LAB-4

BRANCHING AND LOOPING, CODE CONVERSION

Objectives:

In this lab, students will be able to

- Learn different kinds of branching instructions.
- Understand looping code conversion programs

- Write an assembly program to convert a 2 digit hexadecimal number into unpacked ASCII.

```
        AREA RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000          ; stack pointer value when stack is empty
        DCD Reset_Handler       ;reset vector
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
Reset_Handler
        LDR R0,=NUM
        LDR R3,=RESULT
        LDRB R1,[R0]            ; load hex number into register R1
        AND R2,R1,#0x0F         ;mask upper 4 bits
        CMP R2,#09              ;compare the digit with 09
        BLS DOWN                ;if it is lower than 9 then jump to down label
        ADD R2,#07              ;else add 07 to that number
DOWN
        ADD R2,#0x30            ;Add 30H to the number, Ascii value of first digit
        STRB R2,[R3]
        AND R4,R1,#0xF0         ;Mask the second digit
```

```
        MOV R4,R4,LSR#04        ; Shift right by 4 bits
        CMP R4,#09              ; check for >9 or not
        BLS DOWN1
        ADD R3,#07
DOWN1
        ADD R4,#0x30            ; Ascii value of second digit
        STRB R4,[R3,#01]
        NUM DCD 0x000003A
        AREA data, DATA, READWRITE
        RESULT DCD 0
        END
```

❑ Lab Exercises:

1. Write an ARM assembly language program to convert 2-digit hexadecimal number in ASCII unpacked form into its equivalent packed hexadecimal number
2. Write an ARM assembly language program to convert a 32 bit number in the unpacked form into packed form.
3. Write an assembly language program to convert a 2-digit BCD number in to its equivalent hexadecimal number.
4. Write an assembly language program to convert a 2-digit hex number in to its equivalent BCD number.

❑ Additional Exercises:s

1. Write an assembly language program to unpack a 32 bit BCD number into eight 32-bit numbers
2. Write an assembly language program to unpack a 32 bit BCD number into eight 32-bit ASCII numbers
3. Write an assembly language program to find the sum of bits (no. of 1’s) of a 32 bit number available in the memory.