

Lab 09 - Graphic & Arrays

9.1.1

(a)

<pre> 1 MOV R1, #.red 2 STR R1, .Pixel32 3 STR R1, .Pixel33 4 STR R1, .Pixel34 5 STR R1, .Pixel35 6 STR R1, .Pixel36 7 STR R1, .Pixel37 8 STR R1, .Pixel38 9 STR R1, .Pixel39 10 STR R1, .Pixel40 11 STR R1, .Pixel41 12 STR R1, .Pixel42 13 STR R1, .Pixel43 14 STR R1, .Pixel44 15 STR R1, .Pixel45 16 STR R1, .Pixel46 17 STR R1, .Pixel47 18 STR R1, .Pixel48 19 STR R1, .Pixel49 20 STR R1, .Pixel50 21 STR R1, .Pixel51 22 STR R1, .Pixel52 23 STR R1, .Pixel53 24 STR R1, .Pixel54 25 STR R1, .Pixel55 26 STR R1, .Pixel56 27 STR R1, .Pixel57 28 STR R1, .Pixel58 29 STR R1, .Pixel59 30 STR R1, .Pixel60 31 STR R1, .Pixel61 32 STR R1, .Pixel62 33 STR R1, .Pixel63 34 HALT </pre>	<p>The screenshot shows a program execution environment. On the left, a list of assembly instructions is displayed, with line 34 (HALT) highlighted in orange. On the right, the 'Processor' section shows register values (PC, LR, SP, R12-R0) and control bits (Count, Current, Instruction, Status bits). Below this, the 'Input/Output' section displays the message 'Program HALTED. STOP, LOAD or EDIT'. At the bottom, there are buttons for 'Load', 'Save', and 'Edit'.</p>
--	---

(b)

```

1| MOV R1, #.red
2| STR R1, .Pixel15
3| STR R1, .Pixel47
4| STR R1, .Pixel79
5| STR R1, .Pixel111
6| STR R1, .Pixel143
7| STR R1, .Pixel175
8| STR R1, .Pixel207
9| STR R1, .Pixel239
10| STR R1, .Pixel271
11| STR R1, .Pixel303
12| STR R1, .Pixel335
13| STR R1, .Pixel367
14| STR R1, .Pixel399
15| STR R1, .Pixel431
16| STR R1, .Pixel463
17| STR R1, .Pixel495
18| STR R1, .Pixel527
19| STR R1, .Pixel559
20| STR R1, .Pixel591
21| STR R1, .Pixel623
22| STR R1, .Pixel655
23| STR R1, .Pixel687
24| STR R1, .Pixel719
25| STR R1, .Pixel751
26| HALT

```

Program

```

1| MOV R1, #.red
2| STR R1, .Pixel15
3| STR R1, .Pixel47
4| STR R1, .Pixel79
5| STR R1, .Pixel111
6| STR R1, .Pixel143
7| STR R1, .Pixel175
8| STR R1, .Pixel207
9| STR R1, .Pixel239
10| STR R1, .Pixel271
11| STR R1, .Pixel303
12| STR R1, .Pixel335
13| STR R1, .Pixel367
14| STR R1, .Pixel399
15| STR R1, .Pixel431
16| STR R1, .Pixel463
17| STR R1, .Pixel495
18| STR R1, .Pixel527
19| STR R1, .Pixel559
20| STR R1, .Pixel591
21| STR R1, .Pixel623
22| STR R1, .Pixel655
23| STR R1, .Pixel687
24| STR R1, .Pixel719
25| STR R1, .Pixel751
26| HALT

```

Processor

PC	0x00000068
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	0x00000000
R1	0x00ff0000
R0	0x00000000

Count: 26

Current Instruction: #1000070
HALT

Status bits: NZCV
0000

Input/Output

Modifying Program Area

Load Save Submit Revert

9.1.2/ 20 pixels line

Program

```

1| MOV R1, #.PixelScreen // base a
2| MOV R2, #.red
3| MOV R3, #0
4| loop:
5| ADD R4, R1, R3 // calculate
6| STR R2, [R4]
7| ADD R3, R3, #4
8| CMP R3, #80
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	0xffff304c
R3	0x00000050
R2	0x00ff0000
R1	0xffff3000
R0	0x00000000

Count: 104

Current Instruction

Status bits: NZCV
0110

Input/Output

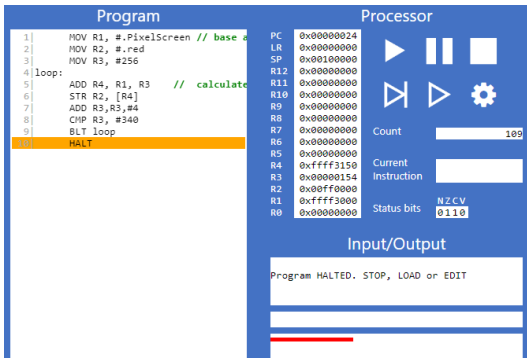
Program HALTED. STOP, LOAD or EDIT

9.1.3/

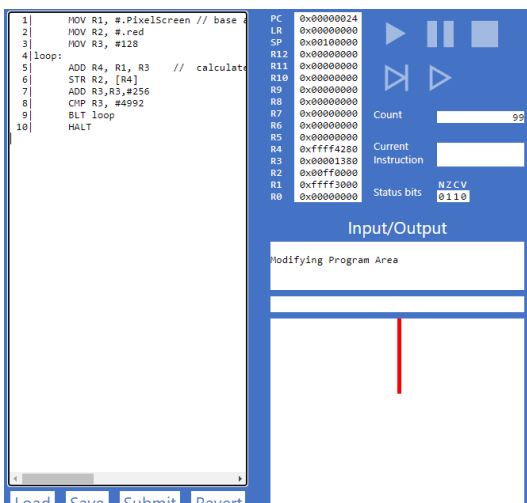
- (a) Explain what specifically makes this code an example of indirect addressing?
How is it using indirect addressing to draw each pixel?

The code above indirectly addresses this because it uses a base address stored in register R1 to access the memory location where pixel values are stored. Instead of directly specifying the memory address for each pixel, the code calculates the address for each pixel via indirect addressing.

- (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

<pre> 1 MOV R1, #.PixelScreen 2 MOV R2, #.red 3 MOV R3, #256 4 loop: 5 ADD R4, R1, R3 6 STR R2, [R4] 7 ADD R3, R3, #4 8 CMP R3, #340 9 BLT loop 10 HALT </pre>	
--	--

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

<pre> 1 MOV R1, #.PixelScreen 2 MOV R2, #.red 3 MOV R3, #128 4 loop: 5 ADD R4, R1, R3 6 STR R2, [R4] 7 ADD R3, R3, #256 8 CMP R3, #4992 9 BLT loop 10 HALT </pre>	
---	--

9.2.1/

```

1|  MOV R1, #.PixelScreen
2|  MOV R2, #.red
3|  MOV R3, #0
4|  MOV R5, #0
5|loop:
6|  ADD R4, R1, R3
7|  STR R2, [R4 + R5]
8|  ADD R3, R3, #4
9|  CMP R3, #80
10|  BLT loop
11|  HALT

```

The screenshot shows a debugger window with assembly code on the left and processor status on the right. The assembly code is as follows:

```

1|  MOV R1, #.PixelScreen
2|  MOV R2, #.red
3|  MOV R3, #0
4|  MOV R5, #0
5|loop:
6|  ADD R4, R1, R3
7|  STR R2, [R4 + R5]
8|  ADD R3, R3, #4
9|  CMP R3, #80
10| BLT loop
11|  HALT

```

The processor status on the right shows:

- PC: 0x00000028
- 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: 105

Current Instruction: [Empty]

Status bits: NZCV 0110

Input/Output: [Empty]

Saving File: [Empty]

Buttons: Load, Save, Edit

9.2.2/

```

1|  MOV R1, #.PixelScreen
2|  MOV R2, #.red
3|  MOV R3, #0
4|  MOV R5, #0
5|loop1:
6|  ADD R4, R1, R3
7|  STR R2, [R4+R5]
8|  ADD R3, R3, #4
9|  CMP R3, #80
10|  BLT loop1
11|  MOV R3, #0
12|  MOV R3, #256
13|loop2:
14|  ADD R4, R1, R3
15|  STR R2, [R4+R5]
16|  ADD R3, R3, #4
17|  CMP R3, #336
18|  BLT loop2
19|  MOV R3, #0
20|  MOV R3, #512
21|loop3:
22|  ADD R4, R1, R3
23|  STR R2, [R4+R5]
24|  ADD R3, R3, #4
25|  CMP R3, #592
26|  BLT loop3
27|  MOV R3, #0
28|  MOV R3, #768
29|loop4:
30|  ADD R4, R1, R3
31|  STR R2, [R4+R5]
32|  ADD R3, R3, #4
33|  CMP R3, #848

```

The screenshot shows a debugger window with assembly code on the left and processor status on the right. The assembly code is as follows:

```

44|  MOV R3, #1280
45|loop6:
46|  ADD R4, R1, R3
47|  STR R2, [R4+R5]
48|  ADD R3, R3, #4
49|  CMP R3, #1360
50|  BLT loop6
51|  MOV R3, #0
52|  MOV R3, #1536
53|loop7:
54|  ADD R4, R1, R3
55|  STR R2, [R4+R5]
56|  ADD R3, R3, #4
57|  CMP R3, #1616
58|  BLT loop7
59|  MOV R3, #0
60|  MOV R3, #1792
61|loop8:
62|  ADD R4, R1, R3
63|  STR R2, [R4+R5]
64|  ADD R3, R3, #4
65|  CMP R3, #1872
66|  BLT loop8
67|  MOV R3, #0
68|  MOV R3, #2048
69|loop9:
70|  ADD R4, R1, R3
71|  STR R2, [R4+R5]
72|  ADD R3, R3, #4
73|  CMP R3, #2128
74|  BLT loop9
75|  MOV R3, #0
76|  MOV R3, #2304
77|loop10:
78|  ADD R4, R1, R3
79|  STR R2, [R4+R5]
80|  ADD R3, R3, #4
81|  CMP R3, #2384
82|  BLT loop10
83|  HALT

```

The processor status on the right shows:

- PC: 0x00000124
- 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: 1023

Current Instruction: [Empty]

Status bits: NZCV 0110

Input/Output: [Empty]

Program HALTED. STOP, LOAD or EDIT

Buttons: Load, Save, Edit

34	BLT loop4	
35	MOV R3,#0	
36	MOV R3,#1024	
37	loop5:	
38	ADD R4,R1,R3	
39	STR R2,[R4+R5]	
40	ADD R3,R3,#4	
41	CMP R3,#1104	
42	BLT loop5	
43	MOV R3,#0	
44	MOV R3,#1280	
45	loop6:	
46	ADD R4,R1,R3	
47	STR R2,[R4+R5]	
48	ADD R3,R3,#4	
49	CMP R3,#1360	
50	BLT loop6	
51	MOV R3,#0	
52	MOV R3,#1536	
53	loop7:	
54	ADD R4,R1,R3	
55	STR R2,[R4+R5]	
56	ADD R3,R3,#4	
57	CMP R3,#1616	
58	BLT loop7	
59	MOV R3,#0	
60	MOV R3,#1792	
61	loop8:	
62	ADD R4,R1,R3	
63	STR R2,[R4+R5]	
64	ADD R3,R3,#4	
65	CMP R3,#1872	
66	BLT loop8	
67	MOV R3,#0	
68	MOV R3,#2048	
69	loop9:	
70	ADD R4,R1,R3	
71	STR R2,[R4+R5]	
72	ADD R3,R3,#4	
73	CMP R3,#2128	
74	BLT loop9	
75	MOV R3,#0	
76	MOV R3,#2304	
77	loop10:	
78	ADD R4,R1,R3	
79	STR R2,[R4+R5]	
80	ADD R3,R3,#4	
81	CMP R3,#2384	
82	BLT loop10	
83	HALT	

9.3.1/

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

Note: if you try and execute this program as it is above, it will cause an error on Line 2 (Unknown Instruction). Why is this ? (HINT: think about what is in memory at that point of the program)

The purpose of `Align256` instruction is to aligns the memory location to a multiple of 256 bytes. It ensure that the following data or code is placed at an address that is a multiple of 256. I would assume the problem with line 2 is that the `arrayLength` is not a multiple of 256 hence causing the error

- (b) Add lines 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)

The screenshot shows a debugger interface with three main panels: Program, Processor, and Memory.

- Program Panel:** Displays assembly code. Line 6, `.ALIGN 256`, is highlighted. The code includes:


```

1| MOV R1,#arrayData
2| MOV R0,#16 // index
3| ADD R1,R1,R0
4| LDR R0,[R1]
5| HALT
6| .ALIGN 256
7| arrayLength: 10
8| arrayData: 9
9| 8
10| 7
11| 6
12| 5
13| 4
14| 3
15| 2
16| 1
17| 0
      
```
- Processor Panel:** Shows the state of registers and processor flags.
 - Registers: R0 through R15 are listed with their current values. R0 contains `0x00000005`.
 - Count: A slider is set to 6.
 - Current Instruction: The instruction being executed is `.ALIGN 256`.
 - Status bits: NZCV are shown as `0000`.
- Memory Panel:** Shows a memory dump starting at address `0x00000000`. The dump shows various memory locations and their contents, with some values highlighted in yellow.

```

1| MOV R1,#arrayData
2| MOV R0,#16 // index
3| ADD R1,R1,R0
4| LDR R0,[R1]
5| HALT
6| .ALIGN 256
7| arrayLength: 10
8| arrayData: 9
9| 8
10| 7
11| 6
12| 5
13| 4
14| 3
15| 2
16| 1
17| 0
      
```

9.3.2/

Program

```

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

```

Processor

PC: 0x00000028
 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: 0x00000024
 R1: 0x00000100
 R0: 0x00000104

Count: 320
 Current Instruction:
 Status bits: NZCV 0110

Input/Output

36
 Program HALTED. STOP, LOAD or EDIT

Memory

000	0x0	0x4	
0x0000	0xe3a00f41	0xe3a01004	0:
0x0001	0xe0822003	0xe2811004	0:
0x0002	0xe50f2114	0xe1000070	0:
0x0003	0x00000000	0x00000000	0:
0x0004	0x00000000	0x00000000	0:
0x0005	0x00000000	0x00000000	0:
0x0006	0x00000000	0x00000000	0:
0x0007	0x00000000	0x00000000	0:
0x0008	0x00000000	0x00000000	0:
0x0009	0x00000000	0x00000000	0:
0x000a	0x00000000	0x00000000	0:
0x000b	0x00000000	0x00000000	0:
0x000c	0x00000000	0x00000000	0:
0x000d	0x00000000	0x00000000	0:
0x000e	0x00000000	0x00000000	0:
0x000f	0x00000000	0x00000000	0:
0x0010	0x0000000a	0x00000009	0:
0x0011	0x00000006	0x00000005	0:
0x0012	0x00000002	0x00000001	0:
0x0013	0x00000000	0x00000000	0:
0x0014	0x00000000	0x00000000	0:
0x0015	0x00000000	0x00000000	0:

Code

```

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

```

9.4.1/

Program

```

1|  MOV R0,#arrayData1
2|  MOV R4,#arrayData2
3|  MOV R1,#36          // index
4| arrayloop:
5|  LDR R3, [R0+R1]      //pointer to
6|  STR R3,[R4]          // display th
7|  STR R3,.WriteUnsignedNum // wr
8|  SUB R1,R1,#4         // subtract t
9|  CMP R1, #0          // R1 - 0
10| BNE arrayloop
11| HALT
12| .ALIGN 256
13| arrayLength1: 10
14| arrayData1: 9
15|  8
16|  7
17|  6
18|  5
19|  4
20|  3
21|  2
22|  1
23|  0
24| arrayLength2:10
25| arrayData2:0

```

Processor

PC: 0x00000028

LR: 0x00000000

SP: 0x00100000

R12: 0x00000000

R11: 0x00000000

R10: 0x00000000

R9: 0x00000000

R8: 0x00000000

R7: 0x00000000

R6: 0x00000000

R5: 0x00000000

R4: 0x00000130

R3: 0x00000008

R2: 0x00000000

R1: 0x00000000

R0: 0x00000104

Count: 58

Current Instruction:

Status bits: NZCV 0100

Input/Output

0 1 2 3 4 5 6 7 8

Program HALTED. STOP, LOAD or EDIT

Memory

000	0x0	0x4
0x0000	0xe3a00f41	0xe3a04e13
0x0001	0xe5843000	0xe50f3108
0x0002	0x1affffff	0xe1000070
0x0003	0x00000000	0x00000000
0x0004	0x00000000	0x00000000
0x0005	0x00000000	0x00000000
0x0006	0x00000000	0x00000000
0x0007	0x00000000	0x00000000
0x0008	0x00000000	0x00000000
0x0009	0x00000000	0x00000000
0x000a	0x00000000	0x00000000
0x000b	0x00000000	0x00000000
0x000c	0x00000000	0x00000000
0x000d	0x00000000	0x00000000
0x000e	0x00000000	0x00000000
0x000f	0x00000000	0x00000000
0x0010	0x0000000a	0x00000009
0x0011	0x00000006	0x00000005
0x0012	0x00000002	0x00000001
0x0013	0x00000008	0x00000000
0x0014	0x00000000	0x00000000
0x0015	0x00000000	0x00000000
0x0016	0x00000000	0x00000000
0x0017	0x00000000	0x00000000
0x0018	0x00000000	0x00000000

Code

```

1|  MOV R0,#arrayData1
2|  MOV R4,#arrayData2
3|  MOV R1,#36          // index
4| arrayloop:
5|  LDR R3, [R0+R1]      //pointer to the array
6|  STR R3,[R4]          // display the value inside of the array
7|  STR R3,.WriteUnsignedNum // write what's inside the array in the
display
8|  SUB R1,R1,#4         // subtract the index of the array by 4 bytes
9|  CMP R1, #0          // R1 - 0
10| BNE arrayloop
11| HALT
12| .ALIGN 256
13| arrayLength1: 10
14| arrayData1: 9
15|  8
16|  7
17|  6
18|  5
19|  4
20|  3
21|  2
22|  1
23|  0
24| arrayLength2:10
25| arrayData2:0

```


9.4.2/

The screenshot displays the Keil uVision IDE interface with three main panels:

- Program:** Shows an ARM assembly program. Line 9, `HALT`, is highlighted in orange. The program includes comments for index, pointer, and subtraction, and defines an array of 10 elements (8 to 0).
- Processor:** Displays the state of the processor registers (PC, LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0). The PC register is 0x00000020. The Count register is 48. The Current Instruction register is empty. The Status bits are NZCV 0100.
- Memory:** Shows a memory dump starting at address 0x00000000. The first few lines show values like 0xe3a00f41, 0xe2411004, and 0xe3510000.

Code

```

1|  MOV R0,#arrayData1
2|  MOV R1,#36      // index
3|arrayloop:
4|  LDR R3, [R0+R1] //pointer to the array
5|  STR R3,.WriteUnsignedNum // write what's inside the array in the
display
6|  SUB R1,R1,#4    // subtract the index of the array by 4 bytes
7|  CMP R1, #0      // R1 - 0
8|  BNE arrayloop
9|  HALT
10| .ALIGN 256
11|arrayLength1: 10
12|arrayData1: 9
13|  8
14|  7
15|  6
16|  5
17|  4
18|  3
19|  2
20|  1
21|  0

```