I. OBJECTIVE:

1. Find the largest/smallest number in an array of size N.
2. Separate Even numbers and odds numbers in an array of size N.

II. PRE-LAB

Explain the addressing modes involved in instructions.

1. Immediate Addressing: is used to load constant values into registers.

   Ex: mov r4, #0x00

2. Literal Addressing: is used to load addresses of data labels into registers.

   Ex: ldr r2, =array

3. Register Indirect Addressing: is used to access values from memory locations pointed by registers, often with post increment to step through arrays.

   Ex: ldr r3, [r2]

4. Logical Immediate Addressing: is used for bitwise operations with immediate values.

   Ex: ands r7, r6, #1

5. Memory Direct Addressing: is used to store values from register into memory locations.

   Ex: str r6, [r5], #4

6. Branch Instructions: 'bgt' & 'bne' branches to specified label if comparision is greater & result is not zero respectively.

For each objective in prelab describe the following points:

- Write the assembly code with description (ex. Mov ax,3000h – ax<-3000h)
- Examine & analyze the input/output of assembly code.

Objective 1:

=) @Largest number from array:

```
.global _start
_start:
        ldr r0, =count
        ldr r1, [r0]
        mov r4, #0x00
        ldr r2, =array
back:   ldr r3, [r2], #4
        cmp r4, r3
        bgt fwd // signed int comparision gives greater than
        mov r4, r3
fwd:    subs r1, r1, #01
        bne back
        str r4, [r2]
exit:   b exit
.data
count:  .word 0x05
array:  .word 0x15, 0x35, 0x45, 0x10, 0x4f
```

=) @Smallest:

```
.global _start
_start: ldr r0, =count
        ldr r1, [r0]
        mov r4, #0x0ff
        ldr r2, =array
back:   ldr r3, [r2], #4
        cmp r4, r3
        blt fwd
        mov r4, r3
fwd:    subs r1, r1, #01
        bne back
        str r4, [r2]
exit:   b exit
.data
count:  .word 0x05
array:  .word 0x15, 0x35, 0x45, 0x10, 0x4f
```

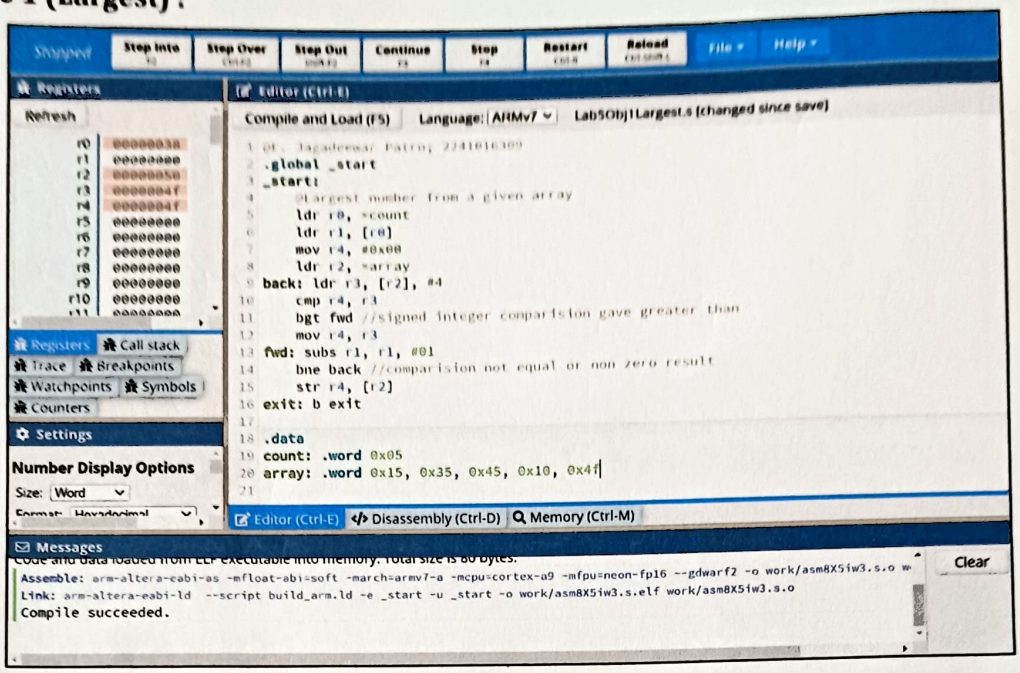
Objective 2:

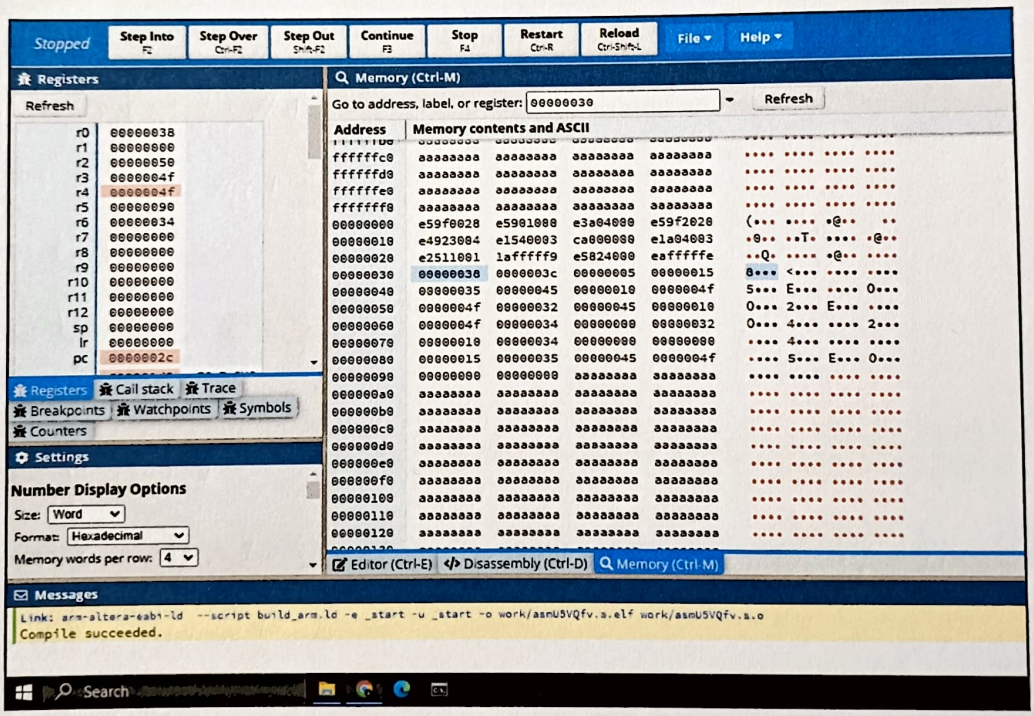
```
.global _start
_start:
    @ separate even and odd numbers from array
    ldr r0, =count
    ldr r1, [r0]
    ldr r3, =array @ r3 = base address of array = array[0]
    ldr r4, =even @ r4 = base address of even data locations. = even[0]
    ldr r5, =odd @ r5 = base address of odd data locations = odd[0]
back: ldr r6, [r3], #4
      ands r7, r6, #1
      beq fwd
      str r6, [r5], #4
      b fwd1
fwd: str r6, [r4], #4
fwd1: subs r1, r1, #01
      bne back
exit: b exit

.data
count: .word 0x07
array: .word 0x15, 0x35, 0x32, 0x45, 0x10, 0x4f, 0x34
even: .word 0, 0, 0, 0, 0
odd: .word 0, 0, 0, 0, 0
```

III. LAB

Objective 1 (Largest) :





From this result I have observed.....

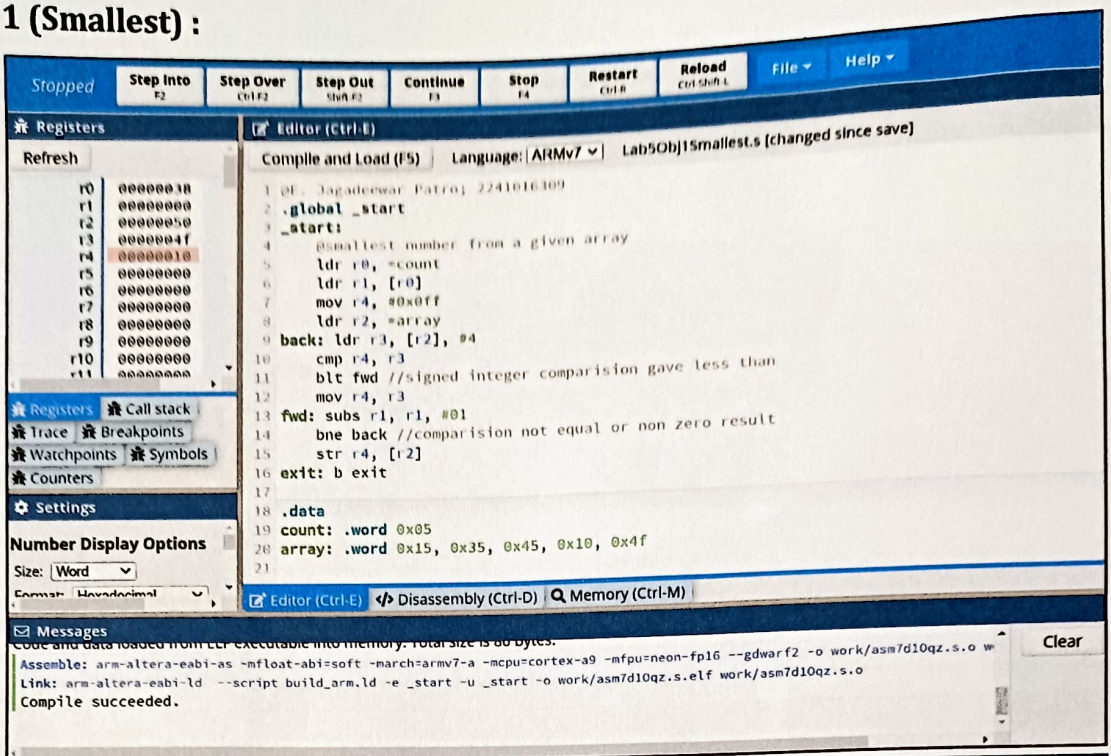
Input:

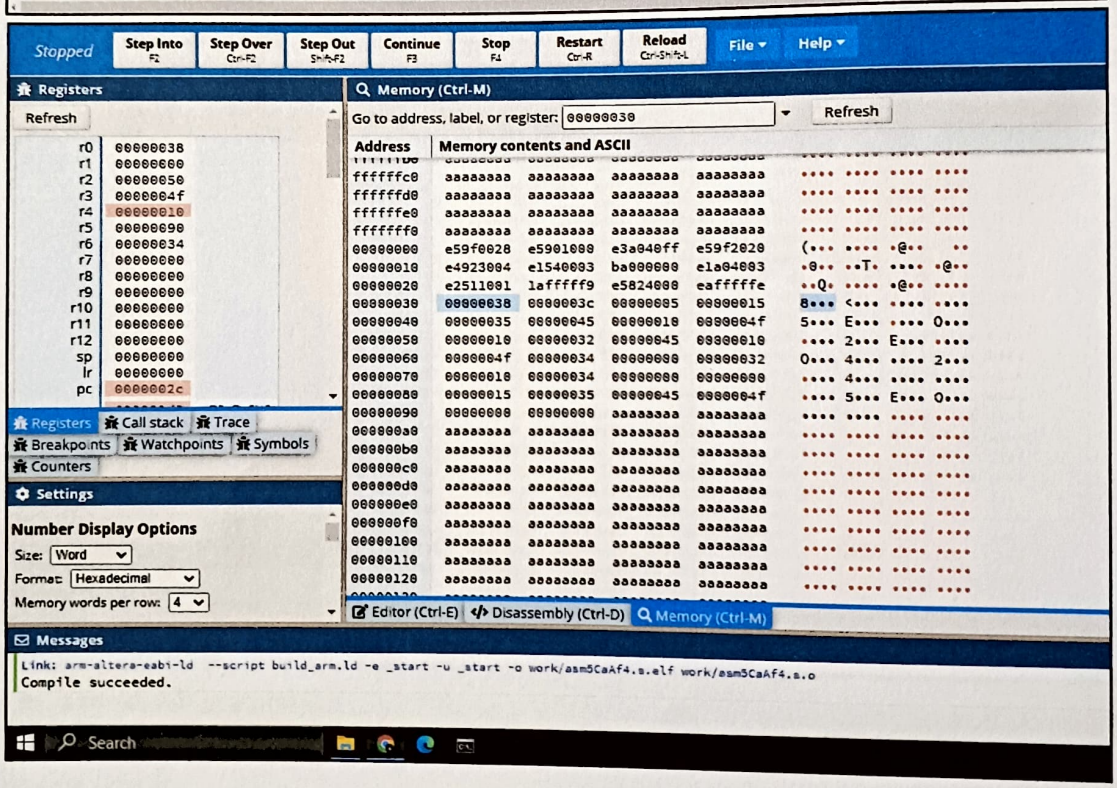
Sl. No.	Memory Location	Operand (Data)
1	0x00000038	0x05
2	0x0000003C	0x15
3	0x00000040	0x35
4	0x00000044	0x45
5	0x00000048	0x10
6	0x0000004C	0x4f

Output:

Sl. No.	Memory Location	Operand (Data)
1	00000050	0x4f

Objective 1 (Smallest) :





From this result I have observed.....

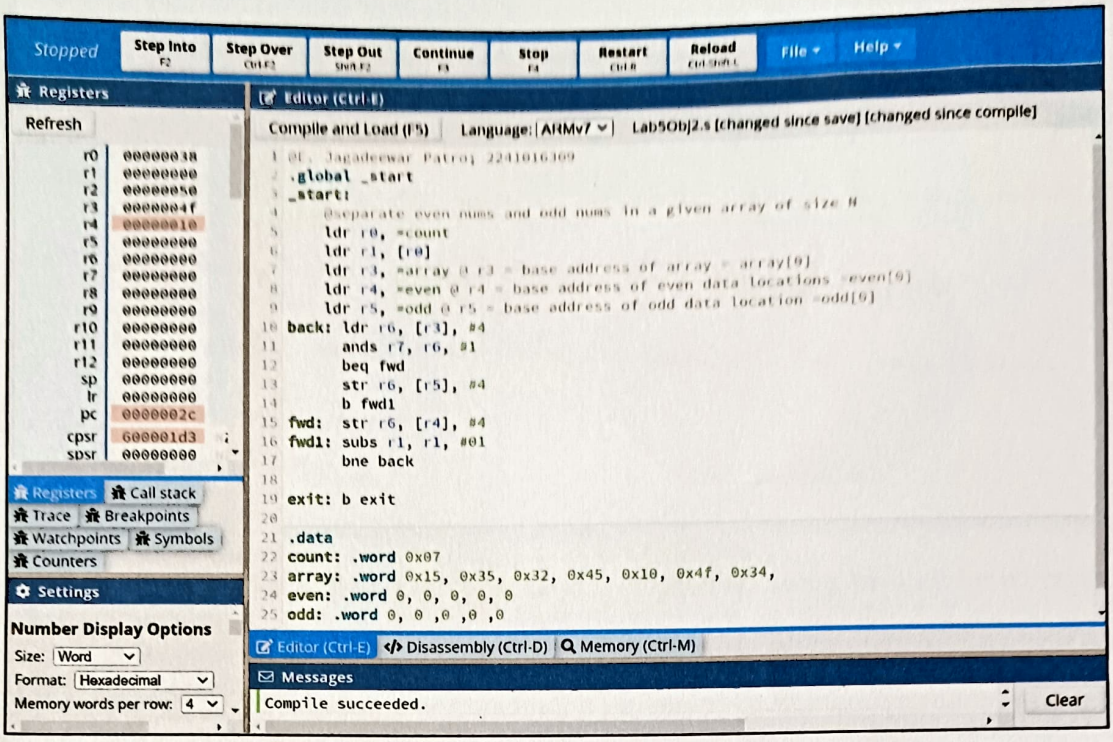
Input:

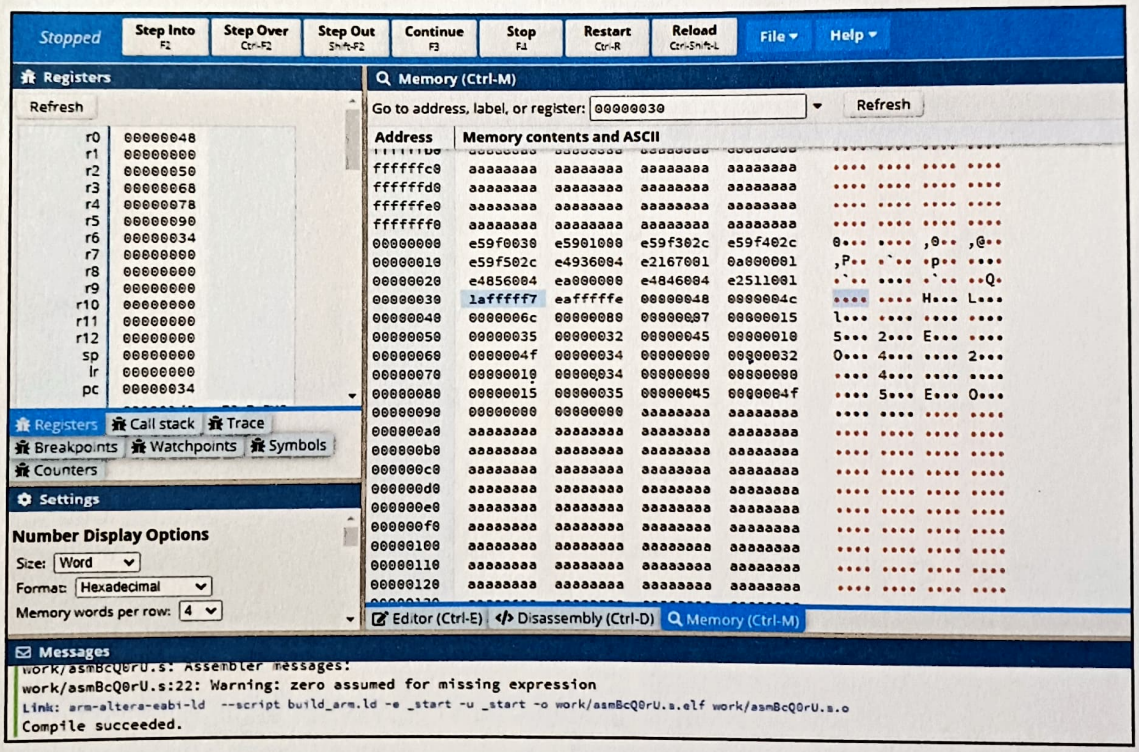
Sl. No.	Memory Location	Operand (Data)
1	00000038	0x05
2	0000003C	0x15
3	00000040	0x35
4	00000044	0x45
5	00000048	0x10
6	0000004C	0x4f

Output:

Sl. No.	Memory Location	Operand (Data)
1	00000050	0x10

Objective 2 :





From this result I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	00000048	0x07
2	0000004C	0x15
3	00000050	0x35
4	00000054	0x32
5	00000058	0x45
6	0000005C	0x10
7	00000060	0x4f
8	00000064	0x34

Output:

Sl. No.	Memory Location	Operand (Data)
1	0000005C	0x10
2	00000060	0x4f
3	00000064	0x34
4	00000068	0x32
5	00000080	0x15
6	00000084	0x35
7	00000088	0x45

IV. CONCLUSION

We have successfully achieved the two primary objectives (i.e, Identifying the largest and smallest numbers in an array of size N, and separating even and odd numbers from an array of size N) by implementing of low-level data manipulation and control flow. This provided practical experience in handling arrays and conditional logic within the constraints of ARM architecture.

V. POST LAB

1. Explain briefly condition codes (flags) of ARM processor.
2. Which condition codes (flags) is considered for the following branch instructions?
   a. B Label
   b. BEQ label
   c. BLT label

1) The condition codes (flags) of ARM Processor are:-

i) N (Negative flag): Set if the result of the operation is negative (i.e MSB=1)

ii) Z (zero flag): Set if the result of the operation is zero.

iii) C (carry flag): Set if the operation result in a carry out from the most significant bit (MSB) for addition, or if a borrow in subtraction.

iv) V (overflow flag): Set if the operation results in an overflow, which occurs when the result of a signed operation exceeds the range that can be represented in the number of bits available.

2) a) B label: No condition codes considered (unconditional)

b) BEQ label: Zero flag (Z)

c) BLT Label: Negative flag (N) and overflow flag (V) in combination (N!=V)