Name: Narasimhan Srikanth Roll Number: EE20B087

EE2016 – Microprocessor Theory and Lab

Experiment 3: ARM Assembly – Computations in ARM

Target of the experiment:

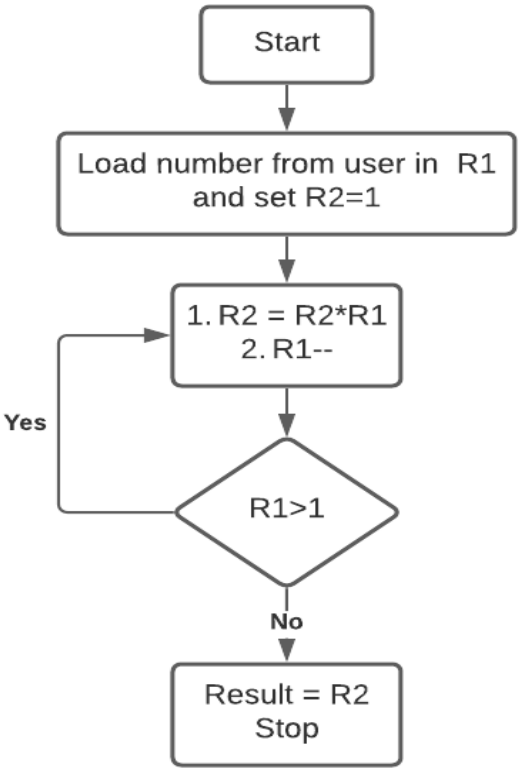
The aim of this experiment is to:
a) learn the architecture of ARM processor
b) learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations
c) go through example programs and (d) write assembly language programs for the given set of (computational) problems

Tasks:

To 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 significant 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. (Implementation should not involve division).

Flowchart and solutions:

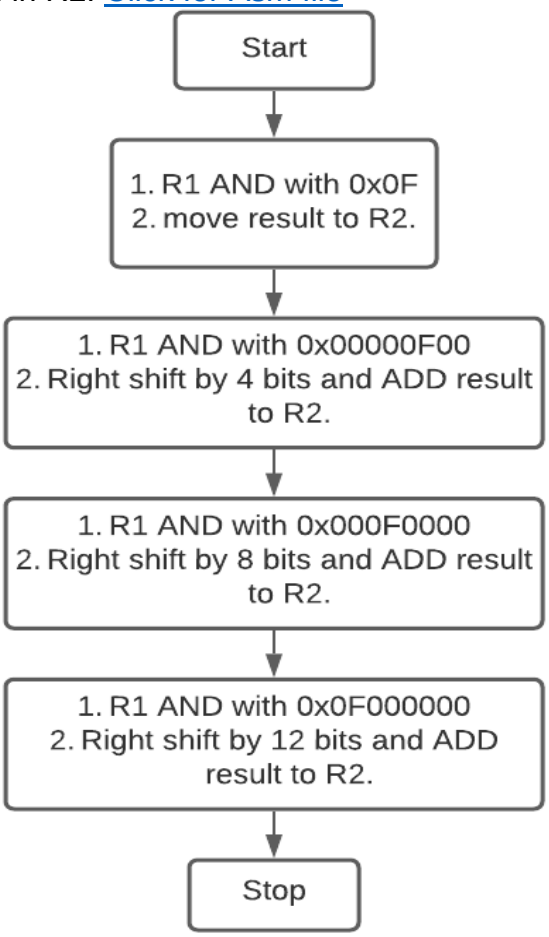
Problem 1: Given the user enters a value, the program must output factorial of that number. The flowchart logic is shown below (in flowchart R1 and R2 are just taken for example). Click for Asm file



```
1  *       to calculate factorial of a number
2
3          TTL     factorial
4          AREA    Program, CODE, READONLY
5          ENTRY
6
7  Main
8          LDR     R1, Value1      ;store 1 in registers R1, R3 and number to find factorial in R2
9          LDR     R3, Value1
10         LDR     R2, Value2
11 Loop
12         MUL     R4, R1, R2      ;Multiply number and store in R1
13         MOV     R1, R4
14         SUB     R2, R3
15         CMP     R2, R3          ;decrement number till it becomes 1. Then break from the loop
16         BNE     Loop
17         STR     R1, Result      ;store answer in Result and also answer is present in R1.
18         SWI     &11
19
20 Value1  DCW     &0001
21         ALIGN
22 Value2  DCW     &0005           ;enter number to find factorial here in hexadecimal
23         ALIGN
24 Result  DCW     0
25         END
```

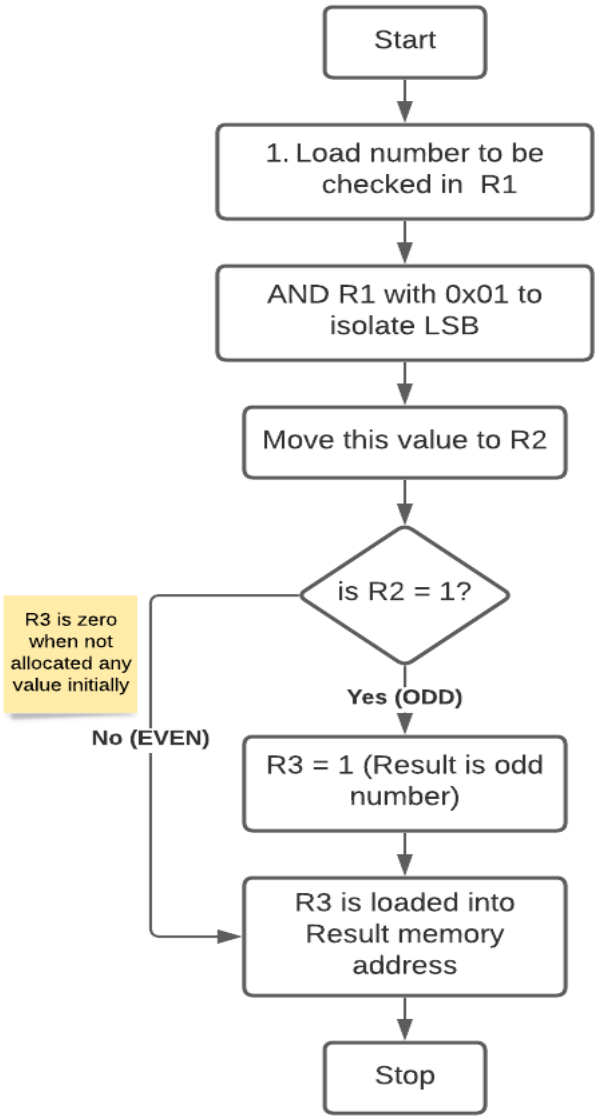
Sample input is 5 and the output loaded in R1 comes out as 120 (0x0078 in hexadecimal number system).

Problem 2: This problem can be neatly explained through an example. If the input is the hexadecimal value 0x0C020B0A, then output is simply the combination of the lower 4 bits of each byte into a 16-bit word, i.e., 0x0000C2BA. User input is loaded in R1 and output is stored in R2. Click for Asm file



```
1  *       Combine the low four bits of each of the four consecutive bytes beginning at LIST into 16bit halfword
2          TTL     16bithalfword
3          AREA    Program, CODE, READONLY
4          ENTRY
5  Main
6          LDR     R1, Valuel              ;value entered by user
7          AND     R2, R1, #0x0000000F     ;ANDing to find the last 4 bits
8          AND     R3, R1, #0x00000F00     ;ANDing to find the 3rd last set of 4 bits
9          MOV     R3, R3, LSR #4
10         ADD     R2, R2, R3
11         AND     R3, R1, #0x000F0000     ;ANDing to find the 5th last set of 4 bits
12         MOV     R3, R3, LSR #8
13         ADD     R2, R2, R3
14         AND     R3, R1, #0x0F000000     ;ANDing to find the 7th last set of 4 bits
15         MOV     R3, R3, LSR #12
16         ADD     R2, R2, R3
17         STR     R2, Result              ;Result available in R2 also
18         SWI &11
19
20 Valuel  DCD     &0C020B0A
21 Result  DCW     0
22         END
23
```

Problem 3: The only bit that will determine whether a binary number is odd or even is the least significant bit (LSB). If LSB is zero, number is even. If LSB is one number is odd. In the program, the output is shown using register R3. If answer is even, R3 is zero. Else R3 is given value one for odd case. Click for Asm file



```
1  *       To find whether 32 bit number is odd or even without division
2          TTL     oddOReven
3          AREA    Program, CODE, READONLY
4          ENTRY
5
6  Main                                    ;the program displays result 0 if EVEN and 1 if ODD
7          LDR     R1, Value1
8          AND     R2, R1, #0x00000001     ;isolating the last bit alone. last bit 1 means odd otherwise even
9          CMP     R2, #1                  ;checking for odd case
10         BNE     leap                    ;if not equal then even and breaks to 12th line
11         LDR     R3, Odd
12 leap    STR     R3, Result              ;R3 contains result
13         SWI     &11
14
15 Value1  DCD     &A234BE11               ;number which we want to check odd or even
16 Odd     DCD     &00000001
17 Result  DCW     0
18         END
```

Inference:

From the above experiments, I was able to explore arithmetic and logical instructions which are available in ARM assembly language. I also got a feel of the IDE used in this week's lab session. Detailed study of the instruction set also gave a lot of insights into the register structure and processor modes of the ARM microprocessor.