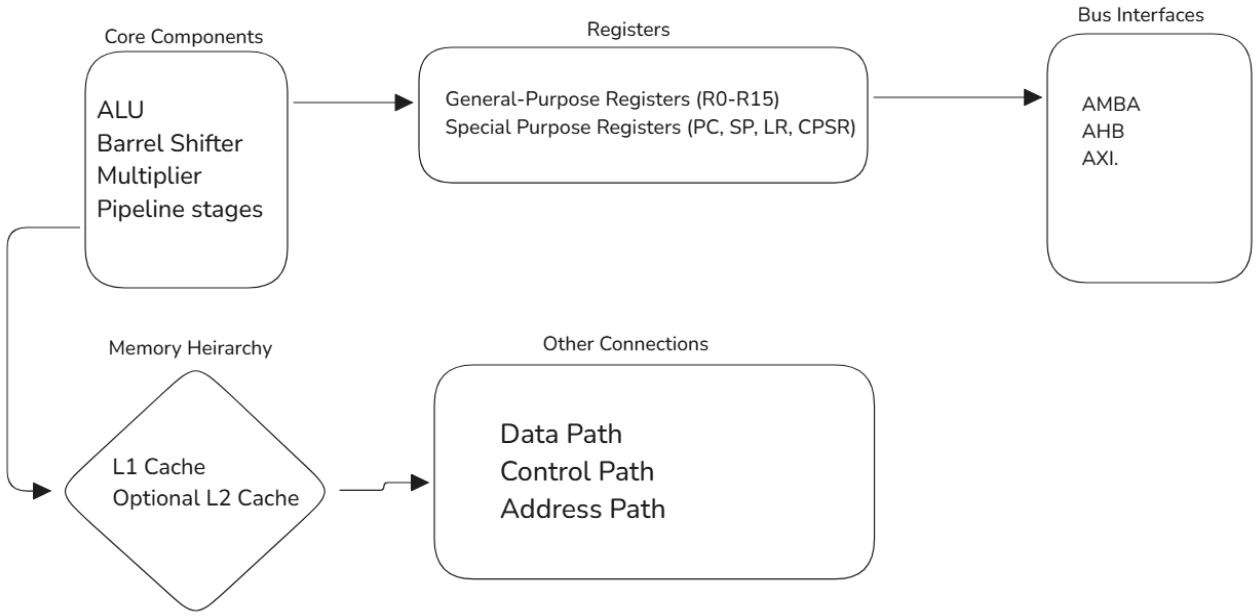
Experiment 1: Understanding ARM Architecture

Objective: Basic Components of the ARM Architecture:

The ARM (Advanced RISC Machine) architecture is a family of Reduced Instruction Set Computing (RISC) architectures designed for efficient performance. The key components include:

1. **Core Processor:**
    - **ALU (Arithmetic Logic Unit):** Executes arithmetic and logical operations.
    - **Barrel Shifter**: Used for efficient data shifts and rotations.
    - **Multiplier:** For rapid multiplication operations.
    - **Pipeline:** Allows instruction-level parallelism (commonly 3, 5, or 8 stages, such as Fetch, Decode, and Execute).
2. **Registers:**
    - **General-Purpose Registers (R0-R15):** Used for data storage and computations.
    - **Program Counter (PC):** Points to the next instruction to be executed.
    - **Link Register (LR):** Stores return addresses for function calls.
    - **Stack Pointer (SP):** Points to the top of the stack.
    - **Current Program Status Register (CPSR): Holds** flags (e.g., Zero, Carry, Overflow) and mode bits.
3. **Memory Interface:**
    - Supports Harvard or Von Neumann memory models.
    - Cache memory (L1 and optionally L2).
4. **Instruction Set:**
    - ARM instructions (32-bit).
    - Thumb instructions (16-bit).
    - Conditional execution for efficiency.
    - Interrupt Controller:
    - Manages hardware and software interrupts.
    - (Advanced Microcontroller Bus Architecture).

Task: Research and draw the ARM processor architecture, labeling its components.

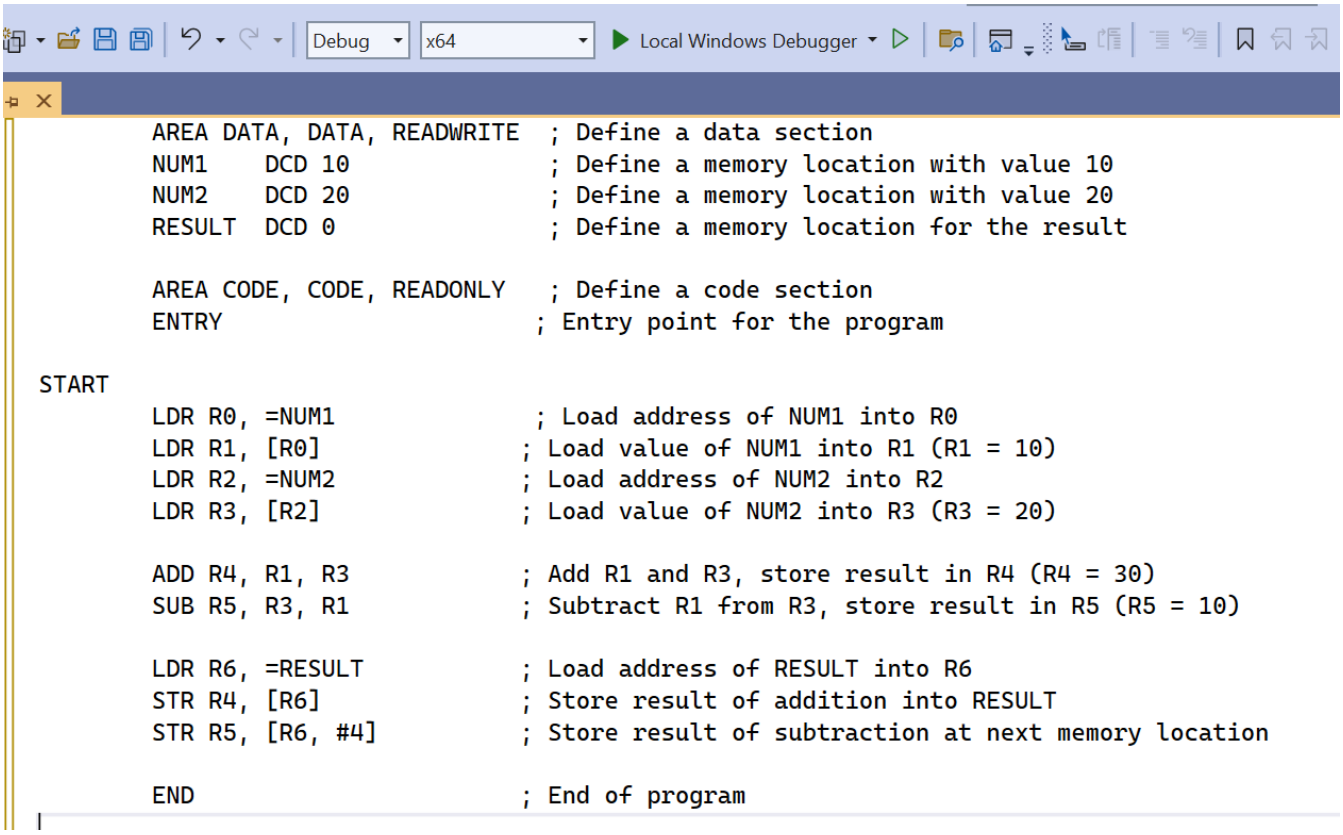


Experiment: 2

Task 1: Load and Store Data using LDR and STR

```
        AREA DATA, DATA, READWRITE  ; Define a data section
        NUM1    DCD 5               ; Define a memory location with value 5
        RESULT  DCD 0               ; Define a memory location for result

        AREA CODE, CODE, READONLY   ; Define a code section
        ENTRY                       ; Entry point for the program

START
        LDR R0, =NUM1               ; Load address of NUM1 into R0
        LDR R1, [R0]                ; Load value at address in R0 into R1 (R1 = 5)
        LDR R2, =RESULT             ; Load address of RESULT into R2
        STR R1, [R2]                ; Store value of R1 into address in R2

        END                         ; End of program
```

Task 2: Perform Basic Arithmetic Operations

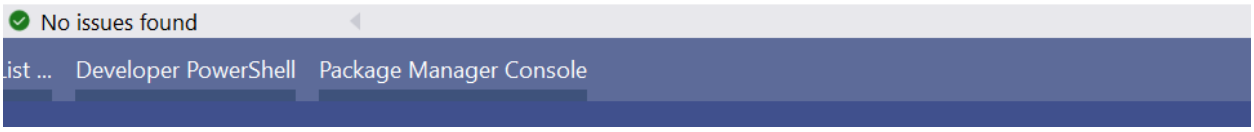



```
        AREA DATA, DATA, READWRITE  ; Define a data section
        NUM1    DCD 10              ; Define a memory location with value 10
        NUM2    DCD 20              ; Define a memory location with value 20
        RESULT  DCD 0               ; Define a memory location for the result

        AREA CODE, CODE, READONLY   ; Define a code section
        ENTRY                       ; Entry point for the program

START
        LDR R0, =NUM1               ; Load address of NUM1 into R0
        LDR R1, [R0]                ; Load value of NUM1 into R1 (R1 = 10)
        LDR R2, =NUM2               ; Load address of NUM2 into R2
        LDR R3, [R2]                ; Load value of NUM2 into R3 (R3 = 20)

        ADD R4, R1, R3              ; Add R1 and R3, store result in R4 (R4 = 30)
        SUB R5, R3, R1              ; Subtract R1 from R3, store result in R5 (R5 = 10)

        LDR R6, =RESULT             ; Load address of RESULT into R6
        STR R4, [R6]                ; Store result of addition into RESULT
        STR R5, [R6, #4]            ; Store result of subtraction at next memory location

        END                         ; End of program
```

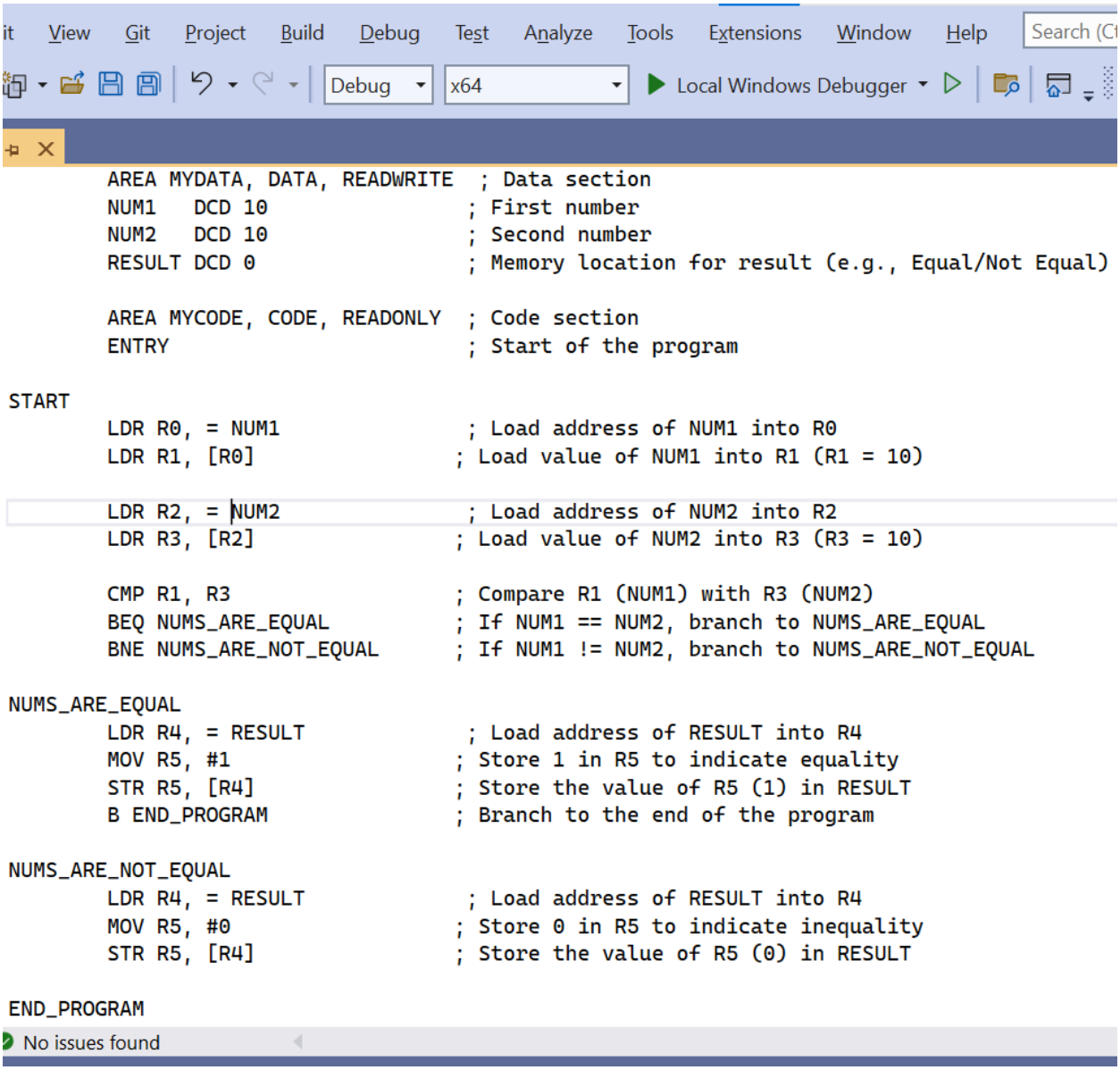
Experiment: 3

Task 1: Compare Two Numbers and Output the Larger Number

```
        AREA MYDATA, DATA, READWRITE  ; Data section
        NUM1   DCD 15                 ; First number
        NUM2   DCD 20                 ; Second number
        RESULT DCD 0                  ; Memory location for the larger number

        AREA MYCODE, CODE, READONLY   ; Code section
        ENTRY                         ; Start of the program

START
        LDR R0, = NUM1                ; Load address of NUM1 into R0
        LDR R1, [R0]                  ; Load value of NUM1 into R1 (R1 = 15)

        LDR R2, = NUM2                ; Load address of NUM2 into R2
        LDR R3, [R2]                  ; Load value of NUM2 into R3 (R3 = 20)

        CMP R1, R3                    ; Compare R1 (NUM1) with R3 (NUM2)
        BGT NUM1_IS_LARGER            ; If NUM1 > NUM2, branch to NUM1_IS_LARGER
        B   NUM2_IS_LARGER            ; Otherwise, branch to NUM2_IS_LARGER

NUM1_IS_LARGER
        LDR R4, = RESULT              ; Load address of RESULT into R4
        STR R1, [R4]                  ; Store NUM1 (R1) in RESULT
        B END_PROGRAM                 ; Branch to the end of the program

NUM2_IS_LARGER
        LDR R4, = RESULT              ; Load address of RESULT into R4
        STR R3, [R4]                  ; Store NUM2 (R3) in RESULT

END_PROGRAM
        END                           ; End of the program
```

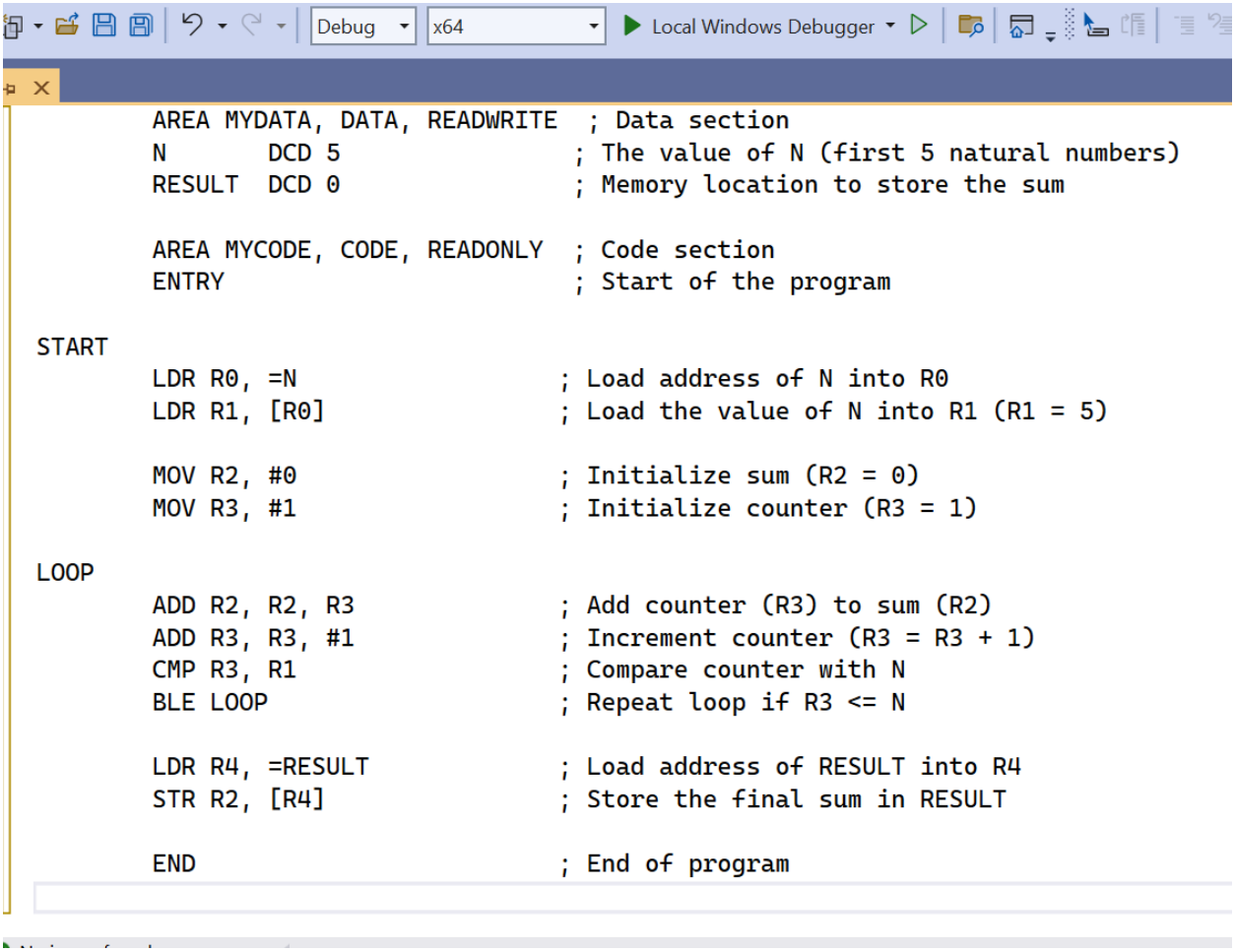


Task 2: Implement a Conditional Block Using CMP, BEQ, BNE



Experiment: 4

Task 1: Calculate the Sum of the First N Natural Numbers




```
        AREA MYDATA, DATA, READWRITE  ; Data section
        N       DCD 5                 ; The value of N (first 5 natural numbers)
        RESULT  DCD 0                 ; Memory location to store the sum

        AREA MYCODE, CODE, READONLY   ; Code section
        ENTRY                         ; Start of the program

START
        LDR R0, =N                    ; Load address of N into R0
        LDR R1, [R0]                  ; Load the value of N into R1 (R1 = 5)

        MOV R2, #0                    ; Initialize sum (R2 = 0)
        MOV R3, #1                    ; Initialize counter (R3 = 1)

LOOP
        ADD R2, R2, R3                ; Add counter (R3) to sum (R2)
        ADD R3, R3, #1                ; Increment counter (R3 = R3 + 1)
        CMP R3, R1                    ; Compare counter with N
        BLE LOOP                      ; Repeat loop if R3 <= N

        LDR R4, =RESULT               ; Load address of RESULT into R4
        STR R2, [R4]                  ; Store the final sum in RESULT

        END                           ; End of program
```

Task 2: Multiplication Using Iterative Addition


```
        AREA MYDATA, DATA, READWRITE  ; Data section
        NUM1    DCD 4                 ; First number
        NUM2    DCD 3                 ; Second number
        RESULT  DCD 0                 ; Memory location for the product

        AREA MYCODE, CODE, READONLY   ; Code section
        ENTRY                         ; Start of the program

START
        LDR R0, =NUM1                 ; Load address of NUM1 into R0
        LDR R1, [R0]                  ; Load value of NUM1 into R1 (R1 = 4)

        LDR R2, =NUM2                 ; Load address of NUM2 into R2
        LDR R3, [R2]                  ; Load value of NUM2 into R3 (R3 = 3)

        MOV R4, #0                    ; Initialize product (R4 = 0)

LOOP
        ADD R4, R4, R1                ; Add NUM1 to the product (R4 = R4 + R1)
        SUBS R3, R3, #1               ; Decrement NUM2 (R3 = R3 - 1)
        BNE LOOP                      ; Repeat loop if R3 != 0

        LDR R5, =RESULT               ; Load address of RESULT into R5
        STR R4, [R5]                  ; Store the final product in RESULT

        END                           ; End of program
```

Experiment: 5

Task 1: Find the Maximum Value in an Array

```
        AREA MYDATA, DATA, READWRITE  ; Data section
        ARRAY   DCD 3, 7, 2, 9, 5    ; Define an array
        SIZE    DCD 5                ; Define the size of the array
        MAX     DCD 0                ; Memory location to store the maximum value

        AREA MYCODE, CODE, READONLY  ; Code section
        ENTRY                        ; Start of the program

START
        LDR R0, =ARRAY               ; Load the address of the array into R0
        LDR R1, =SIZE                ; Load the address of SIZE into R1
        LDR R2, [R1]                 ; Load the size of the array into R2 (R2 = 5)

        LDR R3, [R0]                 ; Load the first element of the array into R3 (MAX = ARRAY[0])

LOOP
        ADD R0, R0, #4               ; Move to the next element in the array
        LDR R4, [R0]                 ; Load the next element into R4
        CMP R3, R4                   ; Compare current MAX (R3) with the element (R4)
        BGT CONTINUE                 ; If MAX > element, skip updating MAX
        MOV R3, R4                   ; Update MAX (R3 = R4)

CONTINUE
        SUBS R2, R2, #1              ; Decrement the counter (R2 = R2 - 1)
        BNE LOOP                     ; Repeat loop if there are more elements

        LDR R5, =MAX                 ; Load the address of MAX into R5
        STR R3, [R5]                 ; Store the final maximum value in MAX

        END                          ; End of program
```

Task 2: Bubble Sort an Array

```
        AREA MYDATA, DATA, READWRITE  ; Data section
        ARRAY   DCD 5, 3, 8, 1, 2    ; Define an array
        SIZE    DCD 5                ; Define the size of the array

        AREA MYCODE, CODE, READONLY  ; Code section
        ENTRY                        ; Start of the program

START
        LDR R0, =ARRAY               ; Load the address of the array into R0
        LDR R1, =SIZE                ; Load the address of SIZE into R1
        LDR R2, [R1]                 ; Load the size of the array into R2 (R2 = 5)

OUTER_LOOP
        MOV R3, R2                   ; Initialize inner loop counter (R3 = SIZE)
        SUB R3, R3, #1               ; R3 = SIZE - 1

INNER_LOOP
        LDR R4, [R0]                 ; Load the first element into R4
        LDR R5, [R0, #4]             ; Load the second element into R5
        CMP R4, R5                   ; Compare the two elements
        BLE SKIP_SWAP                ; If R4 <= R5, skip the swap

        STR R5, [R0]                 ; Swap: Store R5 at the first position
        STR R4, [R0, #4]             ; Swap: Store R4 at the second position

SKIP_SWAP
        ADD R0, R0, #4               ; Move to the next pair
        SUBS R3, R3, #1              ; Decrement the inner loop counter
        BNE INNER_LOOP               ; Repeat inner loop if not done

        LDR R0, =ARRAY               ; Reset the array pointer to the start
        SUBS R2, R2, #1              ; Decrement the outer loop counter
        BNE OUTER_LOOP               ; Repeat outer loop if not done

        END                          ; End of program
```

No issues found