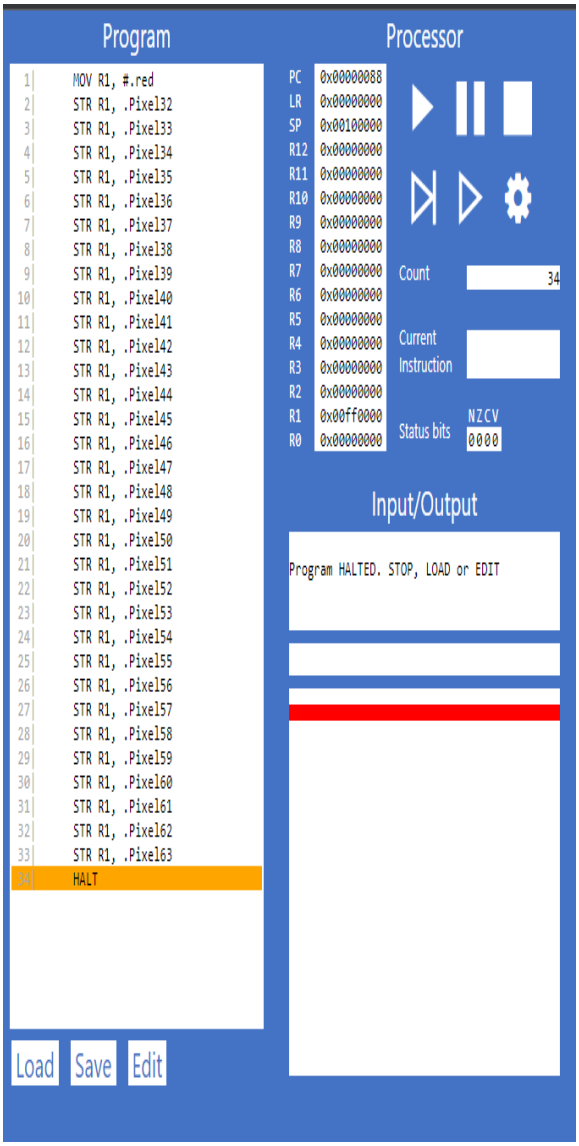
Lab 09 - Graphic & Arrays

9.1.1

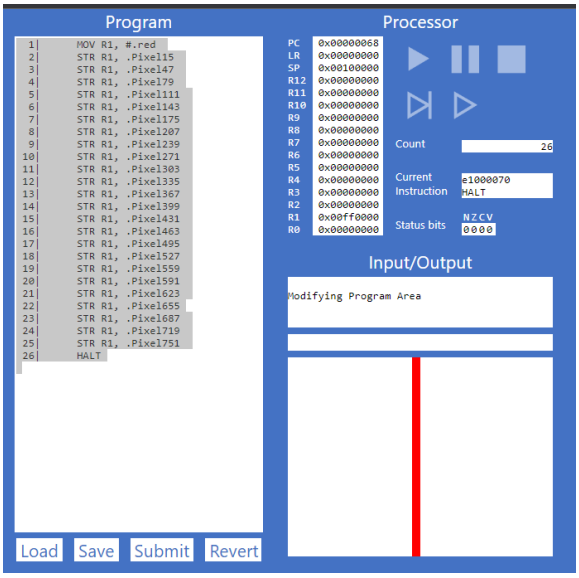
(a)

```
MOV R1, #.red
STR R1, .Pixel32
STR R1, .Pixel33
STR R1, .Pixel34
STR R1, .Pixel35
STR R1, .Pixel36
STR R1, .Pixel37
STR R1, .Pixel38
STR R1, .Pixel39
STR R1, .Pixel40
STR R1, .Pixel41
STR R1, .Pixel42
STR R1, .Pixel43
STR R1, .Pixel44
STR R1, .Pixel45
STR R1, .Pixel46
STR R1, .Pixel47
STR R1, .Pixel48
STR R1, .Pixel49
STR R1, .Pixel50
STR R1, .Pixel51
STR R1, .Pixel52
STR R1, .Pixel53
STR R1, .Pixel54
STR R1, .Pixel55
STR R1, .Pixel56
STR R1, .Pixel57
STR R1, .Pixel58
STR R1, .Pixel59
STR R1, .Pixel60
STR R1, .Pixel61
STR R1, .Pixel62
STR R1, .Pixel63
HALT
```

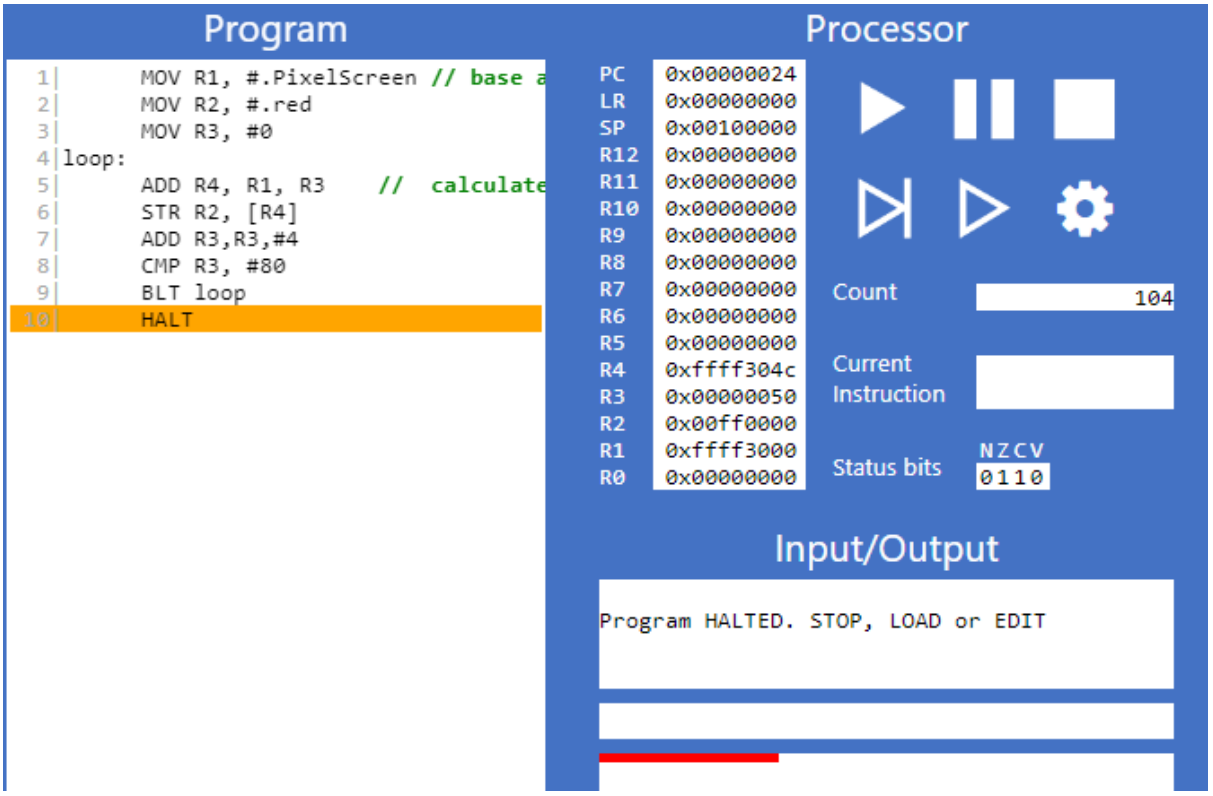


(b)

```
1|     MOV R1, #.red
2|     STR R1, .Pixel15
3|     STR R1, .Pixel47
4|     STR R1, .Pixel79
5|     STR R1, .Pixel111
6|     STR R1, .Pixel143
7|     STR R1, .Pixel175
8|     STR R1, .Pixel207
9|     STR R1, .Pixel239
10|    STR R1, .Pixel271
11|    STR R1, .Pixel303
12|    STR R1, .Pixel335
13|    STR R1, .Pixel367
14|    STR R1, .Pixel399
15|    STR R1, .Pixel431
16|    STR R1, .Pixel463
17|    STR R1, .Pixel495
18|    STR R1, .Pixel527
19|    STR R1, .Pixel559
20|    STR R1, .Pixel591
21|    STR R1, .Pixel623
22|    STR R1, .Pixel655
23|    STR R1, .Pixel687
24|    STR R1, .Pixel719
25|    STR R1, .Pixel751
26|    HALT
```



9.1.2/ 20 pixels line



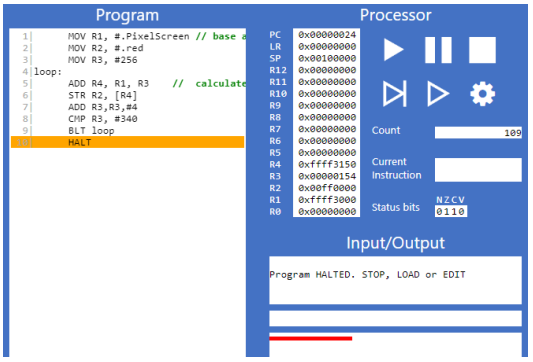
9.1.3/

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

The code above indirectly addresses this because it uses a base address stored in register R1 to access the memory location where pixel values are stored. Instead of directly specifying the memory address for each pixel , the code calculates the address for each pixel via indirect addressing.

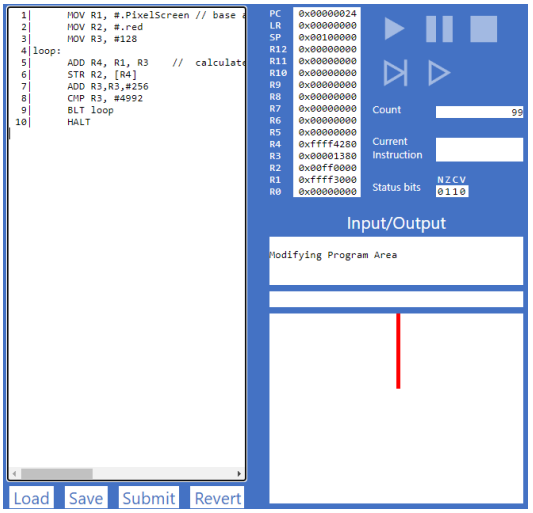
(b) Once you're confident you understand the code , modify the program so that it draws a line of the same length along the second row of the Mid-res display

```
1|      MOV R1, #.PixelScreen
2|      MOV R2, #.red
3|      MOV R3, #256
4|loop:
5|      ADD R4, R1, R3
6|      STR R2, [R4]
7|      ADD R3,R3,#4
8|      CMP R3, #340
9|      BLT loop
10|      HALT
```



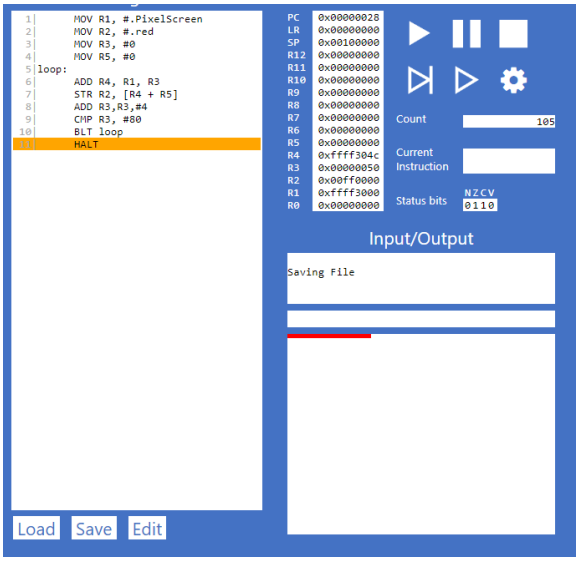
(c) Further modify your program so that it also draws a line of the same length vertically down the middle of the display.

```
1|      MOV R1, #.PixelScreen
2|      MOV R2, #.red
3|      MOV R3, #128
4|loop:
5|      ADD R4, R1, R3
6|      STR R2, [R4]
7|      ADD R3,R3,#256
8|      CMP R3, #4992
9|      BLT loop
10|      HALT
```



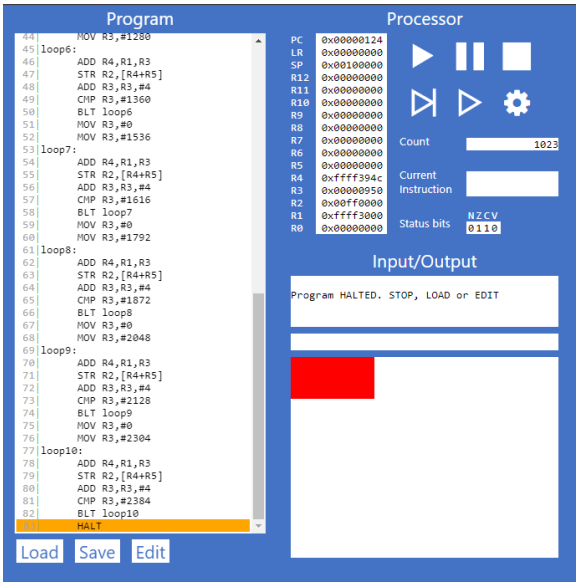
9.2.1/

```
 1|     MOV R1, #.PixelScreen
 2|     MOV R2, #.red
 3|     MOV R3, #0
 4|     MOV R5, #0
 5|loop:
 6|     ADD R4, R1, R3
 7|     STR R2, [R4 + R5]
 8|     ADD R3,R3,#4
 9|     CMP R3, #80
10|     BLT loop
11|     HALT
```



9.2.2/

```
 1|     MOV R1, #.PixelScreen
 2|     MOV R2, #.red
 3|     MOV R3, #0
 4|     MOV R5, #0
 5|loop1:
 6|     ADD R4, R1, R3
 7|     STR R2,[R4+R5]
 8|     ADD R3,R3,#4
 9|     CMP R3, #80
10|     BLT loop1
11|     MOV R3, #0
12|     MOV R3, #256
13|loop2:
14|     ADD R4,R1,R3
15|     STR R2,[R4+R5]
16|     ADD R3,R3,#4
17|     CMP R3, #336
18|     BLT loop2
19|     MOV R3,#0
20|     MOV R3,#512
21|loop3:
22|     ADD R4,R1,R3
23|     STR R2,[R4+R5]
24|     ADD R3,R3,#4
25|     CMP R3, #592
26|     BLT loop3
27|     MOV R3,#0
28|     MOV R3,#768
29|loop4:
30|     ADD R4,R1,R3
31|     STR R2,[R4+R5]
32|     ADD R3,R3,#4
33|     CMP R3,#848
```



```
34|     BLT loop4
35|     MOV R3,#0
36|     MOV R3,#1024
37|loop5:
38|     ADD R4,R1,R3
39|     STR R2, [R4+R5]
40|     ADD R3,R3,#4
41|     CMP R3,#1104
42|     BLT loop5
43|     MOV R3,#0
44|     MOV R3,#1280
45|loop6:
46|     ADD R4,R1,R3
47|     STR R2,[R4+R5]
48|     ADD R3,R3,#4
49|     CMP R3,#1360
50|     BLT loop6
51|     MOV R3,#0
52|     MOV R3,#1536
53|loop7:
54|     ADD R4,R1,R3
55|     STR R2,[R4+R5]
56|     ADD R3,R3,#4
57|     CMP R3,#1616
58|     BLT loop7
59|     MOV R3,#0
60|     MOV R3,#1792
61|loop8:
62|     ADD R4,R1,R3
63|     STR R2,[R4+R5]
64|     ADD R3,R3,#4
65|     CMP R3,#1872
66|     BLT loop8
67|     MOV R3,#0
68|     MOV R3,#2048
69|loop9:
70|     ADD R4,R1,R3
71|     STR R2,[R4+R5]
72|     ADD R3,R3,#4
73|     CMP R3,#2128
74|     BLT loop9
75|     MOV R3,#0
76|     MOV R3,#2304
77|loop10:
78|     ADD R4,R1,R3
79|     STR R2,[R4+R5]
80|     ADD R3,R3,#4
81|     CMP R3,#2384
82|     BLT loop10
83|     HALT
```

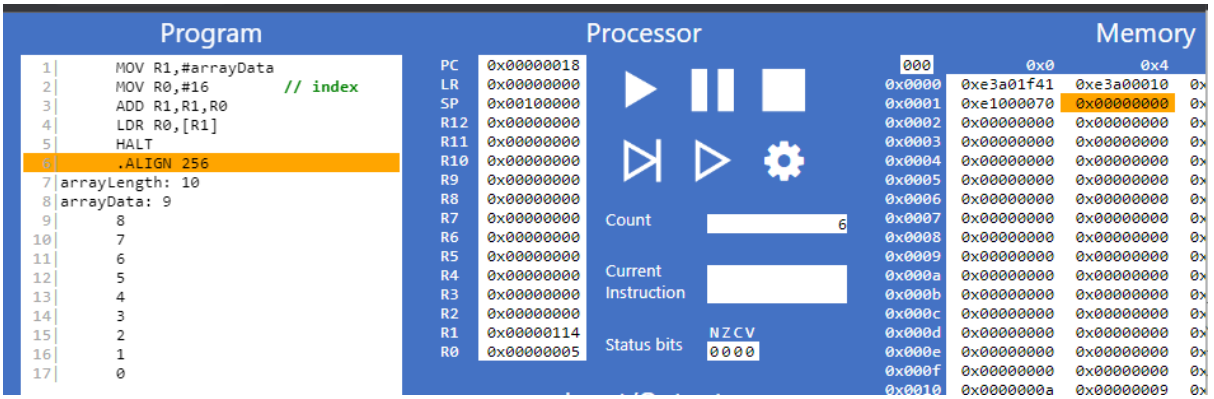
9.3.1/

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

Note: if you try and execute this program as it is above, it will cause an error on Line 2 (Uknown Instruction). Why is this ? (HINT: think about what is in memory at that point of the program)

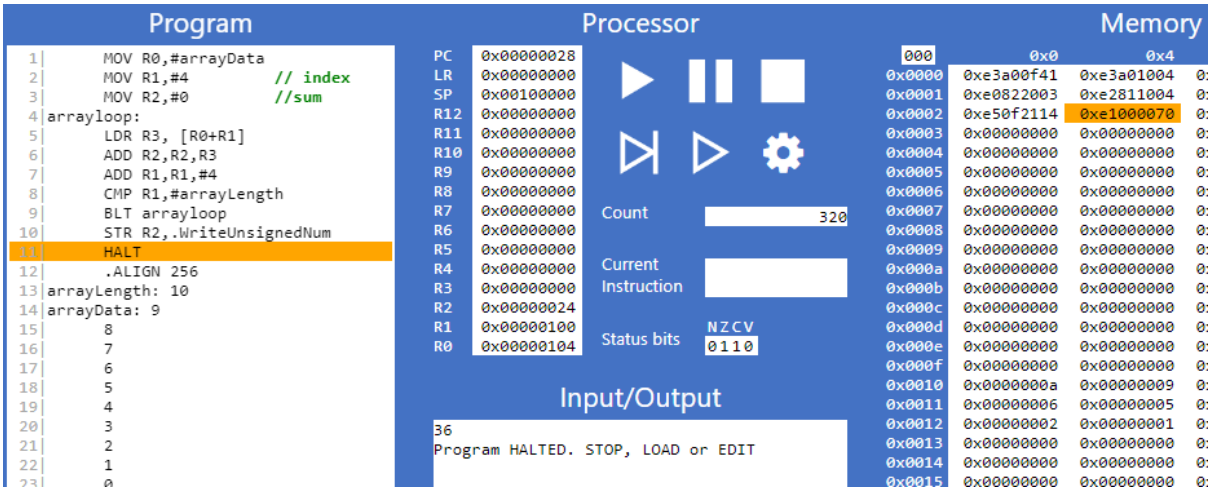
The purpose of Align256 instruction is to aligns the memory location to a multiple of 256 bytes. It ensure that the following data or code is placed at an address that is a multiple of 256. I would assume the problem with line 2 is that the arrayLength is not a multiple of 256 hence causing the error

(b) Add lines 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)




```
1|      MOV R1,#arrayData
2|      MOV R0,#16        // index
3|      ADD R1,R1,R0
4|      LDR R0,[R1]
5|      HALT
6|      .ALIGN 256
7|arrayLength: 10
8|arrayData: 9
9|      8
10|      7
11|      6
12|      5
13|      4
14|      3
15|      2
16|      1
17|      0
```

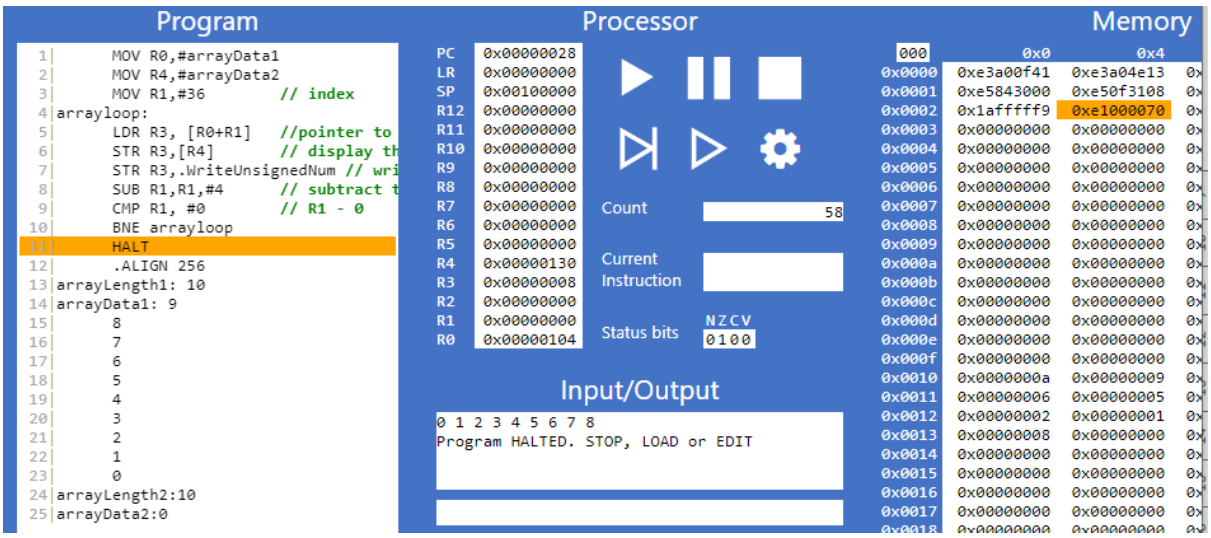
9.3.2/



Code

```
 1|      MOV R0,#arrayData
 2|      MOV R1,#4          // index
 3|      MOV R2,#0          //sum
 4|arrayloop:
 5|      LDR R3, [R0+R1]
 6|      ADD R2,R2,R3
 7|      ADD R1,R1,#4
 8|      CMP R1,#arrayLength
 9|      BLT arrayloop
10|      STR R2,.WriteUnsignedNum
11|      HALT
12|      .ALIGN 256
13|arrayLength: 10
14|arrayData: 9
15|      8
16|      7
17|      6
18|      5
19|      4
20|      3
21|      2
22|      1
23|      0
```

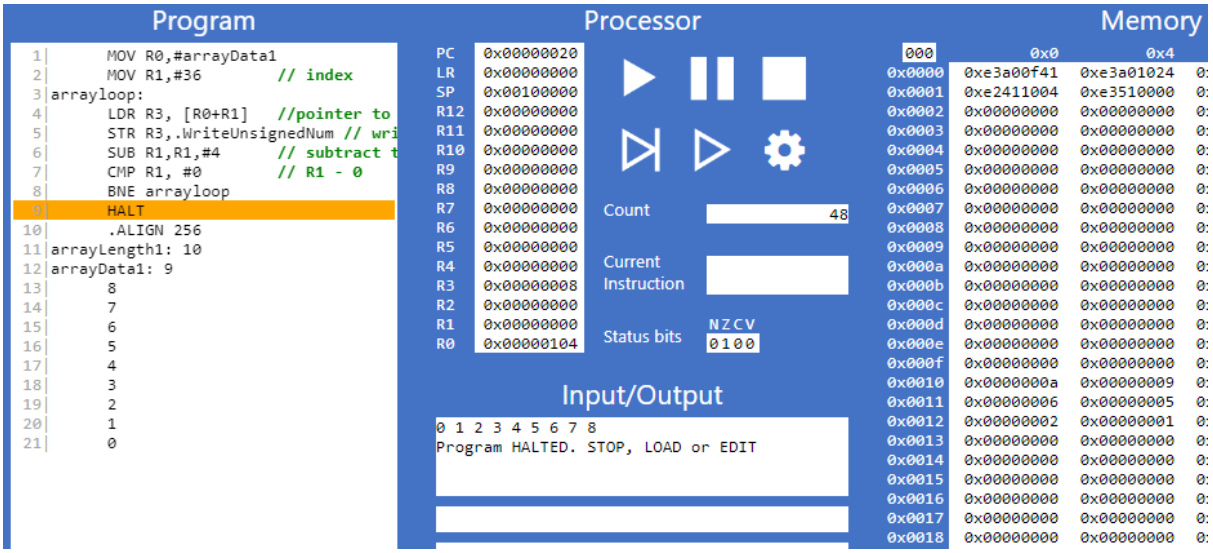
9.4.1/



Code

```
 1|      MOV R0,#arrayData1
 2|      MOV R4,#arrayData2
 3|      MOV R1,#36        // index
 4|arrayloop:
 5|      LDR R3, [R0+R1]   //pointer to the array
 6|      STR R3,[R4]       // display the value inside of the array
 7|      STR R3,.WriteUnsignedNum // write what's inside the array in the
display
 8|      SUB R1,R1,#4      // subtract the index of the array by 4 bytes
 9|      CMP R1, #0        // R1 - 0
10|      BNE arrayloop
11|      HALT
12|      .ALIGN 256
13|arrayLength1: 10
14|arrayData1: 9
15|      8
16|      7
17|      6
18|      5
19|      4
20|      3
21|      2
22|      1
23|      0
24|arrayLength2:10
25|arrayData2:0
```

9.4.2/



Code

```
 1|      MOV R0,#arrayData1
 2|      MOV R1,#36        // index
 3|arrayloop:
 4|      LDR R3, [R0+R1]   //pointer to the array
 5|      STR R3,.WriteUnsignedNum // write what's inside the array in the
display
 6|      SUB R1,R1,#4      // subtract the index of the array by 4 bytes
 7|      CMP R1, #0        // R1 - 0
 8|      BNE arrayloop
 9|      HALT
10|      .ALIGN 256
11|arrayLength1: 10
12|arrayData1: 9
13|      8
14|      7
15|      6
16|      5
17|      4
18|      3
19|      2
20|      1
21|      0
```