NAME: ROHIT KUMAR

ROLL NO.: EE20B111

EE2016 Microprocessor Lab & Theory Jul - Nov 2021

EE Department, IIT Madras.

(Lab Experiment-3 Report)

TITLE: ARM Assembly - Computations in ARM

1.) 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.

2.) Equipment's, Hardware Required:

The list of equipment's, components required are:

1. KEIL 5 IDE for ARM

2. Flash magic software for programming flash memory

3. ARM7 hardware kit

4. USB to serial converter

5. Serial cross cable

It is purely an experiment based on emulation only. (The hardware details given here is to understand the context)

3.) Tasks: Engineering Problems (Questions from the handouts)

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. (You implementation should not involve division).

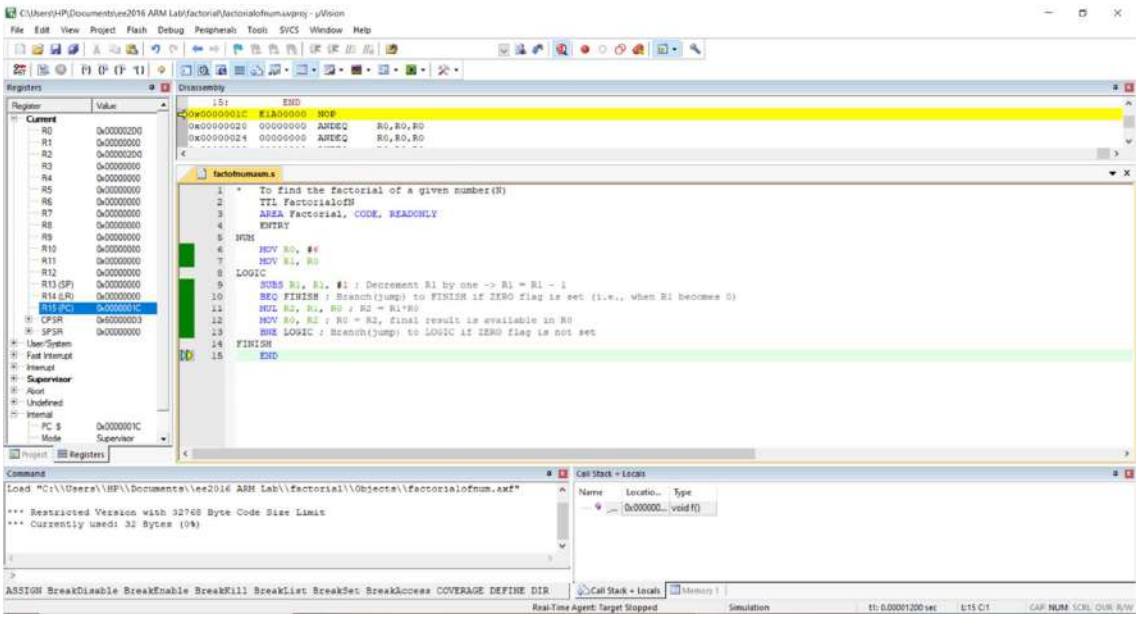
4.) Solutions to the problems:

Problem-1: Factorial of given number:

Code part:

```
*       To find the factorial of a given number(N)

        TTL FactorialofN

        AREA Factorial, CODE, READONLY

        ENTRY

NUM

        MOV R0, #6      ; R0 = 6

        MOV R1, R0      ; R1 = R0 = 6

LOGIC

        SUBS R1, R1, #1  ; Decrement R1 by one -> R1 = R1 - 1

        BEQ FINISH       ; Branch(jump) to FINISH if ZERO flag is set (i.e., when R1 becomes 0)

        MUL R2, R1, R0   ; R2 = R1*R0

        MOV R0, R2       ; R0 = R2, final result is available in R0

        BNE LOGIC   ; Branch(jump) to LOGIC if ZERO flag is not set

FINISH

        END
```

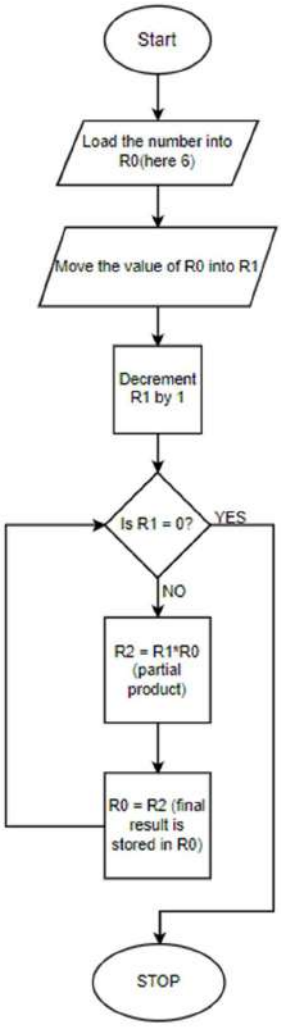
* The output obtained on debugging is:



Since 6! = 720 in decimal system , and 720 is equivalent to 0x2D0 (in hexadecimal).

The result is stored in R0 with a value of 0x000002D0 (equal to 6!).

Flowchart part:



Logic: The main logic is to compute the factorial of the given number, by one by one computing its partial products until when multiplied by one. (6! = ({[(6x5) x (4)]x3}x2)x1 = 6x5x4x3x2x1 = 720).

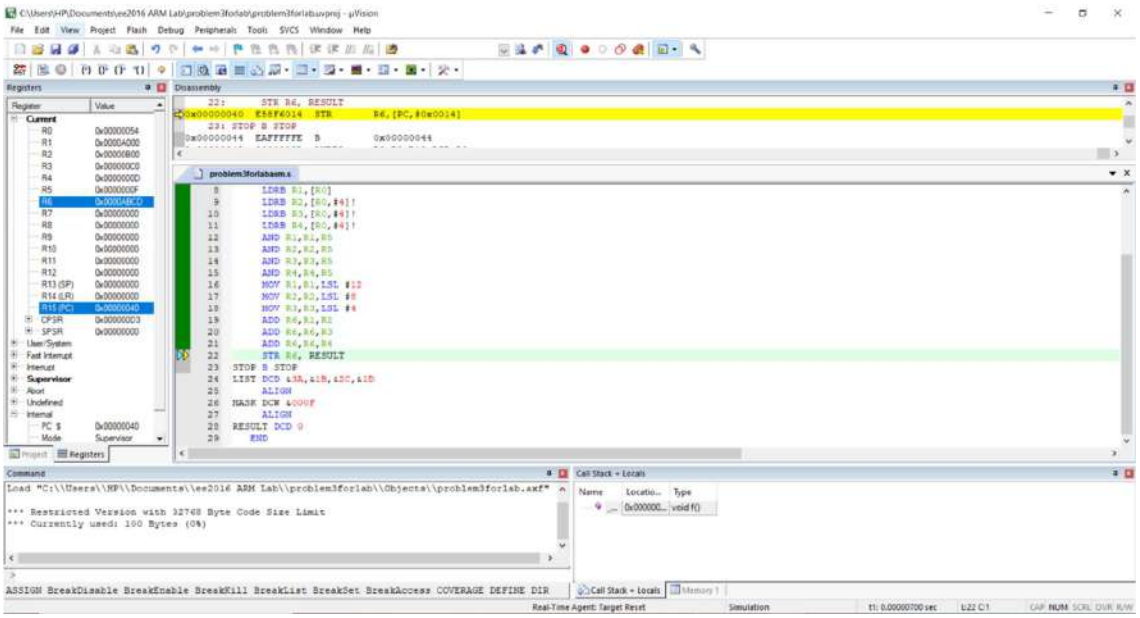
Problem-2: Combine the low four bits of each of the four consecutive bytes:

Code part:

* Combine the low four bits of each of the four consecutive bytes

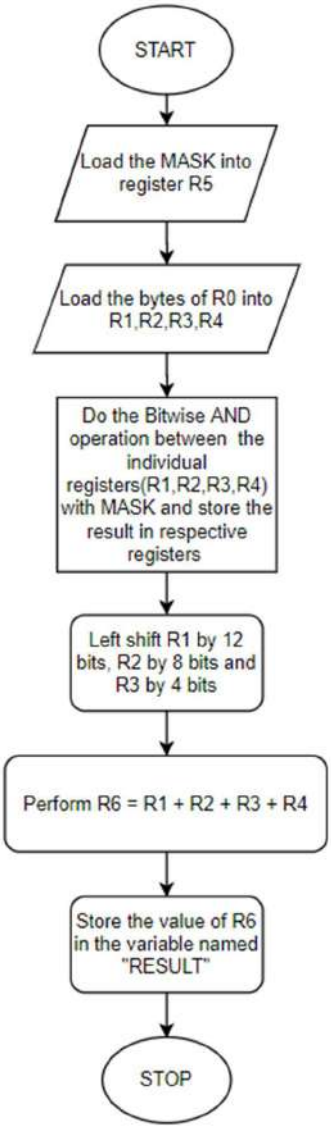
```
        TTL storelowfourbits
        AREA lowfourbit,CODE,READONLY
        ENTRY
START
        LDR R5,MASK
        LDR R0,=LIST
        LDRB R1,[R0]
        LDRB R2,[R0,#4]!
        LDRB R3,[R0,#4]!
        LDRB R4,[R0,#4]!
        AND R1,R1,R5
        AND R2,R2,R5
        AND R3,R3,R5
        AND R4,R4,R5
        MOV R1,R1,LSL #12
        MOV R2,R2,LSL #8
        MOV R3,R3,LSL #4
        ADD R6,R1,R2
        ADD R6,R6,R3
        ADD R6,R6,R4
        STR R6, RESULT
        STOP B STOP
        LIST DCD &3A,&1B,&2C,&1D
        ALIGN
MASK DCW &000F
        ALIGN
RESULT DCD 0
        END
```

* The output obtained on debugging is:



The RESULT is stored in R6 which contains the low four bits. From above, the result(value in R6) for the LIST of 3A, 1B, 2C, 1D is 0x0000ABCD, which are the four low bits as required.

Flowchart part:

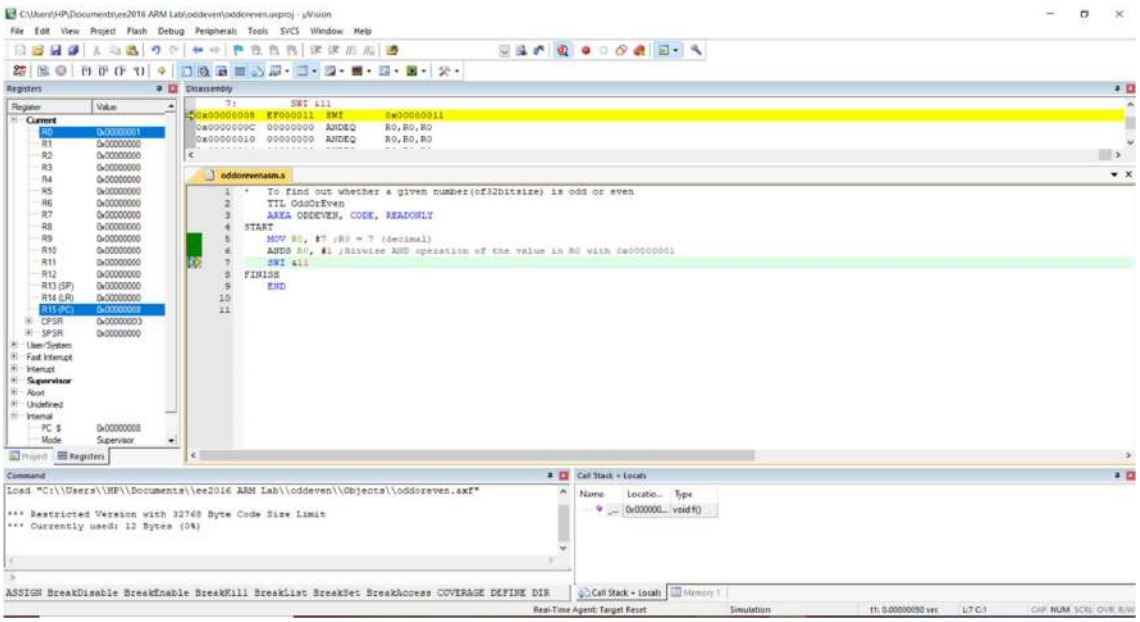


Problem-3: To identify whether a number is even or odd:

Code part:

```
*       To find out whether a given number(of32bitsize) is odd or even
        TTL OddOrEven
        AREA ODDEVEN, CODE, READONLY
START
        MOV R0, #7 ;R0 = 7 (decimal)
        ANDS R0, #1 ;Bitwise AND operation of the value in R0 with 0x00000001
        SWI &11
FINISH
        END
```

* The output obtained on debugging is:



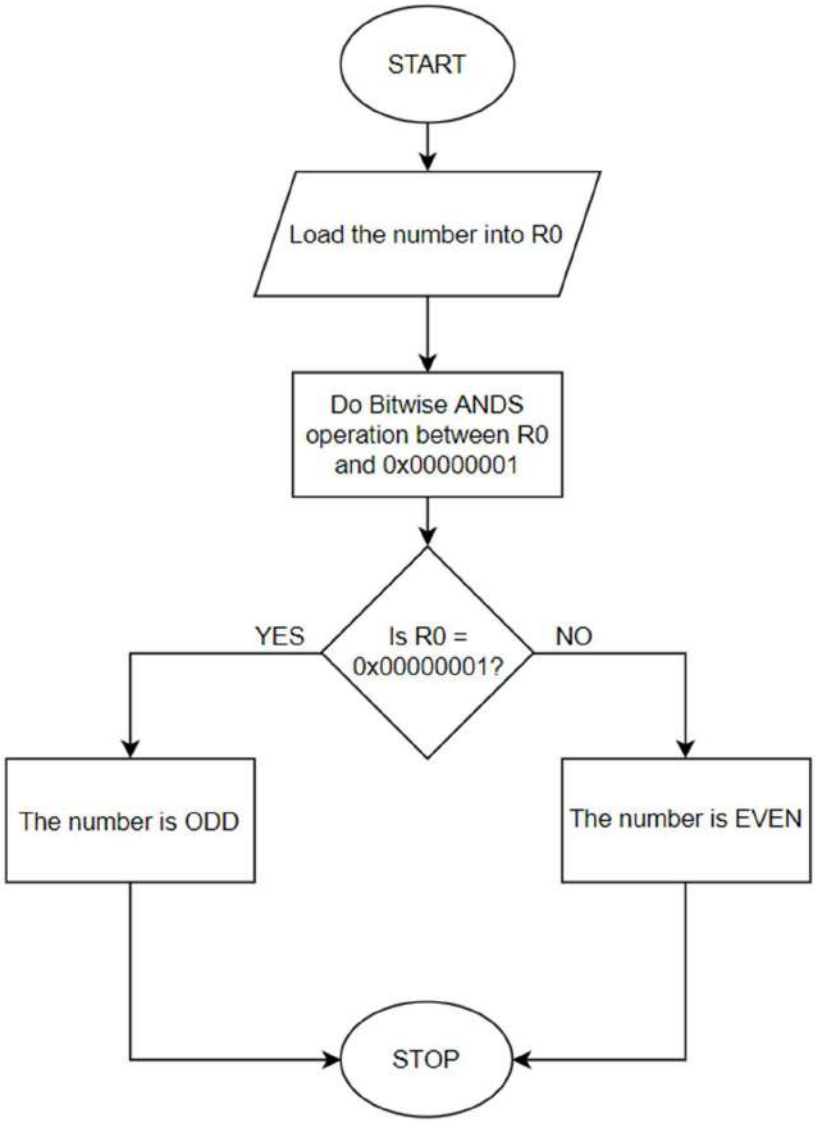
The value stored in R0 determines whether it is even or odd.

If R0 = 0x00000001 (i.e., 1 in decimal) => The number is odd.

If R0 = 0x00000000 (i.e., 0 in decimal) => The number is even.

The value obtained for R0 in above example (number = 7) is 1, this indicates the number(=7) is odd.

Flowchart part:



Logic: The logic behind this is very simple. The LSB bit (last bit) of the number determines whether it is odd or even. If LSB = 1, the number is odd or else even. To determine the LSB, ANDS operation can do this job, the number to which it should be ANDed should be chosen wisely and here it is 0x00000001.

5.) Inferences/Learning from the experiments:

- Learnt about the ARM architecture, and some examples from book WELSH.
- Understood to handle with KEIL U VISION software for our emulation-based experiments purpose for assembly programming.
- Learnt some ARM instruction set to make our programs using simple instructions like MOV, LDR, ADD, AND etc. to branch instruction like BEQ, B, BNE etc. to accomplish our purpose.
- Learnt about the special purpose registers like PC, LR, SP, GPRs etc. and various flags, and how the ARM provides different modes to operate on.