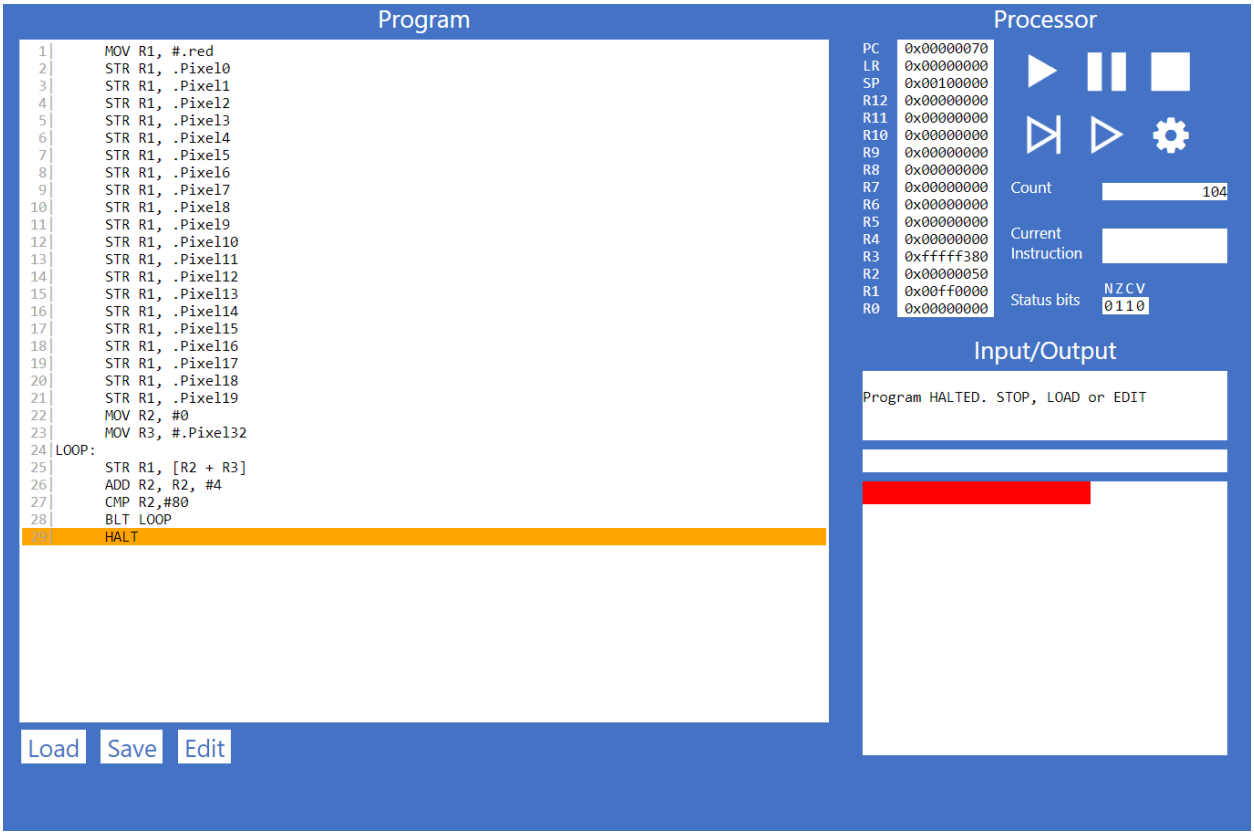
Lab09

Nguyen Nam Tung 103181157

9.1.1.

(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.



1| MOV R1, #.red

2| STR R1, .Pixel0

3| STR R1, .Pixel1

4| STR R1, .Pixel2

5| STR R1, .Pixel3

6| STR R1, .Pixel4

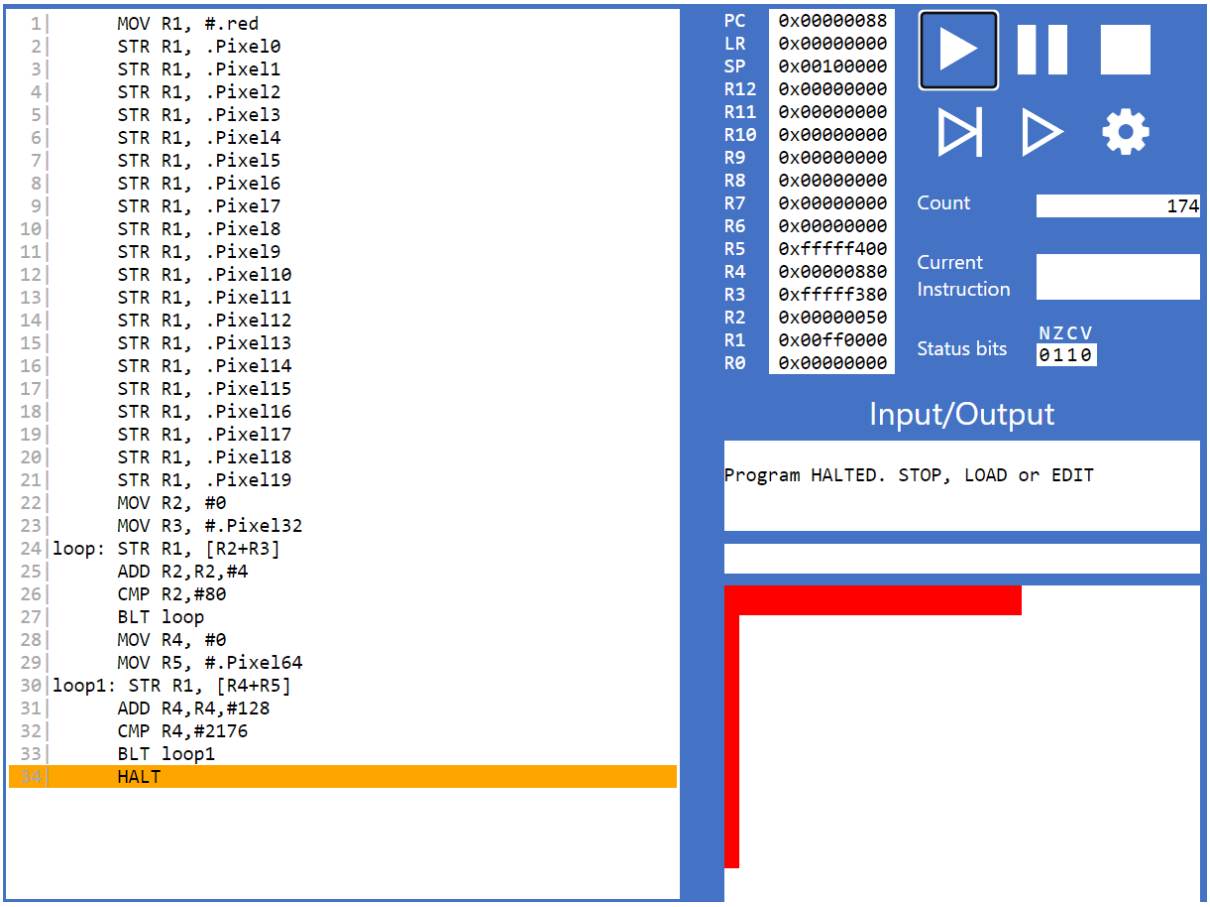
7| STR R1, .Pixel5

8| STR R1, .Pixel6

9| STR R1, .Pixel7

```
10|     STR R1, .Pixel8
11|     STR R1, .Pixel9
12|     STR R1, .Pixel10
13|     STR R1, .Pixel11
14|     STR R1, .Pixel12
15|     STR R1, .Pixel13
16|     STR R1, .Pixel14
17|     STR R1, .Pixel15
18|     STR R1, .Pixel16
19|     STR R1, .Pixel17
20|     STR R1, .Pixel18
21|     STR R1, .Pixel19
22|     MOV R2, #0
23|     MOV R3, #.Pixel32
24|LOOP:
25|     STR R1, [R2 + R3]
26|     ADD R2, R2, #4
27|     CMP R2,#80
28|     BLT LOOP
29|     HALT
```

(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



Program

```
 1|      MOV R1, #.red
 2|      STR R1, .Pixel0
 3|      STR R1, .Pixel1
 4|      STR R1, .Pixel2
 5|      STR R1, .Pixel3
 6|      STR R1, .Pixel4
 7|      STR R1, .Pixel5
 8|      STR R1, .Pixel6
 9|      STR R1, .Pixel7
10|      STR R1, .Pixel8
11|      STR R1, .Pixel9
12|      STR R1, .Pixel10
```

```
13|     STR R1, .Pixel11
14|     STR R1, .Pixel12
15|     STR R1, .Pixel13
16|     STR R1, .Pixel14
17|     STR R1, .Pixel15
18|     STR R1, .Pixel16
19|     STR R1, .Pixel17
20|     STR R1, .Pixel18
21|     STR R1, .Pixel19
22|     MOV R2, #0
23|     MOV R3, #.Pixel32
24|LOOP:
25|     STR R1, [R2 + R3]
26|     ADD R2, R2, #4
27|     CMP R2,#80
28|     BLT LOOP
29|     MOV R4, #0
30|     MOV R5, #.Pixel64
31|LOOP2: STR R1, [R4 + R5]
32|     ADD R4, R4, #128
33|     CMP R4, #2176
34|     BLT LOOP2
35|     HALT
```

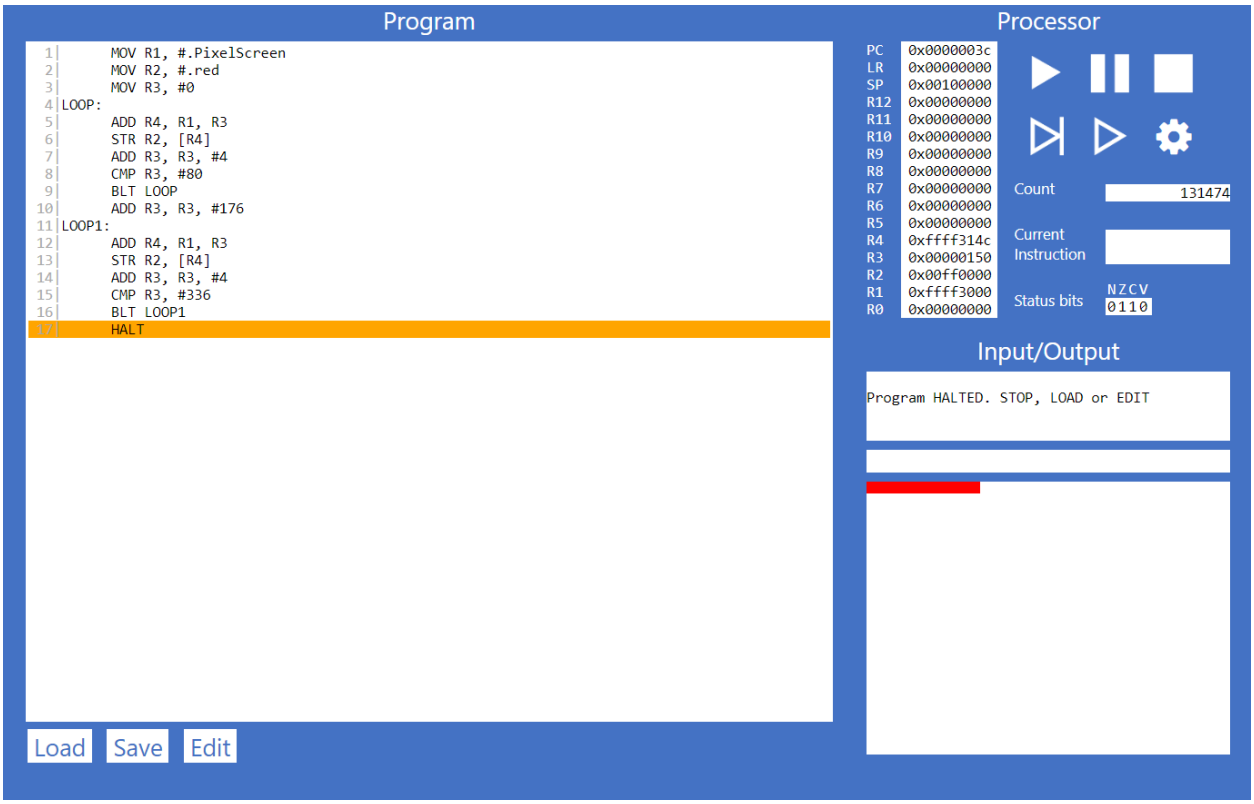
9.1.3

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

Line STR R2 [R4]

This will store the content of the memory of R4 into the memory of R2. It will use indirect addressing to draw each pixel.

(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, #0
 4|LOOP:
 5|     ADD R4, R1, R3
 6|     STR R2, [R4]
 7|     ADD R3, R3, #4
 8|     CMP R3, #80
 9|     BLT LOOP
10|      ADD R3, R3, #176
11|LOOP1:
```

12| ADD R4, R1, R3

13| STR R2, [R4]

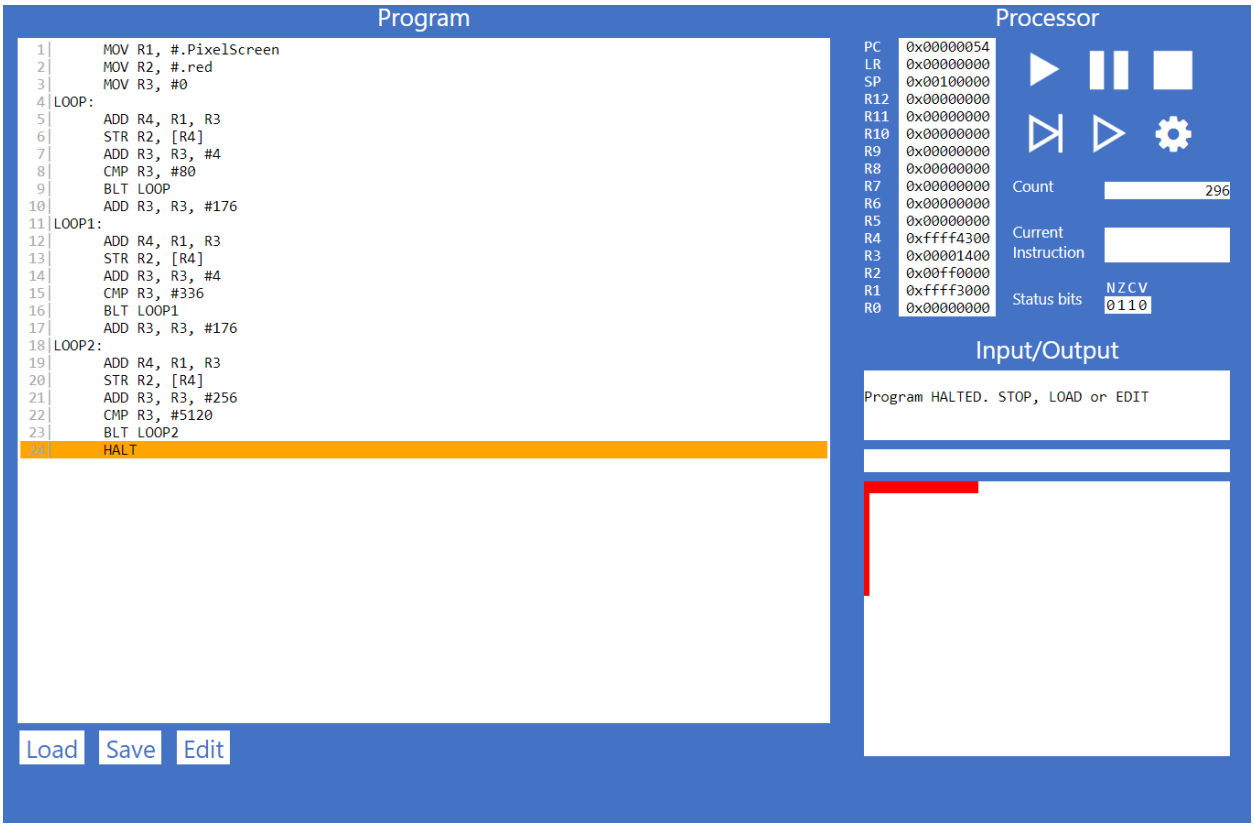
14| ADD R3, R3, #4

15| CMP R3, #336

16| BLT LOOP1

17| 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, #0

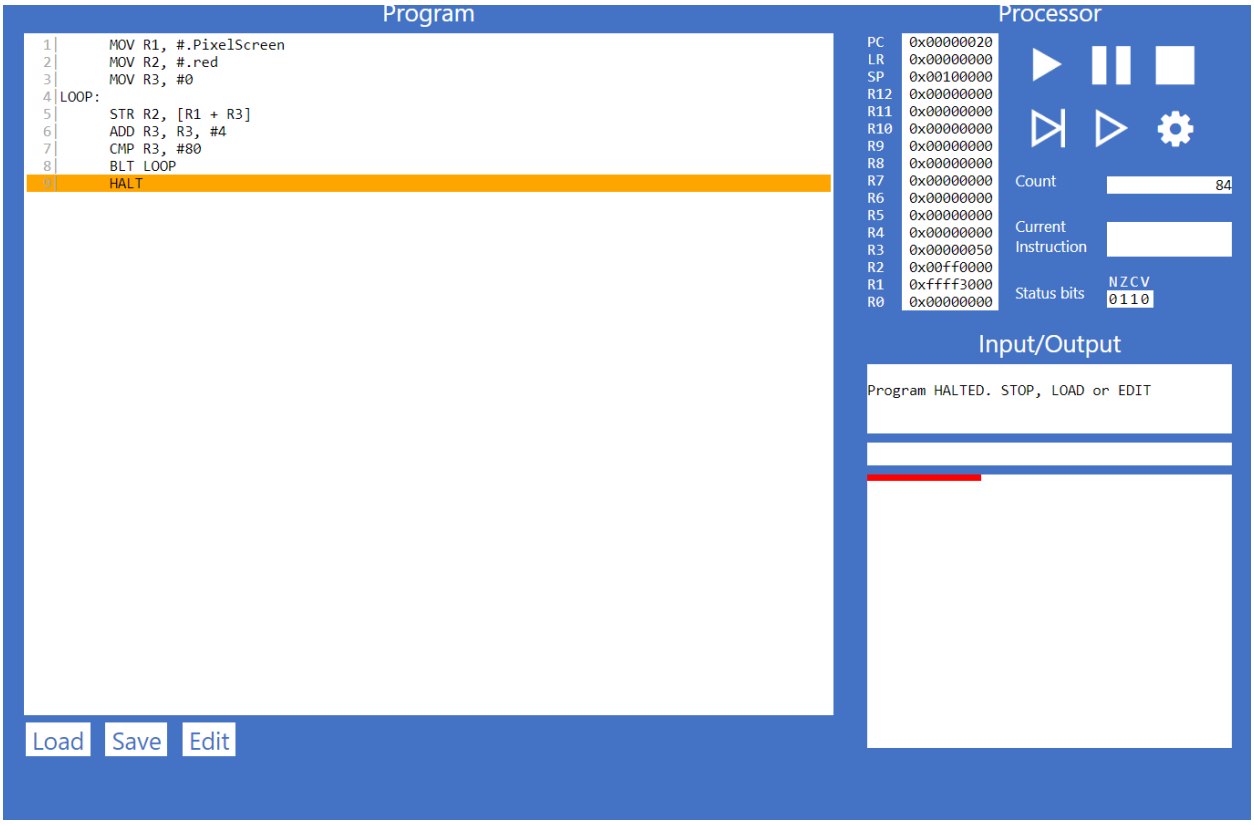
4|LOOP:

5| ADD R4, R1, R3

6| STR R2, [R4]

```
 7|     ADD R3, R3, #4
 8|     CMP R3, #80
 9|     BLT LOOP
10|     ADD R3, R3, #176
11|LOOP1:
12|     ADD R4, R1, R3
13|     STR R2, [R4]
14|     ADD R3, R3, #4
15|     CMP R3, #336
16|     BLT LOOP1
17|     ADD R3, R3, #176
18|LOOP2:
19|     ADD R4, R1, R3
20|     STR R2, [R4]
21|     ADD R3, R3, #256
22|     CMP R3, #5120
23|     BLT LOOP2
24|     HALT
```

9.2.1.



1| MOV R1, #.PixelScreen

2| MOV R2, #.red

3| MOV R3, #0

4|LOOP:

5| STR R2, [R1 + R3]

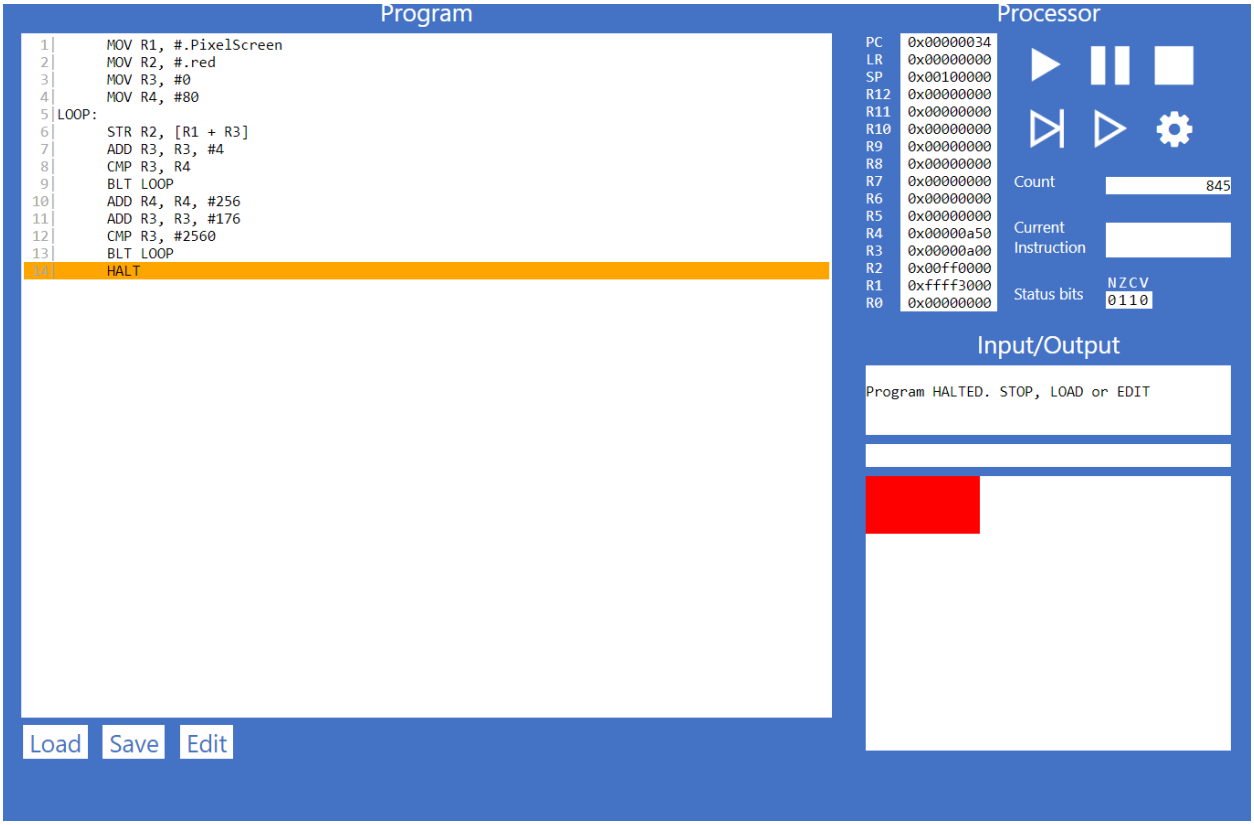
6| ADD R3, R3, #4

7| CMP R3, #80

8| BLT LOOP

9| HALT

9.2.2.



1| MOV R1, #.PixelScreen

2| MOV R2, #.red

3| MOV R3, #0

4| MOV R4, #80

5|LOOP:

6| STR R2, [R1 + R3]

7| ADD R3, R3, #4

8| CMP R3, R4

9| BLT LOOP

10| ADD R4, R4, #256

11| ADD R3, R3, #176

12| CMP R3, #2560

13| BLT LOOP

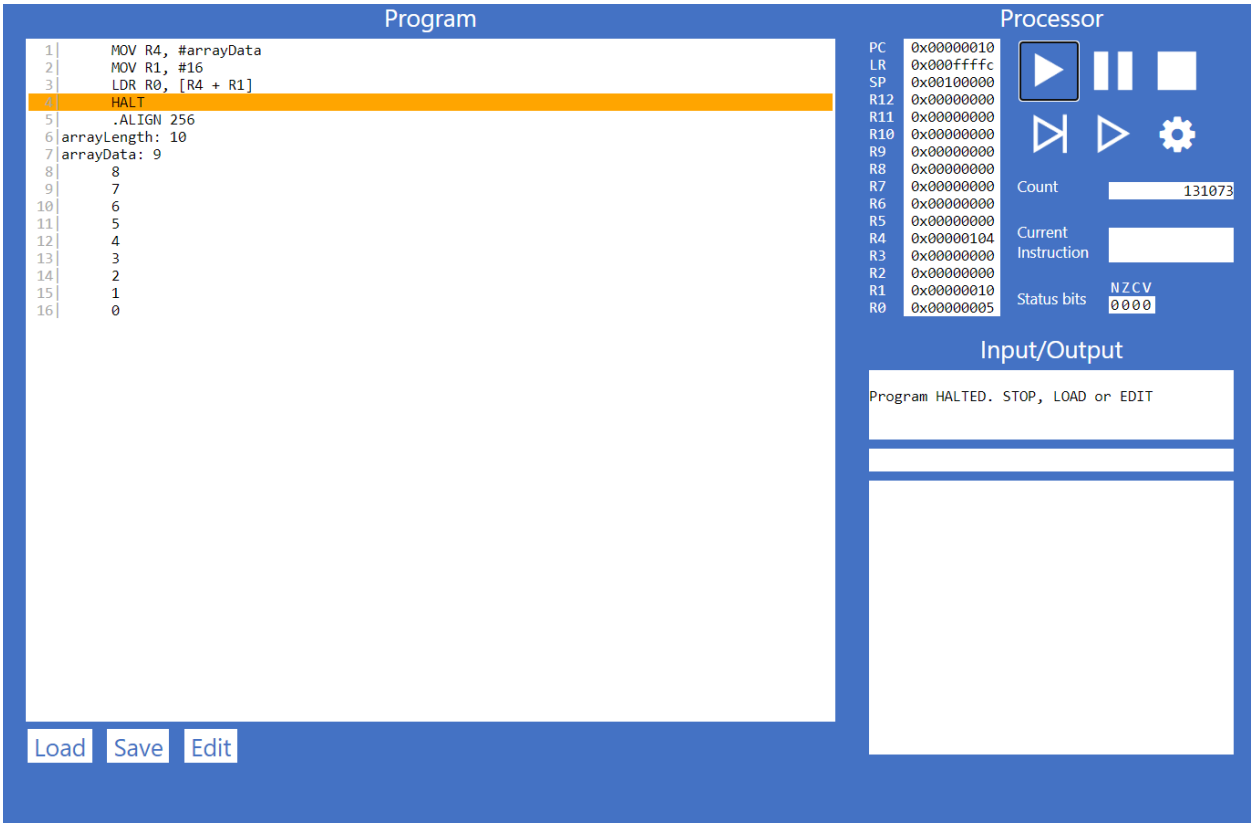
14| HALT

9.3.1.

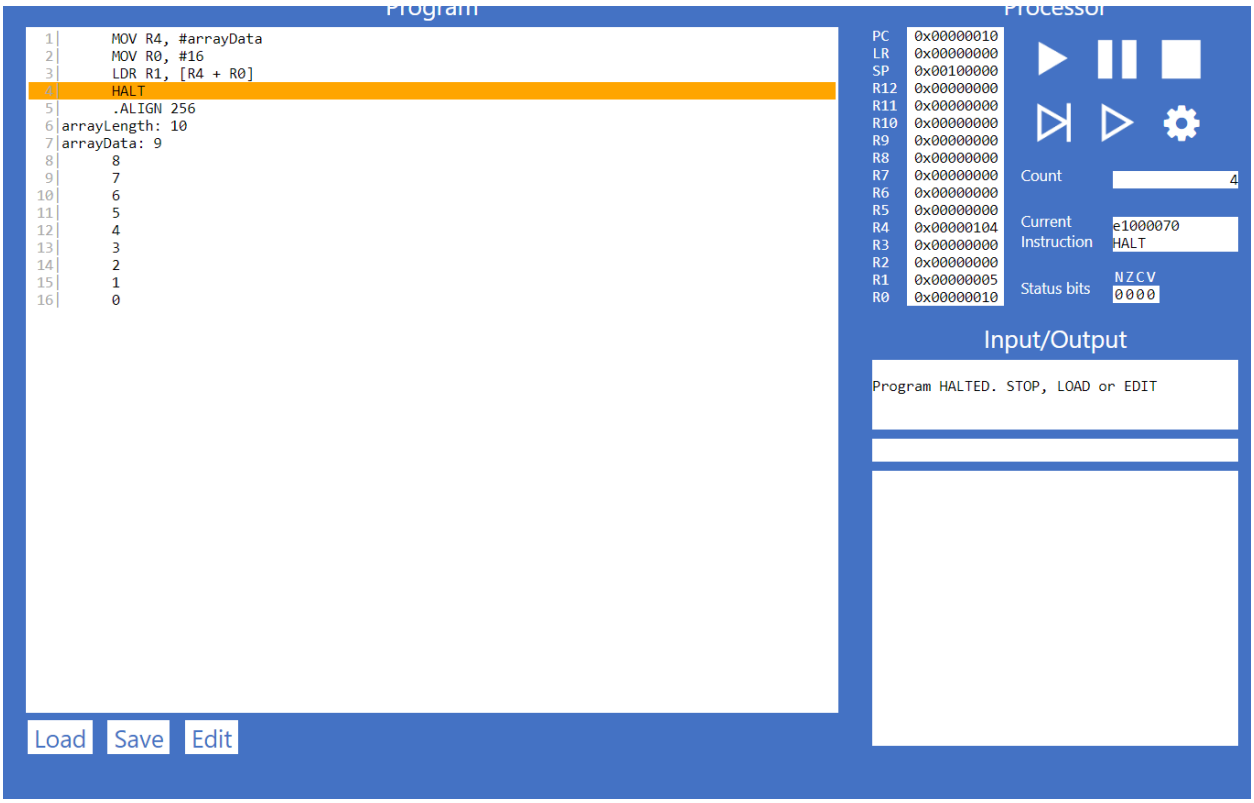
a)

The purpose of .ALIGN 256 will align the data to the next byte address that is divisible by 256

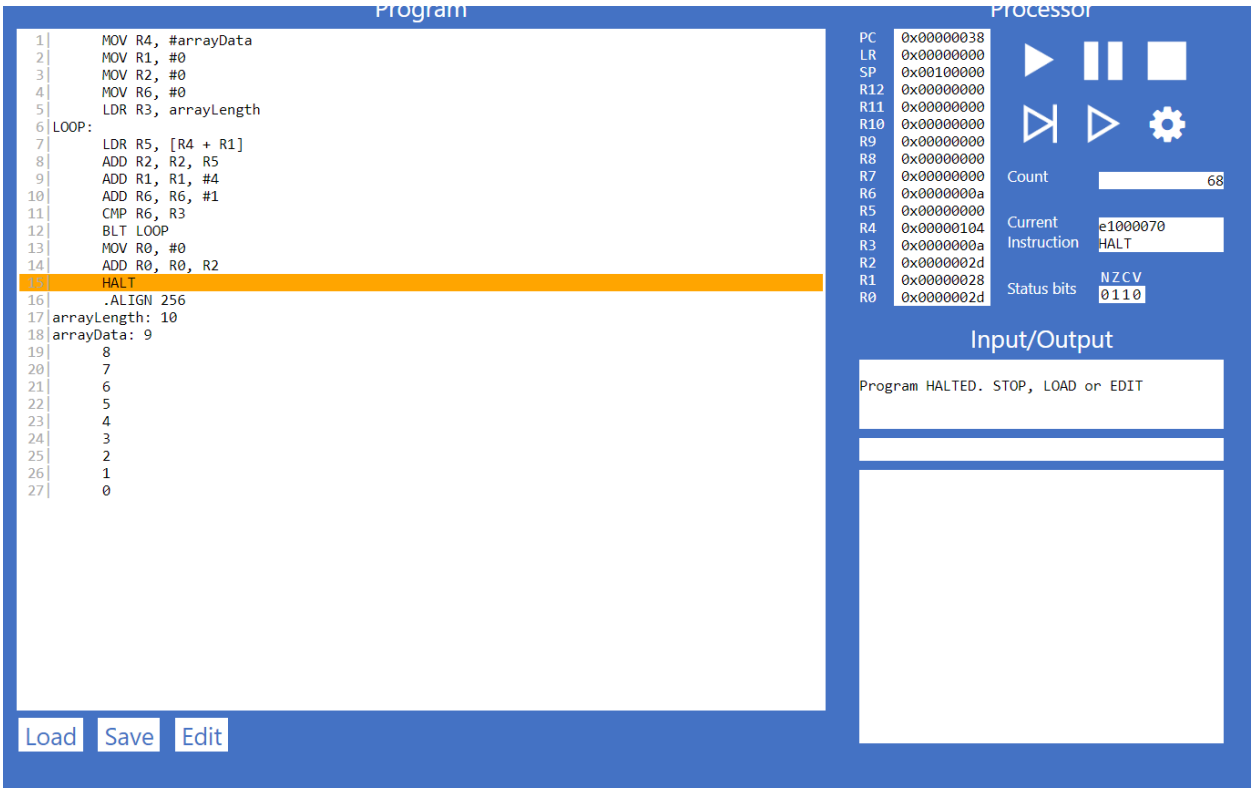
b)



C)



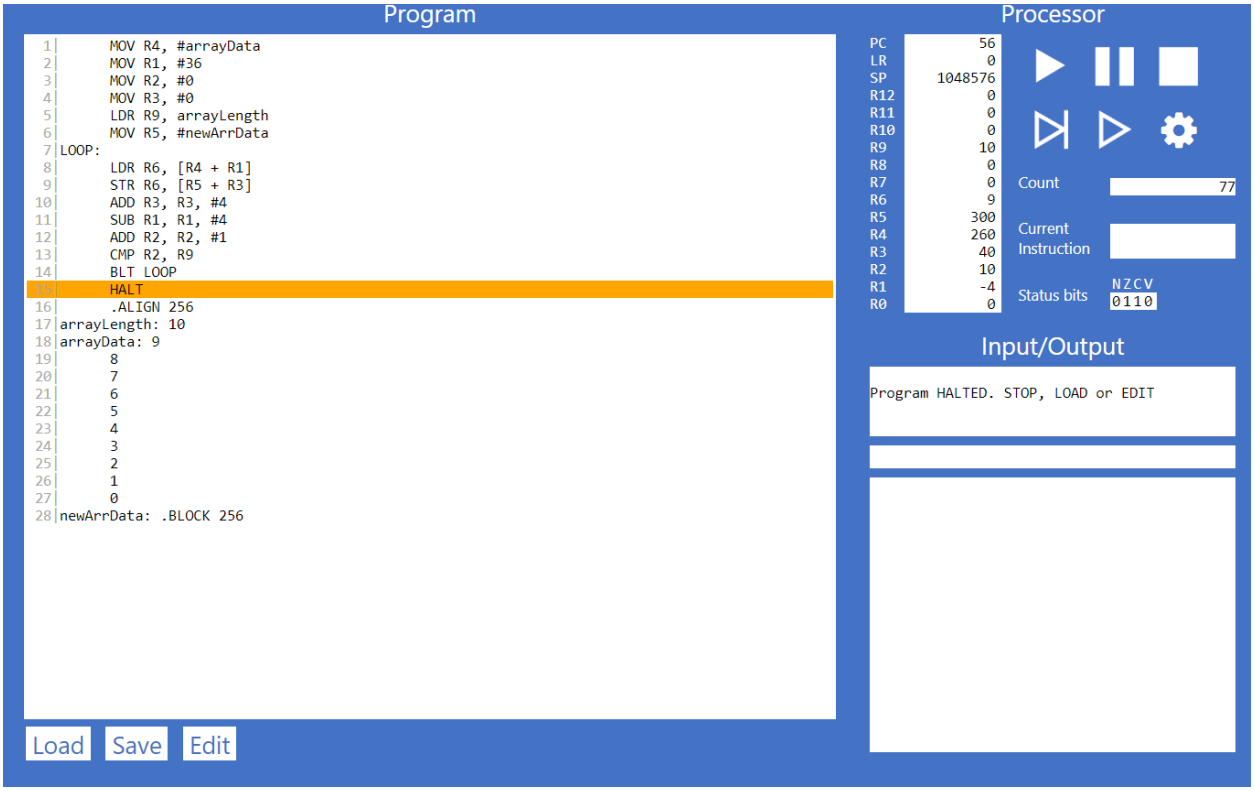
9.3.3.



1| MOV R4, #arrayData

```
 2|     MOV R1, #0
 3|     MOV R2, #0
 4|     MOV R6, #0
 5|     LDR R3, arrayLength
 6|LOOP:
 7|     LDR R5, [R4 + R1]
 8|     ADD R2, R2, R5
 9|     ADD R1, R1, #4
10|     ADD R6, R6, #1
11|     CMP R6, R3
12|     BLT LOOP
13|     MOV R0, #0
14|     ADD R0, R0, R2
15|     HALT
16|     .ALIGN 256
17|arrayLength: 10
18|arrayData: 9
19|     8
20|     7
21|     6
22|     5
23|     4
24|     3
25|     2
26|     1
27|     0
```

9.4.1.



1| MOV R4, #arrayData

2| MOV R1, #36

3| MOV R2, #0

4| MOV R3, #0

5| LDR R9, arrayLength

6| MOV R5, #newArrData

7|LOOP:

8| LDR R6, [R4 + R1]

9| STR R6, [R5 + R3]

10| ADD R3, R3, #4

11| SUB R1, R1, #4

12| ADD R2, R2, #1

13| CMP R2, R9

14| BLT LOOP

15| HALT

```
16|     .ALIGN 256
17|arrayLength: 10
18|arrayData: 9
19|     8
20|     7
21|     6
22|     5
23|     4
24|     3
25|     2
26|     1
27|     0
28|newArrData: .BLOCK 256
```

9.4.2.