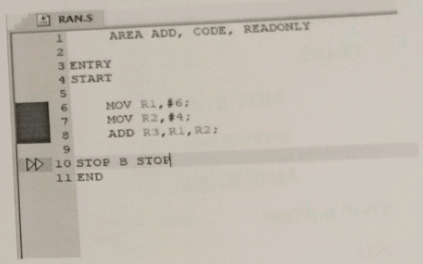
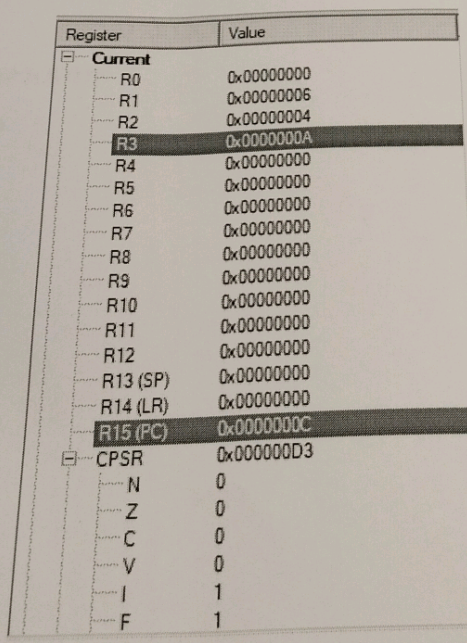
1. Using Keil software, observe the various registers, dump, CPSR, with a simple ALP program.

SIMPLE ALP PROGRAM:-

```
AREA ADD, CODE, READONLY
ENTRY
START
        MOV R1,#6;
        MOV R2,#4;
        ADD R3,R1,R2;
STOP B STOP
END
```



OUTPUT:-



2. Develop and simulate ARM ALP for data transfer, Arithmetic and logical operations.

DATA TRANSFER:-

```
AREA ADD, CODE, READONLY
ENTRY
START
        MOV R1,#6;
        MOV R2,#4;
        MOV R1,R2;
STOP B STOP
END
```

RAN.S

```
1       AREA ADD, CODE, READON
2
3 ENTRY
4 START
5       MOV R1,#6;
6       MOV R2,#4;
7       MOV R1,R2;
8
9
10 STOP B STOP
11 END
```

OUTPUT:-

Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000004
R2	0x00000004
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x0000000C
Mode	Supervisor
States	6
Sec	0.00000050

ARITHMETIC AND LOGICAL OPERATIONS:-

```
AREA ADD, CODE, READONLY
ENTRY
START
        MOV R0,#6;
        MOV R1,#4;
        ADD R2,R1,R0;
        ADC R3,R1,R0;
        SUB R4,R1,R0;
        SBC R5,R1,R0;
        RSB R6,R1,R0;
        RSC R7,R1,R0;
        AND R8,R1,R0;
        ORR R9,R1,R0;
        EOR R10,R1,R0;
        BIC R11,R1,R0;
STOP B STOP
END
```

RAN.S

```
1       AREA ADD, CODE, READONLY
2
3 ENTRY
4 START
5
6       MOV R0,#6;
7       MOV R1,#4;
8       ADD R2,R1,R0;
9       ADC R3,R1,R0;
10      SUB R4,R1,R0;
11      SBC R5,R1,R0;
12      RSB R6,R1,R0;
13      RSC R7,R1,R0;
14      AND R8,R1,R0;
15      ORR R9,R1,R0;
16      EOR R10,R1,R0;
17      BIC R11,R1,R0;
18
19 STOP B STOP
20 END
```

OUTPUT:-

Register	Value
Current	
R0	0x00000006
R1	0x00000004
R2	0x0000000A
R3	0x0000000A
R4	0xFFFFFFFE
R5	0xFFFFFFFD
R6	0x00000002
R7	0x00000001
R8	0x00000004
R9	0x00000006
R10	0x00000002
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000030
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000030
Mode	Supervisor
States	12
Sec	0.00000100

3. Write a program to find the sum of the first 10 integer numbers.

SUM OF THE FIRST 10 INTEGER NUMBERS:-

```
AREA ADD, CODE, READONLY
ENTRY
START
   MOV R0,#10;
   MOV R1,#0;
LOOP
   ADD R1,R0,R1;
   SUBS R0,#01;
   BNE  LOOP;
STOP B STOP
END
```

RAN.S

```
1       AREA ADD, CODE, READONLY
2
3 ENTRY
4 START
5
6       MOV R0,#10;
7       MOV R1,#0;
8 LOOP
9       ADD R1,R0,R1;
10      SUBS R0,#01;
11      BNE LOOP;
12
13 STOP B STOP
14 END
```

OUTPUT:-

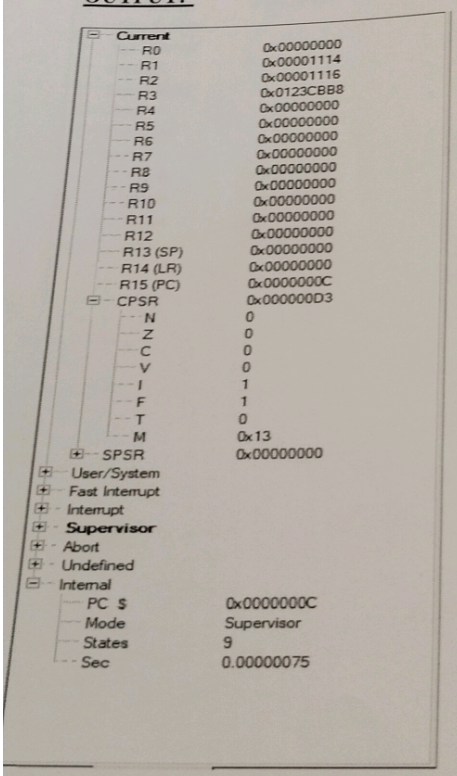
Register	Value
Current	
R0	0x00000000
R1	0x00000037
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x600000D3
N	0
Z	1
C	1
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000010
Mode	Supervisor
States	49
Sec	0.00000408

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

```
AREA MULTIPLICATION, CODE, READONLY
ENTRY
START
        LDR R1,=0x1114
        LDR R2,=0x1116
        MUL R3,R1,R2;
STOP B STOP
END
```

```
1                AREA MULTIPLICATION, CODE, READONLY
2
3 ENTRY
4 START
5       LDR R1,=0x1114
6       LDR R2,=0x1116
7       MUL R3,R1,R2;
8
9 STOP B STOP
10 END
```

OUTPUT:-



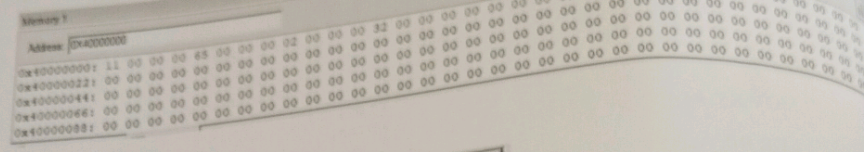
5. Write a program to find the largest or smallest number in an array of 32 numbers.

LARGEST NUMBER:-

```
AREA LARGEST, CODE, READONLY
ENTRY
START
        MOV R5, #4;
        LDR R0, A;
        LDR R2, [R0];
NEXT
        ADD R0, #4;
        LDR R3, [R0];
        CMP R2, R3;
        BHI LARGE;
        MOV R2, R3;
LARGE
        SUB R5, #1;
        BNE NEXT;
        LDR R1, RES;
        STR R2, [R1];
        STOP B STOP
        A DCD 0X40000000;
        RES DCD 0X40000020;
END
```

```
1         AREA LARGEST,CODE,READONLY
2
3 ENTRY
4 START
5
6     MOV R5,#4;
7     LDR R0,A;
8     LDR R2,[R0];
9
10 NEXT
11
12   ADD R0,#4;
13   LDR R3,[R0];
14   CMP R2,R3;
15   BHI LARGE;
16   MOV R2,R3;
17
18 LARGE
19    SUB R5,#1;
20    BNE NEXT ;
21    LDR R1,RES;
22    STR R2,[R1];
23
24 STOP B STOP
25
26 A DCD 0X40000000;
27 RES DCD 0X40000020;
28
29 END
```

OUTPUT:



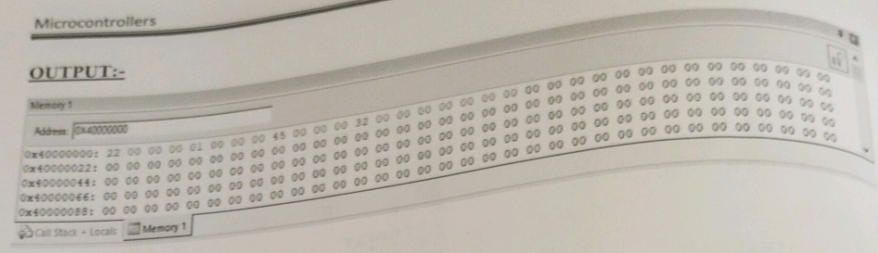
Register	Value
Current	0x40000018
R0	0x00000000
R1	0x00000065
R2	0x00000000
R3	0x00000000
R4	0xFFFFFFFF
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000020
R15 (PC)	0x200000D3
CPSR	0
N	0
Z	1
C	0
V	1
I	1
F	0
T	0x13
M	0x00000000
SPSR	
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000020
Mode	Supervisor
States	74
Sec	0.00000617

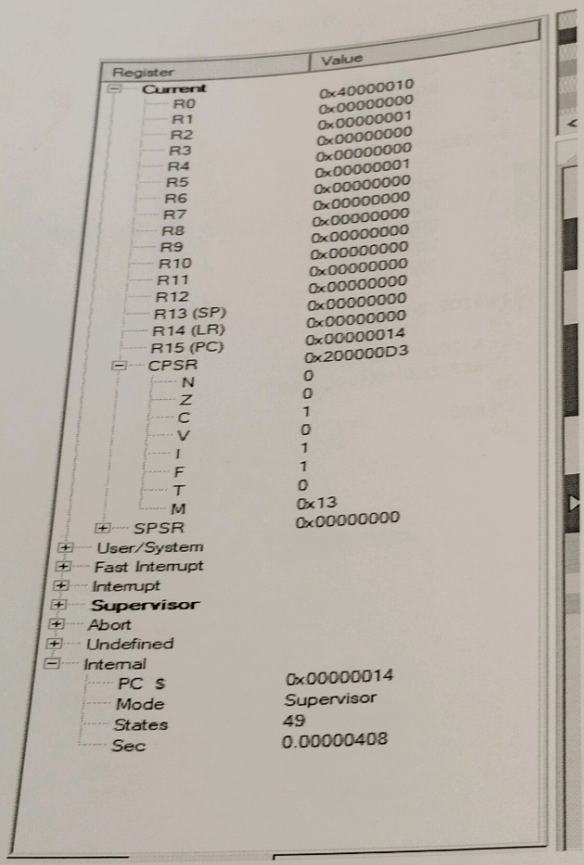
SMALLEST NUMBER:-

```
AREA SMALLEST, CODE, READONLY
START
        MOV R5,#4;
        LDR R0,A;
        LDR R2,[R0];
NEXT
        ADD R0,#4;
        LDR R3,[R0];
        CMP R2,R3;
        BLS SMALL
        MOV R2,R3;
SMALL
        SUB R5,#1;
        CMP R5,#0;
        BNE NEXT
        LDR R1,RES;
        STR R2,[R1];
STOP B STOP
A DCD 0X40000000;
RES DCD 0X40000020;
END
```

```
1          AREA SMALLEST, CODE, READONLY
2  START
3
4      MOV R5,#4;
5      LDR R0,A;
6      LDR R2,[R0];
7
8  NEXT
9
10     ADD R0,#4;
11     LDR R3,[R0];
12     CMP R2,R3;
13     BLS SMALL
14     MOV R2,R3;
15
16 SMALL
17
18     SUB R5,#1;
19     CMP R5,#0;
20     BNE NEXT
21     LDR R1,RES;
22     STR R2,[R1];
23
24 STOP B STOP
25
26 A DCD 0X40000000;
27 RES DCD 0X40000020;
28
29 END
30
```

OUTPUT:-


Memory 1
Address: 0x40000000
0x40000000: 22 00 00 00 01 00 00 00 45 00 00 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000044: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000066: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals
Memory 1


Register
Value
Current
0x40000010
R0
0x00000000
R1
0x00000001
R2
0x00000000
R3
0x00000000
R4
0x00000001
R5
0x00000000
R6
0x00000000
R7
0x00000000
R8
0x00000000
R9
0x00000000
R10
0x00000000
R11
0x00000000
R12
0x00000000
R13 (SP)
0x00000000
R14 (LR)
0x00000014
R15 (PC)
0x200000D3
CPSR
0
N
0
Z
1
C
0
V
1
I
1
F
0
T
0x13
M
0x00000000
SPSR
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
0x00000014
PC $
Supervisor
Mode
49
States
0.00000408
Sec

6. Write a program to count the number of ones and zeros in two consecutive memory locations.

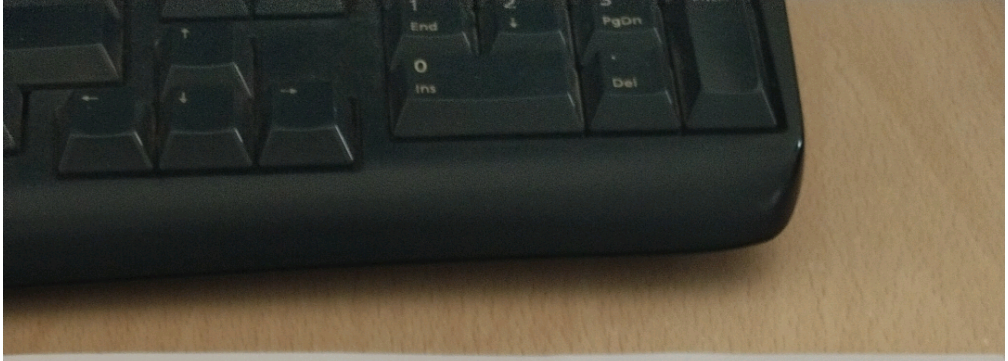
```
AREA ONESANDZERO, CODE, READONLY
ENTRY
        MOV R1, #0; TO COUNT NO OF 1'S
        MOV R2, #0; TO COUNT NO OF 0'S
        MOV R3, #2; NUMBER OF DATA
        LDR R4, =VALUE
LOOP2
        MOV R5, #32;
        LDR R6, [R4];
LOOP0
        MOVS R6, R6, ROR #1;
        BHI ONES
        ADD R2, R2, #1;
        B LOOP1
ONES
        ADD R1, R1,#1;
LOOP1
        SUBS R5, R5, #1;
        BNE LOOP0
        SUBS R3, R3, #1;
        LDR R4, =VALUE1
        BNE LOOP2
STOP B STOP
VALUE DCD 0X3
VALUE1 DCD 0X2
```

```
1          AREA ONESANDZERO,CODE,READONLY
2  ENTRY
3
4      MOV R1,#0; TO COUNT NO OF 1'S
5      MOV R2,#0; TO COUNT NO OF 0'S
6      MOV R3,#2; NUMBER OF DATA
7      LDR R4,=VALUE
8
9  LOOP2
10
11     MOV R5,#32;
12     LDR R6,[R4];
13
14 LOOP0
15
16     MOVS R6,R6,ROR #1;
17     BHI ONES
18     ADD R2,R2,#1;
19     B LOOP1
20
21 ONES
22
23     ADD R1,R1,#1;
24
25 LOOP1
26
27     SUBS R5,R5,#1;
28     BNE LOOP0
29     SUBS R3,R3,#1;
30     LDR R4,=VALUE1
31     BNE LOOP2
32
33 STOP B STOP
34
35 VALUE DCD 0X3
36 VALUE1 DCD 0X2
37
38 END
```

END

OUTPUT:-

Register	Value
Current	
R0	0x00000000
R1	0x00000003
R2	0x0000003D
R3	0x00000000
R4	0x00000048
R5	0x00000000
R6	0x00000002
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
CPSR	0x600000D3
N	0
Z	1
C	1
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000040
Mode	Supervisor
States	707
Sec	0.00005892

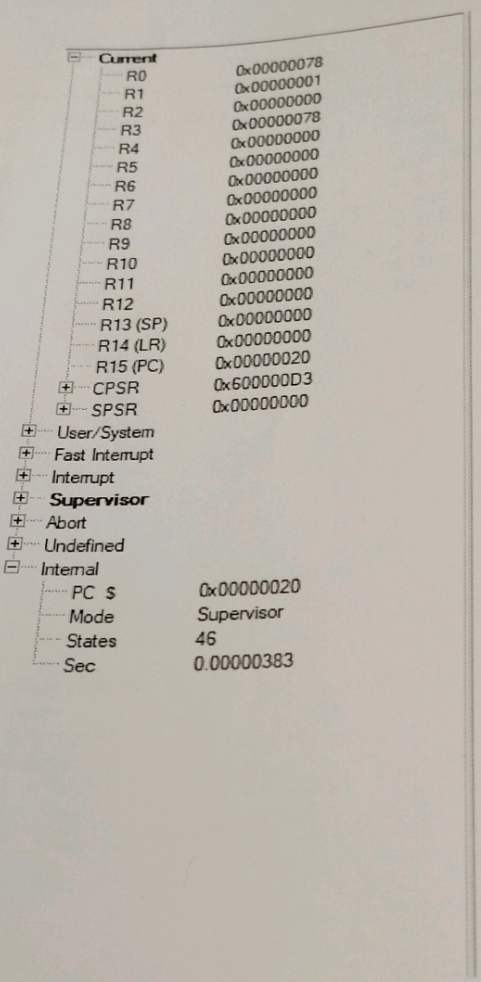


7. Write a program to find the factorial of a number.

```
    AREA ONESANDZERO, CODE, READONLY
ENTRY
    MOV R0, #5;
    MOV R1, R0;
FACT SUBS R1, R1, #1;
    CMP R1, #1;
    BEQ STOP
    MUL R3, R0, R1;
    MOV R0, R3;
    BNE FACT
STOP B STOP
END
```

```
1
2       AREA ONESANDZERO, CODE, READONLY
3  ENTRY
4       MOV R0, #5;
5       MOV R1, R0;
6  FACT SUBS R1, R1, #1;
7       CMP R1, #1;
8       BEQ STOP
9       MUL R3, R0, R1;
10      MOV R0, R3;
11      BNE FACT
12 STOP B STOP
13 END
14
```

OUTPUT:-


Current
R0 0x00000078
R1 0x00000001
R2 0x00000000
R3 0x00000078
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000020
CPSR 0x600000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000020
Mode Supervisor
States 46
Sec 0.00000383

8. Write an ALP to arrange series of 32bit numbers in ascending or descending order

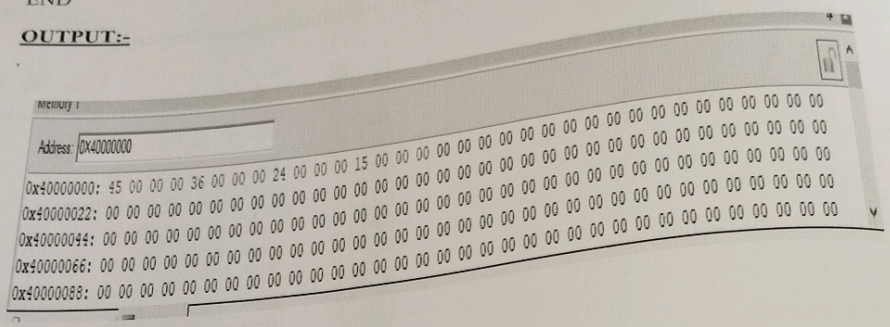
DESCENDING ORDER

```
AREA DECENDING, CODE, READONLY
ENTRY
    MOV R0, #0X04;
NXT
    MOV R1,#03;
    MOV R2,#0X40000000;
COMPARE
    LDR R3,[R2]
    ADD R2,R2,#04;
    LDR R4,[R2]
    CMP R3,R4;
    BGT NOEXCHANGE
    STR R3,[R2];
    SUB R2,R2,#04;
    STR R4,[R2]
    ADD R2,R2,#04
NOEXCHANGE
    SUB R1,R1,#01;
    CMP R1,#00;
    BNE COMPARE
    SUB R0,R0,#01;
    CMP R0,#00
    BNE NXT
STOP B STOP
```

```
1
2     AREA DECENDING,CODE,READONLY
3 ENTRY
4     MOV R0,#0X03;
5 NXT
6     MOV R1,#03;
7     MOV R2,#0X40000000;
8 COMPARE
9     LDR R3,[R2]
10    ADD R2,R2,#04;
11    LDR R4,[R2]
12    CMP R3,R4;
13    BGT NOEXCHANGE
14    STR R3,[R2];
15    SUB R2,R2,#04;
16    STR R4,[R2]
17    ADD R2,R2,#04
18 NOEXCHANGE
19    SUB R1,R1,#01;
20    CMP R1,#00;
21    BNE COMPARE
22    SUB R0,R0,#01;
23    CMP R0,#00
24    BNE NXT
25 STOP B STOP
26 END
27
```

END

OUTPUT:-

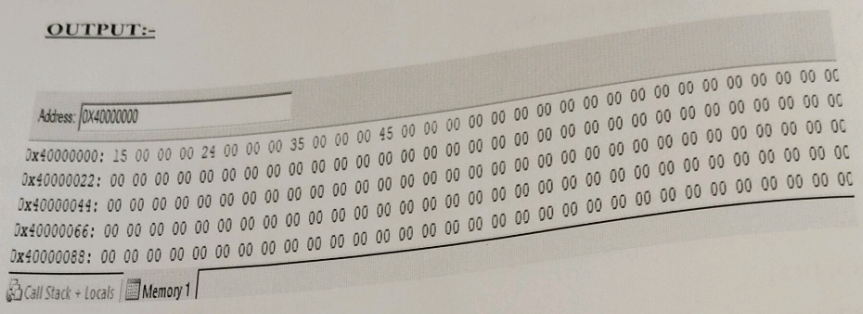

Memory 1
Address: 0X40000000
0x40000000: 45 00 00 00 36 00 00 00 24 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000044: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000066: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ASCENDING ORDER:-

```
AREA DECENDING, CODE, READONLY
ENTRY
      MOV R0, #0X03;
NXT
      MOV R1, #03;
      MOV R2, #0X40000000;
COMPARE
      LDR R3, [R2]
      ADD R2, R2, #04;
      LDR R4, [R2]
      CMP R3, R4;
      BLT NOEXCHANGE
      STR R3, [R2];
      SUB R2, R2, #04;
      STR R4, [R2]
      ADD R2, R2, #04
NOEXCHANGE
      SUB R1, R1,#01;
      CMP R1,#00;
      BNE COMPARE
      SUB R0, R0, #01;
      CMP R0, #00
      BNE NXT
STOP B STOP
END
```

```
1
2       AREA ASCENDING,CODE,READONLY
3  ENTRY
4       MOV R0,#0X03;
5  NXT
6       MOV R1,#03;
7       MOV R2,#0X40000000;
8  COMPARE
9       LDR R3,[R2]
10      ADD R2,R2,#04;
11      LDR R4,[R2]
12      CMP R3,R4;
13      BLT NOEXCHANGE
14      STR R3,[R2];
15      SUB R2,R2,#04;
16      STR R4,[R2]
17      ADD R2,R2,#04
18 NOEXCHANGE
19      SUB R1,R1,#01;
20      CMP R1,#00;
21      BNE COMPARE
22      SUB R0,R0,#01;
23      CMP R0,#00
24      BNE NXT
25 STOP B STOP
26 END
27
```

OUTPUT:-


Address: 0x40000000
0x40000000: 15 00 00 00 24 00 00 00 35 00 00 00 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000044: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000066: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals
Memory 1