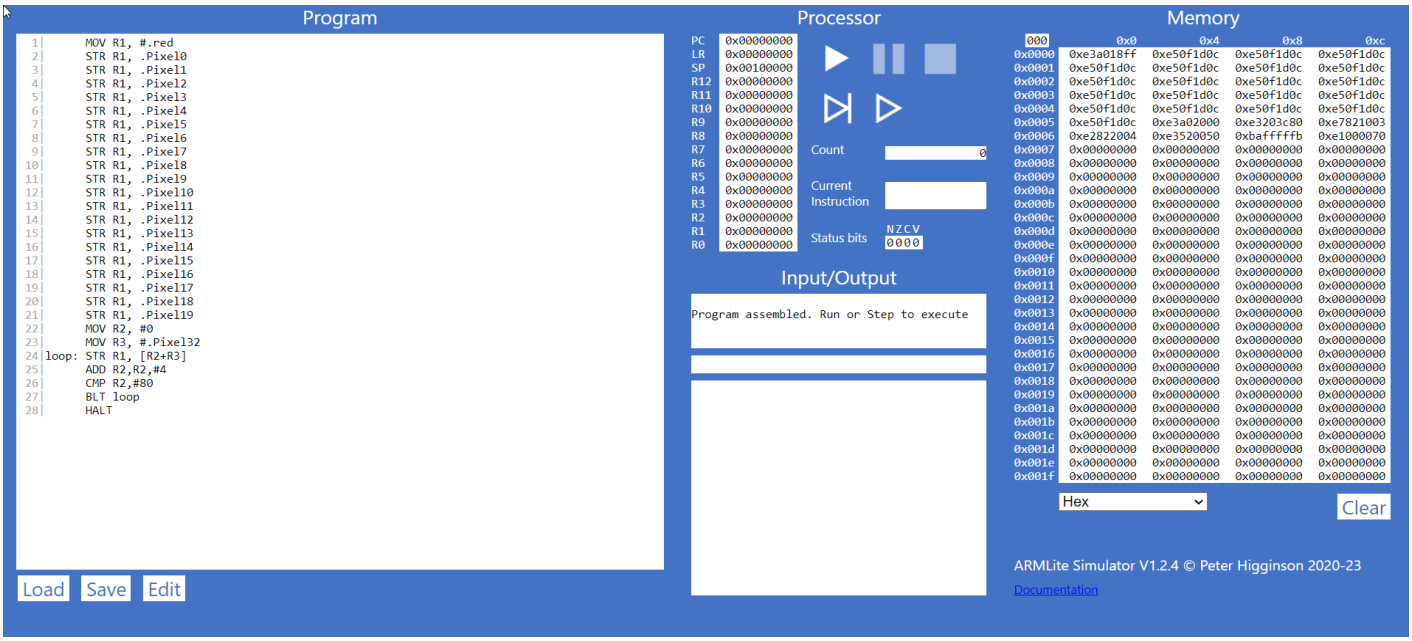
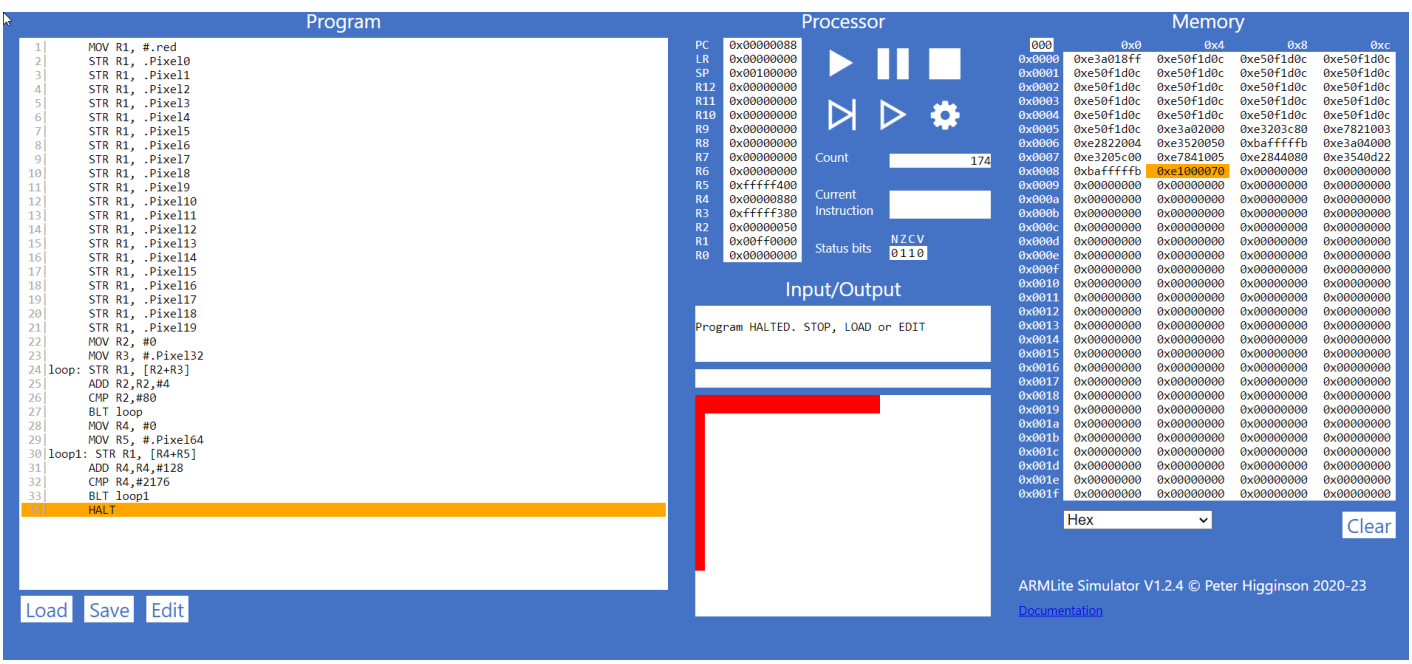
9.1 a) Write a simple ARMlite assembly program that draws a single line of the same length across the second row (starting from the left-most column) in Low-res display mode.



(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.

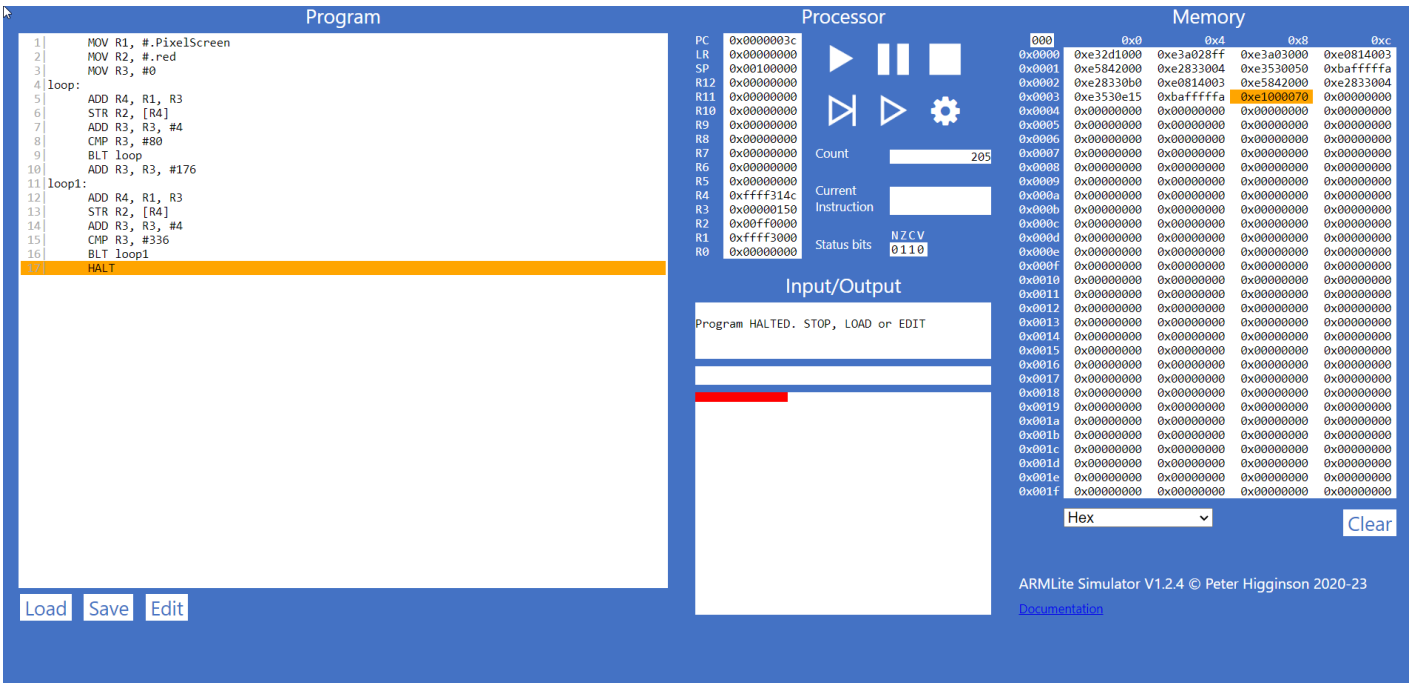


9.1.3

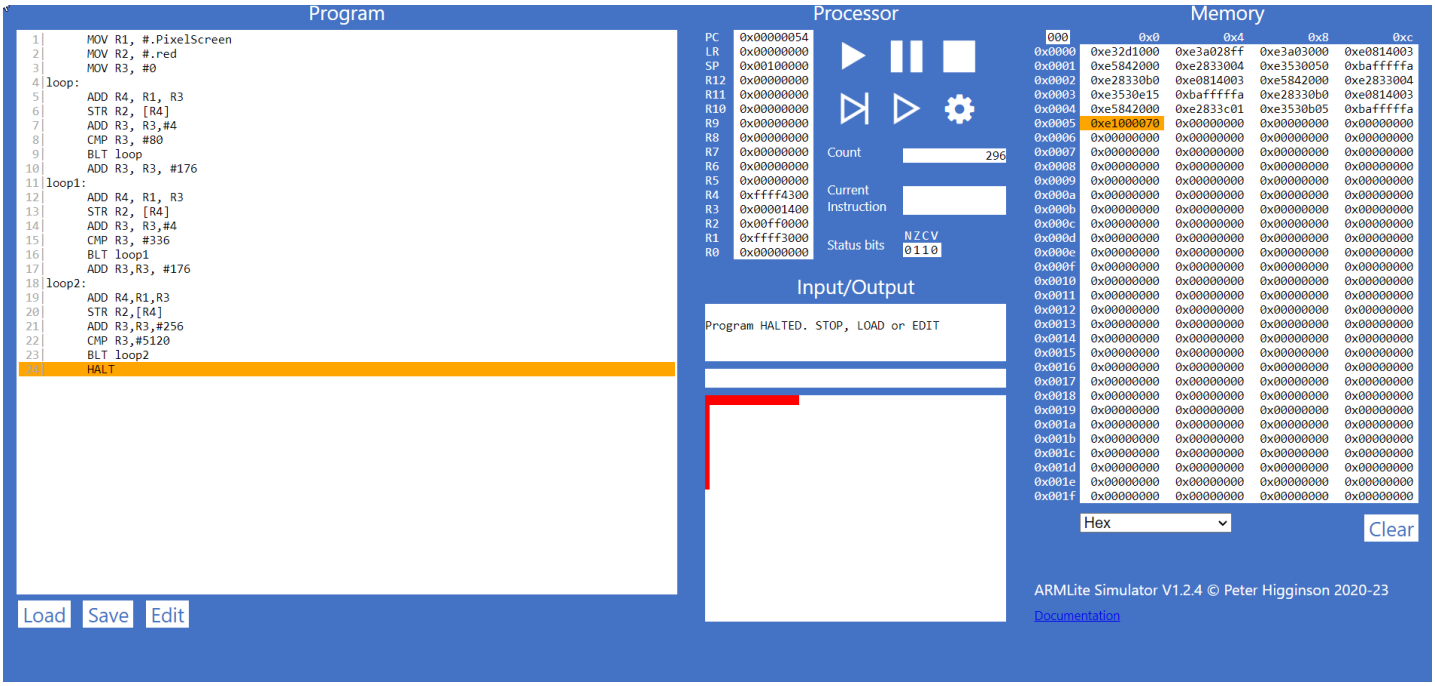
(a) Explain what specifically makes this code an example of indirect addressing ? How is it using indrect addressing to draw each pixel ?

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

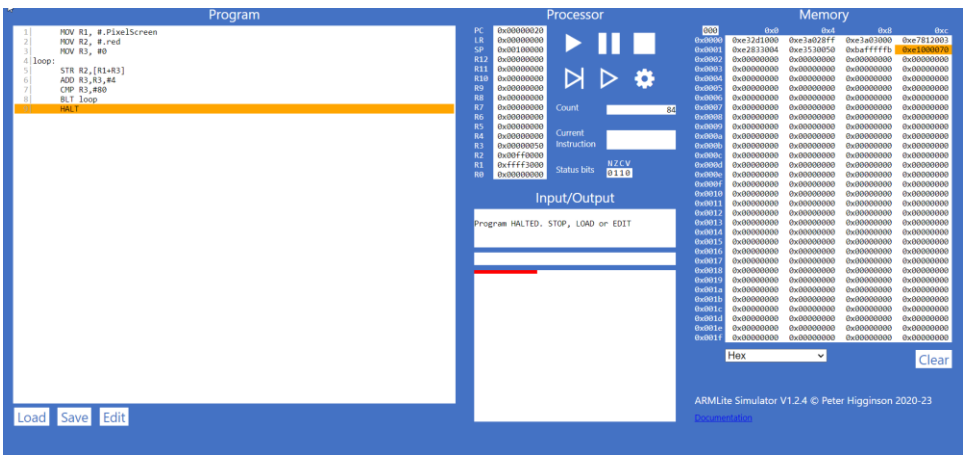
(b) Once you're confident yo understand the code, modify the program so that it draws a line of the same length along the second row of the Mid-res display.



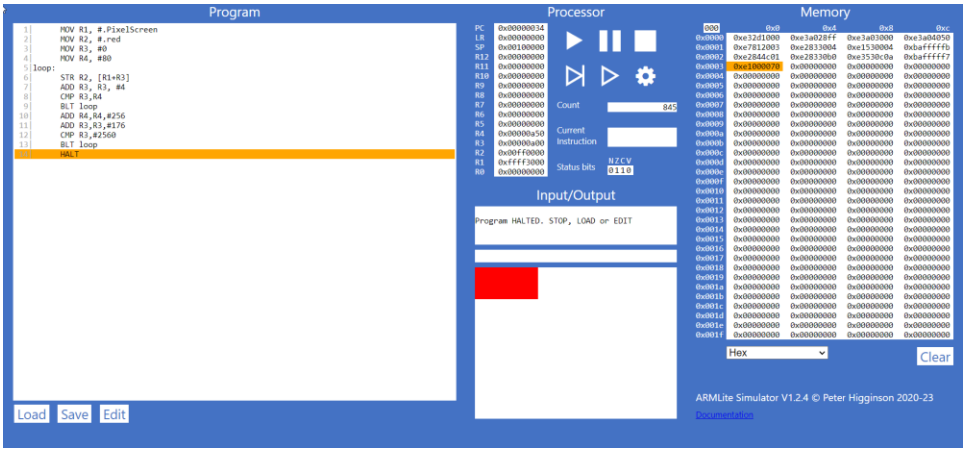
(c) Further modify your program so that it also draws a line of the same length vertically down the middle of the display.


Program
MOV R1, #.PixelScreen
MOV R2, #.red
MOV R3, #0
loop:
ADD R4, R1, R3
STR R2, [R4]
ADD R3, R3,#4
CMP R3, #80
BLT loop
ADD R3, R3, #176
loop1:
ADD R4, R1, R3
STR R2, [R4]
ADD R3, R3,#4
CMP R3, #336
BLT loop1
ADD R3,R3, #176
loop2:
ADD R4,R1,R3
STR R2,[R4]
ADD R3,R3,#256
CMP R3,#5120
BLT loop2
HALT
Load
Save
Edit
Processor
PC 0x00000054
LR 0x00000000
SP 0x00100000
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x00000000
R7 0x00000000
R6 0x00000000
R5 0x00000000
R4 0xffff4300
R3 0x00001400
R2 0x00ff0000
R1 0xffff3000
R0 0x00000000
Count 296
Current Instruction
Status bits NZCV 0110
Input/Output
Program HALTED. STOP, LOAD or EDIT
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe32d1000 0xe3a028ff 0xe3a03000 0xe0814003
0x0001 0xe5842000 0xe2833004 0xe3530050 0xbaffffa
0x0002 0xe28330b0 0xe0814003 0xe5842000 0xe2833004
0x0003 0xe3530e15 0xbaffffa 0xe28330b0 0xe0814003
0x0004 0xe5842000 0xe2833c01 0xe3530b05 0xbaffffa
0x0005 0xe1000070 0x00000000 0x00000000 0x00000000
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

9.2.1


Program
MOV R1, #.PixelScreen
MOV R2, #.red
MOV R3, #0
loop:
STR R2,[R1+R3]
ADD R3,R3,#4
CMP R3,#80
BLT loop
HALT
Load
Save
Edit
Processor
PC 0x00000020
LR 0x00000000
SP 0x00100000
R12 0x00000000
R11 0x00000000
R10 0x00000000
R9 0x00000000
R8 0x00000000
R7 0x00000000
R6 0x00000000
R5 0x00000000
R4 0x00000050
R3 0x00000050
R2 0x00ff0000
R1 0xffff3000
R0 0x00000000
Count 84
Current Instruction
Status bits NZCV 0110
Input/Output
Program HALTED. STOP, LOAD or EDIT
Memory
000 0x0 0x4 0x8 0xc
0x0000 0xe32d1000 0xe3a028ff 0xe3a03000 0xe7812003
0x0001 0xe2833004 0xe3530050 0xbaffffb 0xe1000070
Hex
Clear
ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
Documentation

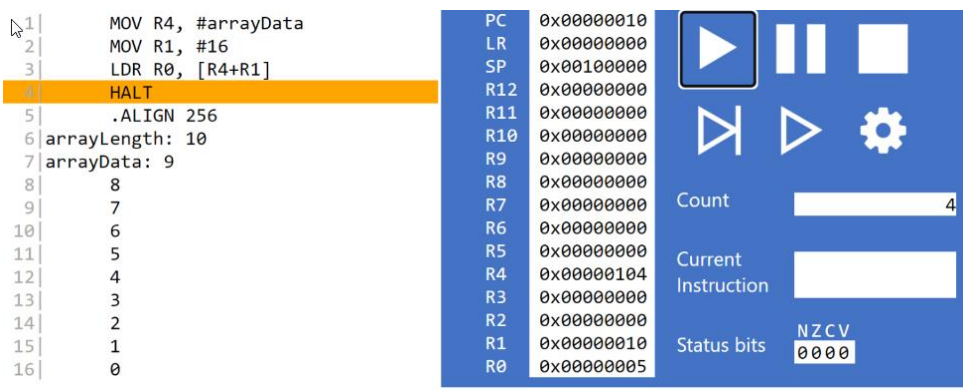
9.2.2



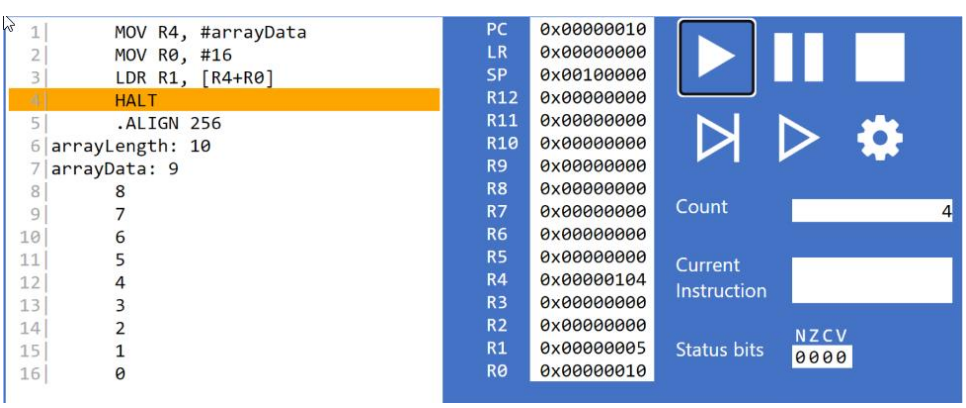
9.3.1

(a) The purpose of. ALIGN 256 will align the following instructions data as to the next byte address that is divisible by 256.

(b)

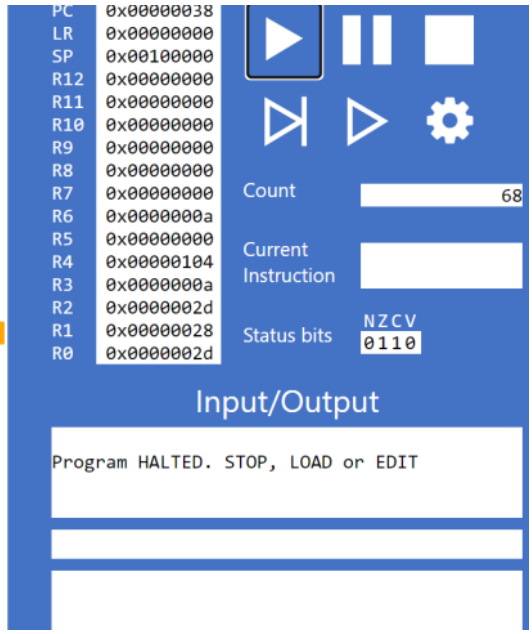


(c)



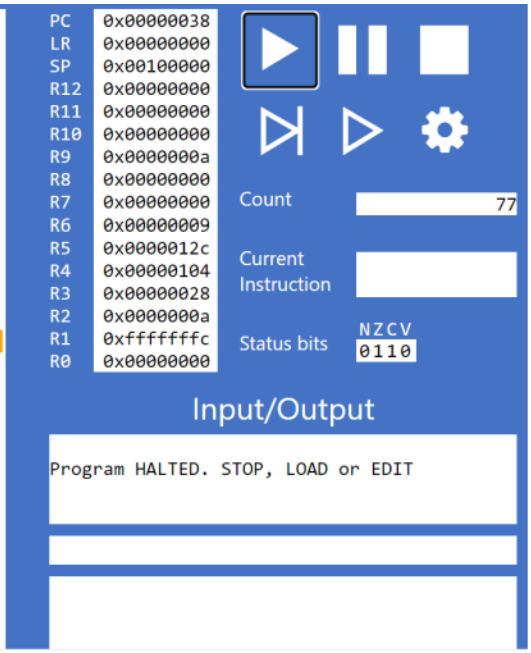
9.3.2

```
        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
```



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
```



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