EMBEDDED PROGRAMMING LAB

LAB-3 DATE:25-09-2024

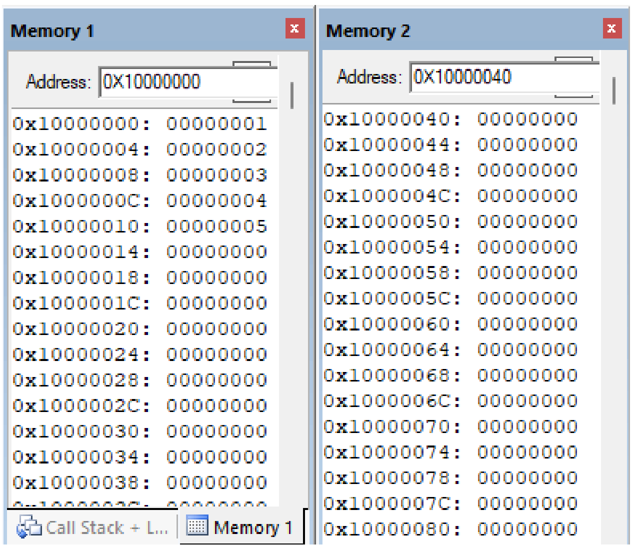
PREETHISH K R

1. Write a program to transfer a data from source location to destination location

Program:

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

    LDR R0,=0X10000000
    LDR R1,=0X10000040
    MOV R2,#5
NEXT LDR R3,[R0],#4
    STR R3,[R1],#4
    SUB R2,#1
    CMP R2,#00
    BNE NEXT
    NOP
    END
```

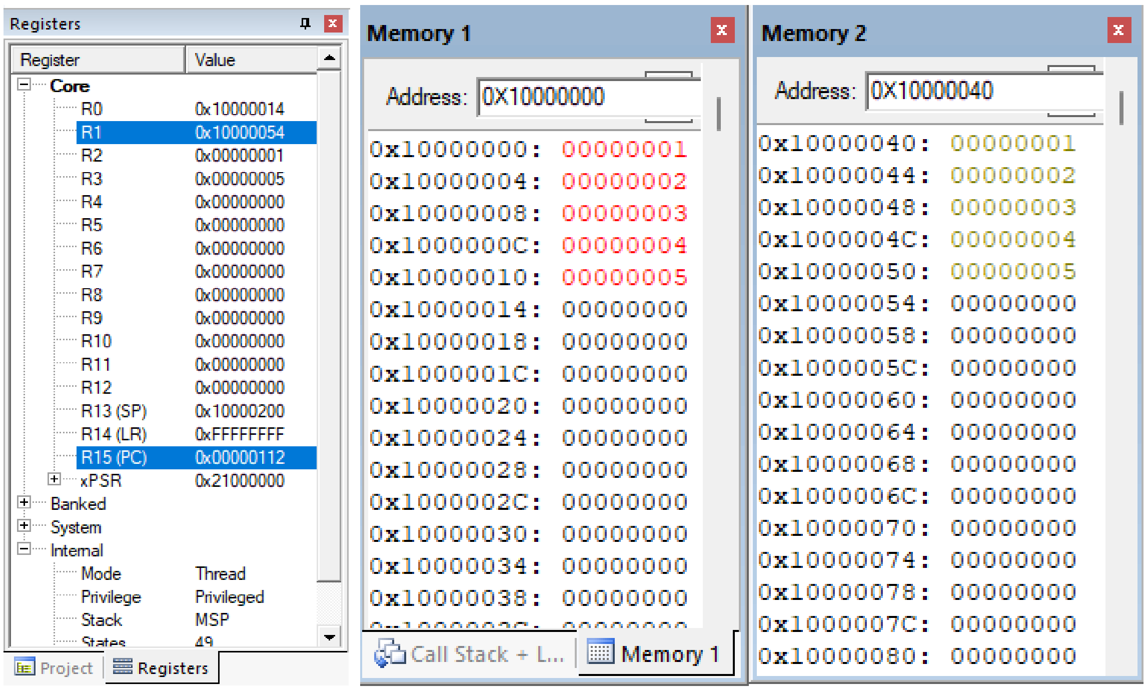
Output:



```
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
```

```
0x10000040: 00000000
0x10000044: 00000000
0x10000048: 00000000
0x1000004C: 00000000
0x10000050: 00000000
0x10000054: 00000000
0x10000058: 00000000
0x1000005C: 00000000
0x10000060: 00000000
0x10000064: 00000000
0x10000068: 00000000
0x1000006C: 00000000
0x10000070: 00000000
0x10000074: 00000000
0x10000078: 00000000
0x1000007C: 00000000
0x10000080: 00000000
```

Fig1.1-Data values entered



Registers

Register	Value
Core	
R0	0x10000014
R1	0x10000054
R2	0x00000001
R3	0x00000005
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000112
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	49

Memory 1

```
Address: 0X10000000
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
```

Memory 2

```
Address: 0X10000040
0x10000040: 00000001
0x10000044: 00000002
0x10000048: 00000003
0x1000004C: 00000004
0x10000050: 00000005
0x10000054: 00000000
0x10000058: 00000000
0x1000005C: 00000000
0x10000060: 00000000
0x10000064: 00000000
0x10000068: 00000000
0x1000006C: 00000000
0x10000070: 00000000
0x10000074: 00000000
0x10000078: 00000000
0x1000007C: 00000000
0x10000080: 00000000
```

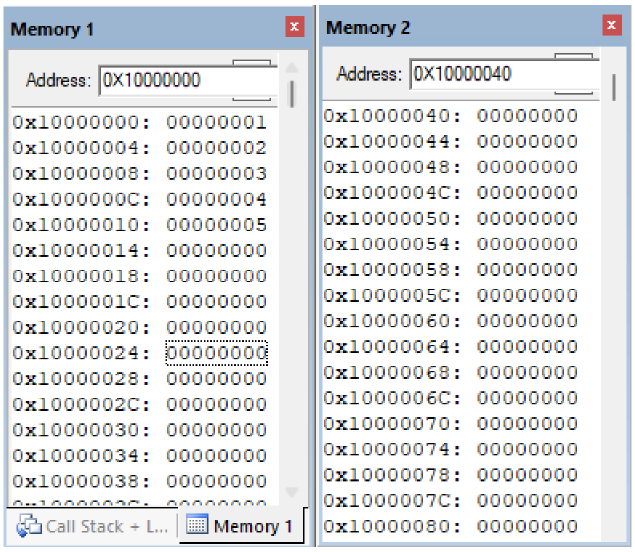
Fig1.2-Result obtained

2. Write a program to exchange the content of memory location storing from 0x10000000 to 0x10000040

Program:

```
        AREA BASIC,CODE,READONLY
        ENTRY
        EXPORT __main
__main
        LDR R0,=0X10000000
        LDR R1,=0X10000040
        MOV R2,#5
AGAIN LDR R3,[R0]
        LDR R4,[R1]
        STR R3,[R1],#4
        STR R4,[R0],#4
        SUB R2,#1
        CMP R2,#00
        BNE AGAIN
        NOP
        END
```

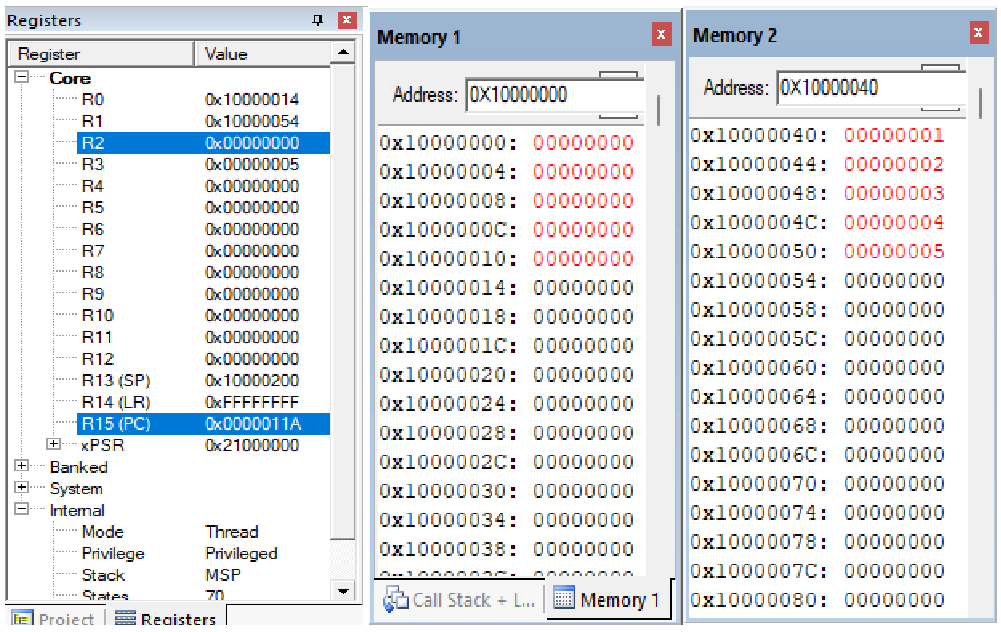
Output:



```
Memory 1
Address: 0X10000000
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
0x10000038: 00000000
```

```
Memory 2
Address: 0X10000040
0x10000040: 00000000
0x10000044: 00000000
0x10000048: 00000000
0x1000004C: 00000000
0x10000050: 00000000
0x10000054: 00000000
0x10000058: 00000000
0x1000005C: 00000000
0x10000060: 00000000
0x10000064: 00000000
0x10000068: 00000000
0x1000006C: 00000000
0x10000070: 00000000
0x10000074: 00000000
0x10000078: 00000000
0x1000007C: 00000000
0x10000080: 00000000
```

Fig2.1-Data values entered



Register	Value
Core	
R0	0x10000014
R1	0x10000054
R2	0x00000000
R3	0x00000005
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0000011A
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	70

```
Memory 1
Address: 0X10000000
0x10000000: 00000000
0x10000004: 00000000
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
```

```
Memory 2
Address: 0X10000040
0x10000040: 00000001
0x10000044: 00000002
0x10000048: 00000003
0x1000004C: 00000004
0x10000050: 00000005
0x10000054: 00000000
0x10000058: 00000000
0x1000005C: 00000000
0x10000060: 00000000
0x10000064: 00000000
0x10000068: 00000000
0x1000006C: 00000000
0x10000070: 00000000
0x10000074: 00000000
0x10000078: 00000000
0x1000007C: 00000000
0x10000080: 00000000
```

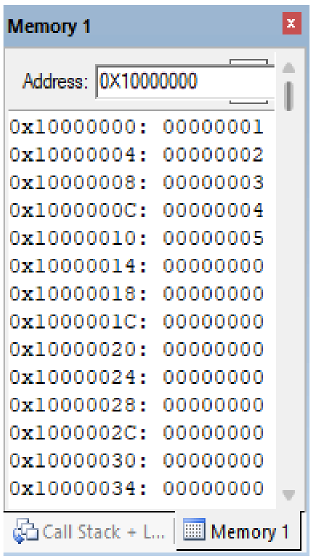
Fig2.2-Result obtained

3. Write a program to find sum of elements in an array starting from 0x10000000

Program:

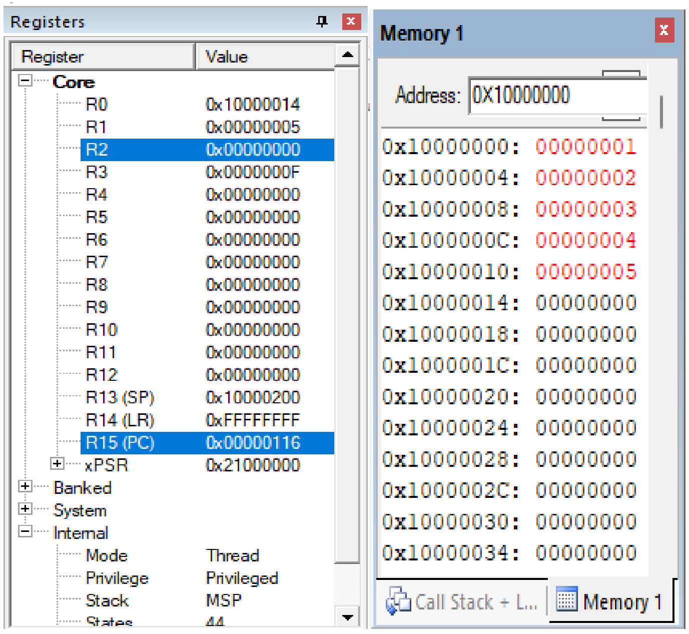
```
        AREA BASIC,CODE,READONLY
        ENTRY
        EXPORT __main
__main
        LDR R0,=0X10000000
        MOV R2,#5
        MOV R3,#00
AGAIN LDR R1,[R0],#4
        ADD R3,R1
        SUB R2,#1
        CMP R2,#00
        BNE  AGAIN
        NOP
        END
```

Output:



```
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
```

Fig3.1-Data values entered



Register	Value
Core	
R0	0x10000014
R1	0x00000005
R2	0x00000000
R3	0x0000000F
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000116
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	44

```
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000000
0x10000018: 00000000
0x1000001C: 00000000
0x10000020: 00000000
0x10000024: 00000000
0x10000028: 00000000
0x1000002C: 00000000
0x10000030: 00000000
0x10000034: 00000000
```

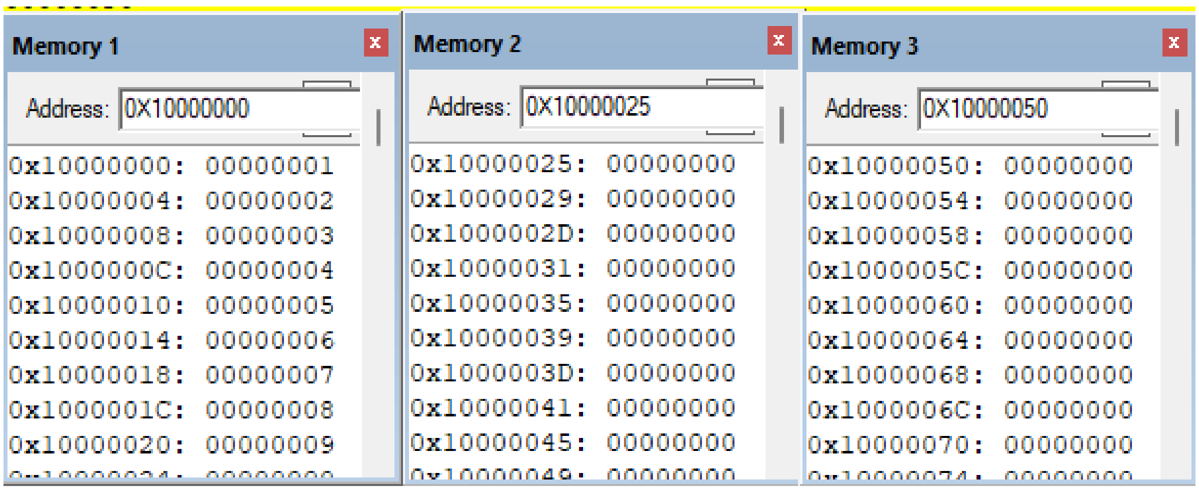
Fig3.2-Result obtained

4. Write a program to separate even and odd numbers from an given array and make separate array for even and odd numbers

Program:

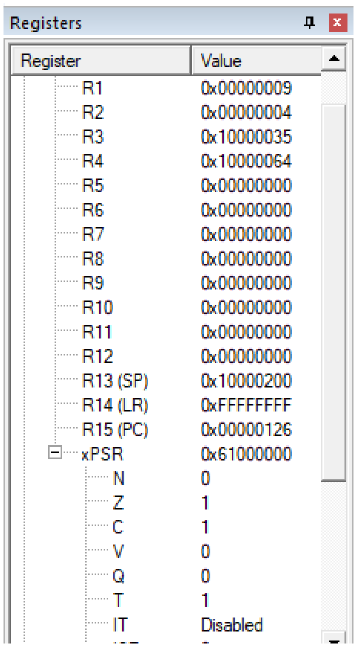
```
    AREA BASIC,CODE,READONLY
    ENTRY
    EXPORT __main
__main

    LDR R3,=0X10000025
    LDR R4,=0X10000050
    MOV R6,#9
    LDR R0,=0X10000000
NEXT LDR R1,[R0],#4
    LSRS R2,R1,#1
    BCS ODDITIS
    STR R1,[R3],#4
    B NEXT1
ODDITIS  STR R1,[R4],#4
NEXT1 SUB R6,#1
    CMP R6,#00
    BNE NEXT
    NOP
    END
```

Output:

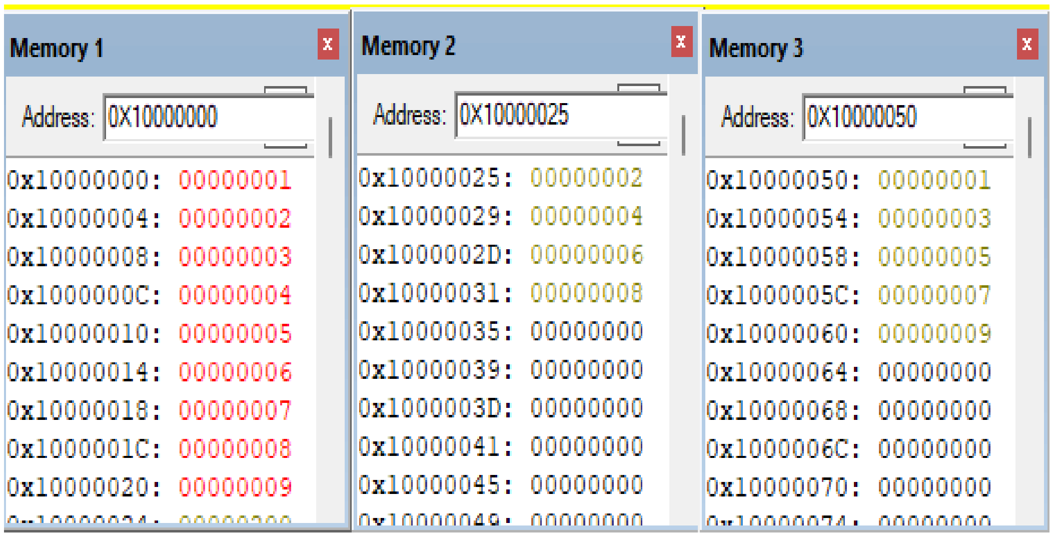


```
Memory 1
Address: 0X10000000
0x10000000: 00000001
0x10000004: 00000002
0x10000008: 00000003
0x1000000C: 00000004
0x10000010: 00000005
0x10000014: 00000006
0x10000018: 00000007
0x1000001C: 00000008
0x10000020: 00000009

Memory 2
Address: 0X10000025
0x10000025: 00000000
0x10000029: 00000000
0x1000002D: 00000000
0x10000031: 00000000
0x10000035: 00000000
0x10000039: 00000000
0x1000003D: 00000000
0x10000041: 00000000
0x10000045: 00000000

Memory 3
Address: 0X10000050
0x10000050: 00000000
0x10000054: 00000000
0x10000058: 00000000
0x1000005C: 00000000
0x10000060: 00000000
0x10000064: 00000000
0x10000068: 00000000
0x1000006C: 00000000
0x10000070: 00000000
```

Fig3.4-Data values entered



```
Registers
Register        Value
  R1            0x00000009
  R2            0x00000004
  R3            0x10000035
  R4            0x10000064
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x10000200
  R14 (LR)      0xFFFFFFFF
  R15 (PC)      0x00000126
  xPSR          0x61000000
    N           0
    Z           1
    C           1
    V           0
    Q           0
    T           1
    IT          Disabled
```



```
Memory 1                     Memory 2                     Memory 3
Address: 0X10000000          Address: 0X10000025          Address: 0X10000050
0x10000000: 00000001         0x10000025: 00000002         0x10000050: 00000001
0x10000004: 00000002         0x10000029: 00000004         0x10000054: 00000003
0x10000008: 00000003         0x1000002D: 00000006         0x10000058: 00000005
0x1000000C: 00000004         0x10000031: 00000008         0x1000005C: 00000007
0x10000010: 00000005         0x10000035: 00000000         0x10000060: 00000009
0x10000014: 00000006         0x10000039: 00000000         0x10000064: 00000000
0x10000018: 00000007         0x1000003D: 00000000         0x10000068: 00000000
0x1000001C: 00000008         0x10000041: 00000000         0x1000006C: 00000000
0x10000020: 00000009         0x10000045: 00000000         0x10000070: 00000000
```

Fig3.4-Result obtained