Experiment No. : 1

Date : 11|3|25

1) a) Write Assembly language program for adding first five natural numbers, store the result in the register.

```
        AREA, SUM, CODE, READONLY
ENTRY
        MOV R0, #0      ; R0 = 0
        MOV R1, #1      ; R1 = 0
BACKK
        ADD R0, R0, #1          ; Increment R0 by 1
        ADD R1, R1, R0          ; Add the numbers, sum is in R1
        CMP R0, #5              ; Check if R0 = 5
        BNE BACKK               ; repeat R0 is not equal to 5
GO B GO                         ; Continue branching to GO
      END                       ; end of the assembly module.
```

Experiment No. : 1

Date : 11-3-25

1. b) Write Assembly longuage Program for adding first 10 odd numbers & store their sum in register.

```
        AREA   SUM, CODE, READONLY
ENTRY
        MOV    R1, #1            ; R1 = 1
        MOV    R2, #9            ; R2 = 9, Counter
        MOV    R3, #1            ; R3 = 1
BACKK
        ADD    R3, R3, #2        ; R3 has the odd number
        ADD    R1, R1, R3        ; R1 contains final sum
        SUBS   R2, R2, #1        ; R2 is the counter for 10 numbers
        BNE    BACKK             ; Repeat addition until R2 = 0
GO  B   GO
        END                      ; End of Assembly module
```

Experiment No. : 2

Date : 18-3-25

2) a) Write ALP to comlpite sum of 5 terms of A.P. where first term is 3 and common difference is 7.

```
        AREA    PROG, CODE, READONLY
ENTRY
        MOV R3, #0          ; stores the sum
        MOV R2, #0          ; Acts as a counter
        MOV R1, #3          ; Calculates the terms in AP
BACKK

        ADD R3, R3, R1      ; add numbers , sum is in R3
        ADD R1, R1, #7      ; Increment R1 by 7
        ADD R2, R2, #1      ; Increment R2 by 1 (counter)
        CMP R2, #5          ; Check if R2 = 5
        BNE BACKK           ; Repeat if R2 note equal to 5
GO  B   GO                  ; Continue branching to GO
        END                 ; End of Assembly Module
```

Experiment No. :..

Date :..

2) b) Write ALP to compute sum of squares of 5 numbers starting from 1. Write & use procedure SQU. Store sum in register.

```
        AREA   PG1, CODE, READONLY
ENTRY
        MOV    R7, #0            ; R7 = 0, stores the sum
        MOV    R2, #1            ; R2 = 1, counter
LOOP    BL     SQU               ; Branch & Link to SQU
        ADD    R7, R7, R4        ; Add result stored in R4 to R7 & store in R7
        ADD    R2, R2, #1        ; Increment R2 by 1
        CMP    R2, #6            ; Check if R2 = 6
        BNE    LOOP              ; repeat if R2 not equal to 6
GO      B      GO                ; Continue branching to GO
SQU     MUL    R4, R2, R2        ; Multibly R2 with R2 & store in R4
        MOV    PC, LR            ; Move value of LR to PC
        END                      ; End of Assembly Module
```

18/3/25

Experiment No. : 3

Date : 25/03/25

3a. ALP to add n even numbers & store result in memory location.

```
            AREA  PROG6, CODE, READONLY

N  RN 1                         ; initialise  N  to  R1
RESULT  RN 2                    ; Result  to  R2
EVEN_NUMBER  RN 3               ; Even-Number  to  R3
 ENTRY
            MOV  N, #5                ; N=5
            MOV  RESULT, #0           ; Result = 0
            MOV  EVEN_NUMBER, #2      ; Even Number = 2
            MOV  R4, #0x4000000       ; [R4] = 0x40000000
LOOP
          ADD  RESULT, RESULT, EVEN_NUMBER  :
          ADD  EVEN_NUMBER, EVEN_NUMBER, #2  ; Even NUMBER R+=2
          SUBS  N, N, #1        ; N=N-1 & Check  N>0
          BNE  LOOP             ; Branch not equal then Loop
          STR  RESULT, [R4]     ; Store Result  address in R4
STOP  B  STOP
  END
```

Experiment No. :

Date : 25/03/25

3b. ALP to generate a ~~g~~ GP ~~ds~~ with limit n. Store result in Memory location.

```
        AREA   ODD CODE, READONLY
A    RN     1                          ; A = R1
D    RN     2                          ; D = R2
N    RN     3                          ; N = R3

ENTRY
        MOV  A, #1                     ; A = 1
        MOV  D, #2                     ; D = 2
        MOV  N, #5                     ; N = 5
        MOV  R5, #0x40000000           ; Address location.
LOOP
                 ; R6= A*D
        MUL  R6, A, D
        MOV  A, R6                     ; A= R6
        STR  A, [R5], #4               ; Store val of A in [R5], [R5] +=4
        SUBS N, N, #1                  ; N -= 1 & check if N>0
        BNE  LOOP                      ; if N<0 break
STOP    B  STOP
END
```

25/3/25

Q.4a) ALP to count the number of zeros and ones in a binary number.

```
            AREA , PROG10 , CODE, READONLY

NUMBER     RN  1  ;  Assigns  NUMBER  to R1
NUMONES    RN  10 ;  Assigns  NUMONES  to R10
NUMZEROS   RN  11 ;  Assigns  NUMZEROS  to R11

ENTRY
        MOV R5 , # 0X40000000  ; R5 stores memory 0X40000000
        LDR NUMBER , = 0XA  ; NUMBER has 0XA  i.e. 10
        MOV NUMONES , # 0  ;  Assigned 0
        MOV NUMZEROS, # 0  ;  Assigned 0

LOOP
       LSRS   NUMBER , #1  ; Logical shift right by 1 bit
       ADDCS  NUMZEROS , #1 ; If carry (c) is set(1) then add 1
       ADDCC  NUMZEROS, #1 ; If carry (c) is clear (0) then add
       CMP   NUMBER  , # 0 ; check if NUMBER = 0
       BNE   LOOP          ; If NUMBER ≠ 0 , then branch loop.
       STR NUMONES , [R5] ;          } Store operation.
       STR  NUMZEROS, [R5, #4];      }

STOP B STOP
      END.
```

Experiment No. :.....4.....

Date :.....01|04|25.....

4b. Write ALP to find the avg. of ten 16 bit nos stored in memory.

```
        AREA   PROG CODE, READONLY
ENTRY
        LDR R7, =TABLE        ; Load address of Table into R7
        MOV R0, #9            ; R0=9, loop counter
        LDRH R1, [R7]         ; Load the first (16-bit) value into R1 from Table
BACKK
        LDRH R2, [R7,#2]!;  load the next value from Table into R2 & increment R1
       ADD R1, R1, R2 ;       Add R1 & R2 and store in R1
       SUBS R0, R0, #1;      Decrement loop counter R0
        BNE BACKK ;          If R0 is not zero, repeat the loop
        MOV R3, #10 ;          Set R3 to 10
        MOV R4 , #0 ;         R4=0, used to store quotient
         MOV R5, R1 ;        Copy the sum from R1 to R5
BACKK1
        SUBS R5,R5,R3 ;       subtract 10 from R5
        ADDPL R4, R4,#1;     If result still +ve, increment R4
        BPL BACKK1    ;       Repeat until R5 becomes -ve
        ADDMI R5, R5, R3 ;   If R5 become -ve, ~~add~~ 10 back to remainder
GO B GO
TABLE DCW 1000, 2564, 8936, 344, 5667, 908, 786, 654, 9871, 456;
                                        Data table containing 10 halfword
                                                       values.
```

Experiment No. : 5a Date : 08/04/2025

Q ALP to find factorial of a number.

```
        AREA PROG12, CODE, READONLY
N RN 1 ; Assign register R1 to variable N
FACT RN 2 ; Assign register R2 to variable FACT
ENTRY
      MOV N, #5   ; N=5
      MOV FACT, #1 ; FACT =1
LOOP
      MUL FACT, N, FACT ; FACT = FACT * N
      SUBS N, N, #1     ; N = N-1, (and sets flag)
      BNE LOOP          ; Branch to loop if N is not zero
STOP B STOP             ; Infinite loop to stop execution.
    END                 ; end of assembly program.
```

Experiment No. : 5

Date : 08/04/25

5b. ALP to generate a fibonacci numbers.

```
        AREA PROG, CODE, READONLY
ENTRY
      MOV R1, #1;    R1 initialised to 1
      LDR R2, =TABLE;   Table address located into R2
      LDR R3, = NUMFIBONACCI; Numfibonacci loaded into R3
     LDRB R6, [R3] ; Byte from Numfibonacci loaded into R3
     STRB R1, [R2], #1; Store R1 to table, increment R2
     MOV R3, #0;    R3 = 0
     MOV R5, #1;    R5 = 1
     SUBS R6, R6, #1 ;   decrement R5
BACKK
    ADD R4, R3, R1;   Calculate next fibonacci
    STRB R4, [R2], #1;   Store R4 to table
    MOV R3, R1;   update R3 to prev fibonacci (R1)
    MOV R1, R4;   update R1 to curr. fibonacci (R4)
    ADD R5, R5, #1;   increment loop counter
    CMP R5, R6 ;   compare counter with limit
    BLS BACKK ;   Branch to BACKK if counter <= limit
GO B GO
NUMFIBONACCI DCB 0x0A; define byte at Numfibonacci with value 0x0A
    AREA NUMBER, DATA, READWRITE; define number
TABLE SPACE 60;  Reserve 60 bytes for table
    END
```

15/4/25