EMBEDDED PROGRAMMING LAB

LAB-5 DATE:15-10-2024

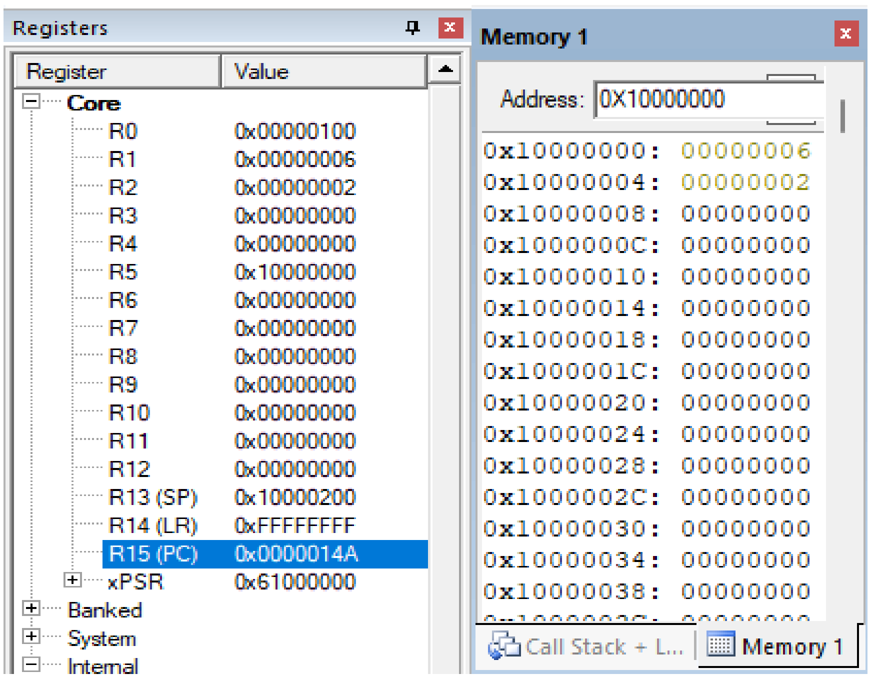
PREETHISH K R

1. Write a program to count number of zeroes and ones given by variable “NUM” And store number of count of ones and zeroes in memory location 0x10000000 onwards.

Program:

```
            AREA BASIC,CODE,READONLY
            ENTRY
            EXPORT __main
NUM DCD 0XED
ONES RN 1
ZEROES RN 2
LOOP RN 4

__main
            MOV R5,#0X10000000
            LDR R0,=NUM
            LDRB R3,[R0]
            MOV ONES, #0
            MOV ZEROES,#0
            MOV LOOP,#8
AGAIN LSRS R3,#1
            ADDCS ONES,#1
            ADDCC ZEROES,#1
            SUB LOOP,#1
            CMP LOOP,#00
            BNE AGAIN
            STRB ONES,[R5]
            STRB ZEROES,[R5,#4]
            NOP
            END
```

Output:



```
Registers
Register        Value
Core
  R0            0x00000100
  R1            0x00000006
  R2            0x00000002
  R3            0x00000000
  R4            0x00000000
  R5            0x10000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x10000200
  R14 (LR)      0xFFFFFFFF
  R15 (PC)      0x0000014A
  xPSR          0x61000000
Banked
System
Internal
```

```
Memory 1
Address: 0X10000000
0x10000000: 00000006
0x10000004: 00000002
0x10000008: 00000000
0x1000000C: 00000000
0x10000010: 00000000
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
```

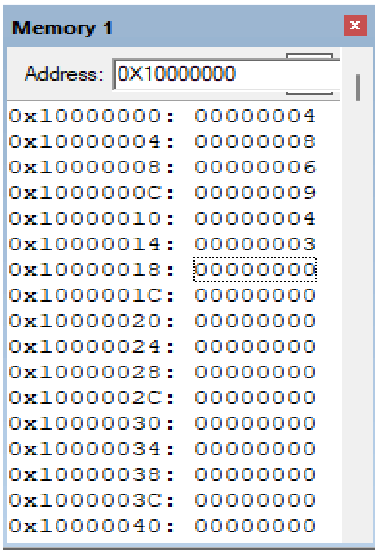
Fig 1.1- Result obtained

2. Write a program find the largest number from given array starting from location 0x10000000 assume that size of array defined as N

Program:

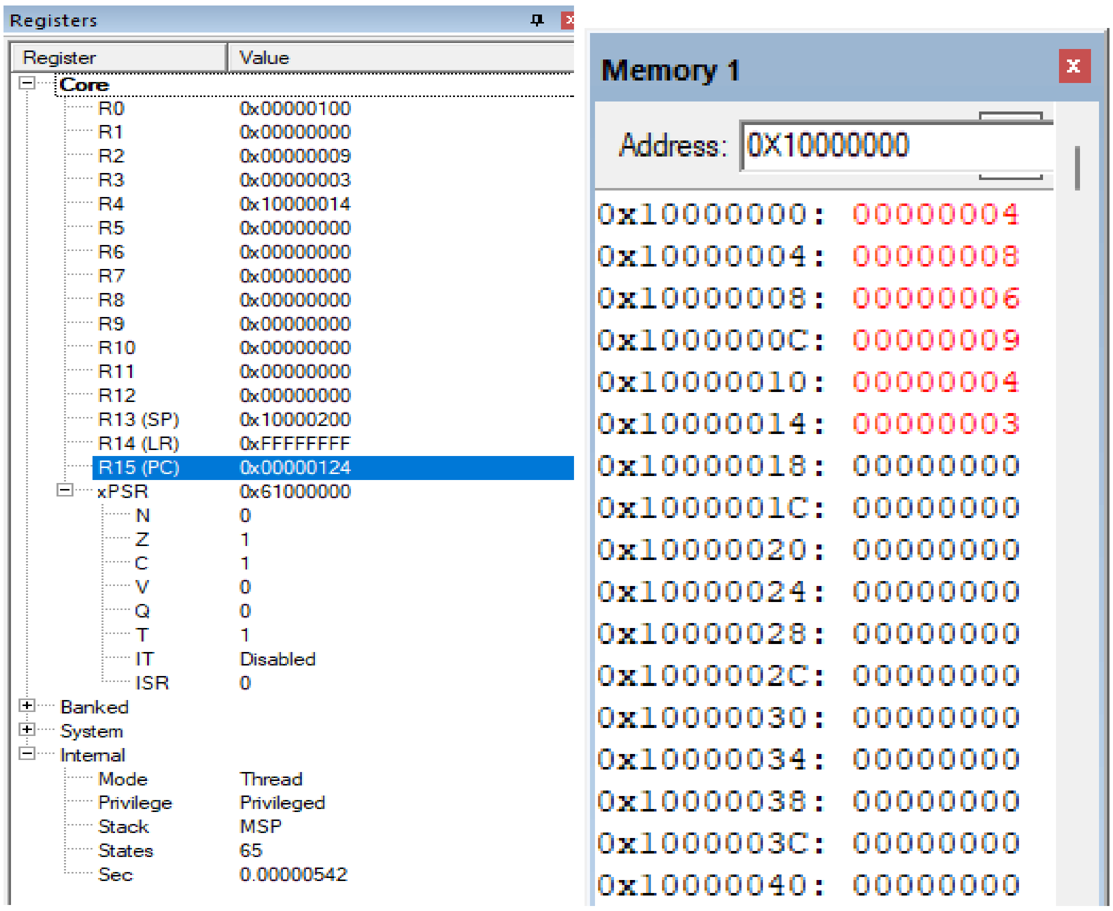
```
    AREA BASIC,CODE,READONLY
    ENTRY
    EXPORT __main
N DCD 5
__main

    LDR R0,=N
    LDRB R1,[R0]

    MOV R4,#0X10000000
    LDRB R2,[R4]
NEXT LDR R3,[R4,#4]!
    CMP R2,R3
    BCS SKIP
    MOV R2,R3
SKIP SUB R1,#1
    CMP R1,#0
    BNE NEXT
    NOP
    END
```

Output:



```
Memory 1
Address: 0X10000000
0x10000000: 00000004
0x10000004: 00000008
0x10000008: 00000006
0x1000000C: 00000009
0x10000010: 00000004
0x10000014: 00000003
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
0x1000003C: 00000000
0x10000040: 00000000
```

Fig 2.1- Data values entered



```
Registers
Register        Value
Core
   R0           0x00000100
   R1           0x00000000
   R2           0x00000009
   R3           0x00000003
   R4           0x10000014
   R5           0x00000000
   R6           0x00000000
   R7           0x00000000
   R8           0x00000000
   R9           0x00000000
   R10          0x00000000
   R11          0x00000000
   R12          0x00000000
   R13 (SP)     0x10000200
   R14 (LR)     0xFFFFFFFF
   R15 (PC)     0x00000124
   xPSR         0x61000000
      N         0
      Z         1
      C         1
      V         0
      Q         0
      T         1
      IT        Disabled
      ISR       0
Banked
System
Internal
   Mode         Thread
   Privilege    Privileged
   Stack        MSP
   States       65
   Sec          0.00000542
```

```
Memory 1
Address: 0X10000000
0x10000000: 00000004
0x10000004: 00000008
0x10000008: 00000006
0x1000000C: 00000009
0x10000010: 00000004
0x10000014: 00000003
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
0x1000003C: 00000000
0x10000040: 00000000
```

Fig 2.2- Result obtained at register R2

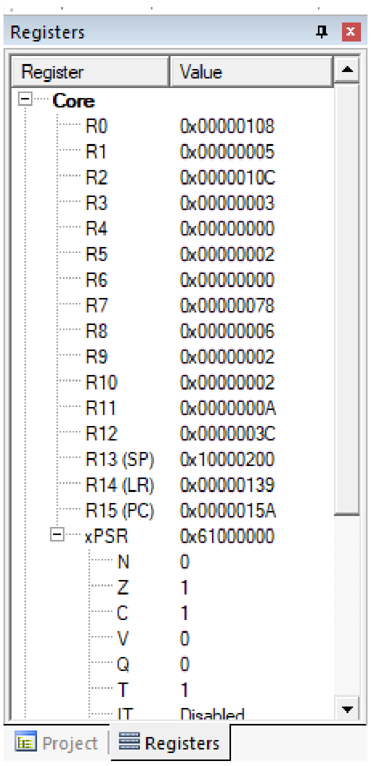
3. Write a program to read variable n and r also to compute NCR and NPR and store the results to variable NCR and NPR use subroutine to find factorial

Program:

```
   AREA BASIC, CODE, READONLY
   ENTRY
   EXPORT __main

N      DCD 5      ;INPUT
R      DCD 3       ;INPUT
NCR    DCD 0
NPR    DCD 0

__main
   LDR R0,=N
   LDRB R1, [R0]
   LDR R2, =R
   LDRB R3, [R2]

   MOV R6, R1
   BL FACT
   MOV R7, R5

   MOV R6, R3
   BL FACT
   MOV R8, R5

   SUB R9, R1, R3
   CMP R9, #0
   BEQ SKIP

   MOV R6, R9
   BL FACT
   MOV R10, R5

   MUL R0, R10, R8
   UDIV R11, R7, R0
   UDIV R12, R7, R10
   B STORE
SKIP
   UDIV R11, R7, R8
   MOV R12, R7

STORE LDR R0,=NCR
        STR R11,[R0]
        LDR R2,=NPR
        STR R12,[R2]

LOOP B LOOP
```

```
FACT
   MOV R5, #1
AGAIN
   MUL R5, R5, R6
   SUB R6, #1
   CMP R6, #0
   BNE AGAIN
   BX LR

   NOP
   END
```

Output:



```
Registers
Register        Value
Core
  R0            0x00000108
  R1            0x00000005
  R2            0x0000010C
  R3            0x00000003
  R4            0x00000000
  R5            0x00000002
  R6            0x00000000
  R7            0x00000078
  R8            0x00000006
  R9            0x00000002
  R10           0x00000002
  R11           0x0000000A
  R12           0x0000003C
  R13 (SP)      0x10000200
  R14 (LR)      0x00000139
  R15 (PC)      0x0000015A
  xPSR          0x61000000
    N           0
    Z           1
    C           1
    V           0
    Q           0
    T           1
    IT          Disabled
Project   Registers
```

Fig 3.1- Result obtained at register R11 (NCR) and R12 (NPR)

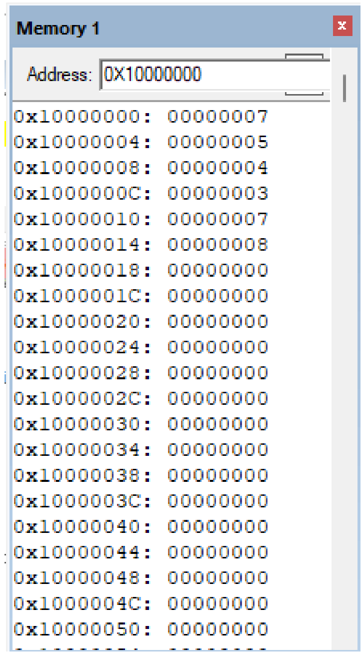
4. Write a program to sort given an array starting from 0x10000000 in ascending order

Program:

```
        AREA BASIC,CODE,READONLY
        ENTRY
        EXPORT __main
N DCD 5
__main

        LDR R0,=N
        LDRB R1,[R0]
        MOV R6,R1
NEXT MOV R7,R6
```

```
        SUB R2,R7,#1
        MOV R3,#0X10000000
REPEAT LDRB R4,[R3]
        LDRB R5,[R3,#4]
        CMP R4,R5
        BCC SKIP
        STR R5,[R3]
        STR R4,[R3,#4]!
SKIP SUBS R2,#1
        BNE REPEAT
        SUBS R1,#1
        BNE NEXT
        NOP
        END
```

Output:

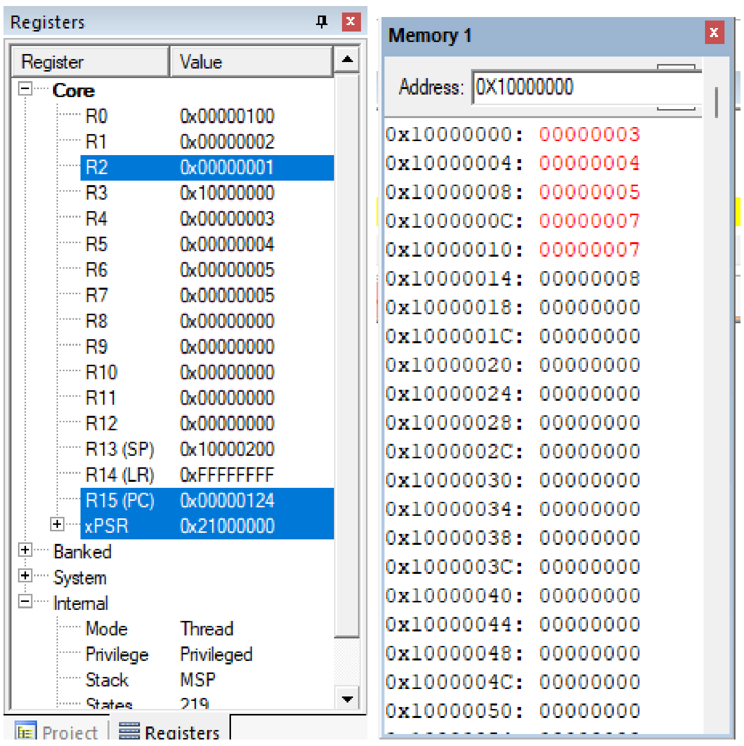


Memory 1

Address: 0X10000000

```
0x10000000: 00000007
0x10000004: 00000005
0x10000008: 00000004
0x1000000C: 00000003
0x10000010: 00000007
0x10000014: 00000008
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
0x1000003C: 00000000
0x10000040: 00000000
0x10000044: 00000000
0x10000048: 00000000
0x1000004C: 00000000
0x10000050: 00000000
```

Fig 4.1-Data values entered



Registers

Register	Value
Core	
R0	0x00000100
R1	0x00000002
R2	0x00000001
R3	0x10000000
R4	0x00000003
R5	0x00000004
R6	0x00000005
R7	0x00000005
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000124
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	219

Memory 1

Address: 0X10000000

```
0x10000000: 00000003
0x10000004: 00000004
0x10000008: 00000005
0x1000000C: 00000007
0x10000010: 00000007
0x10000014: 00000008
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
0x1000003C: 00000000
0x10000040: 00000000
0x10000044: 00000000
0x10000048: 00000000
0x1000004C: 00000000
0x10000050: 00000000
```

Fig 4.2- Result obtained