

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.

```
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
```

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	0xffffffff380
R2	0x00000050
R1	0x00ff0000
R0	0x00000000

▶

⏏

⏏

⏮

⏭

⚙

Count

104

Current Instruction

Status bits

NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

- (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, #.Pixel164
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	0x00000880
R3	0xffffffff380
R2	0x00000050
R1	0x00ff0000
R0	0x00000000

▶

⏏

⏏

⏮

⏭

⚙

Count

174

Current Instruction

Status bits

NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

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

```

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

```

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	0x00000150
R2	0x00ff0000
R1	0xffff3000
R0	0x00000000

Count

205

Current Instruction







Status bits

NZCV
0110

Input/Output

Breakpoint removed at line 15 address 0x00030

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

1	MOV R1, #.PixelScreen	PC	0x00000054	     
2	MOV R2, #.red	LR	0x00000000	
3	MOV R3, #0	SP	0x00100000	
4	loop:	R12	0x00000000	
5	ADD R4, R1, R3	R11	0x00000000	
6	STR R2, [R4]	R10	0x00000000	
7	ADD R3, R3, #4	R9	0x00000000	
8	CMP R3, #80	R8	0x00000000	
9	BLT loop	R7	0x00000000	
10	ADD R3, R3, #176	R6	0x00000000	
11	loop1:	R5	0x00000000	
12	ADD R4, R1, R3	R4	0xfffff4300	
13	STR R2, [R4]	R3	0x00001400	
14	ADD R3, R3, #4	R2	0x00ff0000	
15	CMP R3, #336	R1	0xfffff3000	
16	BLT loop1	R0	0x00000000	
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			

Count 296







Current Instruction

Status bits NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT

Exercise 9.2.1

1	MOV R1, #.PixelScreen	PC	0x00000020	     
2	MOV R2, #.red	LR	0x00000000	
3	MOV R3, #0	SP	0x00100000	
4	loop:	R12	0x00000000	
5	STR R2, [R1+R3]	R11	0x00000000	
6	ADD R3,R3,#4	R10	0x00000000	
7	CMP R3,#80	R9	0x00000000	
8	BLT loop	R8	0x00000000	
9	HALT	R7	0x00000000	
		R6	0x00000000	
		R5	0x00000000	
		R4	0x00000000	
		R3	0x00000050	
		R2	0x00ff0000	
		R1	0xffff3000	
		R0	0x00000000	

Count

Current Instruction

Status bits

N	Z	C	V
0	1	1	0

Input/Output

Program HALTED. STOP, LOAD or EDIT

Excercise 9.2.2

```

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

```

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 e1000070
HALT

Status bits NZCV
0110

Input/Output

Program HALTED. STOP, LOAD or EDIT




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




Current Instruction




Status bits NZCV

Exercise 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

Current Instruction

Status bits NZCV




Exercise 9.3.2




1	MOV R4, #arrayData	PC	0x00000030
2	MOV R0, #0	LR	0x00000000
3	MOV R1, #0	SP	0x00100000
4	MOV R2, #10	R12	0x00000000
5	MOV R3, #0	R11	0x00000000
6	loop:	R10	0x00000000
7	LDR R5, [R4+R1]	R9	0x00000000
8	ADD R0, R0, R5	R8	0x00000000
9	ADD R3, R3, #1	R7	0x00000000
10	ADD R1, R1, #4	R6	0x00000000
11	CMP R3, R2	R5	0x00000000
12	BLT loop	R4	0x00000104
13	HALT	R3	0x0000000a
14	.ALIGN 256	R2	0x0000000a
15	arrayLength: 10	R1	0x00000028
16	arrayData: 9	R0	0x0000002d
17	8	In	
18	7		
19	6		
20	5		
21	4		
22	3		
23	2		
24	1		
25	0		
		Program HALTED.	

Excercise 9.3.3

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

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 NZCV




Input/Output




Program HALTED. STOP, LOAD or EDIT

Exercise 9.4.1

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

PC	0x00000038
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x0000000a
R8	0x00000000
R7	0x00000000
R6	0x00000009
R5	0x0000012c
R4	0x00000104
R3	0x00000028
R2	0x0000000a
R1	0xffffffffc
R0	0x00000000

Count

Current Instruction

Status bits NZCV

Input/Output

Program HALTED. STOP, LOAD or EDIT

Exercise 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