UEC513 – EMBEDDED SYSTEMS

LAB ASSIGNMENT SOLUTIONS

NAME: Digaamber Dhamija **ROLL NO:** 101915008 **BATCH:** 2NC1

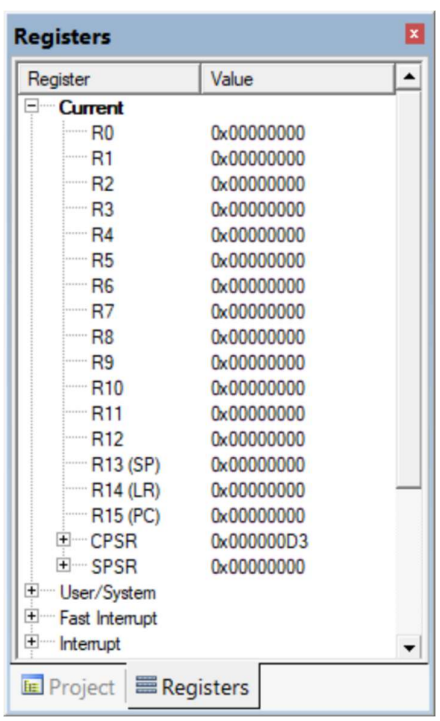
QUES 1 (a): Write a program in ARM assembly language to store the data into one register and then copy that content to all registers.

CODE:

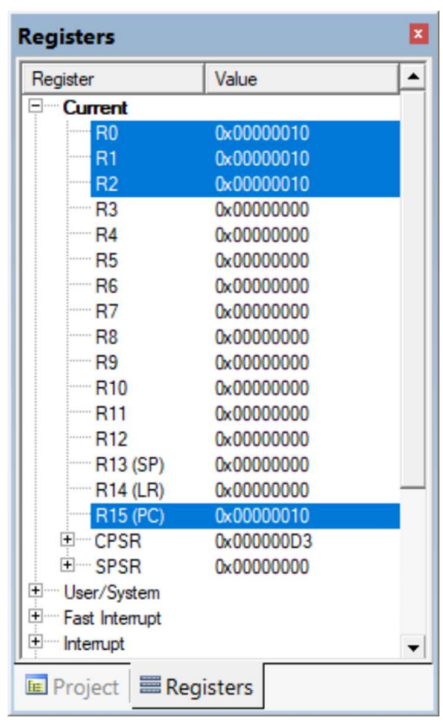
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE
 MOV R1, R0
 MOV R2, R0

 AREA PROGRAM, DATA, READONLY
VALUE DCD &10
 END
```

OUTPUT:

INITIAL



FINAL



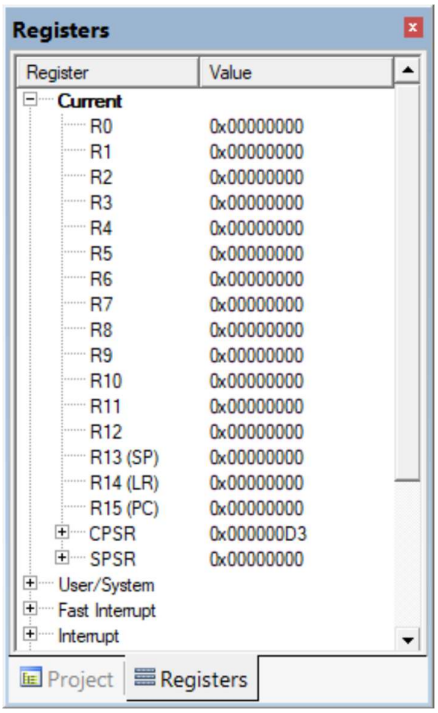
QUES 1 (b): Write a program in ARM assembly language to add two 32 bit numbers using Immediate Addressing Mode.

CODE:

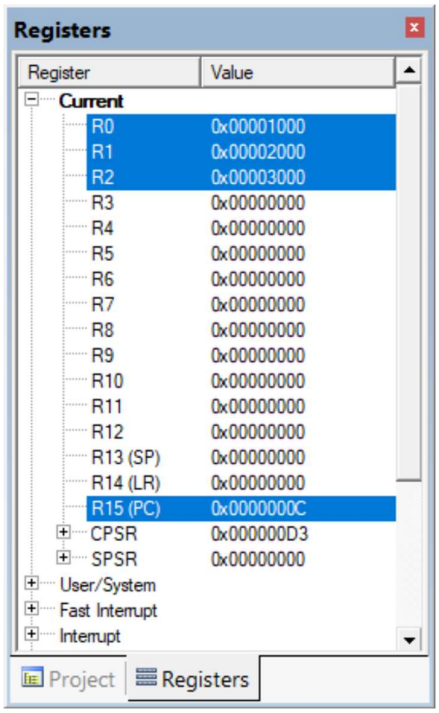
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 MOV R0, #0x00001000
 MOV R1, #0x00002000

 ADD R2, R1, R0
 END
```

OUTPUT:

INITIAL



FINAL



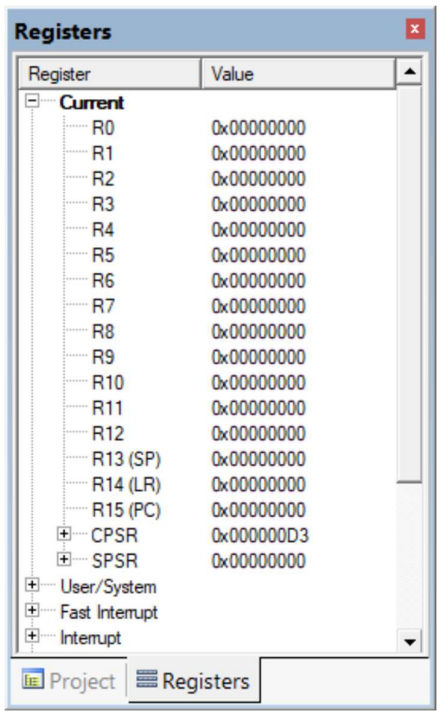
QUES 1 (c): Write a program in ARM assembly language to add two 32 bit numbers using Direct Addressing Mode.

CODE:

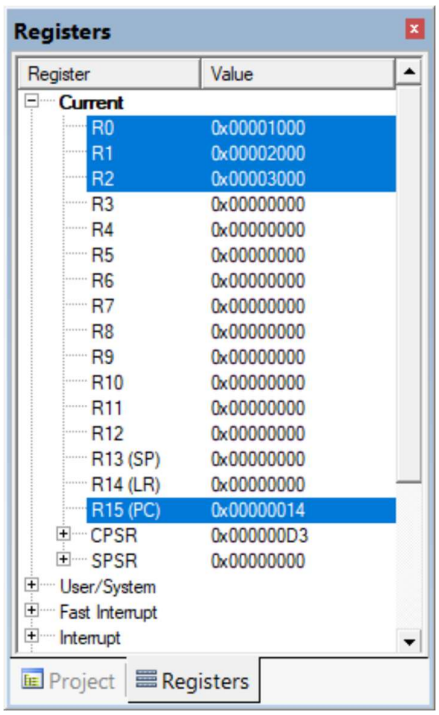
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 LDR R1, VALUE2

 ADD R2, R1, R0

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00002000
 END
```

OUTPUT:

INITIAL



FINAL



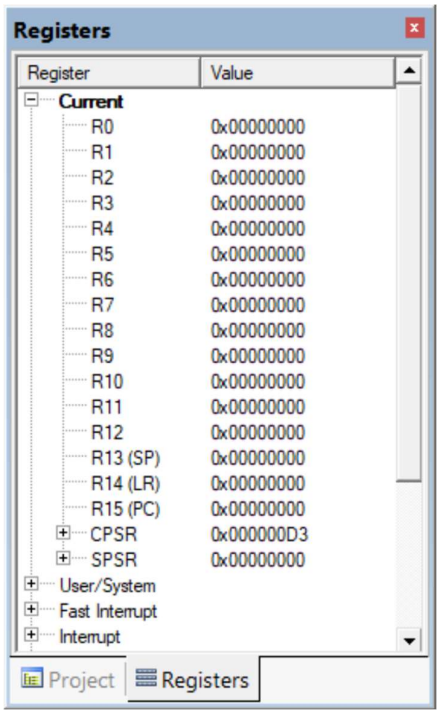
QUES 1 (d): Write a program in ARM assembly language to add two 32 bit numbers using Indirect Addressing Mode.

CODE:

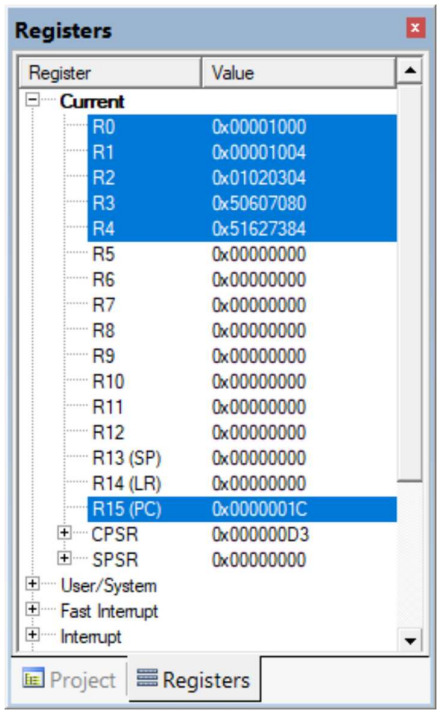
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 LDR R1, VALUE2

 LDR R2, [R0]
 LDR R3, [R1]

 ADD R4, R3, R2

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00001004
 END
```

OUTPUT:

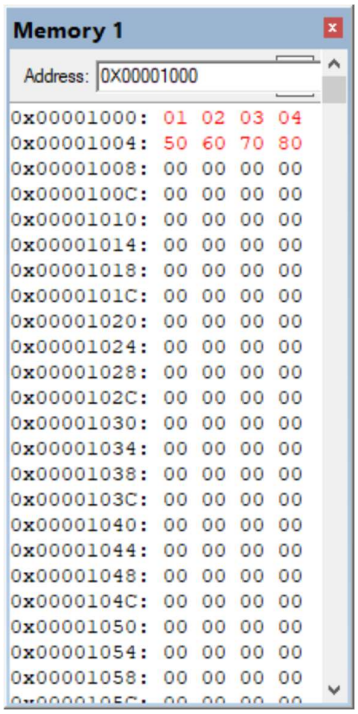
INITIAL



FINAL



MEMORY



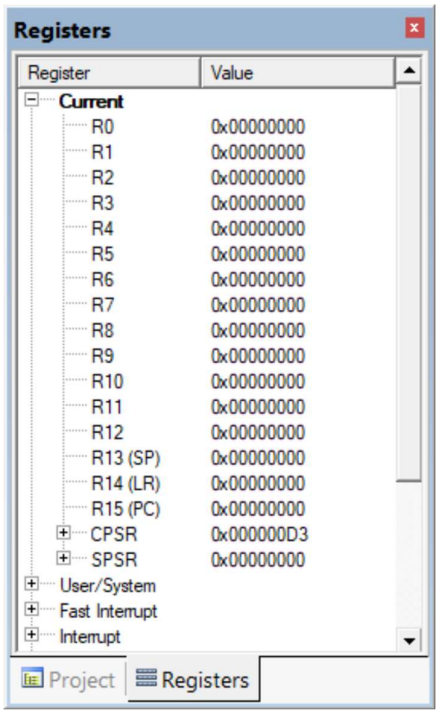
QUES 1 (e): Write a program in ARM assembly language to find the one's complement of a number.

CODE:

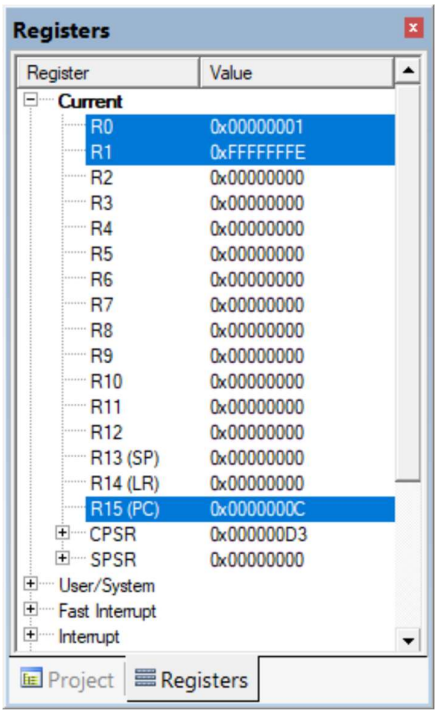
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE
 MVN R1, R0

 AREA PROGRAM, DATA, READONLY
VALUE DCD &00000001
 END
```

OUTPUT:

INITIAL



FINAL



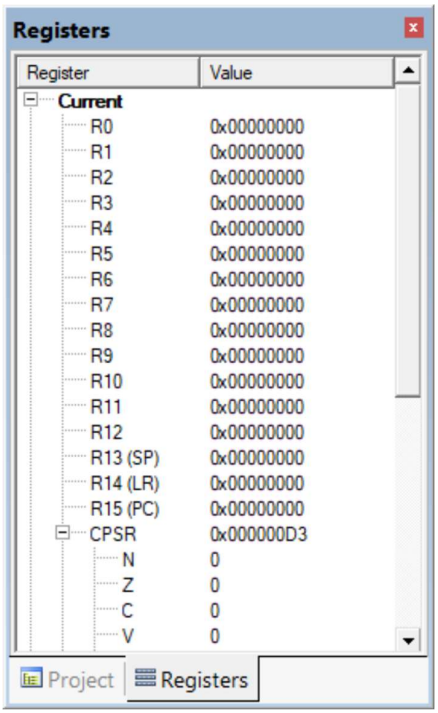
QUES 2: Write a program in ARM assembly language to add 64 bit addition using indirect addressing mode and store the result in memory location.(check the flag status also in CPSR register).

CODE:

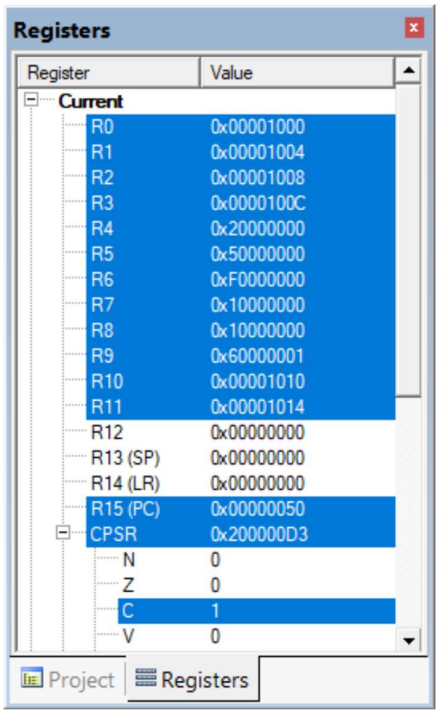
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 LDR R1, VALUE2
 LDR R2, VALUE3
 LDR R3, VALUE4

 LDR R4, [R0]
 LDR R5, [R1]
 LDR R6, [R2]
 LDR R7, [R3]

 ADDS R8, R4, R6
 ADC R9, R5, R7

 LDR R10, VALUE5
 LDR R11, VALUE6

 STR R8, [R10]
 STR R9, [R11]

 AREA PROGRAM, DATA, READONLY

VALUE1 DCD &00001000
VALUE2 DCD &00001004
VALUE3 DCD &00001008
VALUE4 DCD &0000100C
VALUE5 DCD &00001010
VALUE6 DCD &00001014

 END
```

OUTPUT:

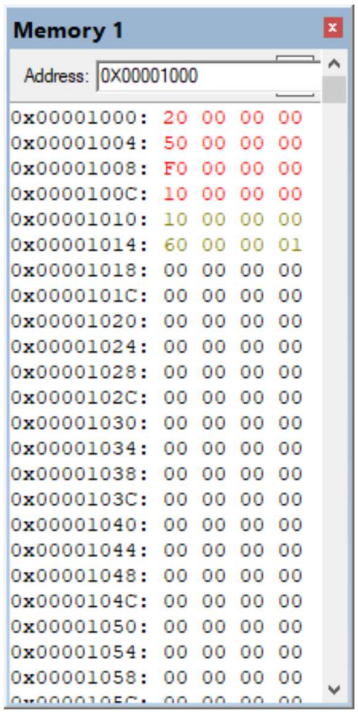
INITIAL



FINAL



MEMORY



QUES 3: Write a program in ARM assembly language to load any register with 32 bit data and perform the following

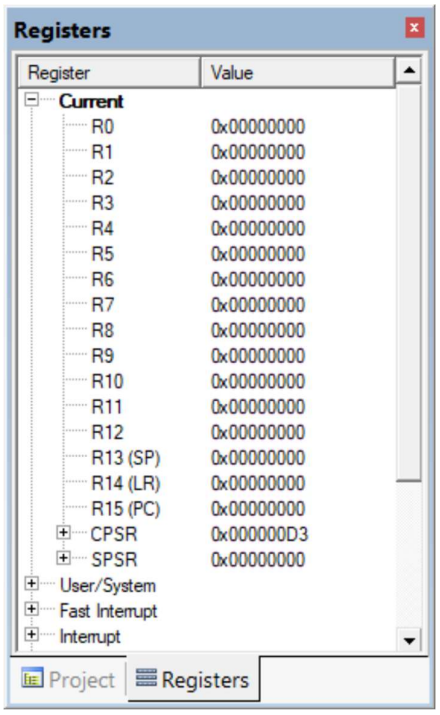
(a): Shift left by 2 bits

CODE:

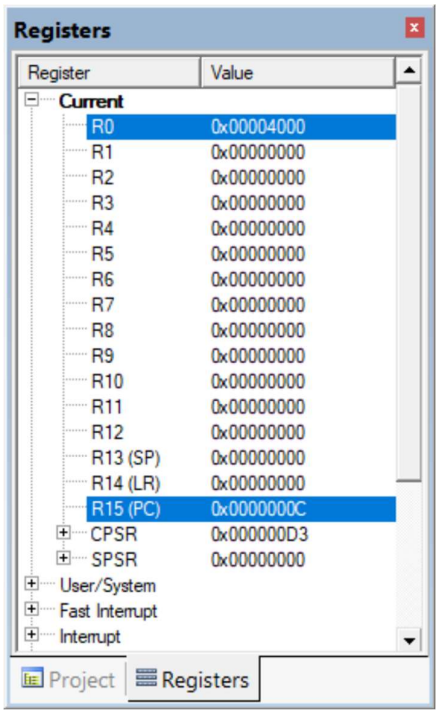
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE
 MOV R0, R0, LSL #2

 AREA PROGRAM, DATA, READONLY
VALUE DCD &00001000
 END
```

OUTPUT:

INITIAL



FINAL



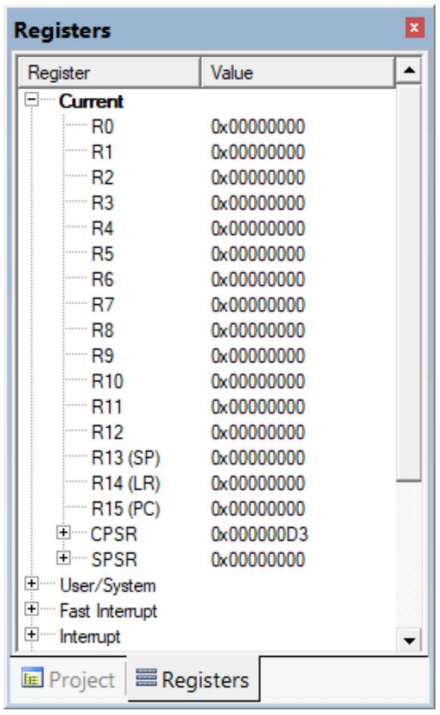
(b): Shift right by the number of bits stored in register R2

CODE:

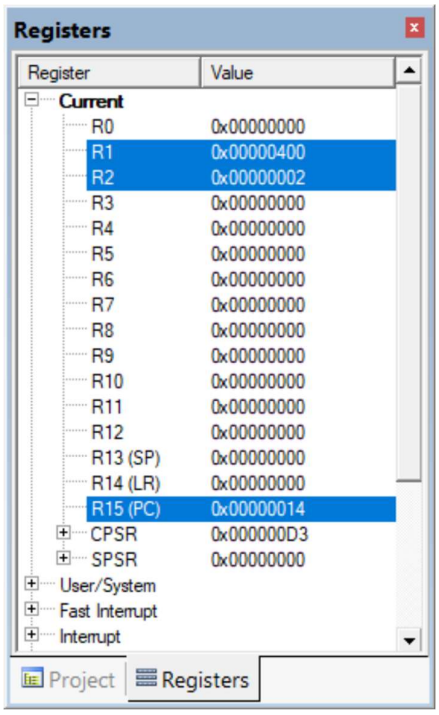
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R1, VALUE1
 LDR R2, VALUE2

 MOV R1, R1, LSR R2

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00000002
 END
```

OUTPUT:

INITIAL



FINAL



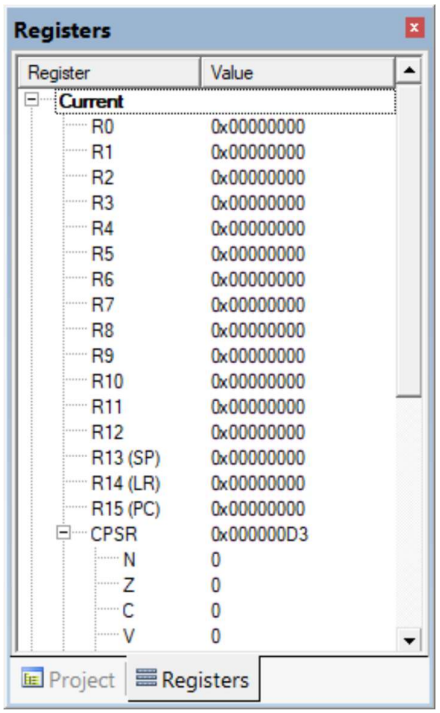
(c): Shift left 5 bits conditionally when zero flag is set

CODE:

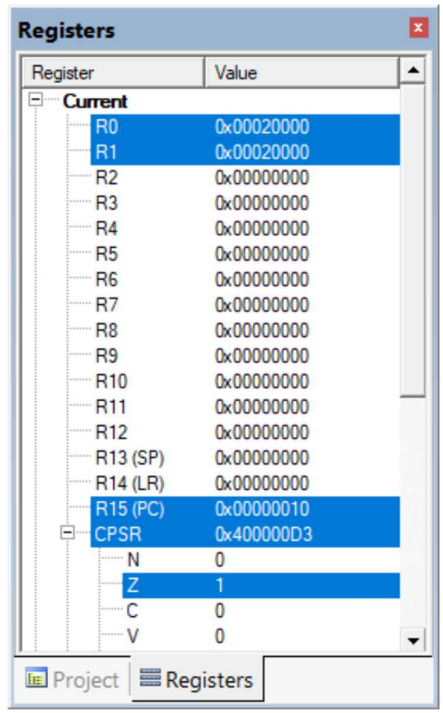
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE
 MOVS R1, #0     ; Setting Zero flag

 MOVEQ R0, R0, LSL #5

 AREA PROGRAM, DATA, READONLY
VALUE DCD &00001000
 END
```

OUTPUT:

INITIAL



FINAL



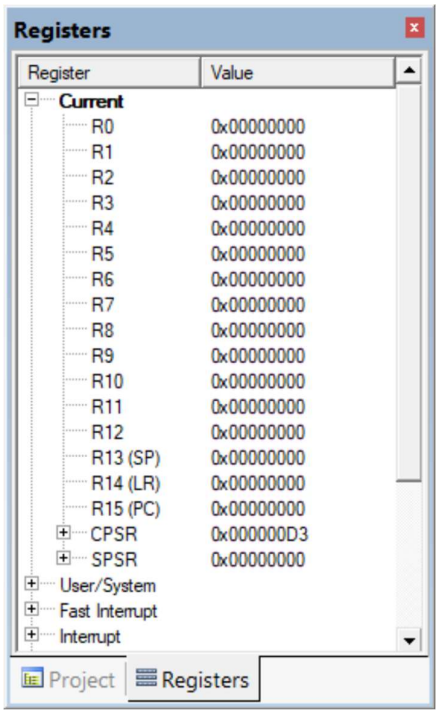
(d): Arithmetic shift right by the value contained in register R2

CODE:

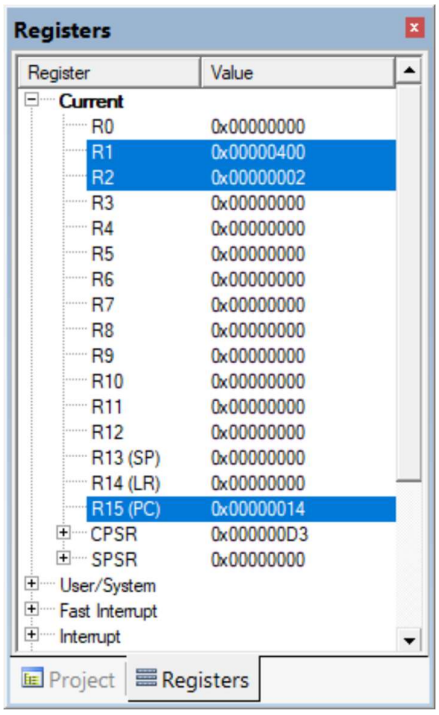
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R1, VALUE1
 LDR R2, VALUE2

 MOV R1, R1, ASR R2

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00000002
 END
```

OUTPUT:

INITIAL



FINAL



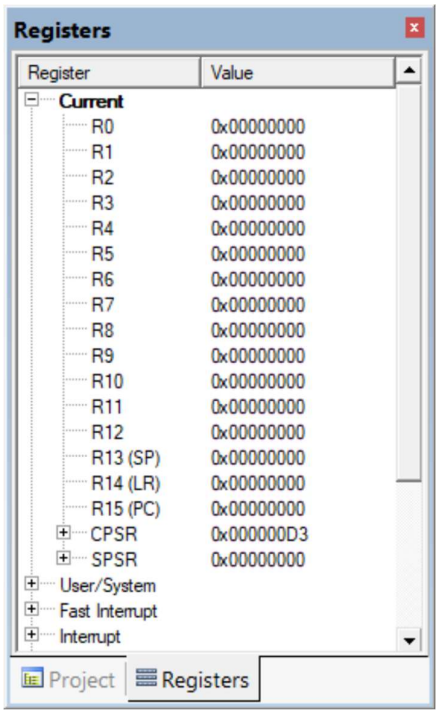
(e): Barrel Shifter

CODE:

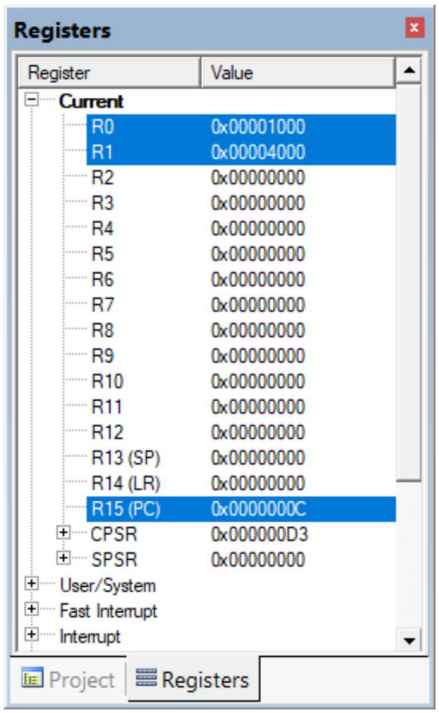
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 MOV R1, R0, LSL #2

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
 END
```

OUTPUT:

INITIAL



FINAL



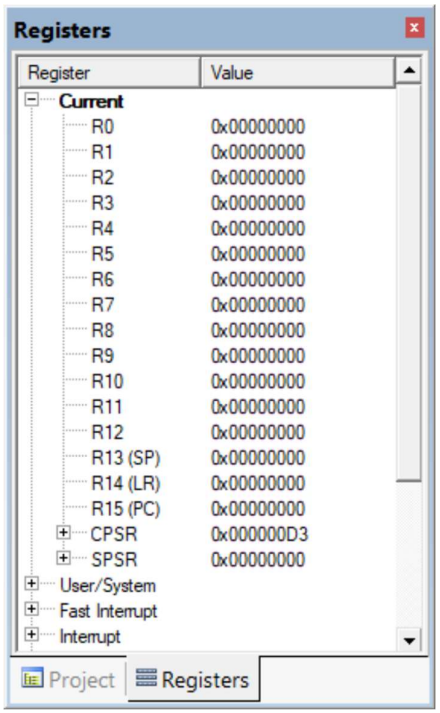
QUES 4 (a): Write a program in ARM assembly language to copy the block of data from source to destination using memory location with loop.(Load and Store Instruction)

CODE:

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 MOV R0, #8
 LDR R1, VALUE1
 LDR R2, VALUE2

LOOP
 LDR R3, [R1], #4
 STR R3, [R2], #4
 SUBS R0, R0, #1
 BNE LOOP

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00002000
 END
```

OUTPUT:

INITIAL

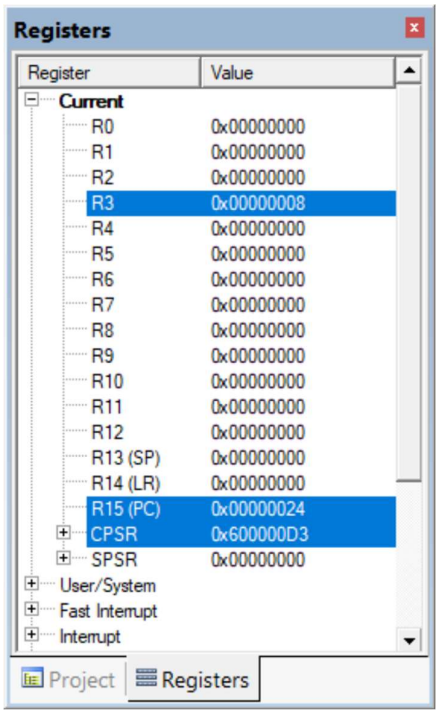


Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
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)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

Project | Registers

FINAL

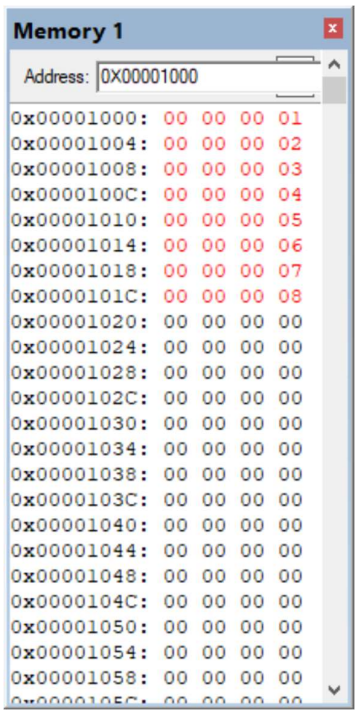


Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000008
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)	0x00000024
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

Project | Registers

MEMORY 1

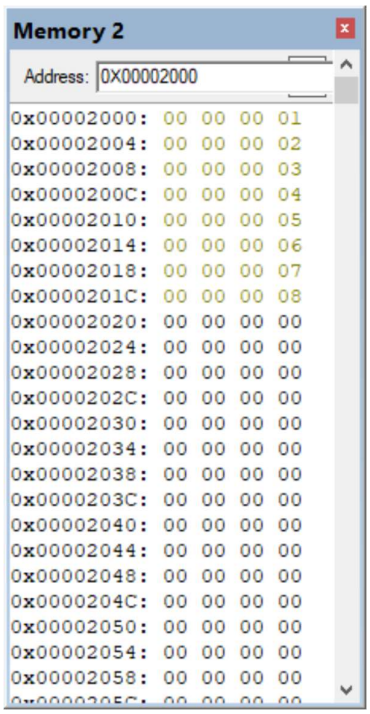


Memory 1

Address: 0X00001000

```
0x00001000: 00 00 00 01
0x00001004: 00 00 00 02
0x00001008: 00 00 00 03
0x0000100C: 00 00 00 04
0x00001010: 00 00 00 05
0x00001014: 00 00 00 06
0x00001018: 00 00 00 07
0x0000101C: 00 00 00 08
0x00001020: 00 00 00 00
0x00001024: 00 00 00 00
0x00001028: 00 00 00 00
0x0000102C: 00 00 00 00
0x00001030: 00 00 00 00
0x00001034: 00 00 00 00
0x00001038: 00 00 00 00
0x0000103C: 00 00 00 00
0x00001040: 00 00 00 00
0x00001044: 00 00 00 00
0x00001048: 00 00 00 00
0x0000104C: 00 00 00 00
0x00001050: 00 00 00 00
0x00001054: 00 00 00 00
0x00001058: 00 00 00 00
```

MEMORY 2



Memory 2

Address: 0X00002000

```
0x00002000: 00 00 00 01
0x00002004: 00 00 00 02
0x00002008: 00 00 00 03
0x0000200C: 00 00 00 04
0x00002010: 00 00 00 05
0x00002014: 00 00 00 06
0x00002018: 00 00 00 07
0x0000201C: 00 00 00 08
0x00002020: 00 00 00 00
0x00002024: 00 00 00 00
0x00002028: 00 00 00 00
0x0000202C: 00 00 00 00
0x00002030: 00 00 00 00
0x00002034: 00 00 00 00
0x00002038: 00 00 00 00
0x0000203C: 00 00 00 00
0x00002040: 00 00 00 00
0x00002044: 00 00 00 00
0x00002048: 00 00 00 00
0x0000204C: 00 00 00 00
0x00002050: 00 00 00 00
0x00002054: 00 00 00 00
0x00002058: 00 00 00 00
```

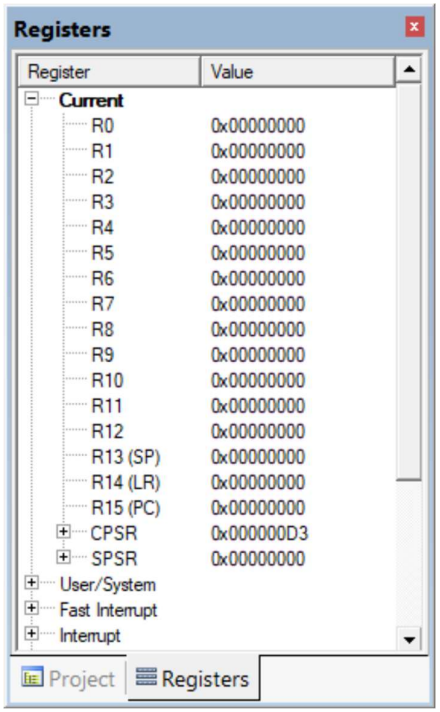
QUES 4 (b): Write a program in ARM assembly language to copy the block of data from source to destination using memory location without loop .(Multiple Load and Store Instruction)

CODE:

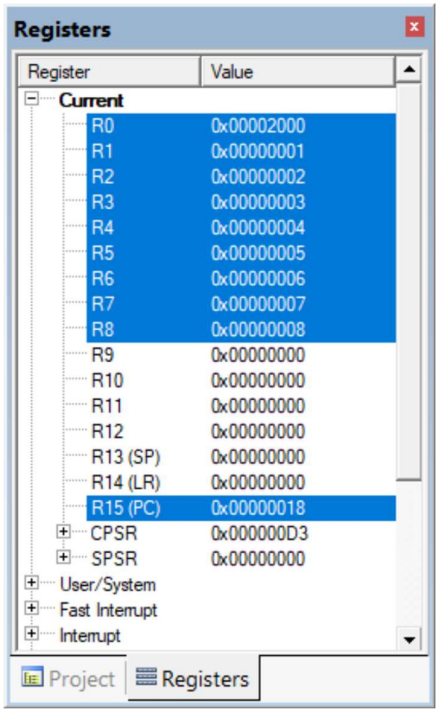
```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 LDMIA R0, {R1-R8}

 LDR R0, VALUE2
 STMIA R0, {R1-R8}

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00001000
VALUE2 DCD &00002000
 END
```

OUTPUT:

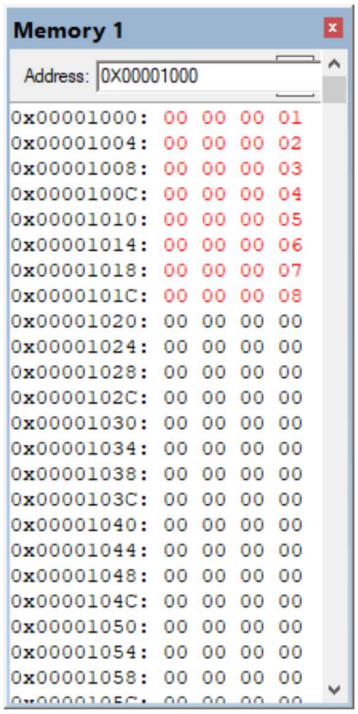
INITIAL



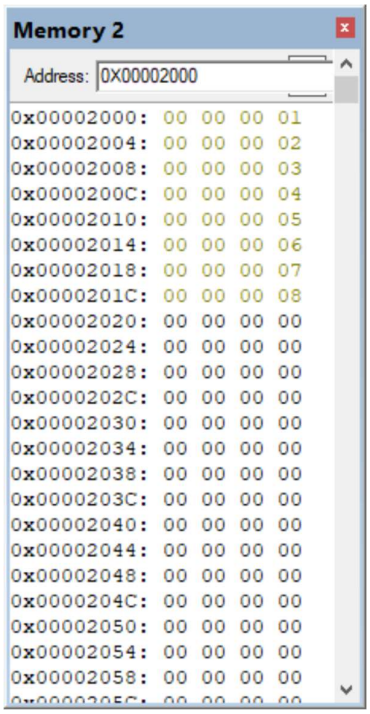
FINAL



MEMORY 1



MEMORY 2



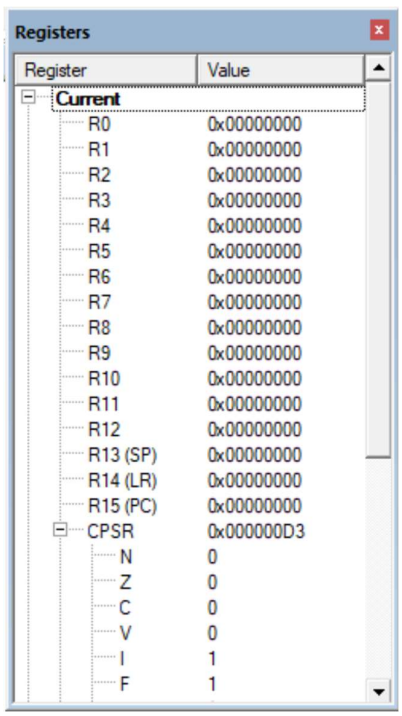
QUES 5: Write a program in ARM assembly language to perform multiplication using repeated addition.

CODE:

```
 AREA PROGRAM, CODE, READONLY
 ENTRY
MAIN
 LDR R0, VALUE1
 LDR R1, VALUE2

LOOP
 ADD R2, R2, R0
 SUBS R1, R1, #1
 BNE LOOP

 AREA PROGRAM, DATA, READONLY
VALUE1 DCD &02
VALUE2 DCD &03
 END
```

OUTPUT:

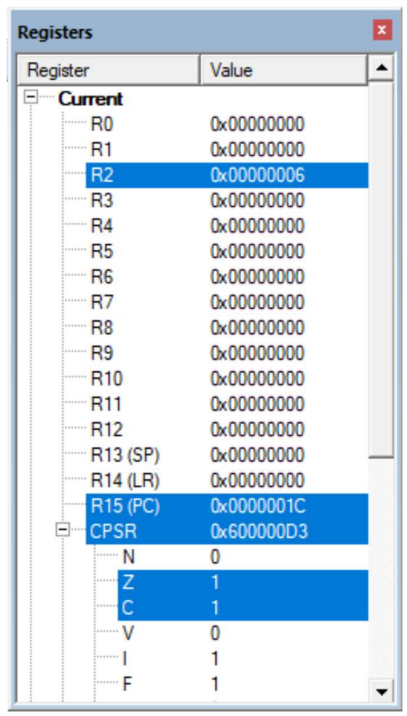
INITIAL



Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
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)	0x00000000
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1

FINAL



Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000006
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)	0x0000001C
CPSR	0x600000D3
N	0
Z	1
C	1
V	0
I	1
F	1