

_Name: Nguyen Quoc Thang

ID: 104193360

Lab 9 : SUBMISSTION

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

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

```
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: STR R1, .Pixel32
23: STR R1, .Pixel33
24: STR R1, .Pixel34
25: STR R1, .Pixel35
26: STR R1, .Pixel36
27: STR R1, .Pixel37
28: STR R1, .Pixel38
29: STR R1, .Pixel39
30: STR R1, .Pixel40
31: STR R1, .Pixel41
32: STR R1, .Pixel42
33: STR R1, .Pixel43
34: STR R1, .Pixel44
35: STR R1, .Pixel45
36: STR R1, .Pixel46
37: STR R1, .Pixel47
38: STR R1, .Pixel48
39: STR R1, .Pixel49
40: STR R1, .Pixel50
41: STR R1, .Pixel51
42: HALT
```

The **Processor** window shows the program is halted at instruction 42. The **Memory** window shows the address range 0x00000000 to 0x0000000F. The **Input/Output** window shows the program is halted. The **Display** window shows a single horizontal line of red pixels on the second row.

- (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 interface. The **Program** window contains the following assembly code:

```
23: STR R1, .Pixel33
24: STR R1, .Pixel34
25: STR R1, .Pixel35
26: STR R1, .Pixel36
27: STR R1, .Pixel37
28: STR R1, .Pixel38
29: STR R1, .Pixel39
30: STR R1, .Pixel40
31: STR R1, .Pixel41
32: STR R1, .Pixel42
33: STR R1, .Pixel43
34: STR R1, .Pixel44
35: STR R1, .Pixel45
36: STR R1, .Pixel46
37: STR R1, .Pixel47
38: STR R1, .Pixel48
39: STR R1, .Pixel49
40: STR R1, .Pixel50
41: STR R1, .Pixel51
42: STR R1, .Pixel64
43: STR R1, .Pixel65
44: STR R1, .Pixel66
45: STR R1, .Pixel67
46: STR R1, .Pixel68
47: STR R1, .Pixel69
48: STR R1, .Pixel70
49: STR R1, .Pixel71
50: STR R1, .Pixel72
51: STR R1, .Pixel73
52: STR R1, .Pixel74
53: STR R1, .Pixel75
54: STR R1, .Pixel76
55: STR R1, .Pixel77
56: STR R1, .Pixel78
57: STR R1, .Pixel79
58: STR R1, .Pixel80
59: STR R1, .Pixel81
60: STR R1, .Pixel82
61: HALT
```

The **Processor** window shows the program is halted at instruction 61. The **Memory** window shows the address range 0x00000000 to 0x0000000F. The **Input/Output** window shows the program is halted. The **Display** window shows a single vertical line of red pixels in the middle of the display.

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 ?

The code snippet is an example of indirect addressing because it uses the values R1 and R3 to calculate the byte offset of each pixel, that byte offset is used as the address, which is stored in R4 and stored the value of each color of the pixel into R2

(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.

The screenshot displays the ARM Lite Simulator V1.2.4 interface. The 'Program' pane on the left shows assembly code for drawing a horizontal line. The 'Processor' pane in the center shows the current state of the processor, including the PC, registers, and status bits. The 'Memory' pane on the right shows a hex dump of memory addresses. The 'Input/Output' pane at the bottom shows the program status and a small display window.

Program

```
1 MOV R1, #PixelScreen // base address of the medium and high res pixel display memory
2 MOV R2, #red
3 MOV R3, #256
4 loop:
5   ADD R4, R1, R3 // calculate the byte offset (R1 + R3) for the next pixel and store
6   STR R2, [R4]
7   ADD R3, R3, #4
8   CMP R3, #336
9   BLT loop
10 HALT
```

Processor

PC: 0x00000024
LR: 0x00000000
SP: 0x00100000
R12: 0x00000000
R11: 0x00000000
R10: 0x00000000
R9: 0x00000000
R8: 0x00000000
R7: 0x00000000
R6: 0x00000000
R5: 0x00000000
R4: 0xffff314c
R3: 0x00000150
R2: 0x00ff0000
R1: 0xffff3000
R0: 0x00000000

Count: 184
Current Instruction:
Status bits: NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Memory

000	0x0	0x4	0x8	0xc
0x0000	0x32d1000	0x3a028ff	0x3a03c01	0xe0814003
0x0001	0x5842000	0xe2833004	0x3530e15	0xbafffffa
0x0002	0x1000070	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](#)

(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, #256
4  MOV R5, #128
5  loop:
6      ADD R6, R1, R5
7      STR R2, [R6]
8      ADD R5, R5, #256
9      CIP R5, #5248
10     BLT loop
11 loop1:
12     ADD R4, R1, R3
13     STR R2, [R4]
14     ADD R3, R3, #4
15     CIP 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: 0xffff4380
R5: 0x00001480
R4: 0xffff314c
R3: 0x00000150
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	0xe3a03c01	0xe3a05080
0x0001	0xe0816005	0xe5862000	0xe2855c01	0xe3550d52
0x0002	0xbaffffff	0xe0814003	0xe5842000	0xe2833004
0x0003	0xe3530e15	0xbaffffff	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](#)

Excercise 9.2.1

Program

```

1  MOV R1, #.PixelScreen
2  MOV R2, #.red
3  MOV R3, #0
4  loop: ADD R4, R1, R3
5      STR R2, [R1 + R3]
6      ADD R3, R3, #4
7      CIP R3, #80
8      BLT loop
9      HALT

```

Load
Save
Edit

Processor

PC: 0x00000024
LR: 0x00000000
SP: 0x00100000
R12: 0x00000000
R11: 0x00000000
R10: 0x00000000
R9: 0x00000000
R8: 0x00000000
R7: 0x00000000
R6: 0x00000000
R5: 0x00000000
R4: 0xffff304c
R3: 0x00000050
R2: 0x00ff0000
R1: 0xffff3000
R0: 0x00000000

Count: 104
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	0xe7812003	0xe2833004	0xe3530050	0xbaffffff
0x0002	0xe1000070	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](#)

Excercise 9.2.2

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

```

1 | MOV R1, #PixelScreen
2 | MOV R2, #red
3 | MOV R3, #0
4 | MOV R6, #80
5 | loop1:
6 |   ADD R4, R1, R3
7 |   STR R2, [R4]
8 |   ADD R3, R3, #4
9 |   CMP R3, R6
10 |  BLT loop1
11 |  BEQ loop2
12 | loop2:
13 |   ADD R3, R3, #176
14 |   ADD R6, R6, #256
15 |   CMP R6, #2384
16 |   BLT loop1
   HALT

```

The 'Processor' window shows the following registers and status bits:

Register	Value
PC	60
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	2384
R5	0
R4	4294916172
R3	2304
R2	16711680
R1	4294914048
R0	0

The 'Memory' window shows a memory dump with the following values:

Address	Value
0x0000	3811381248
0x0001	3766566915
0x0002	3137339386
0x0003	3814067861
0x0004	0
0x0005	0
0x0006	0
0x0007	0
0x0008	0
0x0009	0
0x000a	0
0x000b	0
0x000c	0
0x000d	0
0x000e	0
0x000f	0
0x0010	0
0x0011	0
0x0012	0
0x0013	0
0x0014	0
0x0015	0
0x0016	0
0x0017	0
0x0018	0
0x0019	0
0x001a	0
0x001b	0
0x001c	0
0x001d	0
0x001e	0
0x001f	0

The 'Input/Output' window shows the program status: 'Program HALTED. STOP, LOAD or EDIT'.

Exercise 9.3.1 (a)

The above code defines an array of 10 32 bit integers. What is the purpose of the `.Align 256` instruction ?

- The `.Align 256` instruction is used to align the starting address of `arrayLength` in memory to ensure that the label is located at an address that is divisible by 256.

Exercise 9.3.1 (b)

Add a line of code to the above to read the 5th value of the array to register R0 (i.e., it should use indirect addressing to access the 5th cell in the array)

Program

```

1| MOV R2, #arrayData
2| MOV R1, #16
3| LDR R0, [R2 + 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

```

Load
Save
Edit

Processor

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

▶

⏏

⏮

⏭

⚙

Count

Current Instruction

Status bits N Z C V 0 0 0 0

Input/Output

Program HALTED. STOP, LOAD or EDIT

Memory

000	0x0	0x4	0x8	0xc
0x0000	0xe3a02f41	0xe3a01010	0xe7920001	0xe1800070
0x0001	0x00000000	0x00000000	0x00000000	0x00000000
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	0x0000000a	0x00000009	0x00000008	0x00000007
0x0011	0x00000006	0x00000005	0x00000004	0x00000003
0x0012	0x00000002	0x00000001	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

ARM Lite Simulator V1.2.4 © Peter Higginson 2020-23
[Documentation](#)

MOV R2, #arrayData

MOV R1, #16

LDR R0, [R2 + R1]

HALT

.ALIGN256

arrayLength: 10

arrayData: 9

8

7

6

5

4

3

2

1

0

Exercise 9.3.1 (c)

Now modify your code so that the index to read from in the array is provided in R1.

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

```
1 | MOV R2, #arrayData
2 | MOV R0, #16
3 | LDR R1, [R2 + 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
```

The **Processor** window shows the PC at 0x00000010, LR at 0x00000000, SP at 0x00100000, and registers R12 through R0. The **Memory** window shows the array data starting at 0x00000000, with the value 9 at address 0x00000000. The **Input/Output** window shows the program is HALTED.

Exercise 9.3.2

Now modify your code so that it adds up all the values in the array. Your program should use indexed addressing to access each value and write the result to R0.

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

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

The **Processor** window shows the PC at 0x00000010, LR at 0x00000000, SP at 0x00100000, and registers R12 through R0. The **Memory** window shows the array data starting at 0x00000000, with the value 9 at address 0x00000000. The **Input/Output** window shows the program is HALTED.

Exercise 9.3.3

Program

```

1  MOV R2, #arrayData
2  MOV R1, #0 // INDEX
3  MOV R0, #0 // SUM
4  MOV R4, #0
5  MOV R7, #arrayLength
6  LDR R6, arrayLength
7 arrayLoop:
8  LDR R3, [R2 + R1]
9  ADD R0, R0, R3
10 ADD R1, R1, #4
11 ADD R4, R4, #1
12 CMP R4, R6
13 BLT arrayLoop
14 HALT
15 .ALIGN 256
16 arrayLength: 10
17 arrayData: 9
18 8
19 7
20 6
21 5
22 4
23 3
24 2
25 1
26 0

```

Load Save Edit

Processor

PC: 0x00000034
LR: 0x00000000
SP: 0x00100000
R12: 0x00000000
R11: 0x00000000
R10: 0x00000000
R9: 0x00000000
R8: 0x00000000
R7: 0x00000100
R6: 0x0000000a
R5: 0x00000000
R4: 0x0000000a
R3: 0x00000000
R2: 0x00000104
R1: 0x00000028
R0: 0x00000028

Count: 67
Current Instruction:
Status bits: NZCV 0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Memory

000	0x0	0x4	0x8	0xc
0x0000	0xe3a02f41	0xe3a01000	0xe3a00000	0xe3a04000
0x0001	0xe3a07c01	0xe59f60e4	0xe7923001	0xe0800003
0x0002	0xe2811004	0xe2844001	0xe1540006	0xbaffffff
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	0x0000000a	0x00000009	0x00000008	0x00000007
0x0011	0x00000006	0x00000005	0x00000004	0x00000003
0x0012	0x00000002	0x00000001	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](#)

Exercise 9.4.1

Using the original array definition given in Part 9.3, write an ARMLite program that copies all the values from this array into another array of equal size (in reverse order).

Program

```

1  MOV R2, #arrayData
2  MOV R8, #arrayData2
3  MOV R1, #0 // INDEX
4  MOV R0, #0 // SUM
5  MOV R4, #1
6  MOV R9, #0
7  MOV R5, #0
8  MOV R7, #arrayLength
9  LDR R6, arrayLength
10 arrayLoop:
11 ADD R1, R1, #4
12 ADD R4, R4, #1
13 CMP R4, R6
14 BLT arrayLoop
15 loop:
16 LDR R3, [R2 + R0]
17 STR R3, [R8 + R1]
18 SUB R1, R1, #4
19 ADD R0, R0, #4
20 ADD R5, R5, #1
21 CMP R5, R6
22 BLT loop
23 HALT
24 .ALIGN 256
25 arrayLength: 10
26 arrayData: 9
27 8
28 7
29 6
30 5
31 4
32 3
33 2
34 1
35 0
36 arrayData2: 0
37 0
38 0
39 0

```

Load Save Edit

Processor

PC: 84
LR: 0
SP: 1048576
R12: 0
R11: 0
R10: 0
R9: 0
R8: 380
R7: 256
R6: 10
R5: 10
R4: 10
R3: 0
R2: 260
R1: 4294967292
R0: 40

Count: 116
Current Instruction:
Status bits: NZCV 0110

Input/Output

Saving File

Memory

000	0x0	0x4	0x8	0xc
0x0000	3818925889	3818950475	3818917888	381891379
0x0001	3818930177	3818950656	3818914272	381894553
0x0002	3852427480	3800109960	3800317953	3788378630
0x0003	3137339387	3885117440	3884462081	3795914756
0x0004	3800039428	3800387585	3780444166	3137339384
0x0005	3774873712	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	10	9	8	7
0x0011	6	5	4	3
0x0012	2	1	0	0
0x0013	1	2	3	4
0x0014	5	6	7	8
0x0015	9	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Decimal (unsigned) Clear

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
[Documentation](#)

Exercise 9.4.2

Program

```
1|  MOV R2, #arrayData
2|  MOV R1, #0      // INDEX
3|  MOV R0, #0
4|  MOV R4, #1
5|  MOV R5, #0
6|  LDR R6, arrayLength
7| arrayLoop:
8|  ADD R1, R1, #4
9|  ADD R4, R4, #1
10| CMP R4, R6
11| BLT arrayLoop
12| loop:
13|  LDR R3, [R2 + R1]
14|  LDR R8, [R2 + R0]
15|  STR R3, [R2 + R0]
16|  STR R8, [R2 + R1]
17|  SUB R1, R1, #4
18|  ADD R0, R0, #4
19|  ADD R5, R5, #2
20|  CMP R5, R6
21|  BLT loop
22|  HALT
23|  .ALIGN 256
24| arrayLength: 10
25| arrayData: 9
26|  8
27|  7
28|  6
29|  5
30|  4
31|  3
32|  2
33|  1
34|  0
```

Load Save Edit

Processor

PC 24

LR 0

SP 1048576

R12 0

R11 0

R10 0

R9 0

R8 0

R7 0

R6 10

R5 0

R4 2

R3 0

R2 260

R1 4

R0 0

Count 10

Current Instruction BLT addr

Status bits NZCV 1000

Input/Output

Done instruction BLT addr at line 11

Memory

	0x0	0x4	0x8	0xc
0x0000	3818925889	3818917888	3818913792	381893017
0x0001	3818934272	3852427492	3800109060	380031795
0x0002	3780378630	3137339388	3885117441	3885137920
0x0003	3884068864	3884089345	3795914756	3800039428
0x0004	3800387586	3780444166	3137339382	3774873712
0x0005	0	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	10	9	8	7
0x0011	6	5	4	3
0x0012	2	1	0	0
0x0013	0	0	0	0
0x0014	0	0	0	0
0x0015	0	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Decimal (unsigned) Clear

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23
[Documentation](#)