

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.

The screenshot shows the ARMLite Simulator V1.2.4 interface. The **Program** window contains the following assembly code:

```

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

```

The **Processor** window shows the PC at 0x00000000, LR at 0x00000000, SP at 0x00100000, and registers R1 through R10. The **Memory** window shows a hex dump of memory starting at 0x00000000. The **Input/Output** window shows the program assembled and ready to run. The **Count** is 0, and the **Status bits** are NZCV 0000.

(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.

The screenshot shows the ARMLite Simulator V1.2.4 interface with the assembly program updated to draw a vertical line. The **Program** window contains the following assembly code:

```

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, #.Pixel164
30 loop1: STR R1, [R4,R5]
31 ADD R4, R4, #128
32 CMP R4, #2176
33 BLT loop1
34 HALT

```

The **Processor** window shows the PC at 0x00000008, LR at 0x00000000, SP at 0x00100000, and registers R1 through R10. The **Memory** window shows a hex dump of memory starting at 0x00000000. The **Input/Output** window shows the program HALTED. The **Count** is 174, and the **Status bits** are NZCV 0110. The display shows a red horizontal line across the second row and a red vertical line down the middle of the display.

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 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 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.

Program

```

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

```

Load

Save

Edit

Processor

PC

0x0000003c

LR

0x00000000

SP

0x00100000

R12

0x00000000

R11

0x00000000

R10

0x00000000

R9

0x00000000

R8

0x00000000

R7

0x00000000

R6

0x00000000

R5

0x00000000

R4

0xffff314c

R3

0x0000150

R2

0x00ff0000

R1

0xffff3000

R0

0x00000000

Count

205

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	0xbaffffffa
0x0002	0xe28330b0	0xe0814003	0xe5842000	0xe2833004
0x0003	0xe3530e15	0xbaffffffa	0xe1000070	0x00000000
0x0004	0x00000000	0x00000000	0x00000000	0x00000000
0x0005	0x00000000	0x00000000	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000
0x0007	0x00000000	0x00000000	0x00000000	0x00000000
0x0008	0x00000000	0x00000000	0x00000000	0x00000000
0x0009	0x00000000	0x00000000	0x00000000	0x00000000
0x000a	0x00000000	0x00000000	0x00000000	0x00000000
0x000b	0x00000000	0x00000000	0x00000000	0x00000000
0x000c	0x00000000	0x00000000	0x00000000	0x00000000
0x000d	0x00000000	0x00000000	0x00000000	0x00000000
0x000e	0x00000000	0x00000000	0x00000000	0x00000000
0x000f	0x00000000	0x00000000	0x00000000	0x00000000
0x0010	0x00000000	0x00000000	0x00000000	0x00000000
0x0011	0x00000000	0x00000000	0x00000000	0x00000000
0x0012	0x00000000	0x00000000	0x00000000	0x00000000
0x0013	0x00000000	0x00000000	0x00000000	0x00000000
0x0014	0x00000000	0x00000000	0x00000000	0x00000000
0x0015	0x00000000	0x00000000	0x00000000	0x00000000
0x0016	0x00000000	0x00000000	0x00000000	0x00000000
0x0017	0x00000000	0x00000000	0x00000000	0x00000000
0x0018	0x00000000	0x00000000	0x00000000	0x00000000
0x0019	0x00000000	0x00000000	0x00000000	0x00000000
0x001a	0x00000000	0x00000000	0x00000000	0x00000000
0x001b	0x00000000	0x00000000	0x00000000	0x00000000
0x001c	0x00000000	0x00000000	0x00000000	0x00000000
0x001d	0x00000000	0x00000000	0x00000000	0x00000000
0x001e	0x00000000	0x00000000	0x00000000	0x00000000
0x001f	0x00000000	0x00000000	0x00000000	0x00000000

Hex

Clear

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Documentation

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

Program

```

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

```

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	0xfffff4300
R3	0x00001400
R2	0x00fff0000
R1	0xfffff3000
R0	0x00000000

Count: 296

Current Instruction:

Status bits: NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Memory

	0x0	0x4	0x8	0xc
0x0000	0xe32d1000	0xe3a028ff	0xe3a03000	0xe0814003
0x0001	0xe5842000	0xe2833004	0xe3530050	0xbafffffa
0x0002	0xe28330b0	0xe0814003	0xe5842000	0xe2833004
0x0003	0xe3530e15	0xbaffffffa	0xe28330b0	0xe0814003
0x0004	0xe5842000	0xe2833c01	0xe3530b05	0xbaffffffa
0x0005	0xe1000070	0x00000000	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000
0x0007	0x00000000	0x00000000	0x00000000	0x00000000
0x0008	0x00000000	0x00000000	0x00000000	0x00000000
0x0009	0x00000000	0x00000000	0x00000000	0x00000000
0x000a	0x00000000	0x00000000	0x00000000	0x00000000
0x000b	0x00000000	0x00000000	0x00000000	0x00000000
0x000c	0x00000000	0x00000000	0x00000000	0x00000000
0x000d	0x00000000	0x00000000	0x00000000	0x00000000
0x000e	0x00000000	0x00000000	0x00000000	0x00000000
0x000f	0x00000000	0x00000000	0x00000000	0x00000000
0x0010	0x00000000	0x00000000	0x00000000	0x00000000
0x0011	0x00000000	0x00000000	0x00000000	0x00000000
0x0012	0x00000000	0x00000000	0x00000000	0x00000000
0x0013	0x00000000	0x00000000	0x00000000	0x00000000
0x0014	0x00000000	0x00000000	0x00000000	0x00000000
0x0015	0x00000000	0x00000000	0x00000000	0x00000000
0x0016	0x00000000	0x00000000	0x00000000	0x00000000
0x0017	0x00000000	0x00000000	0x00000000	0x00000000
0x0018	0x00000000	0x00000000	0x00000000	0x00000000
0x0019	0x00000000	0x00000000	0x00000000	0x00000000
0x001a	0x00000000	0x00000000	0x00000000	0x00000000
0x001b	0x00000000	0x00000000	0x00000000	0x00000000
0x001c	0x00000000	0x00000000	0x00000000	0x00000000
0x001d	0x00000000	0x00000000	0x00000000	0x00000000
0x001e	0x00000000	0x00000000	0x00000000	0x00000000
0x001f	0x00000000	0x00000000	0x00000000	0x00000000

Hex
Clear

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

[Documentation](#)

9.2.1

Program

```

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

```

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	0x00000000
R3	0x00000050
R2	0x00fff0000
R1	0xfffff3000
R0	0x00000000

Count: 82

Current Instruction:

Status bits: NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Memory

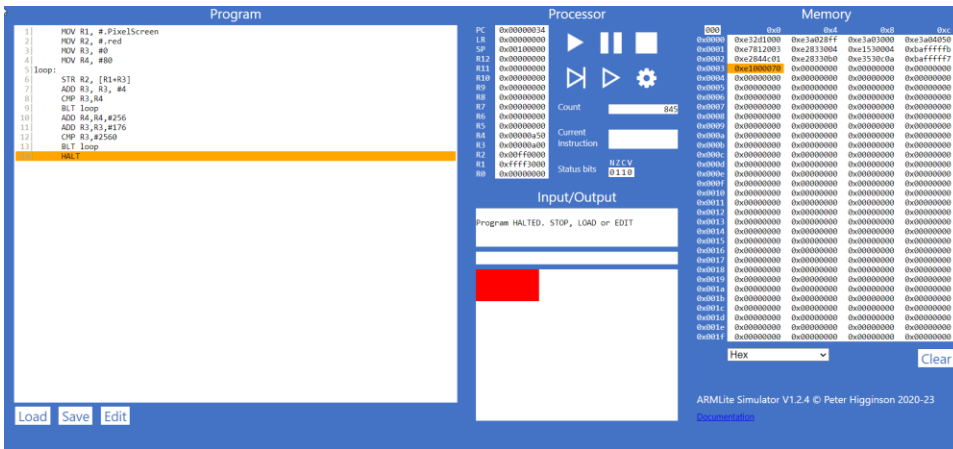
	0x0	0x4	0x8	0xc
0x0000	0xe32d1000	0xe3a028ff	0xe3a03000	0xe7812003
0x0001	0xe2811004	0xe3530050	0xbaffffffa	0xe1000070
0x0002	0x00000000	0x00000000	0x00000000	0x00000000
0x0003	0x00000000	0x00000000	0x00000000	0x00000000
0x0004	0x00000000	0x00000000	0x00000000	0x00000000
0x0005	0x00000000	0x00000000	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000
0x0007	0x00000000	0x00000000	0x00000000	0x00000000
0x0008	0x00000000	0x00000000	0x00000000	0x00000000
0x0009	0x00000000	0x00000000	0x00000000	0x00000000
0x000a	0x00000000	0x00000000	0x00000000	0x00000000
0x000b	0x00000000	0x00000000	0x00000000	0x00000000
0x000c	0x00000000	0x00000000	0x00000000	0x00000000
0x000d	0x00000000	0x00000000	0x00000000	0x00000000
0x000e	0x00000000	0x00000000	0x00000000	0x00000000
0x000f	0x00000000	0x00000000	0x00000000	0x00000000
0x0010	0x00000000	0x00000000	0x00000000	0x00000000
0x0011	0x00000000	0x00000000	0x00000000	0x00000000
0x0012	0x00000000	0x00000000	0x00000000	0x00000000
0x0013	0x00000000	0x00000000	0x00000000	0x00000000
0x0014	0x00000000	0x00000000	0x00000000	0x00000000
0x0015	0x00000000	0x00000000	0x00000000	0x00000000
0x0016	0x00000000	0x00000000	0x00000000	0x00000000
0x0017	0x00000000	0x00000000	0x00000000	0x00000000
0x0018	0x00000000	0x00000000	0x00000000	0x00000000
0x0019	0x00000000	0x00000000	0x00000000	0x00000000
0x001a	0x00000000	0x00000000	0x00000000	0x00000000
0x001b	0x00000000	0x00000000	0x00000000	0x00000000
0x001c	0x00000000	0x00000000	0x00000000	0x00000000
0x001d	0x00000000	0x00000000	0x00000000	0x00000000
0x001e	0x00000000	0x00000000	0x00000000	0x00000000
0x001f	0x00000000	0x00000000	0x00000000	0x00000000

Hex
Clear

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

[Documentation](#)

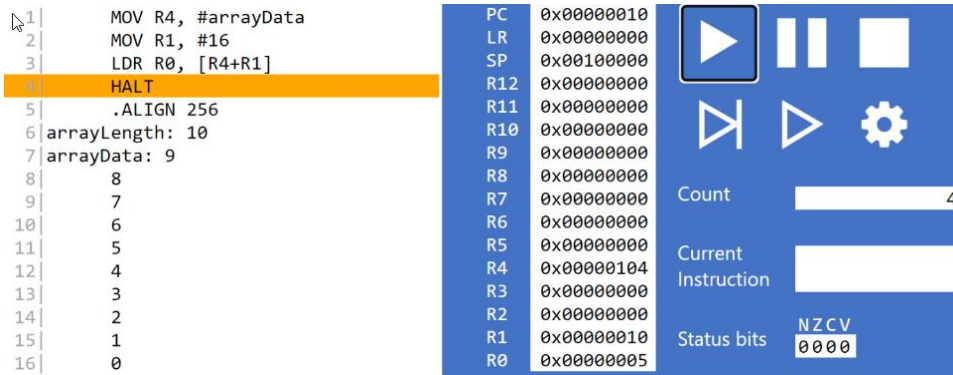
9.2.2



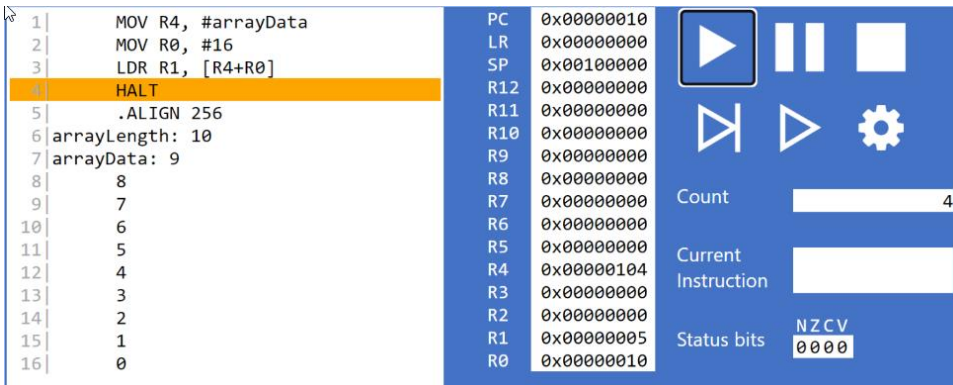
9.3.1

(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

1|
2|
3|
4|
5|
6| loop:
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|

MOV R4, #arrayData
MOV R1, #0
MOV R2, #0
MOV R6, #0
LDR R3, arrayLength
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

PC
LR
SP
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0

0x00000038
0x00000000
0x00100000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x0000000a
0x00000000
0x00000104
0x0000000a
0x0000002d
0x00000028
0x0000002d

▶

⏏

⏏

⏮

⏭

⚙

Count

68

Current Instruction

Status bits

NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

9.4.1

1|
2|
3|
4|
5|
6|
7| loop:
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|

MOV R4, #arrayData
MOV R1, #36
MOV R2, #0
MOV R3, #0
LDR R9, arrayLength
MOV R5, #newArrData
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

PC
LR
SP
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0

0x00000038
0x00000000
0x00100000
0x00000000
0x00000000
0x00000000
0x0000000a
0x00000000
0x00000000
0x00000000
0x00000009
0x0000012c
0x00000104
0x00000028
0x0000000a
0xffffffffc
0x00000000

▶

⏏

⏏

⏮

⏭

⚙

Count

77

Current Instruction

Status bits

NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT




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

Current Instruction

Status bits **NZCV**
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT