EE2016 LAB
EXPERIMENT 6

Abhinav I S
EE23B002

---

1 Introduction

In this experiment, we write ARM assembly programs, to solve 3 Engineering Problems, and simulate the programs in Keil microvision IDE.

2 Objectives

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. 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).

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

3 Procedures

3.1 Factorial of a number using ARM processor

Code is given below

```
* factorial
                TTL FACTORIAL
                AREA Program, CODE, READONLY
                ENTRY

Main
                LDR             R0, Value1
                SWI     &11
```

```
                MOV     R1, R0
LOOP
                SUB             R1, R1, #1
                CMP     R1, #0
                BEQ     loop_end

                MUL     R4, R0, R1
                MOV             R0, R4
                B LOOP



loop_end
                SWI     &11


Value1 DCD              &00000006

END
```

Here, the input is Value1 (6) in this case, After building the code, we can simulate this inside keil microvision application. The output is shown below

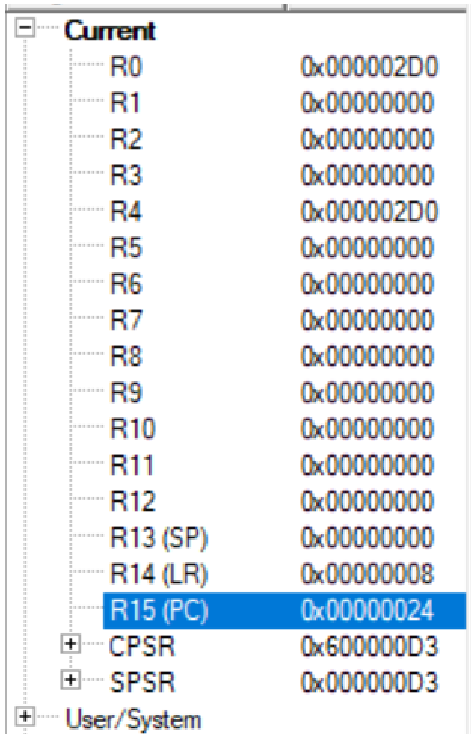

Figure 1: Output (720)

The expected output is $6 * 5 * 4 * 3 * 2 * 1 = 720 = 0x2D0$

3.2 Combining low four bits of each byte

We use an iterating variable to iterate through the 4 elements, and to select the lower bits, we and with 0x0F, and to select the higher bits, 0xF0, we then left shift the result 4 bits, to select the lower bits, we do the exact opposite for select the higher bits.

The code is given below

```
        area list_con ,code ,readonly
        entry
main
        ldr r0, =list
        mov r1, #4
        mov r2, #0
loop
        ldrb r3, [r0], #1
        and r3, r3, #&F
        lsl r2, #4
        orr r2, r2, r3
        subs r1, r1, #1
```

```
        bne loop



                ldr r0, =list
                mov r1, #4
                mov r4, #0


loop1
                ldrb r3, [r0], #1
                and r3, r3, #&F0
                lsr r3, r3, #4
                lsl r4, #4
                orr r4, r4, r3
                subs r1, r1, #1
                bne loop1

                lsl r4, #16
                orr r4, r4, r2

                ldr r3, result
        str r4, [r3]
        swi &11

list    dcb &41
        dcb &32
        dcb &23
        dcb &14
result  dcd &40000000
        end
```

Here the inputs are 41, 32, 23, 14 and hence the expected output is 43211234

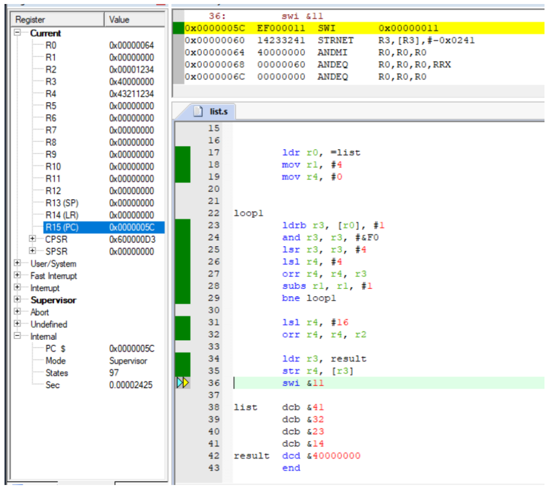

Figure 2: Output in r4

3.3 Checking Evenness

For this program, it is enough to check whether the last bit is set.

The code is given below.

```
        area odd,code,readonly
        entry

main
        MOV R0, #9
        TST R0, #1
        BEQ Bitnotset
        MOV R1, #1
        B exit

Bitnotset
        MOV R1, #2
        B exit

exit
        B exit
        END
```

TST is the instruction used to test the bit

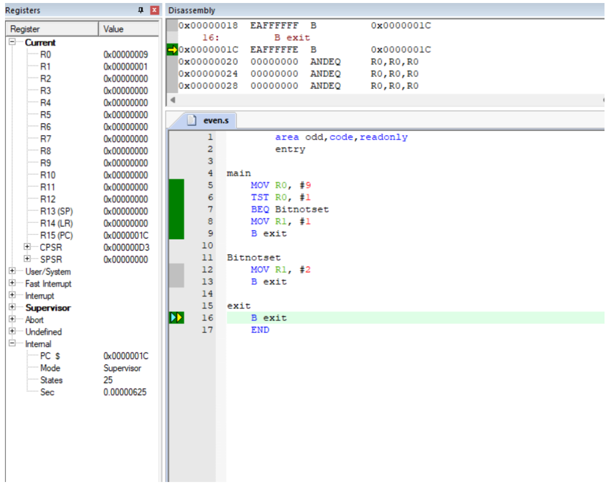

Figure 3: Checking Evenness result stored in Reg R1

R1 contains 1, if it is odd, and 2 if it is even.

4 Conclusion

This experiment provided a comprehensive overview of the ARM architecture and its instruction set, particularly focusing on computational instructions, and proficiency in using the Keil Microvision software