

# Lab09

Nguyen Nam Tung 103181157

## 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 assembly editor interface. The main window is titled "Program" and 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:
25| STR R1, [R2 + R3]
26| ADD R2, R2, #4
27| CMP R2, #80
28| BLT LOOP
29| HALT
```

The "Processor" panel on the right shows the following registers and values:

Register	Value
PC	0x00000070
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0xfffff380
R2	0x00000050
R1	0x00ff0000
R0	0x00000000

The "Input/Output" panel shows the status: "Program HALTED. STOP, LOAD or EDIT".

- 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:
25|  STR R1, [R2 + R3]
26|  ADD R2, R2, #4
27|  CMP R2, #80
28|  BLT LOOP
29|  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

```

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

```

PC	0x00000088
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0xffffffff400
R4	0x000000880
R3	0xffffffff380
R2	0x000000050
R1	0x00ff0000
R0	0x00000000

▶ || □

⏮ ▶ ⚙

Count

Current Instruction

Status bits **NZCV**  
**0110**

Input/Output

Program HALTED. STOP, LOAD or EDIT

## Program

```

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:
25|  STR R1, [R2 + R3]
26|  ADD R2, R2, #4
27|  CMP R2, #80
28|  BLT LOOP
29|  MOV R4, #0
30|  MOV R5, #.Pixel64
31| LOOP2: STR R1, [R4 + R5]
32|  ADD R4, R4, #128
33|  CMP R4, #2176
34|  BLT LOOP2
35|  HALT
```

### 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 ?

Line STR R2 [R4]

This will store the content of the memory of R4 into the memory of R2. It will use indirect addressing to draw each pixel.

(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 shows a program editor window titled "Program" with assembly code. The code is as follows:

```
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
```

The "Processor" panel on the right shows the following registers and values:

Register	Value
PC	0x0000003c
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0xfffff314c
R3	0x00000150
R2	0x00ff0000
R1	0xfffff3000
R0	0x00000000

Control buttons: Play, Pause, Stop, Step Back, Step Forward, Settings.

Count: 131474

Current Instruction: [Empty]

Status bits: NZCV 0110

Input/Output panel: Program HALTED. STOP, LOAD or EDIT

Buttons: Load, Save, Edit

```
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

```

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

The screenshot shows a MIPS assembly simulator interface. The main window is titled "Program" and contains the following assembly code:

```

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

```

The line numbers 1 through 24 are on the left, and the instructions are on the right. Line 24 is highlighted in orange. Below the code are buttons for "Load", "Save", and "Edit".

On the right side, there is a "Processor" panel showing the following registers and values:

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

Below the registers are control buttons: a play button, a pause button, a stop button, a step back button, a step forward button, and a settings gear icon. There are also fields for "Count" (296), "Current Instruction", and "Status bits" (NZCV 0110).

Below the processor panel is an "Input/Output" panel with the text "Program HALTED. STOP, LOAD or EDIT". Below this is a large white area representing the display, with a red L-shaped line drawn in the top-left corner.

```

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
```

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	0x00000000
R1	0xffff3000
R0	0x00000000

▶ || ◻

◀ ▶ ⚙

Count 

84

Current Instruction

Status bits **NZCV**  
**0110**

### Input/Output

Program HALTED. STOP, LOAD or EDIT

```

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

```

9.2.2.



### Program

```

1|    MOV R1, #.PixelScreen
2|    MOV R2, #.red
3|    MOV R3, #0
4|    MOV R4, #80
5| LOOP:
6|    STR R2, [R1 + R3]
7|    ADD R3, R3, #4
8|    CMP R3, R4
9|    BLT LOOP
10|   ADD R4, R4, #256
11|   ADD R3, R3, #176
12|   CMP R3, #2560
13|   BLT LOOP
14|   HALT

```

Load

Save

Edit

### Processor

PC	0x00000034
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000a50
R3	0x00000a00
R2	0x00ff0000
R1	0xffff3000
R0	0x00000000

▶ || ◻

◀ ▶ ⚙

Count 

845

Current Instruction

Status bits 

NZCV 0110

### Input/Output

Program HALTED. STOP, LOAD or EDIT

```

1|    MOV R1, #.PixelScreen
2|    MOV R2, #.red
3|    MOV R3, #0
4|    MOV R4, #80
5| LOOP:
6|    STR R2, [R1 + R3]
7|    ADD R3, R3, #4
8|    CMP R3, R4
9|    BLT LOOP
10|   ADD R4, R4, #256
11|   ADD R3, R3, #176
12|   CMP R3, #2560
13|   BLT LOOP
14|   HALT

```

9.3.1.

a)

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

b)

The image shows a screenshot of an ARM assembly simulator interface. The main window is titled "Program" and displays the following assembly code:

```
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
```

The line 4, containing the instruction `HALT`, is highlighted in orange. Below the code, there are three buttons: "Load", "Save", and "Edit".

On the right side, there is a "Processor" panel. It displays the following registers and their values:

Register	Value
PC	0x00000010
LR	0x000ffffc
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

Below the registers, there are control buttons: a play button (run), a pause button, and a stop button. There are also buttons for single step (left arrow) and step over (right arrow), and a settings gear icon.

Below the control buttons, there are two input fields:

- Count: 131073
- Current Instruction: (empty)

Below these, there are status bits: NZCV 0000.

At the bottom right, there is an "Input/Output" panel. It contains a text box with the message: "Program HALTED. STOP, LOAD or EDIT". Below this text box, there are two empty input fields.

c)

### Program

```

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

```

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	0x00000104
R3	0x00000000
R2	0x00000000
R1	0x00000005
R0	0x00000010

▶ || ◻

⏮ ▶ ⚙

Count

Current

Instruction

Status bits

### Input/Output

Program HALTED. STOP, LOAD or EDIT

9.3.3.

### Program

```

1|  MOV R4, #arrayData
2|  MOV R1, #0
3|  MOV R2, #0
4|  MOV R6, #0
5|  LDR R3, arrayLength
6|  LOOP:
7|      LDR R5, [R4 + R1]
8|      ADD R2, R2, R5
9|      ADD R1, R1, #4
10|     ADD R6, R6, #1
11|     CMP R6, R3
12|     BLT LOOP
13|     MOV R0, #0
14|     ADD R0, R0, R2
15|     HALT
16|     .ALIGN 256
17|     arrayLength: 10
18|     arrayData: 9
19|         8
20|         7
21|         6
22|         5
23|         4
24|         3
25|         2
26|         1
27|         0

```

Load
Save
Edit

### Processor

PC	0x00000038
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x0000000a
R5	0x00000000
R4	0x00000104
R3	0x0000000a
R2	0x0000002d
R1	0x00000028
R0	0x0000002d

▶ || ◻

⏮ ▶ ⚙

Count

Current

Instruction

Status bits

### Input/Output

Program HALTED. STOP, LOAD or EDIT

1| MOV R4, #arrayData

```
2|    MOV R1, #0
3|    MOV R2, #0
4|    MOV R6, #0
5|    LDR R3, arrayLength
6|LOOP:
7|    LDR R5, [R4 + R1]
8|    ADD R2, R2, R5
9|    ADD R1, R1, #4
10|   ADD R6, R6, #1
11|   CMP R6, R3
12|   BLT LOOP
13|   MOV R0, #0
14|   ADD R0, R0, R2
15|   HALT
16|   .ALIGN 256
17|arrayLength: 10
18|arrayData: 9
19|    8
20|    7
21|    6
22|    5
23|    4
24|    3
25|    2
26|    1
27|    0
```

9.4.1.

### Program

```

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

```

Load

Save

Edit

### Processor

PC	56
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	10
R8	0
R7	0
R6	9
R5	300
R4	260
R3	40
R2	10
R1	-4
R0	0

▶ || ◻

⏮ ▶ ⚙

Count 

77

Current Instruction

Status bits 

NZCV 0110

### Input/Output

Program HALTED. STOP, LOAD or EDIT

```

1|    MOV R4, #arrayData
2|    MOV R1, #36
3|    MOV R2, #0
4|    MOV R3, #0
5|    LDR R9, arrayLength
6|    MOV R5, #newArrData
7| LOOP:
8|    LDR R6, [R4 + R1]
9|    STR R6, [R5 + R3]
10|   ADD R3, R3, #4
11|   SUB R1, R1, #4
12|   ADD R2, R2, #1
13|   CMP R2, R9
14|   BLT LOOP
15|   HALT

```

```

16| .ALIGN 256
17| arrayLength: 10
18| arrayData: 9
19| 8
20| 7
21| 6
22| 5
23| 4
24| 3
25| 2
26| 1
27| 0
28| newArrData: .BLOCK 256

```

#### 9.4.2.

<pre> 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 </pre>	<table border="1"> <tr> <td>PC</td> <td>0x00000044</td> </tr> <tr> <td>LR</td> <td>0x00000000</td> </tr> <tr> <td>SP</td> <td>0x00100000</td> </tr> <tr> <td>R12</td> <td>0x00000000</td> </tr> <tr> <td>R11</td> <td>0x00000000</td> </tr> <tr> <td>R10</td> <td>0x00000000</td> </tr> <tr> <td>R9</td> <td>0x00000005</td> </tr> <tr> <td>R8</td> <td>0x00000003</td> </tr> <tr> <td>R7</td> <td>0x0000000c</td> </tr> <tr> <td>R6</td> <td>0x00000004</td> </tr> <tr> <td>R5</td> <td>0x00000005</td> </tr> <tr> <td>R4</td> <td>0x00000104</td> </tr> <tr> <td>R3</td> <td>0x00000014</td> </tr> <tr> <td>R2</td> <td>0x00000005</td> </tr> <tr> <td>R1</td> <td>0x00000010</td> </tr> <tr> <td>R0</td> <td>0x00000000</td> </tr> </table> <div style="margin-top: 10px;"> <div> </div> <div> </div> </div> <div style="margin-top: 10px;"> <div>Count <span style="border: 1px solid black; padding: 2px 10px;">53</span></div> <div>Current Instruction <span style="border: 1px solid black; padding: 2px 10px;"></span></div> <div>Status bits <span style="border: 1px solid black; padding: 2px 10px;">NZCV 0110</span></div> </div> <div style="margin-top: 10px; text-align: center;"> <h3>Input/Output</h3> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Program HALTED. STOP, LOAD or EDIT </div> <div style="border: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border: 1px solid black; height: 20px;"></div> </div>	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
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																																