Experiment 3: ARM Assembly - Computations in ARM

- **KEERTHANA RACHURI**
- **EE20B102**

Brief outline of the target in the experiment:

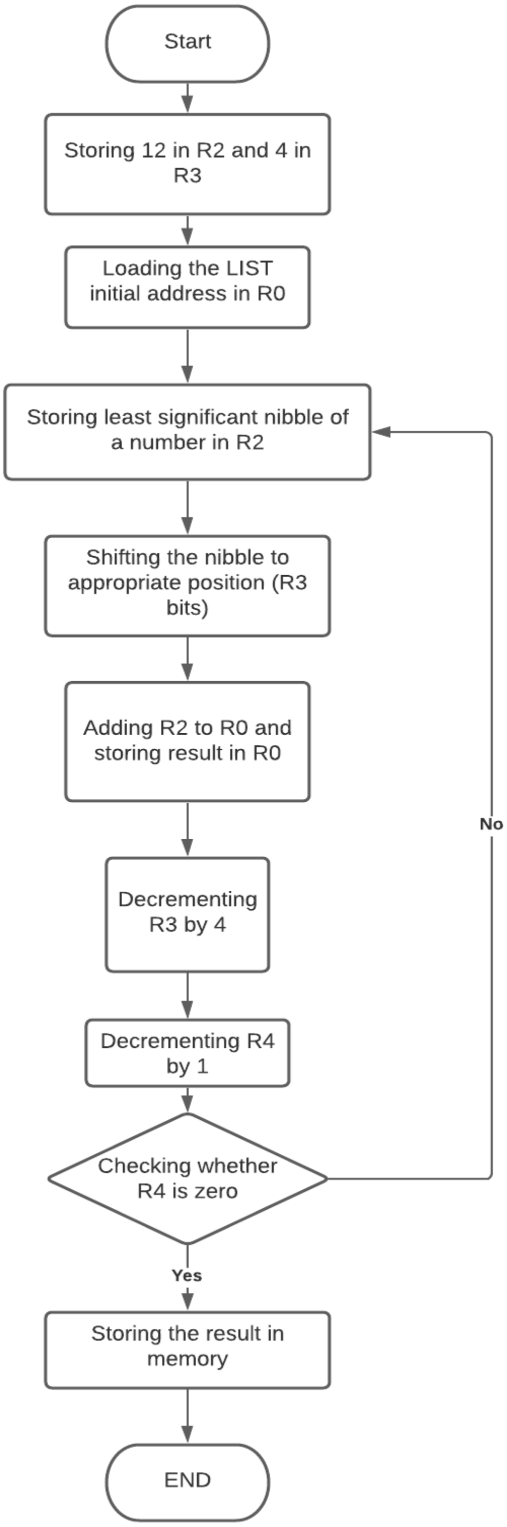
▪ Learn the architecture of ARM processor

▪ Learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations

▪ Write assembly language programs for the given set of (computational) problems

Questions

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

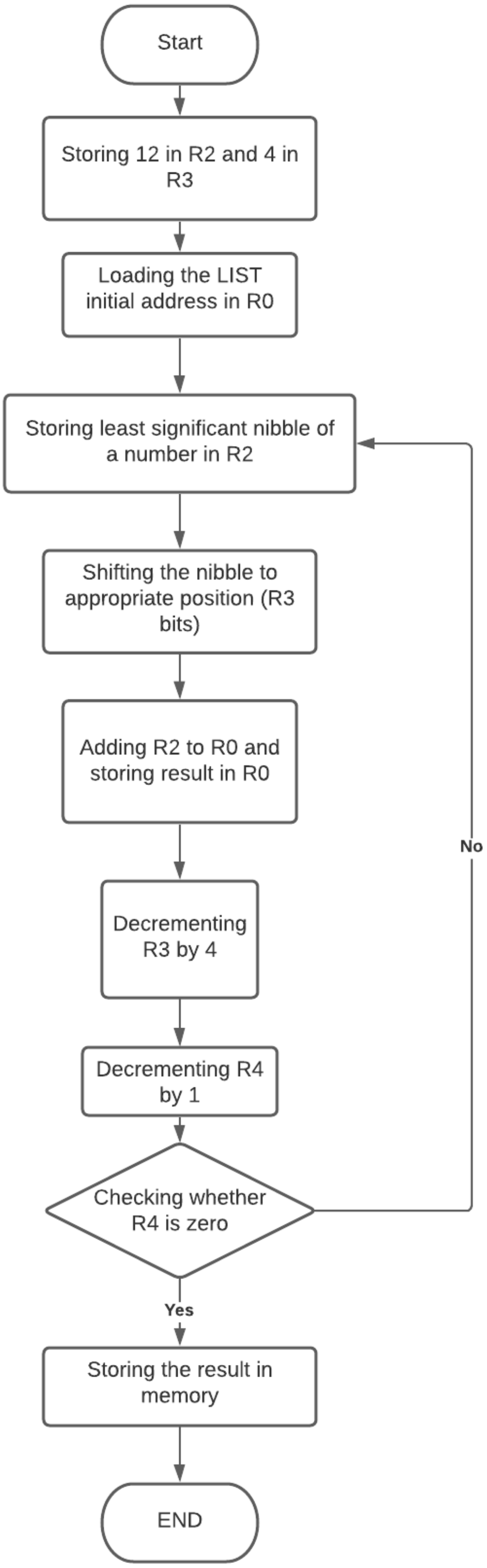

Start
Storing 12 in R2 and 4 in R3
Loading the LIST initial address in R0
Storing least significant nibble of a number in R2
Shifting the nibble to appropriate position (R3 bits)
Adding R2 to R0 and storing result in R0
Decrementing R3 by 4
Decrementing R4 by 1
Checking whether R4 is zero
No
Yes
Storing the result in memory
END

B. code:

```
TTL factorial AREA abc,CODE,READONLY ;

ENTRY LDR R0,NUM1 ; Loading desired number into R1

MOV R3,#1 ;loading 1 in R3


 AGAIN          MUL R5,R3,R0 ; Multiplying R3 and R0 and storing in R5

 MOV R3,R5 ; Moving R5 into R3

SUB R0,R0,#01 ; decrementing R0 by 1

 CMP R0,#01 ; comparing R0 by 1 so that to stop when R0 becomes 1

BNE AGAIN ;

 LDR R2,RESULT

STR R3,[R2] ; final factorial value stored in R4

SWI &11

NUM1 DCW &4 ;

align

RESULT DCD &40000000

END
```

2. 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 significant nibble of the result. Store the result in the 32-bit variable RESULT.

a. **Flow chart:**

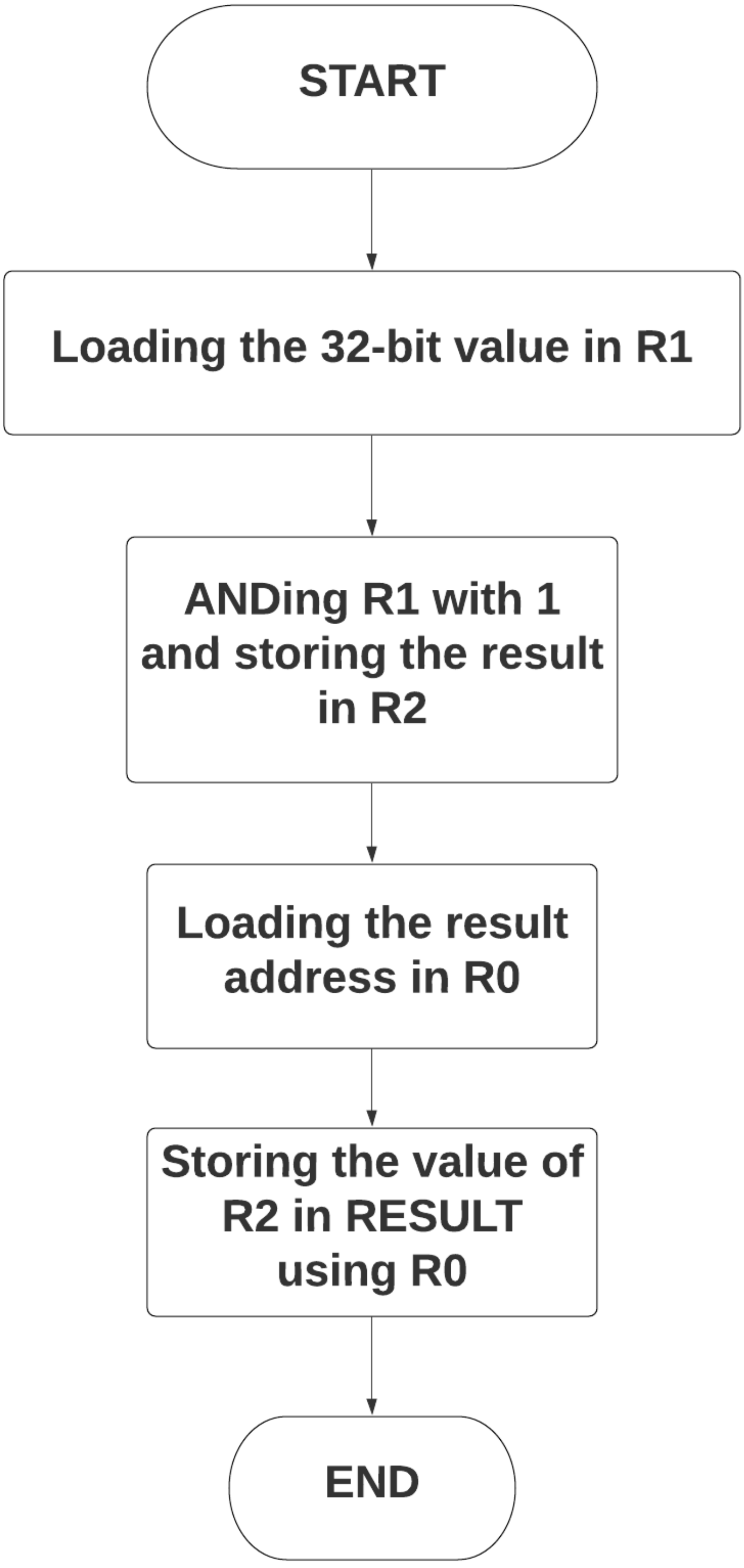

Start
Storing 12 in R2 and 4 in R3
Loading the LIST initial address in R0
Storing least significant nibble of a number in R2
Shifting the nibble to appropriate position (R3 bits)
Adding R2 to R0 and storing result in R0
Decrementing R3 by 4
Decrementing R4 by 1
Checking whether R4 is zero
No
Yes
Storing the result in memory
END

b. **Code:**

```
TTL 16-bit half word
AREA abc,CODE,READONLY ;
           ENTRY
          ADR R0,LIST
          LDR R1,[R0] ; storing the first value in LIST
          MOV R2,#0
          AND R3,R1,#&0F ;
          ADD R2,R2,R3  ; adding  the value of R3 to R2
          MOV R5,#3  ; loading 3 in R5 , R5 acts a counter
BACK LDR R1,[R0,#4]! ; storing the values of LIST in R1
          AND R3,R1,#&0F ; clearing all the bits in R1 other than R
          MOV R2,R2,LSL#4 ; shifting the value of R2 by 4 bits
          ADD R2,R2,R3  ;
          SUB R5,R5,#1 ; decrementing the counter
         CMP R5,#0 ; checking the value of counter
          BNE BACK
          LDR R6,RESULT
          STR R2,[R6]  ; storing final result in R6
          SWI &11
     LIST  DCD &32,&43,&54,&EE
              ALIGN
    RESULT DCD &40000000
END
```

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

a. **Flow chart:**


START
Loading the 32-bit value in R1
ANDing R1 with 1 and storing the result in R2
Loading the result address in R0
Storing the value of R2 in RESULT using R0
END

b. Code:

```
  TTL  ODD or EVEN
AREA abc,CODE,READONLY ;
          ENTRY
         LDR R0,NUM1 ; stroring number in R)
         AND R1,R0,#0X1 ; clearing all the bits other than least significant bit

         CMP R1,#0X1 ; comparing R1 with 1
         BEQ ODD
         MOV R2,#0X00          ; if R2 contains 0 then the number is even
         LDR R0,RESULT
         STR R2,[R0] ; storing final value in R2
          B STOP

ODD MOV R2,#0X01; if R2 contains 1 then the number is odd

      LDR R0,RESULT
         STR R2,[R0] ; storing final value in R2
         SWI &11

NUM1 DCD &7978FFE3
   Align

RESULT DCD &40000000

STOP B STOP

         END
```

MY LEARNINGS FROM THE EXPERIMENT:

- I have learnt how to use basic instructions in ARM assembly.
- I have learnt how to make loops work using branch instructions and status flags.
- I have learnt how to write data into program memory using Definite constant directive (DCD) / Definite constant Word (DCW).
- I have learnt how to access program memory using OFFSET addressing.
- I have learnt about logical and arithmetic shifting of registers using LSL,ROR,LSR,ASR mnemonics.
- I have learnt about the role and usage of R13, R14, R15 in a program.