Experiment 1

Write an ALP to i) Multiply two 16-bit binary numbers. ii) Add two 64-bit numbers

---

i) ALP to multiply two 16 bit binary numbers.

```
AREA Addition, CODE, READONLY

ENTRY

MOV R2, #0002

MOV R1, #0002

MUL R2, R2, R1

L B L

END
```

OUTPUT: R2=0004

---

ii) **Add two 64-bit numbers**

```
AREA Addition, CODE, READONLY

ENTRY

MOV R0, #0X40000000

LDR R1, [R0]

MOV R0, #0X40000004

LDR R2, [R0]

ADD R3, R1, R2

MOV R0, #0X40000008

STR R3, [R0]

NOP
```

```
NOP

END
```

---

2 Write an ALP to find the sum of first 10 integer numbers.

```
AREA ArryAdd, CODE, READONLY

ENTRY

MOV R0, #05

MOV R1, #00

MOV R2, #0X40000000

LDR R3, [R2]

LOOP ADD R2, R2, #04

LDR R4, [R2]

ADD R3, R3, R4

SUB R0, R0, #01

CMP R0, #00

BNE LOOP

MOV R5, #0X4000001C

STR R3,[R5]

NOP

NOP

END
```

---

3 Write an ALP to find factorial of a number.

```
AREA Factorial, CODE, READONLY
```

```
ENTRY

MOV R0, #05

MOV R1, R0

CMP R0, #01

MOV R2, #01

BLE STOP

LOOP SUB R1, R1, #01

CMP R1, #01

BEQ STOP

MUL R2, R0, R1

MOV R0, R2

B LOOP

STOP NOP

END
```

---

4 Write an ALP to add an array of 16-bit numbers and store the 32-bit result in internal RAM.

```
AREA ADDITION,CODE,READONLY

ENTRY

MOV R5,#6

MOV R0,#0

LDR R1,=VALUE

LOOP LDRH R3,[R1],#2

ADD R0,R0,R3
```

```
SUBS R5,R5,#1

CMP R5,#0

BNE LOOP

LDR R4,=RESULT

STR R0,[R4]

STOP B STOP

VALUE DCD 0x0000,0x1111,0x2222,0x3333,0xAAAA,0xBBBB,0xCCCC

AREA INFO,DATA,READWRITE

RESULT DCD 0x00000000

END
```

5.Write an ALP to find the square of a number (1 to 10) using look-up table.

```
AREA Square, CODE, READONLY

ENTRY

LDR R0, =TABLE ;Load the starting address of the Lookup Table

LDR R1, =6 ;Load no whose square is to be found

MOV R1, R1, LSL#0X2 ;Generate address corresponding to the square of the given no

ADD R0, R0, R1 ;Load address of element in Lookup Table

LDR R3, [R0] ;Get square of given no in R3

NOP

NOP

NOP

;Lookup Table contains SQUARES of nos from 0 to 10 (in HEX)
```

```
TABLE DCD 0X00000000 ;SQUARE OF 0=0

DCD 0X00000001 ;SQUARE OF 1=1

DCD 0X00000004 ;SQUARE OF 2=4

DCD 0X00000009 ;SQUARE OF 3=9

DCD 0X00000010 ;SQUARE OF 4=16

DCD 0X00000019 ;SQUARE OF 5=25

DCD 0X00000024 ;SQUARE OF 6=36

DCD 0X00000031 ;SQUARE OF 7=49

DCD 0X00000040 ;SQUARE OF 8=64

DCD 0X00000051 ;SQUARE OF 9=81

DCD 0X00000064 ;SQUARE OF 10=100

END
```

6.Write an ALP to find the largest/smallest number in an array of 32 numbers.

```
i. AREA Largest, CODE, READONLY

ENTRY

MOV R5, #05

MOV R0, #0X40000000

LDR R1, [R0]

LOOP ADD R0, R0, #04

LDR R2, [R0]

CMP R1, R2

BHI SKIP
```

```
MOV R1, R2
SKIP SUBS R5, R5, #01
CMP R5, #00
BHI LOOP
MOV R0, #0X4000001C
STR R1,[R0]
END
```

ii. AREA Smallest, CODE, READONLY

```
ENTRY
MOV R5, #05
MOV R0, #0X40000000
LDR R1, [R0]
LOOP ADD R0, R0, #04
LDR R2, [R0]
CMP R1, R2
BLE SKIP
MOV R1, R2
SKIP SUB R5, R5, #01
CMP R5, #00
BNE LOOP
MOV R0, #0X4000001C
STR R1,[R0]
END
```

7. Write an ALP to arrange a series of 32-bit numbers in ascending/descending order.

i. AREA ASCENDING,CODE,READONLY

```
ENTRY

MOV R0,#05

AGAIN MOV R1,#05

MOV R2,#0x40000000

LOOP LDR R3,[R2]

ADD R2,R2,#04

LDR R4,[R2]

CMP R3,R4

BGT SKIP

STR R3,[R2]

SUB R2,R2,#04

STR R4,[R2]

ADD R2,R2,#04

SKIP SUB R1,R1,#01

CMP R1,#00

BNE LOOP

SUB R0,R0,#01

CMP R0,#00

BNE AGAIN

NOP

NOP
```

```
END
```

ii. AREA DESCENDING,CODE,READONLY

```
ENTRY
MOV R0,#05
AGAIN MOV R1,#05
MOV R2,#0x40000000
LOOP LDR R3,[R2]
ADD R2,R2,#04
LDR R4,[R2]
CMP R3,R4
BLT SKIP
STR R3,[R2]
SUB R2,R2,#04
STR R4,[R2]
ADD R2,R2,#04
SKIP SUB R1,R1,#01
CMP R1,#00
BNE LOOP
SUB R0,R0,#01
CMP R0,#00
BNE AGAIN
NOP
NOP
END
```

Expt.8.

i) Write an ALP to count the number of ones and zeros in two consecutive memory locations.

```
AREA OnesAndZeros, CODE, READONLY

ENTRY

MOV R6, #0X40000000 ;Load the two numbers

MOV R2, #02 ;2 memory location counter

MOV R0, #00 ;Counter for ZEROS

MOV R1, #00 ;Counter for ONES

LOOP LDR R4, [R6]

MOV R3, #32 ;32 bits counter

HERE MOVS R4, R4, ROR #01

BHI ONE

ADD R0, R0, #01

B SKIP

ONE ADD R1, R1, #01

SKIP SUB R3, R3, #01

CMP R3, #00

BNE HERE

ADD R6, R6, #04

SUB R2, R2, #01

CMP R2, #00

BNE LOOP
```

```
NOP

NOP

END
```

ii. Write an ALP to Scan a series of 32-bit numbers to find how many are negative.

```
AREA NoOfNegativeNumbers, CODE, READONLY

ENTRY

MOV R0, #05

MOV R2, #0X40000000 ;Loading the 5 numbers in this memory location

LDR R1, [R2]

LOOP MOVS R1, R1, LSL #01

BLS SKIP

ADD R3, R3, #01

SKIP ADD R2, R2, #04

LDR R1, [R2]

SUB R0, R0, #01

CMP R0, #00

BNE LOOP

NOP

NOP

END
```