Lab1

Q1)

```
        AREA  RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0X10001000
        DCD Reset_Handler
        ALIGN
        AREA  mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        MOV R0,#3
        MOV R1, #10
        ADD R0, R0, R1
STOP B STOP
        END
```

Q2)

```
        AREA RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0X10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        LDR R3, [R0]
        STR R3, [R1]
STOP
        B STOP
SRC DCD 10
        AREA mydata, DATA, READWRITE
DST DCD 0
        END
```

Lab2

1.

```
                AREA RESET,DATA,READONLY
        EXPORT __Vectors
```

```
__Vectors
        DCD 0X10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode,CODE,READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        MOV R0,#0x1
        LDR R3,=data_location
        LDR R4, [R3]
        MOV R5,R4
        ; Exchanging R6 and R7 using R8
        MOV R6,#0x69
        MOV R7,#0x49
        MOV R8,R6
        MOV R6,R7
        MOV R7,R8
STOP
        B STOP
        AREA mydata, DATA, READWRITE
data_location
        DCD 0x4
        END
```

2.

```
        AREA RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0X10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        LDR R3, [R0]
        STR R3, [R1]
STOP
```

```
        B STOP
        AREA mydata, DATA, READWRITE
SRC DCD 10
DST DCD 0
        END
```

3.

```
        AREA RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0X10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        MOV R4,#10
LOOP
        LDR R3, [R0]
        ADD R0,#4
        STR R3, [R1]
        ADD R1,R1,#4
        SUBS R4,R4,#1
        BNE LOOP
STOP
        B STOP
SRC DCD 1,2,3,4,5,6,7,8,9,0xA
        AREA mydata, DATA, READWRITE
DST DCD 0
        END
```

4.

```
#Reverse an array of 10 32-bit numbers in the memory.
      AREA RESET, DATA, READONLY
      EXPORT __Vectors
__Vectors
      DCD 0X10001000
      DCD Reset_Handler
      ALIGN
      AREA mycode, CODE, READONLY
```

```
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0, =ARRAY              ; Load address of the start of the array into R0
        LDR R1, =ARRAY + 36           ; Load address of the last element (end of the array) into
R1
        MOV R2, #5                  ; Set loop counter to 5 (half the array size)

LOOP
        LDR R3, [R0]                ; Load value at the start pointer into R3
        LDR R4, [R1]                ; Load value at the end pointer into R4
        STR R4, [R0]                ; Store value in R4 at the start pointer
        STR R3, [R1]                ; Store value in R3 at the end pointer
        ADD R0, R0, #4                ; Increment start pointer (move to the next element)
        SUB R1, R1, #4                ; Decrement end pointer (move to the previous element)
        SUBS R2, R2, #1               ; Decrement the loop counter and update flags
        BNE LOOP                    ; Repeat if counter is not zero

STOP    B STOP                        ; Infinite loop to stop execution

        AREA mydata, DATA, READWRITE
ARRAY   DCD 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ; Define an array of ten 32-bit numbers
        END
```

LAB 3

1.

```
        AREA RESET,DATA,READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000
        DCD Reset_Handler

        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0,=VALUE1
        LDR R1,[R0]
        LDR R0,=VALUE2
        LDR R3,[R0]
        SUBS R6,R1,R3
```

```
        LDR R2,=RESULT
        STR R6,[R2]
STOP
        B STOP

VALUE1 DCD 0x12345678
VALUE2 DCD 0x12340000
        AREA data,DATA,READWRITE
RESULT DCD 0
        END
```

2.

```
        AREA RESET,DATA,READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000
        DCD Reset_Handler

        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0,=VALUE1
        LDR R1,[R0]
        LDR R0,=VALUE2
        LDR R3,[R0]
        MUL R6,R1,R3
        LDR R2,=RESULT
        STR R6,[R2]
STOP
        B STOP

VALUE1 DCD 0x00000008
VALUE2 DCD 0x00000005
        AREA data,DATA,READWRITE
RESULT DCD 0
        END
```

3.

```
        AREA RESET,DATA,READONLY
        EXPORT __Vectors
__Vectors
```

```
        DCD 0x40001000
        DCD Reset_Handler

        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0,=VALUE1
        LDR R1,[R0]
        LDR R0,=VALUE2
        LDR R3,[R0]
        MOV R4,#0  ;Quotient
LOOP
        CMP R1,R3
        BCC DONE
        SUB R1,R1,R3
        ADD R4,R4,#1
        B LOOP
DONE
        LDR R2,=QUOTIENT
        STR R4,[R2] ;Quotient
        LDR R2,=REMAINDER
        STR R1,[R2] ;Remainder

STOP
        B STOP

VALUE1 DCD 32
VALUE2 DCD 5
        AREA data,DATA,READWRITE
QUOTIENT DCD 0
REMAINDER DCD 0
        END
```

4.

```
        AREA RESET,DATA,READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000
        DCD Reset_Handler

        ALIGN
        AREA mycode, CODE, READONLY
```

```
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0,=NUM
        LDR R3,[R0]
        MLA R1,R3,R3,R3
        LSR R1,#1
        LDR R2,=RESULT
        STR R1,[R2]
STOP
        B STOP
NUM DCD 5
        AREA data,DATA,READWRITE
RESULT DCD 0
        END
```

Additional

1.

```
    AREA RESET,DATA,READONLY
    EXPORT __Vectors
__Vectors
    DCD 0x40001000
    DCD Reset_Handler

    ALIGN
    AREA mycode, CODE, READONLY
    ENTRY
    EXPORT Reset_Handler
Reset_Handler
    LDR R0, =NUMBERS      ; Load address of the numbers array
    MOV R1, #10           ; Loop counter (10 numbers)
    MOV R2, #0            ; Initialize sum to 0

LOOP
    LDR R3, [R0], #4      ; Load value from address in R0, then increment R0 by 4
    ADD R2, R2, R3        ; Add value to sum (R2)
    SUBS R1, R1, #1       ; Decrement counter
    BNE LOOP              ; Repeat until R1 == 0

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

```
STOP    B STOP              ; Stop execution

NUMBERS DCD 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  ; Array of 10 numbers

      AREA data,DATA,READWRITE
RESULT  DCD 0               ; Store sum here

      END
```

2.

```
                AREA RESET,DATA,READONLY
                EXPORT __Vectors

__Vectors
                DCD 0x40001000
                DCD Reset_Handler

                ALIGN
                AREA mycode,CODE,READONLY
                ENTRY
                EXPORT Reset_Handler
Reset_Handler
                LDR R0, =FIB_SERIES
                MOV R1,#0     ; First Fibonacci number (F(0) = 0)
                MOV R2,#1     ; Second Fibonacci number (F(1) = 1)
                MOV R3,#10    ; Number of terms to generate

                STR R1,[R0],#4          ; Store F(0) at memory location and increment R0
                STR R2,[R0],#4          ; Store F(1) at next memory location and increment R0
                SUBS R3,R3,#2           ; Reduce loop counter (since first two values are already
stored)

LOOP
                ADD R4,R1,R2            ; Compute next Fibonacci number
                STR R4,[R0],#4  ; Store computed Fibonacci number in memory
                MOV R1, R2      ; Update F(n-2) = F(n-1)
                MOV R2, R4      ; Update F(n-1) = F(n)
                SUBS R3,R3,#1
                BNE LOOP

STOP
                B STOP

                AREA data,DATA,READWRITE
```

```
FIB_SERIES  SPACE 40   ; Reserve space for 10 Fibonacci numbers (10 × 4 bytes = 40 bytes)

                END
```

3.

This program calculates the **Greatest Common Divisor (GCD)** using **Euclidean Algorithm** and **Least Common Multiple (LCM)** using the formula:

$$LCM(a,b) = \frac{a \times b}{GCD(a,b)}$$

Program Logic

1. **Compute GCD using Euclidean Algorithm:**
   - Repeat `a = a - b` (or `b = b - a`) until one number becomes `0`.
   - The remaining number is the **GCD**.
2. **Compute LCM using the formula:**
   - $\text{LCM} = \frac{A \times B}{GCD}$
   - Perform multiplication first.
   - Then divide by GCD.

```
        AREA RESET,DATA,READONLY
        EXPORT __Vectors

__Vectors
        DCD 0x40001000
        DCD Reset_Handler

        ALIGN
        AREA mycode,CODE,READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0, =VALUE1       ; Load address of VALUE1
        LDR R1, [R0]          ; Load VALUE1 into R1 (A)

        LDR R0, =VALUE2       ; Load address of VALUE2
        LDR R2, [R0]          ; Load VALUE2 into R2 (B)
```

```
        MOV R3, R1          ; Copy A to R3 (for LCM calculation)
        MOV R4, R2          ; Copy B to R4 (for LCM calculation)

        ; Compute GCD using Euclidean Algorithm
GCD_LOOP
        CMP R1, R2          ; Compare A and B
        BEQ GCD_DONE        ; If A == B, GCD found
        BHI SUB_A           ; If A > B, subtract B from A
        SUB R2, R2, R1      ; Else, subtract A from B
        B GCD_LOOP

SUB_A
        SUB R1, R1, R2      ; A = A - B
        B GCD_LOOP

GCD_DONE
        ; Store GCD result
        LDR R0, =GCD_RESULT
        STR R1, [R0]        ; Store GCD (R1 now holds GCD)

        ; Compute LCM = (A * B) / GCD
        MUL R5, R3, R4      ; R5 = A * B
        UDIV R6, R5, R1     ; R6 = (A * B) / GCD

        ; Store LCM result
        LDR R0, =LCM_RESULT
        STR R6, [R0]        ; Store LCM result

STOP    B STOP

        AREA data,DATA,READWRITE
VALUE1  DCD 12              ; First 8-bit number
VALUE2  DCD 18              ; Second 8-bit number
GCD_RESULT DCD 0            ; Store GCD result
LCM_RESULT DCD 0            ; Store LCM result

        END
```