT
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe3a04f41 0xe3a01000 0xe3a00000 0xe7943001
0x0001 0xe0800003 0xe2811004 0xe3510c01 0xbafffffa
0x0002 0xe1000070 0x00000000 0x00000000 0x00000000
0x0003 0x00000000 0x00000000 0x00000000 0x00000000
0x0004 0x00000000 0x00000000 0x00000000 0x00000000
0x0005 0x00000000 0x00000000 0x00000000 0x00000000
0x0006 0x00000000 0x00000000 0x00000000 0x00000000
0x0007 0x00000000 0x00000000 0x00000000 0x00000000
0x0008 0x00000000 0x00000000 0x00000000 0x00000000
0x0009 0x00000000 0x00000000 0x00000000 0x00000000
0x000a 0x00000000 0x00000000 0x00000000 0x00000000
0x000b 0x00000000 0x00000000 0x00000000 0x00000000
0x000c 0x00000000 0x00000000 0x00000000 0x00000000
0x000d 0x00000000 0x00000000 0x00000000 0x00000000
0x000e 0x00000000 0x00000000 0x00000000 0x00000000
0x000f 0x00000000 0x00000000 0x00000000 0x00000000
0x0010 0x0000000a 0x00000009 0x00000008 0x00000007
0x0011 0x00000006 0x00000005 0x00000004 0x00000003
0x0012 0x00000002 0x00000001 0x00000000 0x00000000
0x0013 0x00000000 0x00000000 0x00000000 0x00000000
0x0014 0x00000000 0x00000000 0x00000000 0x00000000
0x0015 0x00000000 0x00000000 0x00000000 0x00000000
0x0016 0x00000000 0x00000000 0x00000000 0x00000000
0x0017 0x00000000 0x00000000 0x00000000 0x00000000
0x0018 0x00000000 0x00000000 0x00000000 0x00000000
0x0019 0x00000000 0x00000000 0x00000000 0x00000000
0x001a 0x00000000 0x00000000 0x00000000 0x00000000
0x001b 0x00000000 0x00000000 0x00000000 0x00000000
0x001c 0x00000000 0x00000000 0x00000000 0x00000000
0x001d 0x00000000 0x00000000 0x00000000 0x00000000
0x001e 0x00000000 0x00000000 0x00000000 0x00000000
0x001f 0x00000000 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2 © Peter Higginson 2020
Documentation