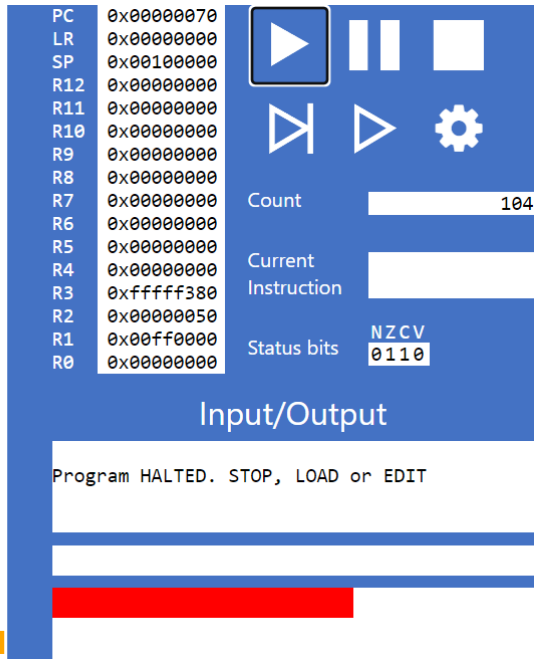
Name: Tran Duc Anh Dang

Exercise 9.1.1

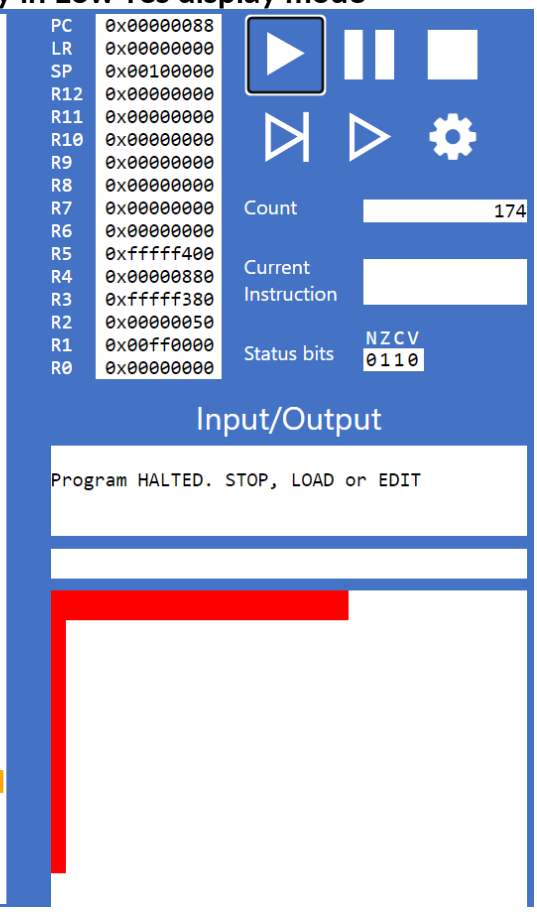
(a) Write a simple ARMlite assembly program that draws a single line of the same length across the second row (starting from the left-most column) in Low-res display mode.

```
        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
        MOV R2, #0
        MOV R3, #.Pixel32
loop:   STR R1, [R2+R3]
        ADD R2,R2,#4
        CMP R2,#80
        BLT loop
        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

```
        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
        MOV R2, #0
        MOV R3, #.Pixel32
loop:   STR R1, [R2+R3]
        ADD R2,R2,#4
        CMP R2,#80
        BLT loop
        MOV R4, #0
        MOV R5, #.Pixel64
loop1:  STR R1, [R4+R5]
        ADD R4,R4,#128
        CMP R4,#2176
        BLT loop1
        HALT
```

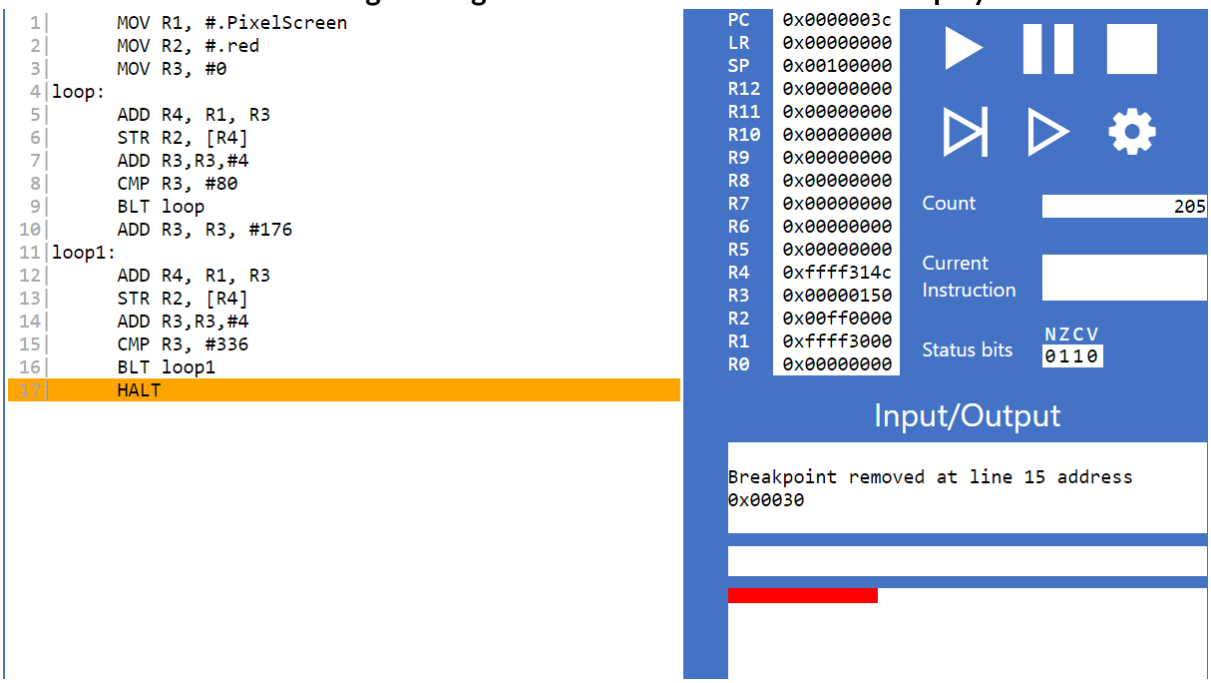


Exercise 9.1.3

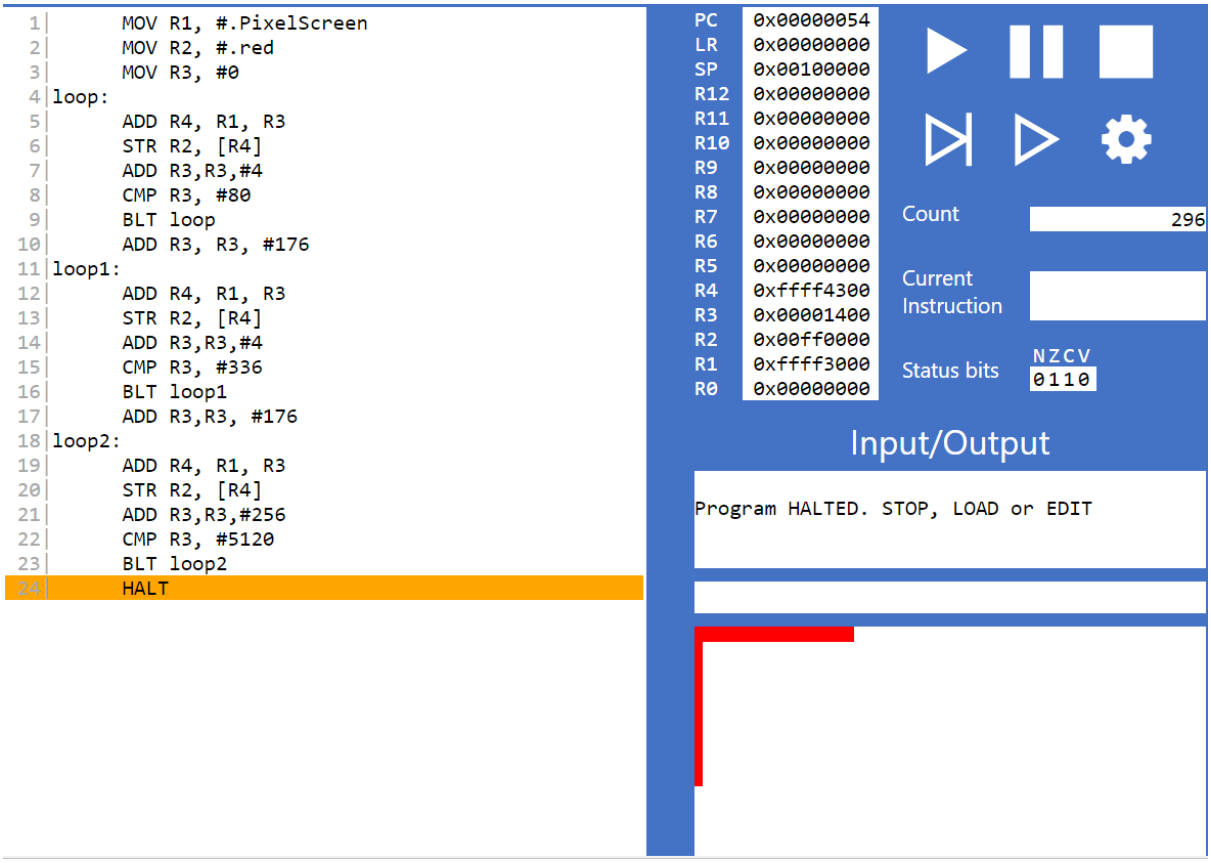
(a) **Explain what specifically makes this code an example of indirect addressing ? How is it using indirect addressing to draw each pixel ?**

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

(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**



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



Exercise 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
```

PC	0x00000020
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0x00000050
R2	0x00ff0000
R1	0xffff3000
R0	0x00000000

Count 84

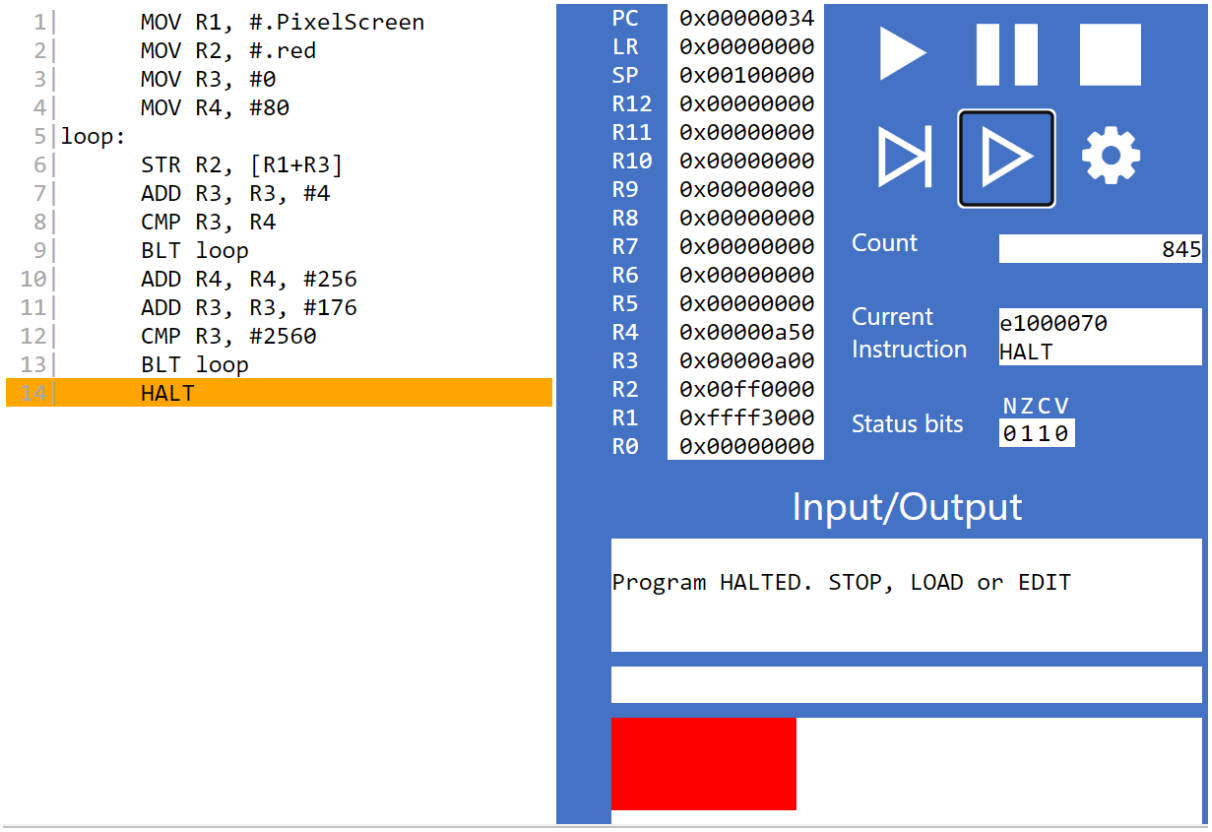
Current Instruction

Status bits NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Excercise 9.2.2



Excercise 9.3.1 (a)

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

Excercise 9.3.1 (b)

```
1        MOV R4, #arrayData
2        MOV R1, #16
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
```

PC	0x00000010
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	0x00000010
R0	0x00000005

Count 4

Current Instruction

Status bits NZCV 0000

Excercise 9.3.1 (c)

```
1        MOV R4, #arrayData
2        MOV R0, #16
3        LDR R1, [R4+R0]
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
```

PC	0x00000010
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	0x00000005
R0	0x00000010

Count 4

Current Instruction

Status bits NZCV 0000

Excercise 9.3.2

```
        MOV R4, #arrayData
        MOV R0, #0
        MOV R1, #0
        MOV R2, #10
        MOV R3, #0
loop:
        LDR R5, [R4+R1]
        ADD R0, R0, R5
        ADD R3, R3, #1
        ADD R1, R1, #4
        CMP R3, R2
        BLT loop
        HALT
        .ALIGN 256
arrayLength: 10
arrayData: 9
        8
        7
        6
        5
        4
        3
        2
        1
        0
```

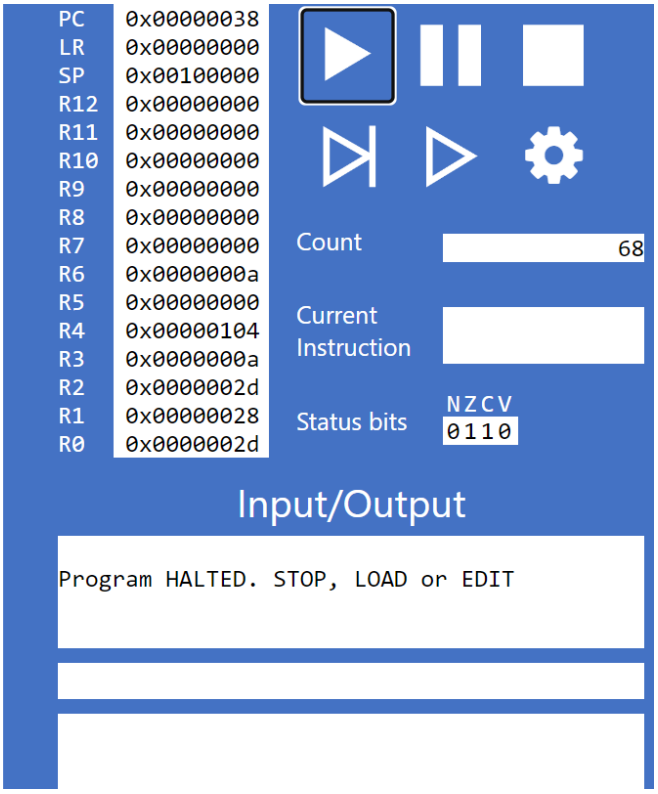
PC	0x00000030
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000104
R3	0x0000000a
R2	0x0000000a
R1	0x00000028
R0	0x0000002d

In

```
Program HALTED.
```

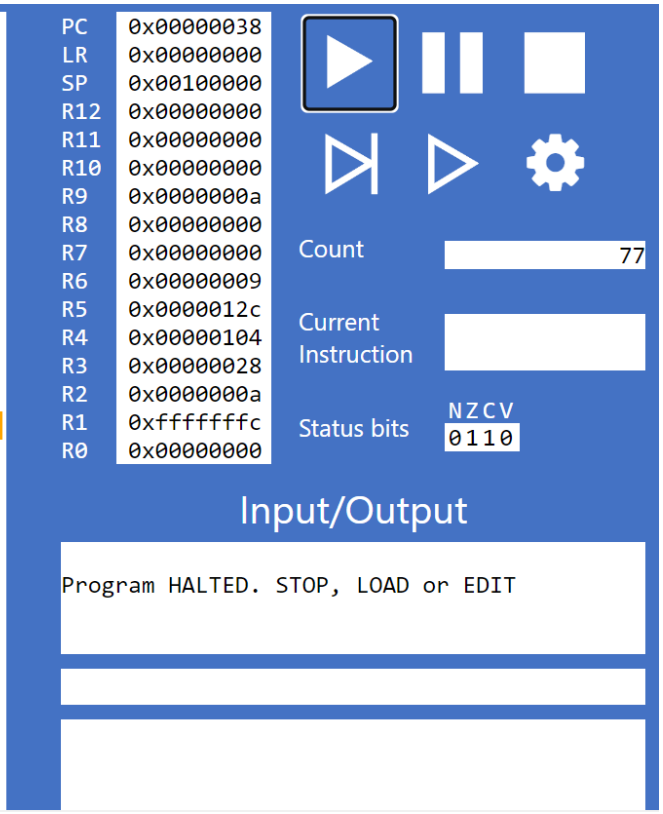
Excercise 9.3.3

```
        MOV R4, #arrayData
        MOV R1, #0
        MOV R2, #0
        MOV R6, #0
        LDR R3, arrayLength
loop:
        LDR R5, [R4+R1]
        ADD R2, R2, R5
        ADD R1, R1, #4
        ADD R6, R6, #1
        CMP R6, R3
        BLT loop
        MOV R0, #0
        ADD R0, R0, R2
        HALT
        .ALIGN 256
arrayLength: 10
arrayData: 9
        8
        7
        6
        5
        4
        3
        2
        1
        0
```



Excercise 9.4.1

```
        MOV R4, #arrayData
        MOV R1, #36
        MOV R2, #0
        MOV R3, #0
        LDR R9, arrayLength
        MOV R5, #newArrData
loop:
        LDR R6, [R4+R1]
        STR R6, [R5+R3]
        ADD R3, R3, #4
        SUB R1, R1, #4
        ADD R2, R2, #1
        CMP R2, R9
        BLT loop
        HALT
        .ALIGN 256
arrayLength: 10
arrayData: 9
        8
        7
        6
        5
        4
        3
        2
        1
        0
newArrData: .BLOCK 256
```



Excercise 9.4.2

```
1        MOV R4, #arrayData
2        MOV R1, #36
3        MOV R2, #0
4        MOV R3, #0
5        MOV R9, #5
6  loop:
7        LDR R5, [R4+R3]
8        LDR R6, [R4+R1]
9        STR R6, [R4+R3]
10       STR R5, [R4+R1]
11       ADD R3, R3, #4
12       SUB R1, R1, #4
13       ADD R2, R2, #1
14       CMP R2, R9
15       BLT loop
16       MOV R7, #12
17       LDR R8, [R4+R7]
18       HALT
19       .ALIGN 256
20 arrayLength: 10
21 arrayData: 9
22       8
23       7
24       6
25       5
26       4
27       3
28       2
29       1
30       0
```

PC	0x00000044
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000005
R8	0x00000003
R7	0x0000000c
R6	0x00000004
R5	0x00000005
R4	0x00000104
R3	0x00000014
R2	0x00000005
R1	0x00000010
R0	0x00000000

Count 53

Current Instruction

Status bits NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT