Experiment No. : 6a

Date : 15/4/25

ALP to find /calculate nPr

```
        AREA PROG16, CODE, READONLY
DIVIDEND RN 1;  Assign R1 to name DIVIDEND
DIVISOR  RN 2;      "   R2  "   "   DIVISOR
QUOTIENT RN 3;      "   R3  "   "   QUOTIENT
REMAINDER RN 4;     "   R4  "   "   REMAINDER
N RN 10   ;         "   R10 "   "   N (total ! items)
R RN 11   ;         "   R11 "   "   R (items to choose)
NDR RN 12 ;         "   R12 "   "   NDR
ENTRY
        MOV N, #6 ; N=6
        MOV R, #3 ; R=3
        LDR R5, =0X40000000 ; Load address 0X40000000 into R5
        SUB NDR,N,R ; Subtract R from N and store in NDR
        MOV DIVIDEND,N ; Move value of N to DIVIDEND
        BL FACT      ; Branch with link to subroutine FACT
        MOV N,DIVISOR ; Move the value of DIVISOR to N.
        MOV DIVIDEND ,NDR ; Move the value of NDR to DIVIDEND.
        BL FACT      ; Branch with link to FACT
        MOV DIVIDEND,N; Move N to DIVIDEND.
        BL DIV       ; Branch with link to DIV.
        STR QUOTIENT , [R5] ; store value in quotient into
                              ML pointed by R5.

STOP B STOP.
```

Experiment No. ..... Date : .....

```
FACT  MOV  DIVISOR , #1 ; DIVISOR = 1
LOOP2 MUL  DIVISOR, DIVIDEND , DIVISOR ; Multiply DIVIDEND with
                DIVISOR and store in DIVISOR
      SUBS DIVIDEND, DIVIDEND , #1 ; subtract 1
      BNE  LOOP2      ; Branch to LOOP2 if zero flag not set
      BX   LR         ; Branch and Exchange to the
                        address stored in LR.

DIV   MOV  QUOTIENT , #0 ; QUOTIENT = 0
LOOP3 SUBS DIVIDEND, DIVIDEND , DIVISOR ; Substract
      DIVISOR from DIVIDEND
      ADDPL QUOTIENT , QUOTIENT , #1 ; if result of substraction
                was positive or zero, increment QUOTIENT
      BPL LOOP3 ; Branch to LOOP3 if the result of substraction
                was positive or zero.
      ADDMI REMAINDER, DIVIDEND , DIVISOR ;
BX LR ; Branch and Exchange to address stored in LR
      END
```

Experiment No. 6b

Date : 15/4/25

ALP to calculate nCr

```
        AREA PROG17, CODE, READONLY
DIVIDEND RN 1 ; Assign R1 to name DIVIDEND
DIVISOR  RN 2 ;   "  R2  "  "    DIVISOR
QUOTIENT RN 3 ;   "  R3  "  "    QUOTIENT
REMAINDER RN 4 ;  "  R4  "  "    REMAINDER
N        RN 10 ;  "  R10 "  "    N
R        RN 11 ;  "  R11 "  "    R
NDR      RN 12 ;  "  R12 "  "    NDR
ENTRY
        LDR R5, = 0X 40000000 ; load address in R5
        MOV N, #6             ; N=6
        MOV R, # 3            ; R=3
        SUB NDR, N, R         ; NDR = N-R
        MOV DIVIDEND : N      ; DIVIDEND = N
        BL FACT               ; call FACT
        MOV N, DIVISOR        ; N = DIVISOR
        MOV .DIVIDEND , R     ; DIVIDEND = R
        BL FACT               ; call FACT

        MOV R, DIVISOR        ; R = DIVISOR
        MOV DIVIDEND , NDR    ; DIVIDEND = NDR
        B L FACT              ; call FACT
        MOV DIVIDEND , N      ; DIVIDEND = N
        MUL DIVISOR , R, DIVISOR ; DIVISOR = fact of R *
        BL DIV.                    fact of (N-R)
```

```
        STR QUOTIENT, [R5] ; result stored in m1 of R5
STOP B STOP
FACT MOV DIVISOR, #1      ; DIVISOR = 1
LOOP2 MUL DIVISOR, DIVIDEND, DIVISOR ; find recursive product
      SUBS DIVIDEND, DIVIDEND, #1 ; decrement DIVIDEND
      BNE LOOP2
      BX LR ; return from subroutine.
DIV   MOV QUOTIENT, #0
LOOP3 SUBS DIVIDEND, DIVIDEND, DIVISOR ; dividend - divisor
      ADDPL QUOTIENT, QUOTIENT, #1 ; increment quotient
                                     if result positive.
      BPL LOOP3
      ADDMI REMAINDER, DIVIDEND, DIVISOR ; if subtraction is
      BX LR                                -ve find remainder.
      END.
```

Experiment No. : 7a.

Date : 22/4/25

ALP to find sum of digits of a number.

```
        AREA PROG7A, CODE, READONLY
DIVIDEND RN 1 ; Assign R1 to name DIVIDEND
DIVISOR RN 2 ;      "  R2  "   "   DIVISOR
QUOTIENT RN 3 ;     "  R3  "   "   QUOTIENT
REMAINDER RN 4;     "  R4  "   "   REMAINDER
RESULT RN 5;        "  R5  "   "   RESULT
RE ENTRY
        LDR DIVIDEND, =12345 ; Add load 12345 into DIVIDEND
        MOV DIVISOR, #10     ; DIVISOR = 10
        MOV RESULT, #0       ; initialize RESULT as 0
LOOP
        BL DIV ; Branch with link to DIV subroutine
        ADD RESULT, REMAINDER, RESULT ; add REMAINDER to
                    current REMAINDER & store in RESULT
        CMP QUOTIENT, #0 ; Quotient compare with 0
        MOVNE DIVIDEND, QUOTIENT ; if Quotient not equal to 0,
                    move it to DIVIDEND.
        BNE LOOP     ; Branch to loop if not equal.
STOP B STOP
DIV
        MOV QUOTIENT, #0 ; Initialize QUOTIENT = 0
LOOP2
        SUBS DIVIDEND, DIVIDEND, DIVISOR; Subtract DIVISOR
                    from DIVIDEND and store in DIVIDEND.
        ADDPL QUOTIENT, QUOTIENT, #1 ; if the result of subtract-
                    was 0 or +ve increment
                    QUOTIENT by 1.
```

Experiment No. ..... Date : .....

BPL LOOP2 ; Branch to LOOP2 if the result of subtraction was +ve or 0.

ADDMI REMAINDER, DIVIDEND, DIVISOR ; if the result of subs. was -ve, add DIVISOR back to DIVIDEND to get correct REMAINDER.

BX LR ; Branch back to the calling instruction using Link Register.

END.

2/14

Experiment No. : 7 (b) Date : 29/4/25

Q Write an ALP to check whether the given number is palindrome or not.

```
        AREA PROG14, CODE, READONLY
ENTRY
        LDR R1, = 12321  ; R1 = 12321
        MOV R6, R1       ; R6 = R1 (original)
        MOV R2, #10      ; R2 = 10 (divisor)
        MOV R5, #0       ; R5 = 0 (reversed number)
        MOV R10, #10     ; R10 = 10 (for multiplication)
LOOP
        BL DIV           ; Call division subroutine.
        MLA R5, R10, R5, R4 ; R5 = (R10 * R5) + R4 (build reversed no.)
        CMP R3, #0       ; compare quotient with 0.
        MOVNE R1, R3     ; If Quotient != 0, R1 = quotient.
        BNE LOOP         ; If Quotient != 0, loop again.
        CMP R5, R6       ; Compare reversed with original.
        MOVEQ R7, #1     ; If equal, R7 = 1 (palindrome)
        MOVNE R7, #0     ; If not equal, R7 = 0
STOP B STOP              ;     Halt program.
DIV                      ; Division subroutine
    MOV R3, #0           ;   R3 = 0 (Quotient)
LOOP2
    SUBS R1, R1, R2      ; R1 = R1 - R2 (subtract divisor)
    ADDPL R3, R3, #1     ; If R1 >= 0, R3 = R3 + 1 (increment Quotien
    BPL LOOP2            ; If R1 >= 0, loop again
    ADDMI R4, R1, R2     ; If R1 < 0, R4 = R1 + R2 (remainder)
    BX LR                ; Return from subroutine
END
```