Lab 6
I pledge my honor that I have abided by the Stevens Honor system

Question 1:

```
1 // lab 6
3 .text
 4 .global _start
 5 .extern printf
 6 _start:
          //loading addresses of variables
          ADR X10, i
9
          ADR X11, f
          ADR X12, g
10
11
          //loading values of variables
          LDUR X13, [X10, #0] // X13 = i
12
          LDUR X14, [X11, \#0] // X14 = f
13
14
          LDUR X15, [X12, #0] // X15 = g
15
          //math
16
          SUB X13, X13, #4
          CBZ X13, Else
17
          //else (i != 4)
19
          SUB X15, X15, #2
20
          STUR X15, [X11, #0]
21
22
          //print value of f
23
          ADR X0, msg
          LDUR X1, [X11, #0]
24
25
          BL printf
26
          //exit call
27
28
          MOV X0, #0
29
          MOV W8, #93
          SVC #0
30
31 Else:
32
          // (i==4)
33
          ADD X15, X15, #1
34
          STUR X15, [X11, #0]
          ADR Xp, msg
LDUR X1, [X11, #0]
35
36
37
          BL printf
          MOV X0, #0
38
          MOV W8, #93
39
          SVC #0
40
41
42 .data
43
          i: .quad 5
44
          f: .quad
45
          g: .quad 6
          msg: .ascii "f = %d\n\0"
46
47
48 .end
```

```
hyogwah@hyo:~/assembly/lab6$ aarch64-linux-gnu-as 1.s -o 1.o
hyogwah@hyo:~/assembly/lab6$ aarch64-linux-gnu-ld 1.o -lc
hyogwah@hyo:~/assembly/lab6$ qemu-aarch64 a.out
f = 4
hyogwah@hyo:~/assembly/lab6$ aarch64-linux-gnu-as 1.s -o 1.o
hyogwah@hyo:~/assembly/lab6$ aarch64-linux-gnu-ld 1.o -lc
hyogwah@hyo:~/assembly/lab6$ qemu-aarch64 a.out
f = 7
```

Question 2:

```
1 // lab 6
 2 / question 2
 3
 4
 5 .text
6 .global _start
7 .extern printf
 8 _start:
 9
           //load address/value
           ADR X10, a
10
11
           ADR X11, b
           ADR X12, c
12
          LDUR X13, [X10, #0]
LDUR X14, [X11, #0]
13
14
15
           LDUR X15, [X12, #0]
16
           ADD X13, X13, X14 //should add a+b to prep for
17
18
           SUB X13, X13, \#14 // a+b == 14 check
19
           CBZ X13, Else
20
           // when a+b != 14
21
           MOV X15, #-2
22
           STUR X15, [X12, #0]
23
24
           ADR X0, msg
25
           LDUR X1, [X12, #0]
26
           BL printf
27
28
           MOV X0, #0
           MOV W8, #93
29
30
           SVC #0
31 Else: // when a+b == 14
           //c=3
33
           MOV X15, #3
34
           STUR X15, [X12, #0]
35
           ADR X0, msg
           LDUR X1, [X12, #0]
36
37
           BL printf
           MOV X0, #0
38
39
           MOV W8, #93
40
           SVC #0
41 .data
42
           a: .quad 6
43
           b: .quad 7
           c: .quad 0
44
           msg: .ascii "c = %d\n\0"
45
46
47 .end
```

```
2 // question 2
 5 .text
 6 .global _start
 7 .extern printf
 8 start:
 9
          //load address/value
10
         ADR X10, a
11
         ADR X11, b
12
         ADR X12, c
13
         LDUR X13, [X10, #0]
14
         LDUR X14, [X11, #0]
15
         LDUR X15, [X12, #0]
16
17
         ADD X13, X13, X14 //should add a+b to prep for
         SUB X13, X13, #14 // a+b == 14 check
18
19
         CBZ X13, Else
20
21
         // when a+b != 14
22
         MOV X15, #-2
                                                                  I
23
         STUR X15, [X12, #0]
24
         ADR X0, msg
25
         LDUR X1, [X12, #0]
26
         BL printf
27
28
         MOV X0, #0
29
         MOV W8, #93
30
         SVC #0
31 Else: // when a+b == 14
32
         //c=3
33
         MOV X15, #3
         STUR X15, [X12, #0]
34
35
         ADR X0, msg
36
         LDUR X1, [X12, #0]
37
         BL printf
38
         MOV X0, #0
39
         MOV W8, #93
40
         SVC #0
41 .data
42
         a: .quad 8
43
         b: .quad 7
44
         c: .quad 0
         msg: .ascii "c = %d\n\0"
45
46
47 .end
hyogwah@hyo:~/assembly/lab6/q2$ aarch64-linux-gnu-as 2.s -o 2.o
hyogwah@hyo:~/assembly/lab6/q2$ aarch64-linux-gnu-ld 2.o -lc
hyogwah@hyo:~/assembly/lab6/q2$ qemu-aarch64 a.out
c = 3
hyogwah@hyo:~/assembly/lab6/q2$ aarch64-linux-gnu-as 2.s -o 2.o
hyogwah@hyo:~/assembly/lab6/q2$ aarch64-linux-gnu-ld 2.o -lc
hyogwah@hyo:~/assembly/lab6/q2$ qemu-aarch64 a.out
```

I showed multiple scenarios

c = -2

1 // lab 6

Question 3:

```
hyogwah@hyo:~/assembly/lab6/q3$ aarch64-linux-gnu-as 3.s -o 3.o
hyogwah@hyo:~/assembly/lab6/q3$ qemu-aarch64 ./a.out
1792
```

```
//lab 6
2 .text
 3 .global _start
4 .extern printf
7 AddArray:
          ADR XO, arr // base of global array
8
          MOV X1, 0 // index
9
10
          ADR X12, val
11
          LSL X2, X1, #3 //bitshift 3 to the left, (multiplying by 8) X2 == offset
12 L1:
          ADD X9, X0, X2 //X9 = base + offset (exact location of element == address on source)
13
          LDUR X10, [X9, 0] //load value of X9 = arr[i]
14
15
          ADD X12, X10, X12
16
17
          ADD X1, X1, 1
18
          CMP X1, 10 //compare X1 and 10
19
          B.NE L1 //branch not equal
20
21
          ADR X0, str
22
          LDUR X1, [X12, 0]
23
          BL printf
          MOV X0, 0
24
25
          MOV W8, 93
26
          SVC 0
27
28 _start:
29
30
          BL AddArray
          // exit call
31
          MOV XO, 0
32
          MOV W8, 93
33
34
          SVC 0
35
36
37
38 .data
39 val: .quad 0
40 arr: .quad 1,2,3,4,5,6,7,8,9,10
41 str: .ascii "%d\n\0" // quad, use %ld to print
42
43 .end
```

I feel like my code for question 3 and 4 should be correct, I added some comments, can you please double check and see what's going wrong? I'm trying to load the address of arr[i] into 10, so I can add it to X12, which was my accumulator.

Question 4:

```
1 //lab 6
 2 .text
 3 .global _start
 4 .extern printf
 7 AddArray:
            ADR X0, arr // base of global array MOV X1, 0 // index
 9
            ADR X12, val
10
11
            LSL X2, X1, #3 //bitshift 3 to the left, (multiplying by 8) X2 == offset
12 L1:
            ADD X9, X0, X2 //X9 = base + offset (exact location of element == address on source) LDUR X10, [X9, 0] //load value of X9 = arr[i]
13
14
15
            ADD X12, X10, X12
16
            ADD &1, X1, 1
CMP X1, 8 //compare X1 and 10
17
18
            B.NE L1 //branch not equal
19
20
21
            ADR X0, str
22
            LDUR X1, [X12, 0]
23
            BL printf
24
            MOV XO, 0
            MOV W8, 93
25
26
            SVC 0
27
28 _start:
29
30
            BL AddArray
31
            // exit call
            MOV XO, 0
32
33
34
            MOV W8, 93
            SVC 0
35
36
37
38 .data
40 arr: .quad 1,0,4,5,8,3,1,3
41 str: .ascii "%d\n\0" // quad, use %ld to print
42
43 .end
44
```

```
hyogwah@hyo:~/assembly/lab6/q3$ aarch64-linux-gnu-ld 3.o -lc
hyogwah@hyo:~/assembly/lab6/q3$ qemu-aarch64 ./a.out
0
```