**ARM Assembly – Computations in ARM**

**Aim:** a) Learn the ARM architecture

b) Learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations

**Tasks:**

1. Compute the factorial of a given number N using ARM processor through assembly programming
2. Flowchart:

Start

Load R0 (R0 = 1)

Load R1 (R1 = N + 1)

Load R3 (R3 = 1)

Multiply R3, R0 and store in R3

(R3 = R0 × R3)

Increment R0

(Loop counter)

NO

Is R0 equal to R1

YES

End

1. Code:

        ; Result Will be stored in R3

        TTL     factorial

        AREA    Program, CODE, READONLY

Main    MOV R0, #1      ; Initialize

        LDR R1, num     ; Upper limit of loop

        ADD R1, R1, #1

        MOV R3, #1      ; Intialize result (0! = 1)

loop    MUL R3, R0, R3  ; Mulitply R3 with R0 & store result in R3

        ADD R0, R0, #1  ; Increase count by 1

        CMP R0, R1      ; Compare with upper limit

        BNE loop        ; Branch if not equal

        SWI &11         ; Break Point

num     DCW &6

        align

        END

1. Combine the low four bits of each of the four consecutive bytes beginning at LIST into one of 16-bit halfword. The value of the LIST goes into the most significant nibble of the result. Store the result in the 32-bit variable RESULT
2. Flowchart:

* R0 contains address & points to values in LIST and gets increased by 1 in each loop
* R1 is used to store the no of LEFT SHIFTS required in each loop. Initially R1=12.

Each time loop gets executed R1 gets decreased by 4 (12 8 4 0). Once R1 becomes negative (-4), NEGATIVE status Bit will be set and loop stops

* In each loop the appropriately left shifted value is added to R5 and will be stored R5
* Final Result will be stored at address 0x40000000 (RESULT = R5)

START

Load LIST address in R0

Set R1 = 12 (No of left shifts, initially 12)

Also, R5 = 0 (RESULT)

Load R2 ( R2 = [R0] )

R0 = R0 + 1

Store Last four bits

R2 = R2 & 0x0F

Left shift value and add it to R5

R5 = R5 + Left shifted value

Decrement R1 by 4

R1 = R1 – 0x04

NO

Is R1 Negative

YES

Store result at 0x40000000

STOP

1. Code:

        TTL     Combine low four bits of four BYTES

        AREA    Program, CODE, READONLY

Main    ADR R0, LIST            ; Load LIST Address

        MOV R1, #12             ; R1 stores no of Left SHIFTS (Initially 12)

        MOV R5, #0              ; RESULT in R5. Initialize with ZERO

loop    LDRB R2, [R0], #1       ; Read BYTE at address R0, then R0 = R0 + 1

        AND R2, R2, #0xF        ; R2 = R2 & 0xF (BITWISE AND)

        ADD R5, R5, R2, LSL R1  ; R5 = R5 + (value in R2 shifted by R1 times)

        SUBS R1, #0x4           ; R1 = R1 - 4, SET Negative STATUS Bit if R1 < 0

        BMI stop                ; STOP if value is negative, LOOP Ends

        B loop

stop    LDR R0, RESULT

        STR R5, [R0]            ; Store RESULT at address 0x40000000

        SWI &11

LIST    DCB &1A, &1F, &1B, &1C  ; List of Values

RESULT  DCD &40000000

        END

1. Given a 32-bit number, identify whether it is an even or odd without using divison
2. Flowchart:

START

Load R0 (R0 = value)

Store LSB in R1

(R1 = R0 & 0x01)

END

1. Code:

        TTL     ODD or EVEN

        AREA    Program, CODE, READONLY

Main    LDR R0, value   ; Store Input

        AND R1, R0, #1  ; Bitwise AND

                        ; If R1 is ZERO, Value is even

                        ; If R1 is ONE, Value is odd

        SWI &11

value   DCW &9

        align

        END

**Inferences:**

* ARM supports seven processor modes. The modes other than User mode are known as privileged modes and have full access to system resources
* ARM is a RISC architecture with fixed 32-bit length instruction size
* Operands are passed through Barrel Shifter which means they can be modified before it is used
* Immediate operand values can be loaded into registers using MOV instruction
* Only instructions having an S qualifier affects the flag bits
* In ARM, instructions can have conditional field which allows them to be executed conditionally