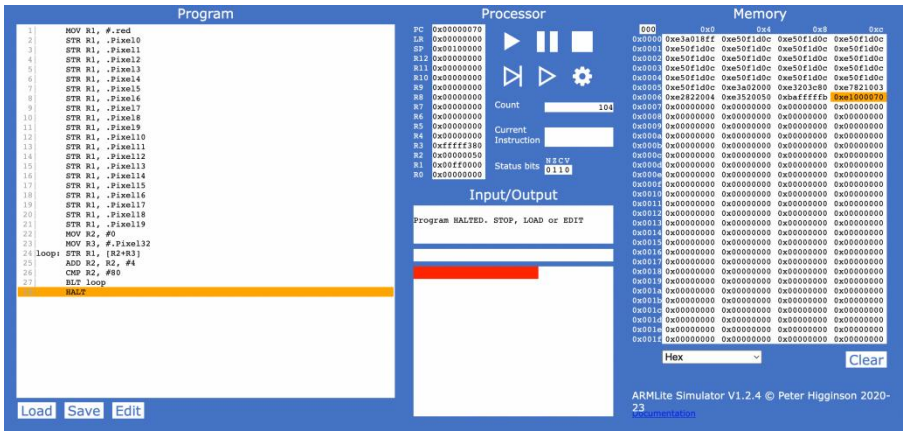
Student's ID: 104209393
Student's name: Ai Vi Tran

9.1.1:
(a)

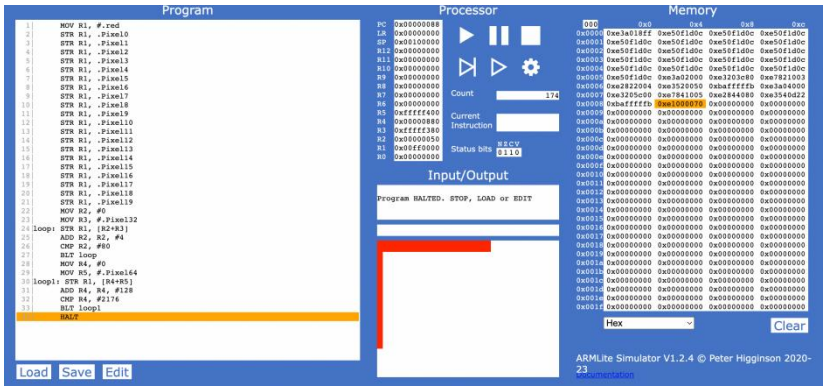
```
 1|     MOV R1, #.red
 2|     STR R1, .Pixel0
 3|     STR R1, .Pixel1
 4|     STR R1, .Pixel2
 5|     STR R1, .Pixel3
 6|     STR R1, .Pixel4
 7|     STR R1, .Pixel5
 8|     STR R1, .Pixel6
 9|     STR R1, .Pixel7
10|     STR R1, .Pixel8
11|     STR R1, .Pixel9
12|     STR R1, .Pixel10
13|     STR 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: STR R1, [R2+R3]
25|     ADD R2, R2, #4
26|     CMP R2, #80
27|     BLT loop
28|     HALT
```



(b)

```
 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: STR R1, [R2+R3]
25|     ADD R2, R2, #4
26|     CMP R2, #80
27|     BLT loop
28|     MOV R4, #0
29|     MOV R5, #.Pixel64
30|loop1: STR R1, [R4+R5]
31|     ADD R4, R4, #128
32|     CMP R4, #2176
33|     BLT loop1
34|     HALT
```

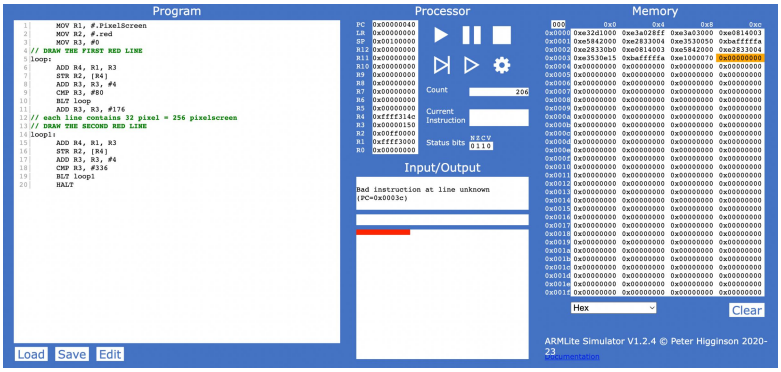


9.1.3:

(a) This code is an example of indirect addressing because there is a line STR R2, [R4]. This will store the content that the memory of R4 into memory of the R2, It use indirect addressing to draw each pixel because the memory of R4 will change every single loop base on R3 and the R2 which have the value .red will store.

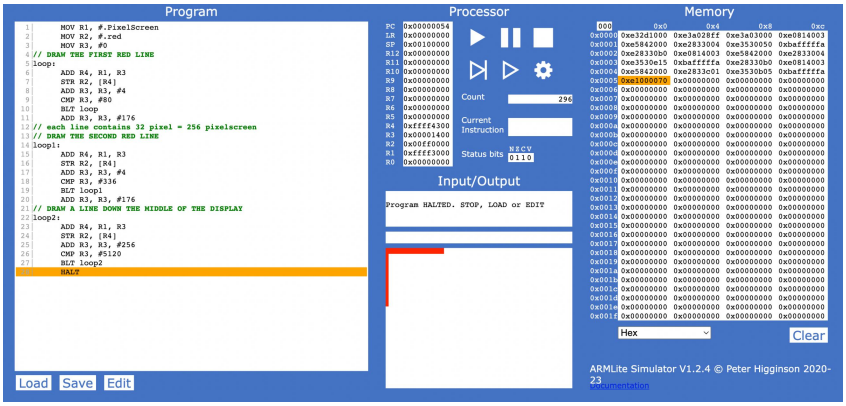
(b)

```
1|     MOV R1, #.PixelScreen
2|     MOV R2, #.red
3|     MOV R3, #0
4|// DRAW THE FIRST RED LINE
5|loop:
6|     ADD R4, R1, R3
7|     STR R2, [R4]
8|     ADD R3, R3, #4
9|     CMP R3, #80
10|     BLT loop
11|     ADD R3, R3, #176
12|// each line contains 32 pixel = 256 pixelscreen
13|// DRAW THE SECOND RED LINE
14|loop1:
15|     ADD R4, R1, R3
16|     STR R2, [R4]
17|     ADD R3, R3, #4
18|     CMP R3, #336
19|     BLT loop1
20|     HALT
```



(c)

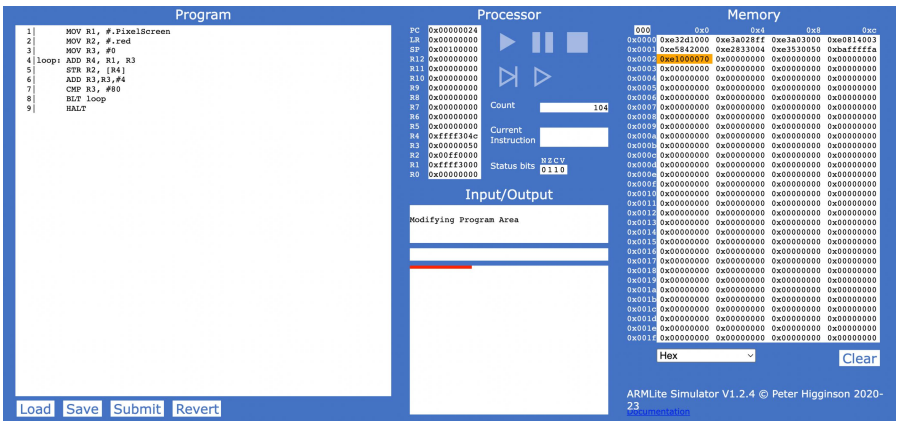
```
 1|      MOV R1, #.PixelScreen
 2|      MOV R2, #.red
 3|      MOV R3, #0
 4|// DRAW THE FIRST RED LINE
 5|loop:
 6|      ADD R4, R1, R3
 7|      STR R2, [R4]
 8|      ADD R3, R3, #4
 9|      CMP R3, #80
10|      BLT loop
11|      ADD R3, R3, #176
12|// each line contains 32 pixel = 256 pixelscreen
13|// DRAW THE SECOND RED LINE
14|loop1:
15|      ADD R4, R1, R3
16|      STR R2, [R4]
17|      ADD R3, R3, #4
18|      CMP R3, #336
19|      BLT loop1
20|      ADD R3, R3, #176
21|// DRAW A LINE DOWN THE MIDDLE OF THE DISPLAY
22|loop2:
23|      ADD R4, R1, R3
24|      STR R2, [R4]
25|      ADD R3, R3, #256
26|      CMP R3, #5120
27|      BLT loop2
28|      HALT
```



9.2.1:

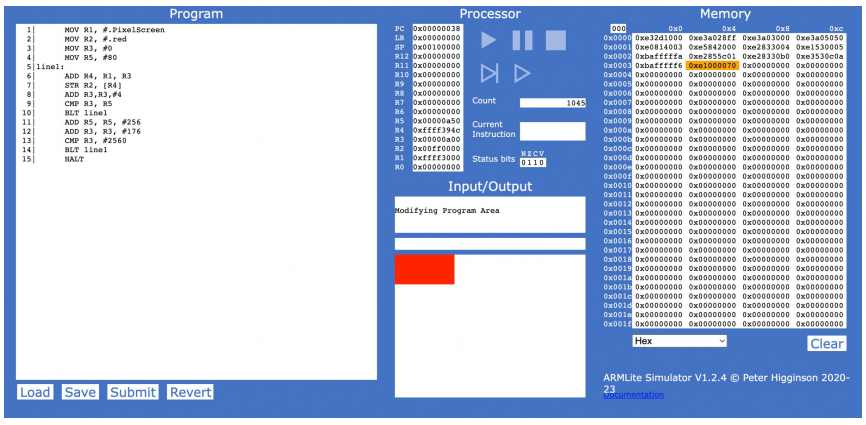
```
1|      MOV R1, #.PixelScreen
 2|      MOV R2, #.red
 3|      MOV R3, #0
 4|loop: ADD R4, R1, R3
 5|      STR R2, [R4]
```

```
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 R5, #80
5|line1:
6|      ADD R4, R1, R3
7|      STR R2, [R4]
8|      ADD R3,R3,#4
9|      CMP R3, R5
10|     BLT line1
11|     ADD R5, R5, #256
12|     ADD R3, R3, #176
13|     CMP R3, #2560
14|     BLT line1
15|     HALT
```



9.3.1:

(a): The purpose of using .Align 256 instruction is to ensure that the array starts at a memory address that is a multiple of 256.

(b):

```
1|      MOV R4, #arrayData
2|      MOV R1, #16       //array starts from 0
3|      LDR R0, [R4+R1]
4|      HALT
5|      .ALIGN 256
6|arrayLength: 10
7|arrayData: 9
8|      8
```

9| 7
10| 6
11| 5
12| 4
13| 3
14| 2
15| 1
16| 0

9.3.2: