ARM ASSEMBLY PROGRAMMING

Aim:-

To (a) learn the architecture of ARM processor (b) learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations (c) go through example programs and (d) write assembly language programs for the given set of (computational) problems

Tasks:

Engineering Problem Solve the following engineering problems using ARM through assembly programs

1. Compute the factorial of a given number using ARM processor through assembly programming

2. Combine the low four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword. The value at LIST goes into the most signicant nibble of the result. Store the result in the 32-bit variable RESULT.

3. Given a 32 bit number, identify whether it is an even or odd. (You implementation should not involve division).

SOLUTIONS:-

1) **FLOW CHART**
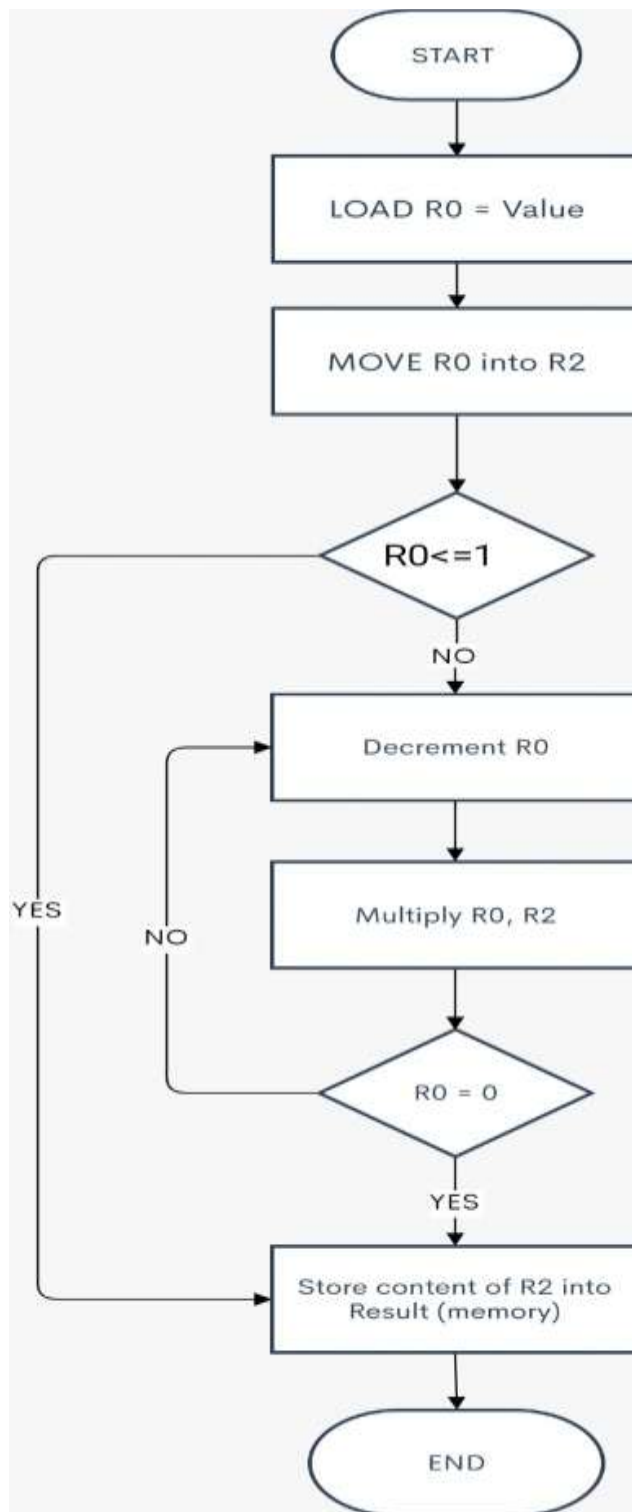
CODE:

```
*       FACTORIAL OF A NUMBER
        TTL    factorial
        AREA Program, CODE, READONLY
        ENTRY

Main          LDR    R0, Value
              MOV  R2, R0

              CMP  R0, #1          ; CHECK R0 WITH DECIMAL VALUE 1
              BGT  FACT            ; IF VALUE >1 THEN JMP TO FACT

              MOV  R2, #1
              STR  R2, RESULT      ; For VALUE== 1 OR 0 we are storing
                                   ; RESULT=1(ie.,0!=1!=1)
              SWI &11

FACT   SUB  R0, #1                 ; USING R0 AS A COUNTER

REPEAT        MUL  R2, R0,R2          ; R2=R0*R2
              SUBS R0, R0, #1         ; R0=R0-1 SET THE FLAG RESGISTER VALUES
              BNE  REPEAT
              STR  R2, RESULT

HERE   B      HERE
Value  DCD    &4
RESULT DCD  0
              END
```
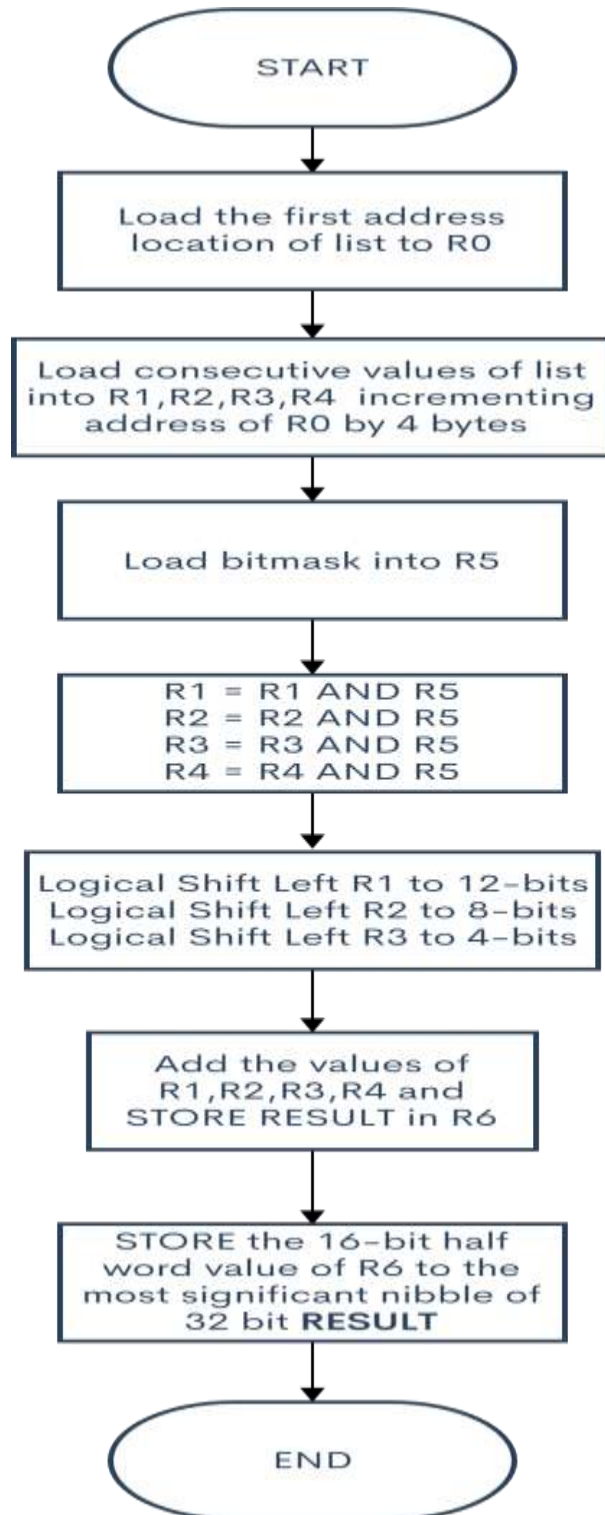
2)
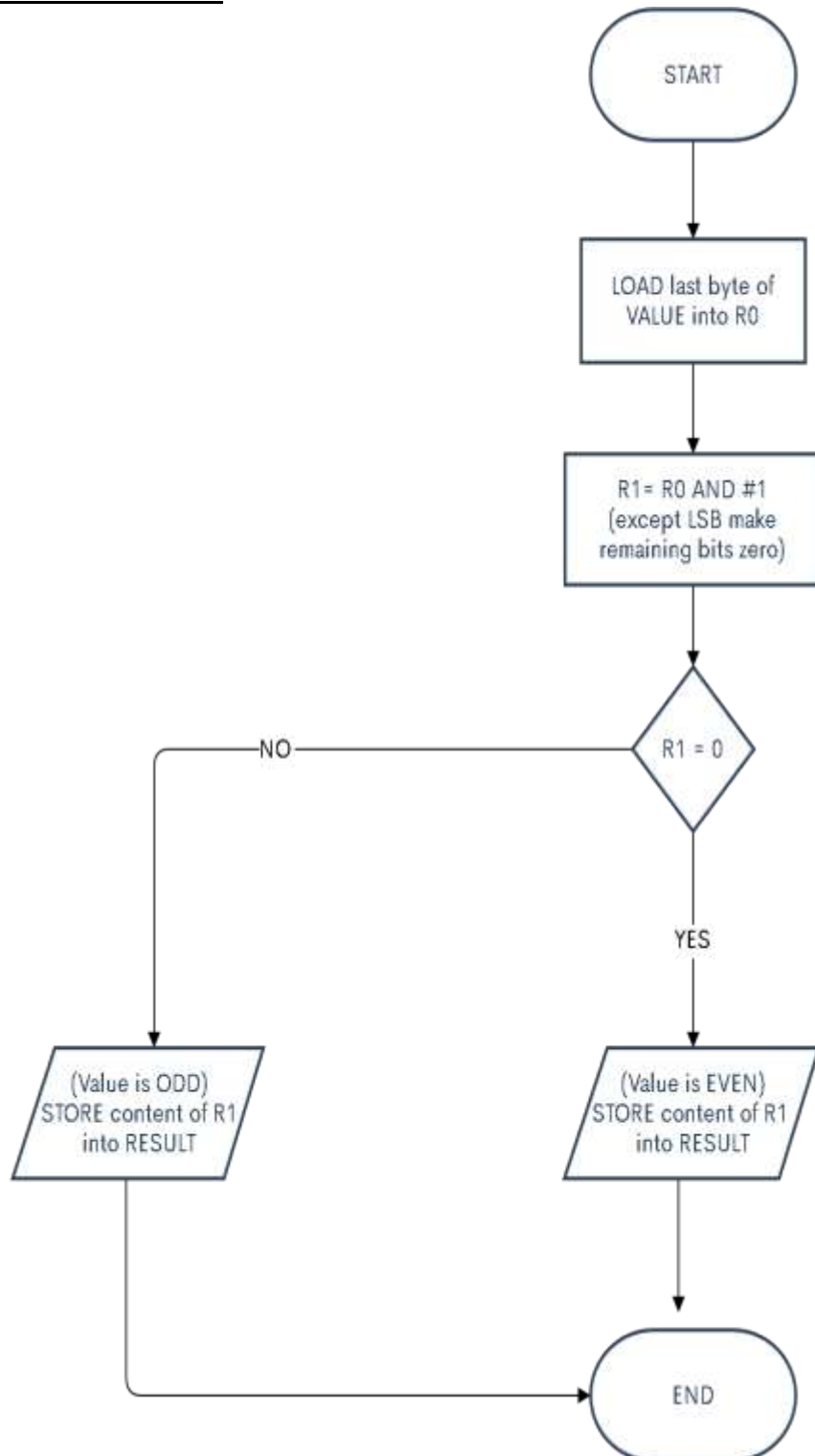
FLOW CHART:

CODE:

```
AREA PROGRAM, CODE, READONLY
    ENTRY

MAIN
     LDR R5, Mask
    LDR R0, =Value       ; pointing to first address  location of value
     LDRB R1, [R0]        ; loading last byte of hex 3B
     LDRB R2, [R0, #4]!   ; loading last byte of hex 4C by incrementing address of R0 by
4 bytes
     LDRB R3, [R0, #4]!
     LDRB R4, [R0, #4]!
     AND R1, R1, R5       ;obtaining last four bits of R1 using bitmask register R5 and
storing result back to R1
     AND R2, R2, R5       ;obtaining last four bits of R2 using bitmask register R5 and
storing result back to R2
     AND R3, R3, R5
     AND R4, R4, R5
     MOV R1, R1, LSL #12  ; first four bits of 16 bit halfword
     MOV R2, R2, LSL #8   ; second four bits of 16bit halfword
     MOV R3, R3, LSL #4   ; next four bits of 16 bit halfword
     ADD R6, R1, R2       ; adding all values in four registers to obtain 16bit halfword
     ADD R6, R6, R3
     ADD R6, R6, R4
     STR R6, Result       ; storing the contents of R6 into the most significant nibble
RESULT(32bit)
HERE    B    HERE


Mask    DCW &000F
            ALIGN
Value   DCD &3B, &4C, &1D, &46
            ALIGN
Result      DCD 0
                    END
```

3)

FLOW CHART:-

```
                    ┌─────────────┐
                    │    START    │
                    └──────┬──────┘
                           │
                           ▼
                  ┌──────────────────┐
                  │  LOAD last byte of│
                  │  VALUE into R0    │
                  └────────┬──────────┘
                           │
                           ▼
                 ┌────────────────────┐
                 │  R1= R0 AND #1     │
                 │  (except LSB make  │
                 │  remaining bits zero)│
                 └─────────┬──────────┘
                           │
                           ▼
                          ◇
              ──NO──────  R1 = 0
                           ◇
                          YES
```

| (Value is ODD) STORE content of R1 into RESULT | (Value is EVEN) STORE content of R1 into RESULT |

```
                    ┌─────────────┐
                    │     END     │
                    └─────────────┘
```

CODE:-

```
*       to find a number is even or odd
        TTL EVEN_ODD
        AREA PROGRAM, CODE, READONLY
        ENTRY
MAIN
        LDRB R0, VALUE              ; loading last byte of R0
        ANDS R1, R0, #1            ; performing AND operetion for R1 and R0, also updating
status resgister
        STR R1,       RESULT        ;storing the final value of R1 back to the
                                    ; RESULT(result=0 says EVEN)
                                    ; result=1 says the value is ODD
HERE    B HERE
VALUE DCD &10
        ALIGN
RESULT DCD &0

        END
```

Learning Outcomes:

I have learnt the architecture of ARM processor and basics of ARM instruction set, in particular the ARM instructions pertaining to computations. Also we have applied this basic knowledge of ARM instruction set in writing assembly language programs for the given set of computational problems.

ARM assembly language programming instructions are easier to understand by the user and also easy to implement.
1) I have learnt how to find FACTORIAL of a number ARM assembly language
2) I have learnt how to add last four bits of a byte from the consecutive bytes of the list. Stored the 16-bit value obtained in most significant nibble of the result.
3) I also learnt how to check, "If the given number is EVEN or ODD" using ARM assembly language programming.