

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:

```

1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE
5  MOV R1, R0
6  MOV R2, R0
7
8  AREA PROGRAM, DATA, READONLY
9  VALUE DCD &10
10 END

```

OUTPUT:

INITIAL

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	

FINAL

Register	Value
Current	
R0	0x00000010
R1	0x00000010
R2	0x00000010
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	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  MOV R0, #0x00001000
5  MOV R1, #0x00002000
6
7  ADD R2, R1, R0
8  END
```

OUTPUT:

INITIAL

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	

FINAL

Register	Value
Current	
R0	0x00001000
R1	0x00002000
R2	0x00003000
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	

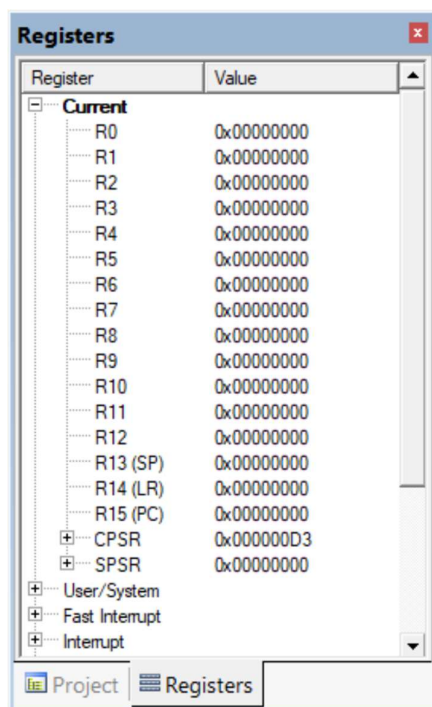
QUES 1 (c): Write a program in ARM assembly language to add two 32 bit numbers using Direct Addressing Mode.

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE1
5  LDR R1, VALUE2
6
7  ADD R2, R1, R0
8
9  AREA PROGRAM, DATA, READONLY
10 VALUE1 DCD &00001000
11 VALUE2 DCD &00002000
12 END
```

OUTPUT:

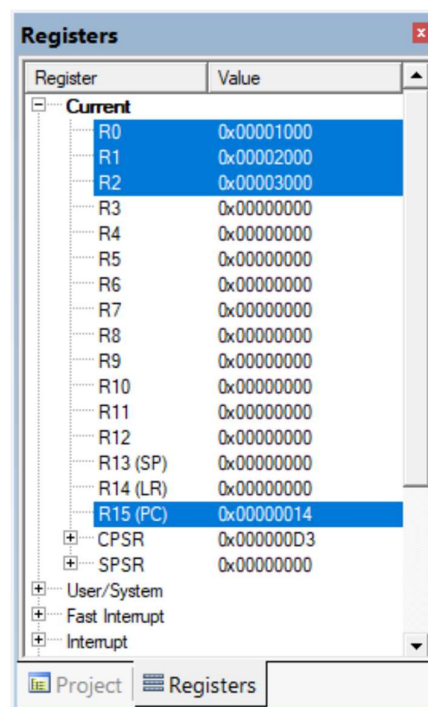
INITIAL



The 'Registers' window displays the initial state of the ARM registers. All registers (R0 through R15) and the CPSR/SPSR are set to 0x00000000. The PC (R15) is set to 0x00000000.

Register	Value
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	

FINAL



The 'Registers' window displays the final state of the ARM registers after execution. R0 contains 0x00001000, R1 contains 0x00002000, and R2 contains 0x00003000. The PC (R15) has updated to 0x00000014. The CPSR remains at 0x000000D3.

Register	Value
R0	0x00001000
R1	0x00002000
R2	0x00003000
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)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE1
5  LDR R1, VALUE2
6
7  LDR R2, [R0]
8  LDR R3, [R1]
9
10 ADD R4, R3, R2
11
12  AREA PROGRAM, DATA, READONLY
13 VALUE1 DCD &00001000
14 VALUE2 DCD &00001004
15  END
```

OUTPUT:

INITIAL

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	

FINAL

Register	Value
Current	
R0	0x00001000
R1	0x00001004
R2	0x01020304
R3	0x50607080
R4	0x51627384
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	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

MEMORY

Address	Value
0x00001000	01 02 03 04
0x00001004	50 60 70 80
0x00001008	00 00 00 00
0x0000100C	00 00 00 00
0x00001010	00 00 00 00
0x00001014	00 00 00 00
0x00001018	00 00 00 00
0x0000101C	00 00 00 00
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
0x0000105C	00 00 00 00

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE
5  MVN R1, R0
6
7  AREA PROGRAM, DATA, READONLY
8  VALUE DCD &00000001
9  END
```

OUTPUT:

INITIAL

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	

FINAL

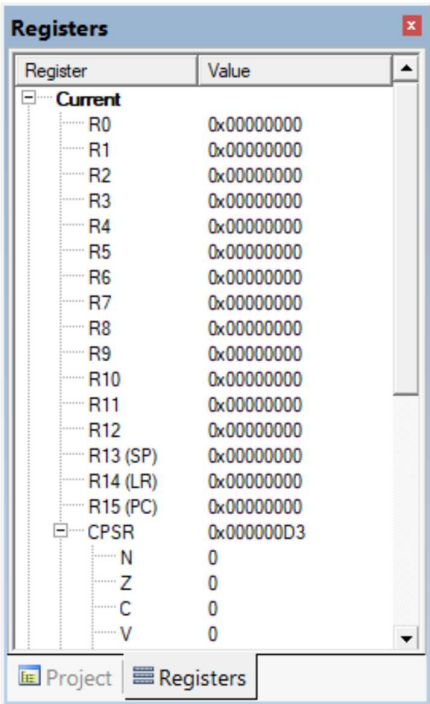
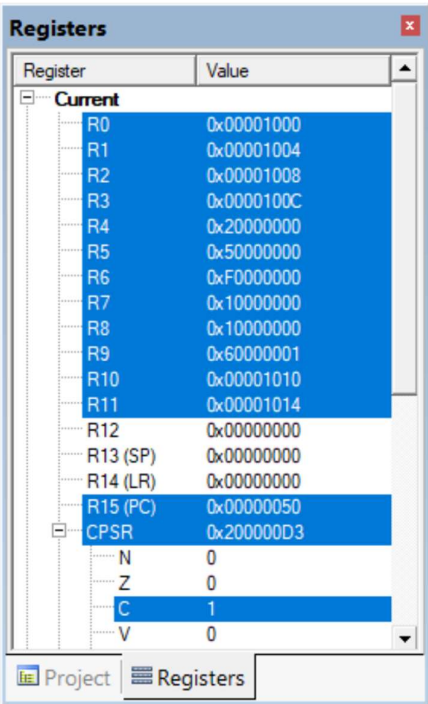
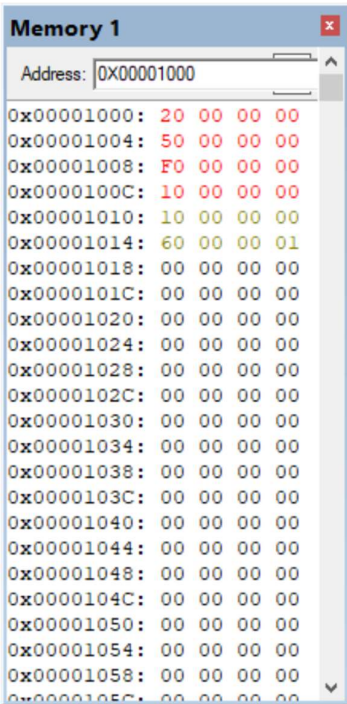
Register	Value
Current	
R0	0x00000001
R1	0xFFFFFFFF
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

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:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE1
5  LDR R1, VALUE2
6  LDR R2, VALUE3
7  LDR R3, VALUE4
8
9  LDR R4, [R0]
10 LDR R5, [R1]
11 LDR R6, [R2]
12 LDR R7, [R3]
13
14 ADDS R8, R4, R6
15 ADC R9, R5, R7
16
17 LDR R10, VALUE5
18 LDR R11, VALUE6
19
20 STR R8, [R10]
21 STR R9, [R11]
22
23 AREA PROGRAM, DATA, READONLY
24
25 VALUE1 DCD &00001000
26 VALUE2 DCD &00001004
27 VALUE3 DCD &00001008
28 VALUE4 DCD &0000100C
29 VALUE5 DCD &00001010
30 VALUE6 DCD &00001014
31
32 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:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE
5  MOV R0, R0, LSL #2
6
7  AREA PROGRAM, DATA, READONLY
8  VALUE DCD &00001000
9  END
```

OUTPUT:

INITIAL		FINAL	
Registers		Registers	
Register	Value	Register	Value
Current		Current	
R0	0x00000000	R0	0x00004000
R1	0x00000000	R1	0x00000000
R2	0x00000000	R2	0x00000000
R3	0x00000000	R3	0x00000000
R4	0x00000000	R4	0x00000000
R5	0x00000000	R5	0x00000000
R6	0x00000000	R6	0x00000000
R7	0x00000000	R7	0x00000000
R8	0x00000000	R8	0x00000000
R9	0x00000000	R9	0x00000000
R10	0x00000000	R10	0x00000000
R11	0x00000000	R11	0x00000000
R12	0x00000000	R12	0x00000000
R13 (SP)	0x00000000	R13 (SP)	0x00000000
R14 (LR)	0x00000000	R14 (LR)	0x00000000
R15 (PC)	0x00000000	R15 (PC)	0x0000000C
CPSR	0x000000D3	CPSR	0x000000D3
SPSR	0x00000000	SPSR	0x00000000
User/System		User/System	
Fast Interrupt		Fast Interrupt	
Interrupt		Interrupt	

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R1, VALUE1
5  LDR R2, VALUE2
6
7  MOV R1, R1, LSR R2
8
9  AREA PROGRAM, DATA, READONLY
10 VALUE1 DCD &00001000
11 VALUE2 DCD &00000002
12 END
```


OUTPUT:

INITIAL		FINAL	
Registers		Registers	
Register	Value	Register	Value
Current		Current	
R0	0x00000000	R0	0x00000000
R1	0x00000000	R1	0x00000400
R2	0x00000000	R2	0x00000002
R3	0x00000000	R3	0x00000000
R4	0x00000000	R4	0x00000000
R5	0x00000000	R5	0x00000000
R6	0x00000000	R6	0x00000000
R7	0x00000000	R7	0x00000000
R8	0x00000000	R8	0x00000000
R9	0x00000000	R9	0x00000000
R10	0x00000000	R10	0x00000000
R11	0x00000000	R11	0x00000000
R12	0x00000000	R12	0x00000000
R13 (SP)	0x00000000	R13 (SP)	0x00000000
R14 (LR)	0x00000000	R14 (LR)	0x00000000
R15 (PC)	0x00000000	R15 (PC)	0x00000014
CPSR	0x000000D3	CPSR	0x000000D3
SPSR	0x00000000	SPSR	0x00000000
User/System		User/System	
Fast Interrupt		Fast Interrupt	
Interrupt		Interrupt	

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE
5  MOVS R1, #0      ; Setting Zero flag
6
7  MOVEQ R0, R0, LSL #5
8
9  AREA PROGRAM, DATA, READONLY
10 VALUE DCD &00001000
11 END
```

OUTPUT:

INITIAL		FINAL	
Registers		Registers	
Register	Value	Register	Value
Current		Current	
R0	0x00000000	R0	0x00020000
R1	0x00000000	R1	0x00020000
R2	0x00000000	R2	0x00000000
R3	0x00000000	R3	0x00000000
R4	0x00000000	R4	0x00000000
R5	0x00000000	R5	0x00000000
R6	0x00000000	R6	0x00000000
R7	0x00000000	R7	0x00000000
R8	0x00000000	R8	0x00000000
R9	0x00000000	R9	0x00000000
R10	0x00000000	R10	0x00000000
R11	0x00000000	R11	0x00000000
R12	0x00000000	R12	0x00000000
R13 (SP)	0x00000000	R13 (SP)	0x00000000
R14 (LR)	0x00000000	R14 (LR)	0x00000000
R15 (PC)	0x00000000	R15 (PC)	0x00000010
CPSR	0x000000D3	CPSR	0x400000D3
N	0	N	0
Z	0	Z	1
C	0	C	0
V	0	V	0

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

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R1, VALUE1
5  LDR R2, VALUE2
6
7  MOV R1, R1, ASR R2
8
9  AREA PROGRAM, DATA, READONLY
10 VALUE1 DCD &00001000
11 VALUE2 DCD &00000002
12 END
```

OUTPUT:

INITIAL		FINAL	
Registers		Registers	
Register	Value	Register	Value
Current		Current	
R0	0x00000000	R0	0x00000000
R1	0x00000000	R1	0x00000400
R2	0x00000000	R2	0x00000002
R3	0x00000000	R3	0x00000000
R4	0x00000000	R4	0x00000000
R5	0x00000000	R5	0x00000000
R6	0x00000000	R6	0x00000000
R7	0x00000000	R7	0x00000000
R8	0x00000000	R8	0x00000000
R9	0x00000000	R9	0x00000000
R10	0x00000000	R10	0x00000000
R11	0x00000000	R11	0x00000000
R12	0x00000000	R12	0x00000000
R13 (SP)	0x00000000	R13 (SP)	0x00000000
R14 (LR)	0x00000000	R14 (LR)	0x00000000
R15 (PC)	0x00000000	R15 (PC)	0x00000014
CPSR	0x000000D3	CPSR	0x000000D3
SPSR	0x00000000	SPSR	0x00000000
User/System		User/System	
Fast Interrupt		Fast Interrupt	
Interrupt		Interrupt	

(e): Barrel Shifter

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE1
5  MOV R1, R0, LSL #2
6
7  AREA PROGRAM, DATA, READONLY
8  VALUE1 DCD &00001000
9  END
```

OUTPUT:

INITIAL		FINAL	
Registers		Registers	
Register	Value	Register	Value
Current		Current	
R0	0x00000000	R0	0x00001000
R1	0x00000000	R1	0x00004000
R2	0x00000000	R2	0x00000000
R3	0x00000000	R3	0x00000000
R4	0x00000000	R4	0x00000000
R5	0x00000000	R5	0x00000000
R6	0x00000000	R6	0x00000000
R7	0x00000000	R7	0x00000000
R8	0x00000000	R8	0x00000000
R9	0x00000000	R9	0x00000000
R10	0x00000000	R10	0x00000000
R11	0x00000000	R11	0x00000000
R12	0x00000000	R12	0x00000000
R13 (SP)	0x00000000	R13 (SP)	0x00000000
R14 (LR)	0x00000000	R14 (LR)	0x00000000
R15 (PC)	0x00000000	R15 (PC)	0x0000000C
CPSR	0x000000D3	CPSR	0x000000D3
SPSR	0x00000000	SPSR	0x00000000
User/System		User/System	
Fast Interrupt		Fast Interrupt	
Interrupt		Interrupt	

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:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  MOV R0, #8
5  LDR R1, VALUE1
6  LDR R2, VALUE2
7
8  LOOP
9  LDR R3, [R1], #4
10 STR R3, [R2], #4
11 SUBS R0, R0, #1
12 BNE LOOP
13
14 AREA PROGRAM, DATA, READONLY
15 VALUE1 DCD &00001000
16 VALUE2 DCD &00002000
17 END
```

OUTPUT:

INITIAL

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	

FINAL

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	

MEMORY 1

Address	Value
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
0x0000105C	00 00 00 00

MEMORY 2

Address	Value
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
0x0000205C	00 00 00 00

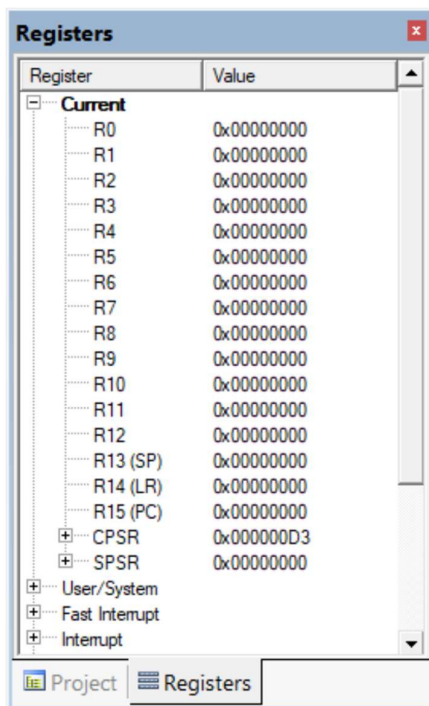
QUES 4 (b): Write a program in ARM assembly language to copy the block of data from source to destination using memory location without loop .(Multiple Load and Store Instruction)

CODE:

```
1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  MAIN
4  LDR R0, VALUE1
5  LDMIA R0, {R1-R8}
6
7  LDR R0, VALUE2
8  STMIA R0, {R1-R8}
9
10 AREA PROGRAM, DATA, READONLY
11 VALUE1 DCD &00001000
12 VALUE2 DCD &00002000
13 END
```

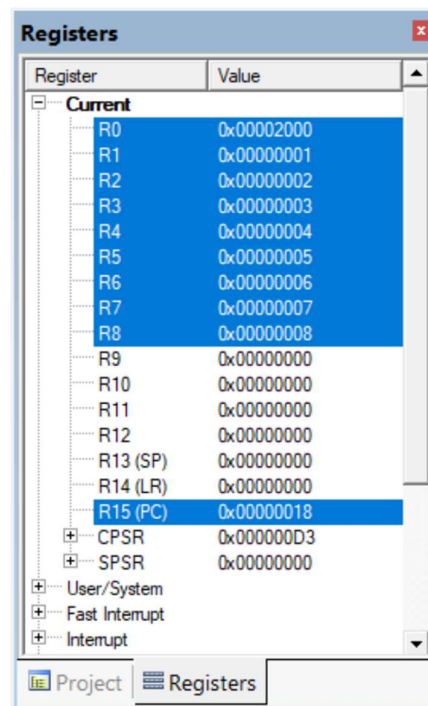
OUTPUT:

INITIAL



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	

FINAL



Register	Value
Current	
R0	0x00002000
R1	0x00000001
R2	0x00000002
R3	0x00000003
R4	0x00000004
R5	0x00000005
R6	0x00000006
R7	0x00000007
R8	0x00000008
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	

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
0x0000105C:	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
0x0000205C:	00	00	00	00

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

CODE:

```
1 AREA PROGRAM, CODE, READONLY
2 ENTRY
3 MAIN
4 LDR R0, VALUE1
5 LDR R1, VALUE2
6
7 LOOP
8 ADD R2, R2, R0
9 SUBS R1, R1, #1
10 BNE LOOP
11
12 AREA PROGRAM, DATA, READONLY
13 VALUE1 DCD &02
14 VALUE2 DCD &03
15 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

