# EE2016_MUP_LAB_EXPERIMENT_06_ARM_EMULATION
MIRUDHULA J | EE23B046

## OBJECTIVES:

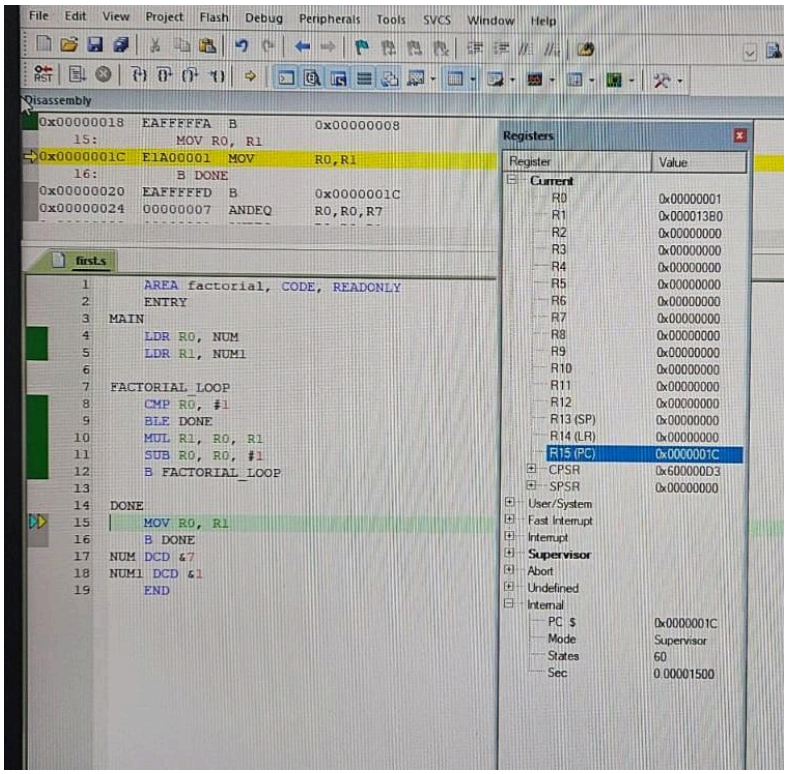
- Learn the architecture of ARM processor
- Learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations
- write assembly language programs for the given set of problems

## CODE USED:

**PROBLEM_1:** Compute The Factorial of a given number using ARM processor through assembly programming.

```
        AREA FACTORIAL, CODE, READONLY
        ENTRY
MAIN
        LDR R0, NUM
        LDR R1, NUM1

FACTORIAL_LOOP
        CMP r0, #1 ; Compare if r0 is 1
        BLE DONE ; If r0 <= 1, end loop
        MUL r1, r0, r1 ; Multiply r1 by r0
        SUB r0, r0, #1 ; Decrement r0 by 1
        B FACTORIAL_LOOP ; Repeat loop

DONE
        MOV r0, r1 ; r0 now holds the factorial result
        B DONE
NUM DCD &9
NUM1 DCD &1
        END
```

**OUTPUT:**



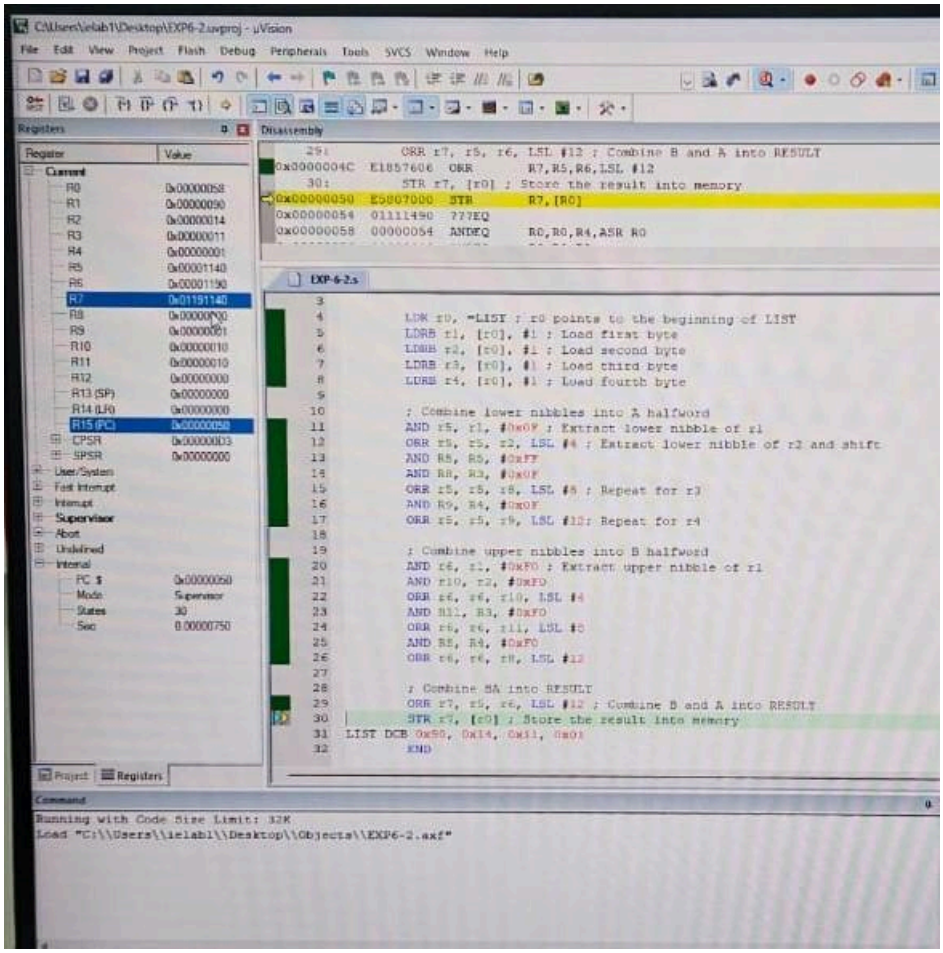

**The output is seen in register R0**

**PROBLEM_2:** Combine the low four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword. Call it as A halfword. Similarly, combine the higher four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword, which is called as 'B' halfword. Combine BA (in that order) and store the result in the 32-bit variable RESULT. (Meaning store B in the higher 16-bit and A in the lower 16-bit of RESULT).

```
        AREA COMBINE_BYTES, CODE, READONLY
        ENTRY

        LDR r0, =LIST ; r0 points to the beginning of LIST
        LDRB r1, [r0], #1 ; Load first byte
        LDRB r2, [r0], #1 ; Load second byte
        LDRB r3, [r0], #1 ; Load third byte
        LDRB r4, [r0], #1 ; Load fourth byte

        ; Combine lower nibbles into A halfword
        AND r5, r1, #0x0F ; Extract lower nibble of r1
        ORR r5, r5, r2, LSL #4 ; Extract lower nibble of r2 and shift
AND R5, R5, #0xFF
AND R8, R3, #0x0F
        ORR r5, r5, r8, LSL #8 ; Repeat for r3
AND R9, R4, #0x0F
        ORR r5, r5, r9, LSL #12; Repeat for r4
```

```
; Combine upper nibbles into B halfword
        AND r6, r1, #0xF0 ; Extract upper nibble of r1
AND r10, r2, #0xF0
        ORR r6, r6, r10, LSL #4
AND R11, R3, #0xF0
        ORR r6, r6, r11, LSL #8
AND R8, R4, #0xF0
        ORR r6, r6, r8, LSL #12

        ; Combine BA into RESULT
        ORR r7, r5, r6, LSL #12 ; Combine B and A into RESULT
        STR r7, [r0] ; Store the result into memory
LIST DCB 0x12, 0x34, 0x56, 0x78
        END
```

**OUTPUT:**

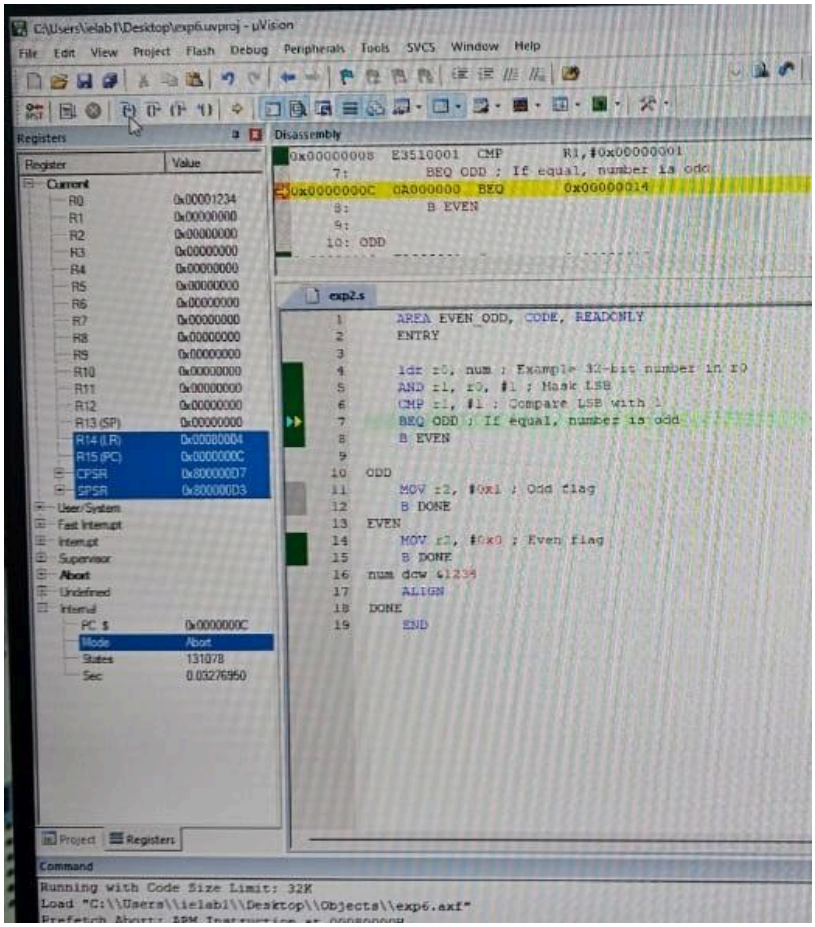



**The output can be seen in register R7**

**PROBLEM_3:** Given a 32 bit number, identify whether it is an even or odd. (You implementation should not involve division).

```
    AREA EVEN_ODD, CODE, READONLY
    ENTRY

    ldr r0, num ; Example 32-bit number in r0
    AND r1, r0, #1 ; Mask LSB
    CMP r1, #1 ; Compare LSB with 1
BEQ ODD ; If equal, number is odd
B EVEN

ODD
MOV r2, #0x1 ; Odd flag
B DONE
EVEN
    MOV r2, #0x0 ; Even flag
B DONE
num dcw &1233
ALIGN
DONE
    END
```

**OUTPUT:**




**The output can be seen in register R2**

## PROCEDURE FOLLOWED:

- Wrote the assembly programs for the above problems (one at a time).
- Entered the above program in KEIL software, edit and compile / assemble.
- Run it in the 'debug' mode to see what's happening to the registers.
- Finally, demonstrated its working, toTA

## MY CONTRIBUTION:

- I was responsible for 2nd and 3rd question coding and implementation.