Experiment 3: ARM Assembly – Computations in ARM

Andapally Snehitha
EE20B008

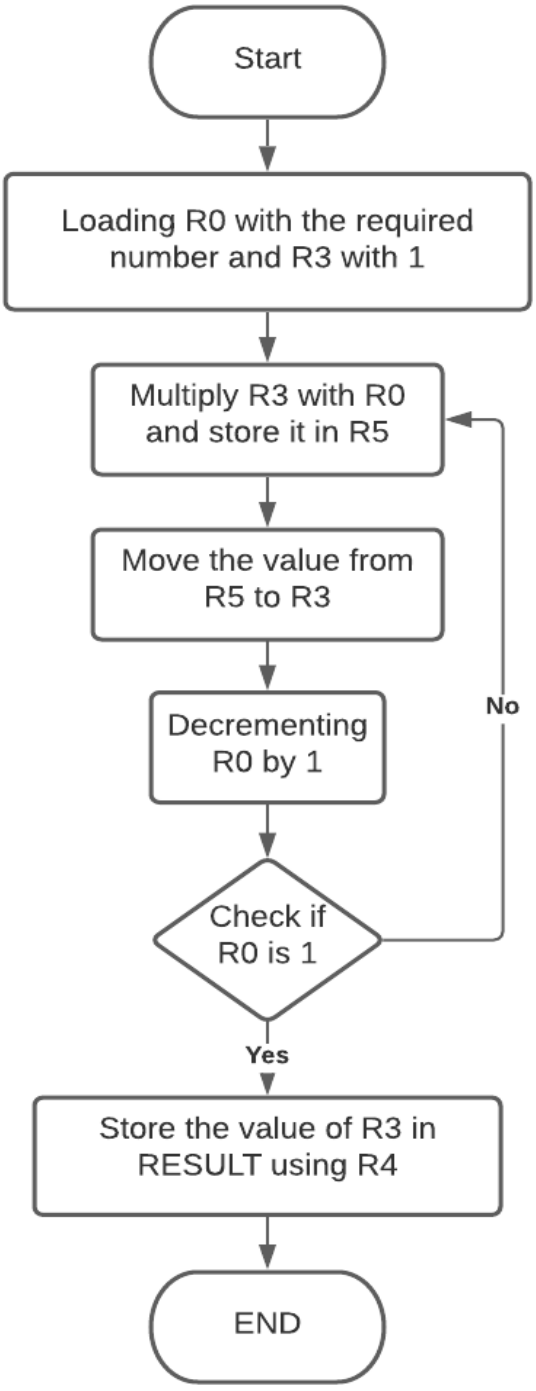
Brief outline of the target in the experiment:

- Learn the architecture of ARM processor.
- Learn basics of ARM instruction set, in particular the ARM instructionspertaining 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. Flow Chart

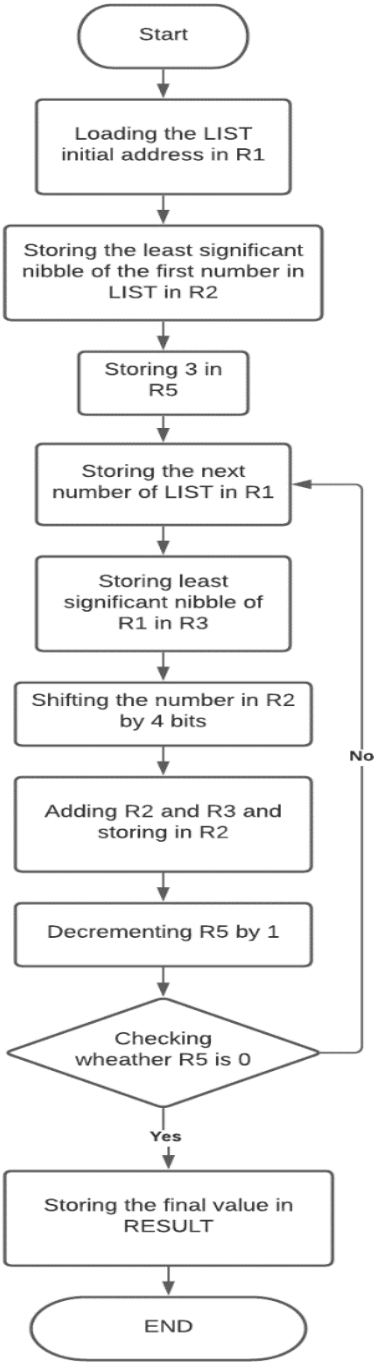

Start
Loading R0 with the required number and R3 with 1
Multiply R3 with R0 and store it in R5
Move the value from R5 to R3
Decrementing R0 by 1
Check if R0 is 1
No
Yes
Store the value of R3 in RESULT using R4
END

b. Code

```
AREA abc, CODE, READONLY;
      ENTRY

      LDR R0, NUM1     ; Loads the number in R0 for which factorial need
to be calculated
      MOV R3, #1       ; Storing 1 in R3


AGAIN       MUL R5, R3, R0
            MOV R3, R5
            SUB R0, R0, #1   ; decrementing R0 by 1
            CMP R0, #1       ; comparing R0 with 1 so that to stop when R0
becomes 1
            BNE AGAIN        ; if not equal, then AGAIN
            MOV R4, R3       ; the final factorial value is stored in R4

            SWI &11

NUM1 DCW &5
            ALIGN


            END
```

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

a. Flow Chart

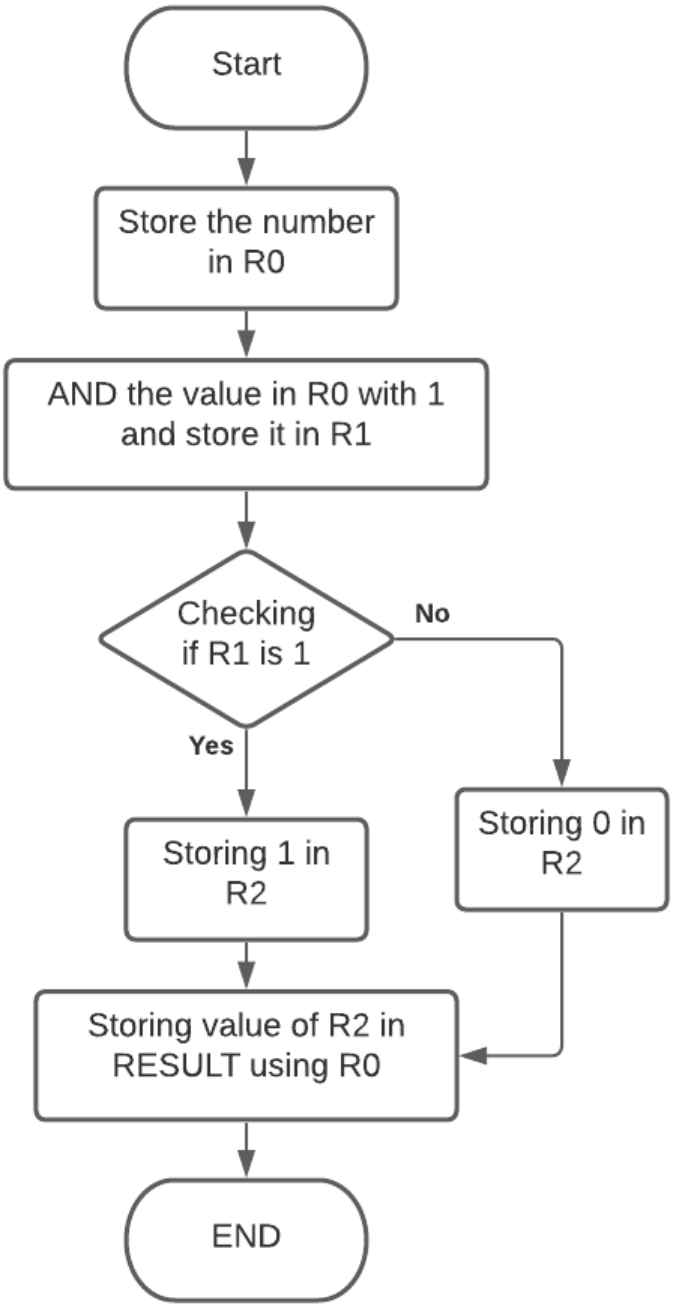

Start
Loading the LIST initial address in R1
Storing the least significant nibble of the first number in LIST in R2
Storing 3 in R5
Storing the next number of LIST in R1
Storing least significant nibble of R1 in R3
Shifting the number in R2 by 4 bits
Adding R2 and R3 and storing in R2
Decrementing R5 by 1
Checking wheather R5 is 0
No
Yes
Storing the final value in RESULT
END

b. Code

```
AREA abc, CODE, READONLY;
ENTRY
ADR R0,LIST
LDR R1,[R0]      ; storing the first value in LIST
MOV R2,#0
AND R3,R1,#&0F   ; clearing all bits other than least significant bit
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 &16, &54, &47, &CC
ALIGN
RESULT DCD &40000000

END
```

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

a. Flow chart


Start
Store the number in R0
AND the value in R0 with 1 and store it in R1
Checking if R1 is 1
No
Yes
Storing 1 in R2
Storing 0 in R2
Storing value of R2 in RESULT using R0
END

b. Code

```
AREA abc, CODE, READONLY;
      ENTRY

      LDR R0, NUM1
      AND R1, R0, #0X1 ; bitwise AND operation
      CMP R1, #0X1           ; comparing R1 with 1 so that to stop when R1
becomes 1
      BEQ ODD                ; if equal, then ODD
      MOV R2, #0X00    ; if R2 contains 0 then the number is EVEN
       B STOP
ODD   MOV R2, #0X01    ; if R2 contains 1 then the number is ODD
      SWI &11

NUM1 DCW &9
      ALIGN
  STOP B STOP

        END
```

Inferences:

- I have learnt how to use basic instructions in ARM assembly.
- I have learnt about the role and usage of R13, R14, R15 in a program.
- I have learnt how to access program memory using OFFSET addressing.
- I have learnt how to make loops work using branch instructions and status flags.