Mohammad Hashemi | 97243073

Instructor's Name: Dr. Dara Rahmati

April 28, 2021

HW05 Report

Microprocessor and Assembly Language Course - Spring 2021

1. **Which of the followings are illegal?**

a) MOV R2, #0x50000

Illegal. We cannot load values larger than 0xFF (255) into registers R0 to R12 using MOV instruction.

b) MOV R1, 255

Illegal. If 255 is assumed to be represented in decimal format, the correct form of this instruction is: MOV R1, #255

c) MOV 123, 0x50

Illegal. We cannot put an immediate value into a constant :|

d) MOV R2, #0x50

Correct.

e) MOV R17, #25

Illegal. We do not have R17 in ARM.

f) MOV R1, #0x00

Correct.

g) MOV R23, #0xF5

Illegal. We do not have R23 in ARM.

2. Find the C flag value after each of the following codes.

a) LDR R0, =0xFFFFFF54

LDR R5, =0xFFFFFFC4

ADDS R2, R5, R0

Carry flag will be set to 1.

The result of this addition operation is R2 = 0xFFFFFF18 and a carry is propagated from the second least bit to MSB.

b) MOV R3, #0

LDR R6, =0xFFFFFFFF

ADDS R3, R3, R6

Carry flag will be set to 0.

The result of this addition operation is R1 = 0xFFFFFFFF and there is no carry which will be propagated beyond the bits.

3. Write a program to calculate the GCD of two given numbers.

```
        AREA      RESET, CODE, READONLY


        ENTRY
start
        MOV       R1, #12     ; R1 = 12
        MOV       R2, #8      ; R2 = 8
GCD
      CMP         R1, R2
      SUBGT       R1, R1, R2  ; R1 = R1 - R2
      SUBLT       R2, R2, R1  ; R2 = R2 - R1
      BNE         GCD

      MOV         R0, R1      ; R0 = GCD(R1, R2)
```

And the result of the code above is : GCD(12, 8) = 4

4. Write a program to compute factorial(reversed(n)).

```
            AREA     RESET, CODE, READONLY


            ENTRY
start
            MOV    R0, #0xE0000000  ; = 1110 0000 0000 0000 0000 0000 0000 0000
            MOV    R1, #0
            MOV    R2, #32          ; since we have 32-bit register

REVERESE_LOOP
            LSRS   R0, #1
            ADDCS  R1, R1, #1       ; add if carry flag is set
            LSL    R1, #1
            SUB    R2, R2, #1
            CMP    R2, #1
            BNE    REVERESE_LOOP
            ADD    R1, R1, #1
            MOV    R0, R1           ; R0 = Reverse(n)
                                    ; = 0000 0000 0000 0000 0000 0000 0000 0111
            MOV    R10, #1
FACTORIAL
            CMP    R0, #1
            BEQ    DONE
            MUL          R10, R0, R10
            SUB          R0, R0, #1
            B            FACTORIAL
DONE
```

For an example, we initialize R0 to 0xE000000 which is equal to 7 as decimal and the reverse of this number is put in R1. And the factorial of 7 is 5040 which is set into the R10.

5. Write a program to count the number of "101" occurrence in a binary number.

```
            AREA     RESET, CODE, READONLY


            ENTRY
start
            LDR    R10, =97243073   ; input     = 0000 0101 1100 1011 1100 1111 1100 0001
            MOV    R0, #0           ; counter
            MOV    R1, #7           ; 7 = 111 in binary
            MOV    R2, #5           ; 5 = 101 in binary
            MOV    R3, #29          ; 29 = 32 - 3
LOOP
            AND    R4, R10, R1      ; fetch the three least bits in R10
            CMP    R4, R2
            ADDEQ  R0, #1
            LSL    R1, #1           ; next 3 bits in the input number
            LSL    R2, #1           ; shift the pattern 1 to the left
            SUB    R3, R3, #1       ; loop counter
            CMP    R3, #0           ; end loop
            BGT    LOOP
```

Number of 101 occurrence in 0000 0101 1100 1011 1100 1111 1100 0001 is 2 and the result is set into the R0 after the execution of the code above.

6. Write a program to toggle the i'th to j'th bits.

```
                  AREA     RESET, CODE, READONLY
                  ENTRY
start


                  MOV    R0, #0xFFFFFFFF
                  MOV    R1, R0
                  MOV    R2, #3                       ; R2 = i
                  MOV    R3, #8                       ; R3 = j
                  MOV    R4, #171                     ; input    = 0000 0000 1010 1011
                  MOV    R5, #31
                  LSL    R0, R2                       ; R0 = R0 << i
                  SUB    R3, R5, R3
                  LSR    R1, R3                       ; R0 = R0 >> j
                  AND    R1, R1, R0                   ; R1 = 0000 0001 1111 1000
                  EOR    R4, R4, R1                   ; R4 = result = 0000 0001 0101 0011
```

The result of this program are shown in the comments in the source code above.