Austin Petersen
Ayden Dauenhauer
Lab 1
ELEN 120

Problem 1

1. What is the result in R0 after the following pseudo instruction is executed?
   LDR r0, =0x15
   R0 holds 0x00000015

2. What is the address that corresponds to the label label1?
   The address is 0x000001E8

3. What is the value in r1 after the second ADD instruction is executed?
   0x1E4 is in r1

4. What is the address of the instruction: ADD r1, #4?
   The address is 0x1D4

5. What is the value stored in the PC before the instruction: ADD r1, #4 is executed?
   0x1E0 is stored in the PC before

6. What is the final result in R3 after the program reaches the endless loop?
   The final result is 0x1F

7. Add meaningful comments to the program.

```
        AREA main, CODE, READONLY
        EXPORT __main
        ENTRY

__main PROC

        LDR r0, =0x15        // Assigns 0x15 to r0
        LDR r1, =label1 // Assigns the value stored in the address of label1 to r1
        LDR r2, [r1]   // Copies r1 into r2
        ADD r3, r0, r2        // r3 = r0 + r2
        ADD r1, #4            // r1 = r1 + 4
        LDR r4, [r1]   // Copies r1 into r4
```

```
        ADD r3, r4              // r3 = r3 + r4
endless B endless   // Endless loop
        ENDP

        ALIGN
label1 DCD 0x06
P1 DCD 0x04

        END
```

Problem 2

1. Describe what the program does in a single statement.
   The program adds 1 to -1.

2. What are the addresses of each of the 4 bytes of memory loaded into r2 by the LDR instruction?
   The address is in 0x1D8, 0x1D9,

3. Can we replace the MVN instruction with NEG and get the same result? Explain.
   No, MVN will put 0 in R3 and NEG puts 1 in R3. NEG performs 2's complement and thus adds 1.

4. Can we replace the MVN instruction with NOT and get the same result? Explain
   NOT does not exist as a valid instruction in ARM Assembly, so it cannot replace MVN.

Problem 3

1. What are the addresses of the stored values 3, 4, and 5 in list?
   The address of the stored values is 0x1EC

2. What is loaded into r2 after the first LDR instruction is executed?
   0x0 is in r2 after the first LDR instruction

3. What does the following instruction do? LSL r2, r2, #1
   LSL is logical shift left, so it shifted the value 0x3 to 0x6. In binary, this went from 0011 and was shifted to 0110.

4. Check the contents of the word at memory location 0x1F0 after the program is executed. What is the new value stored in memory at this location?

   The 0x4 is changed to 0x8 and 0x5 is changed to 0xA

5. Do the following instructions produce a different result for the same initial value in register r2?
   - MOV r2, r2, LSL #1
   - LSL r2, r2, #1

   These both shift the bit to the left so they produce the same result.

Problem 4

```
                AREA main, CODE, READONLY
                EXPORT __main
                ENTRY
__main PROC

        LDR r1, =num1           // Loads the address of num1 into r1
        LDR r2, =num2           // Loads the address of num2 into r2
        LDR r3, [r1]        // Loads the value of r1 into r3
        LDR r4, [r2]        // Loads the value of r2 into r4
        MUL r5, r3, r3          // Computes the square of r3 and stores in r5
        MUL r6, r4, r4          // Computes the square of r4 and stores in r6
        ADD r7, r5, r6          // Adds r5 and r6 and stores in r7
        STR r7, [r2, #4]    // Stores the final result into memory

endless B endless

        ENDP

        ALIGN
num1 DCD 0x05
num2 DCD 0x03
result DCD 0x22

        END
```