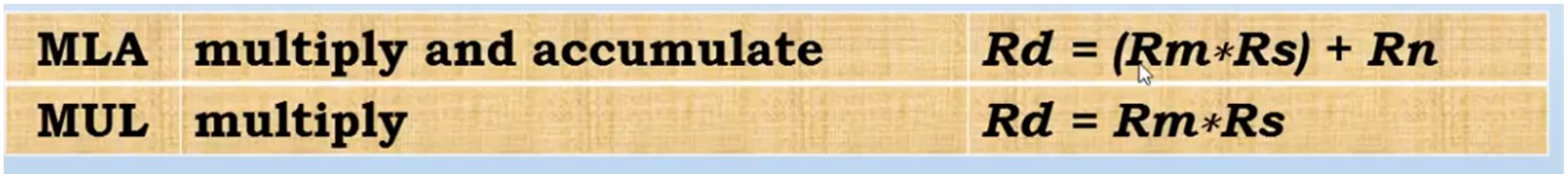
ARM Assembly

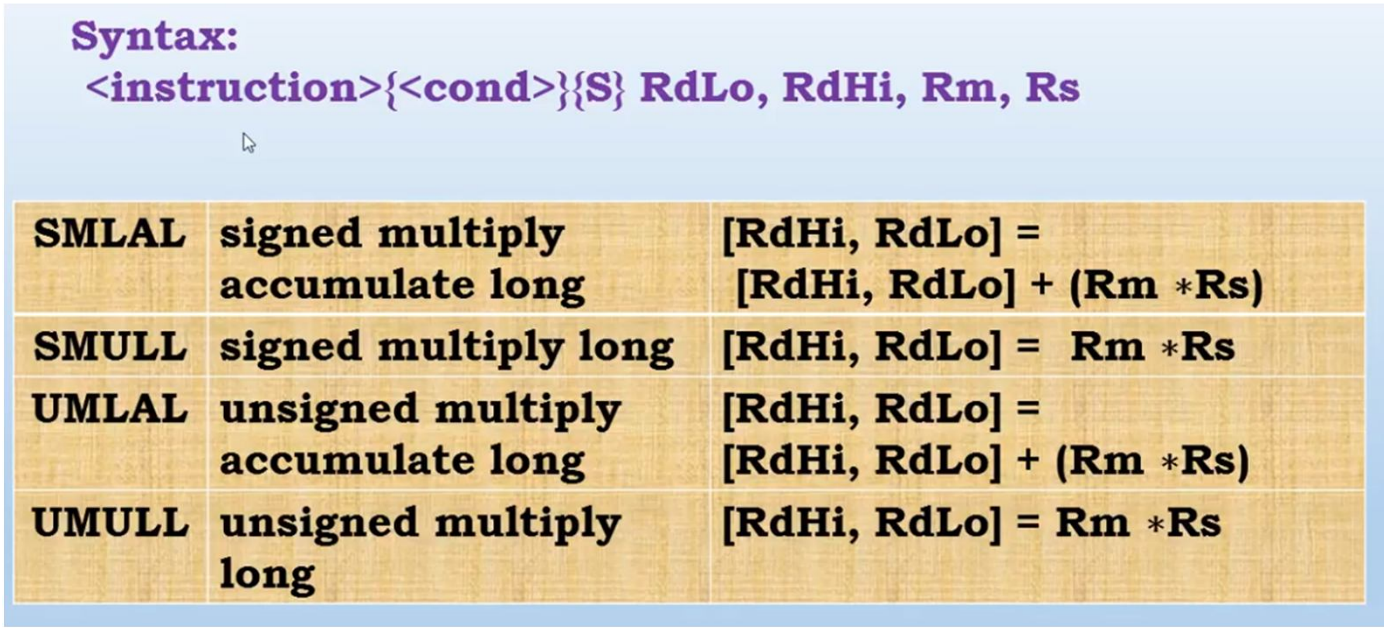
A.Baskar
Dept of CSE
Amrita Vishwa Vidyapeetham

```
AREA   Factorial,CODE,READONLY
       ENTRY
       MOV     R0,#4
       BL      FACT
B1     B   B1


FACT
    MOV         R3, #1
loop
    CMP        R0, #1
    MULGT      R3, R0, R3
    SUBGT       R0, R0, #1
    BGT         loop
    MOV         PC,R14
```



MLA	**multiply and accumulate**	$Rd = (Rm*Rs) + Rn$
MUL	**multiply**	$Rd = Rm*Rs$

Long Multiply 64 bit



Syntax:
<instruction>{<cond>}{S} RdLo, RdHi, Rm, Rs

SMLAL	**signed multiply accumulate long**	**[RdHi, RdLo] = [RdHi, RdLo] + (Rm *Rs)**
SMULL	**signed multiply long**	**[RdHi, RdLo] = Rm *Rs**
UMLAL	**unsigned multiply accumulate long**	**[RdHi, RdLo] = [RdHi, RdLo] + (Rm *Rs)**
UMULL	**unsigned multiply long**	**[RdHi, RdLo] = Rm *Rs**

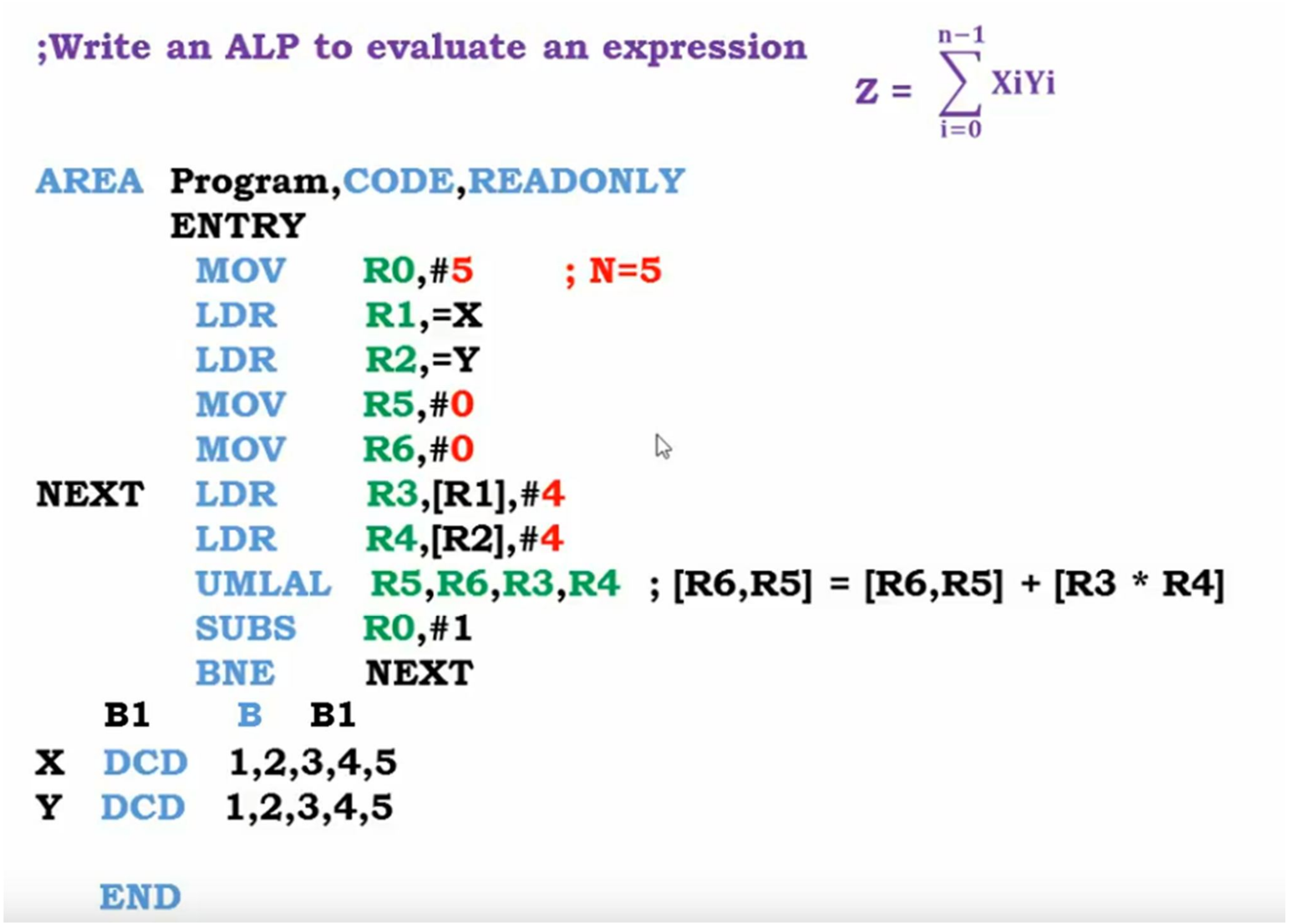
PRE:

```
r0 = 0x00000000
r1 = 0x00000000
r2 = 0xf0000002
r3 = 0x00000002
```

```
UMULL  r0, r1, r2, r3      ; [r1,r0] = r2*r3
```

POST:

```
r0 = 0xe0000004 ; = RdLo
r1 = 0x00000001 ; = RdHi
```

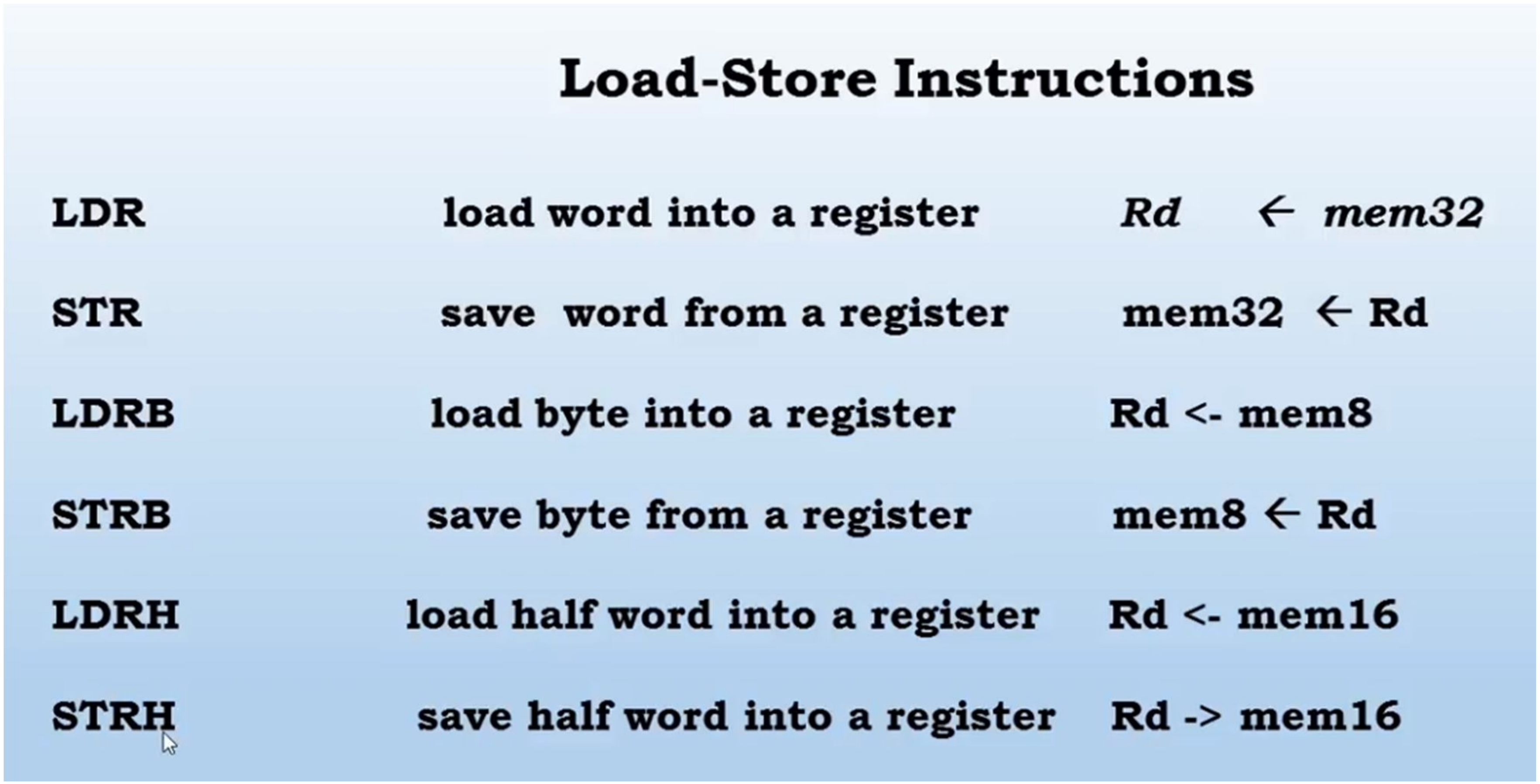


;Write an ALP to evaluate an expression

$$Z = \sum_{i=0}^{n-1} X_i Y_i$$

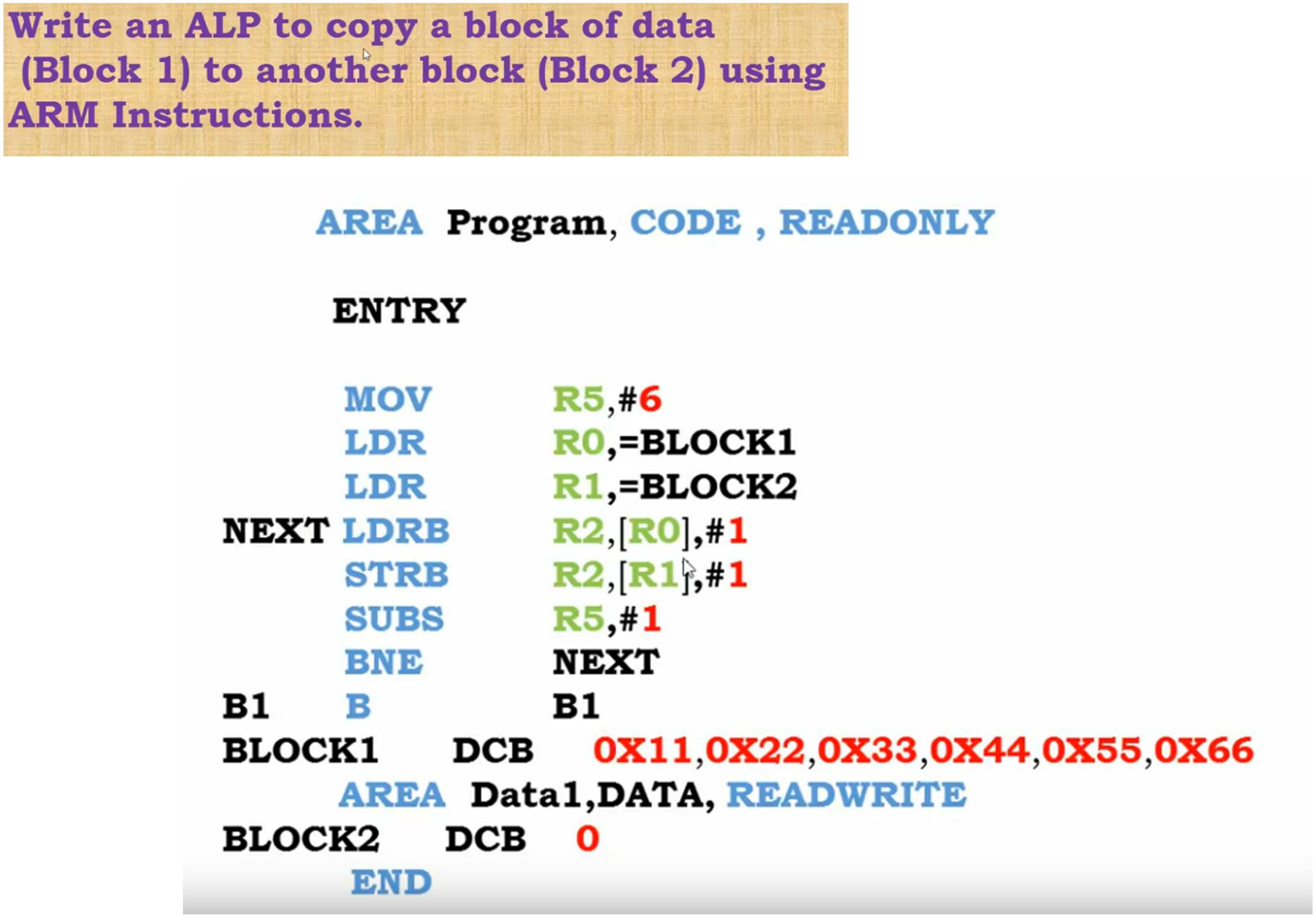
```
AREA   Program,CODE,READONLY
       ENTRY
        MOV     R0,#5       ; N=5
        LDR     R1,=X
        LDR     R2,=Y
        MOV     R5,#0
        MOV     R6,#0
NEXT    LDR     R3,[R1],#4
        LDR     R4,[R2],#4
        UMLAL   R5,R6,R3,R4  ; [R6,R5] = [R6,R5] + [R3 * R4]
        SUBS    R0,#1
        BNE     NEXT
    B1      B   B1
X   DCD   1,2,3,4,5
Y   DCD   1,2,3,4,5

    END
```

Assembler directive: DCD-Define constant for double word



Load-Store Instructions

LDR	load word into a register	*Rd ← mem32*
STR	save word from a register	mem32 ← Rd
LDRB	load byte into a register	Rd <- mem8
STRB	save byte from a register	mem8 ← Rd
LDRH	load half word into a register	Rd <- mem16
STRH	save half word into a register	Rd -> mem16

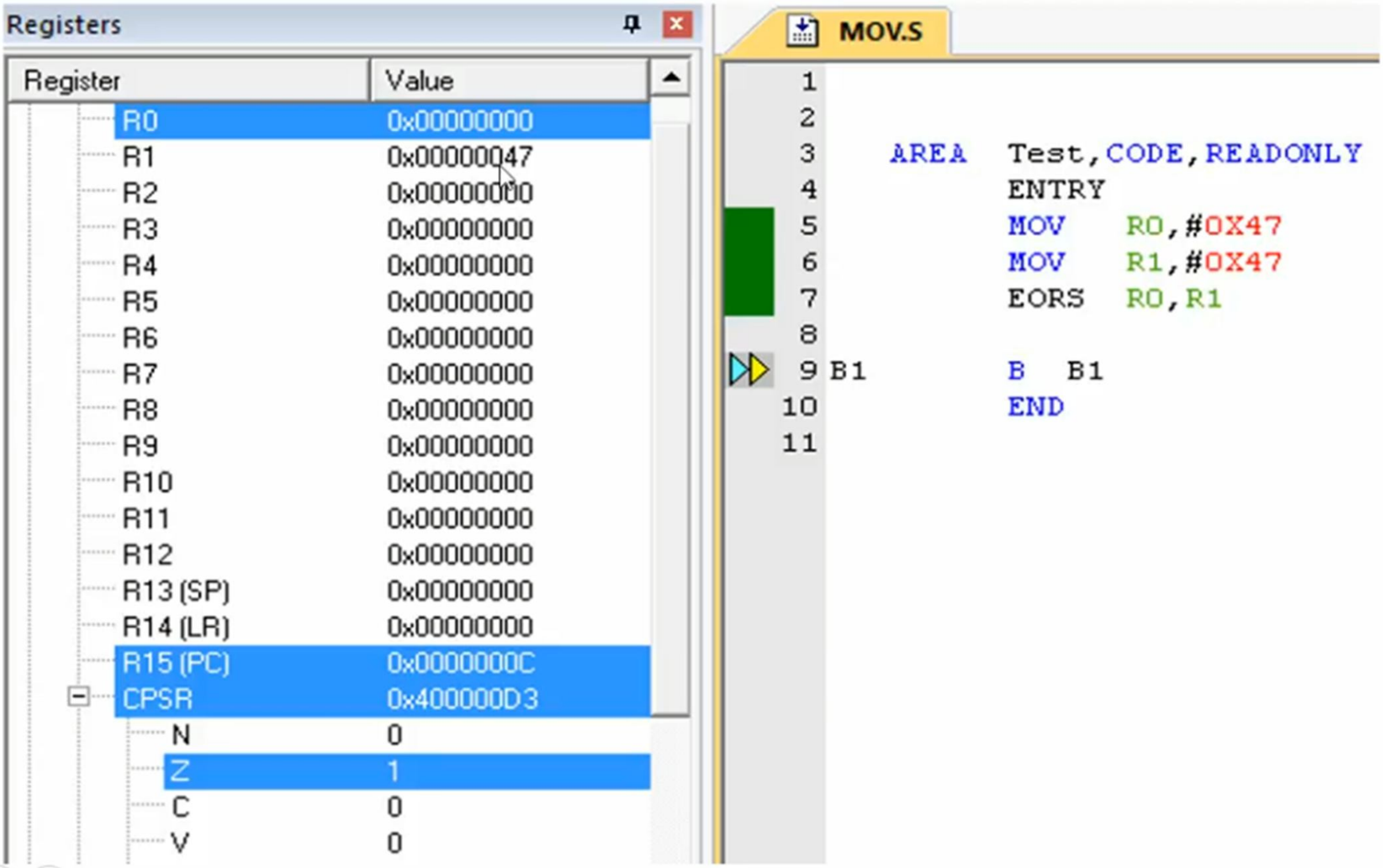


Write an ALP to copy a block of data (Block 1) to another block (Block 2) using ARM Instructions.

```
       AREA  Program, CODE , READONLY

        ENTRY

        MOV         R5,#6
        LDR         R0,=BLOCK1
        LDR         R1,=BLOCK2
NEXT    LDRB        R2,[R0],#1
        STRB        R2,[R1],#1
        SUBS        R5,#1
        BNE         NEXT
B1      B           B1
BLOCK1      DCB    0X11,0X22,0X33,0X44,0X55,0X66
        AREA  Data1,DATA, READWRITE
BLOCK2      DCB    0
        END
```

CMN	**compare negated**	***flags set as a result of Rn + N***
CMP	**compare**	***flags set as a result of Rn − N***
TEQ	**test for equality of two 32-bit values**	***flags set as a result of Rn ∧ N***
TST	**test bits of a 32-bit value**	***flags set as a result of Rn & N***

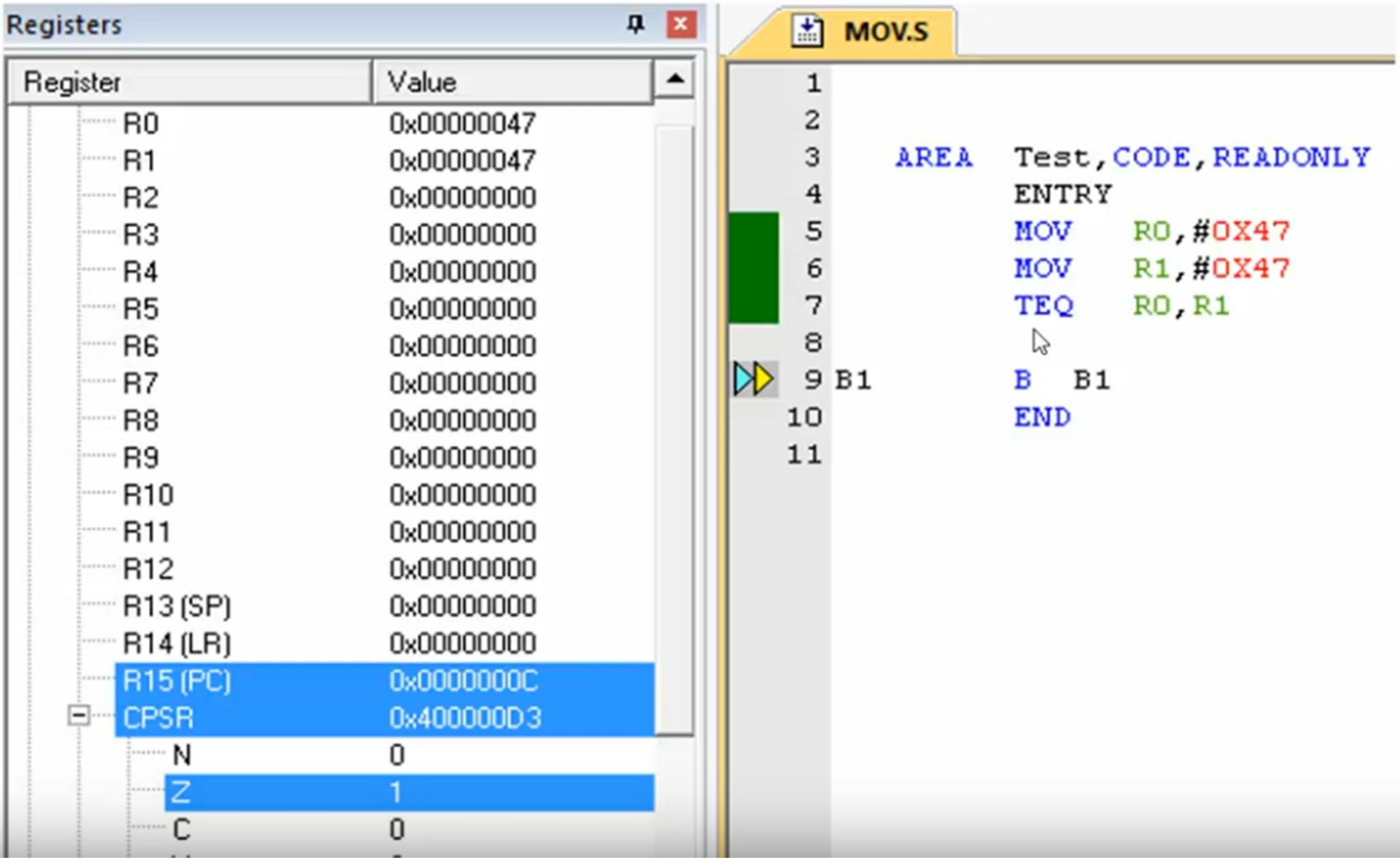
Comparison between EORS and TEQ Instruction



Register	Value
R0	0x00000000
R1	0x00000047
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	0x400000D3
N	0
Z	1
C	0
V	0

```


    AREA    Test,CODE,READONLY
            ENTRY
            MOV     R0,#0X47
            MOV     R1,#0X47
            EORS    R0,R1

B1          B   B1
            END

```

Comparison between EORS and TEQ Instruction



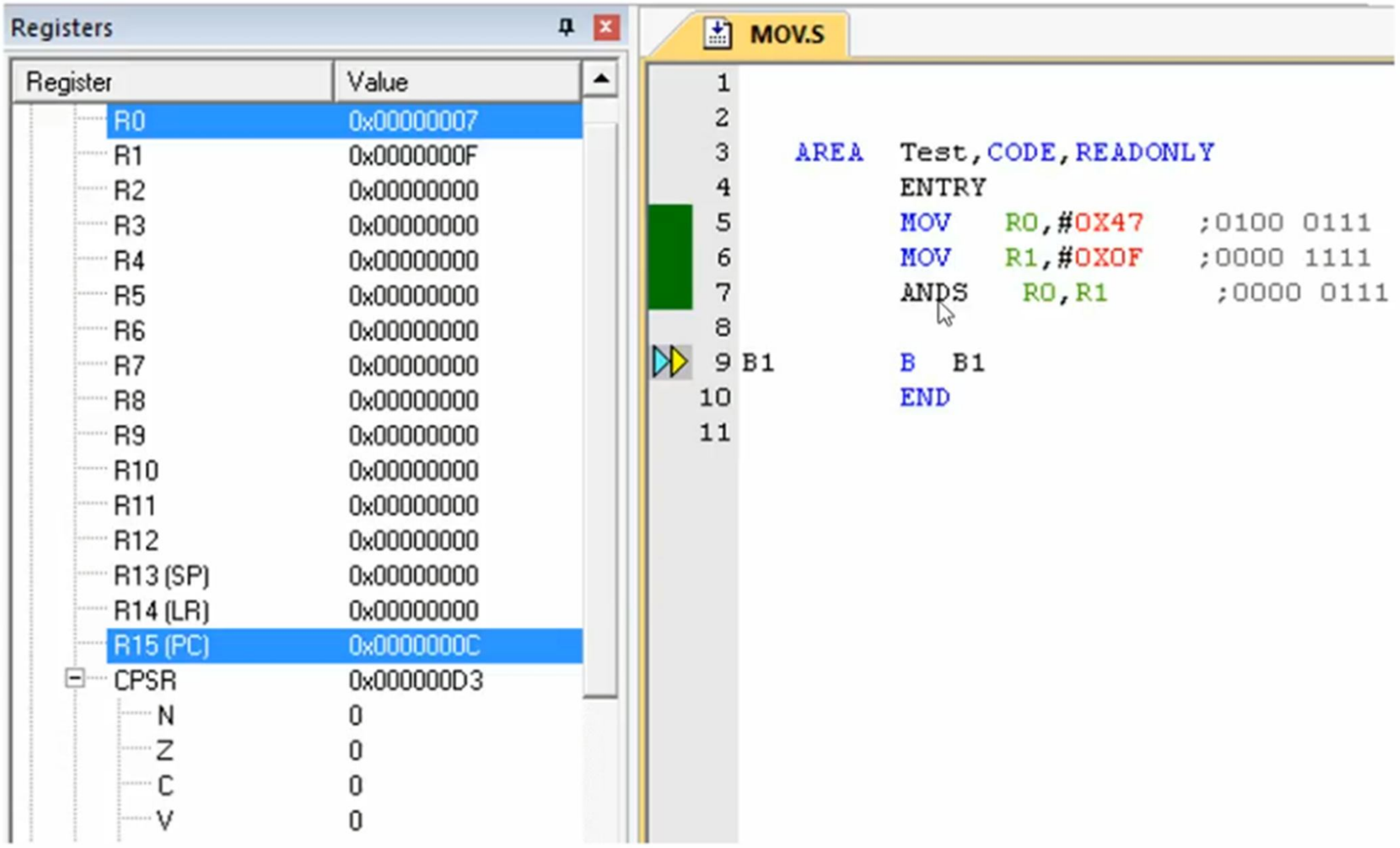
Registers

Register	Value
R0	0x00000047
R1	0x00000047
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	0x400000D3
N	0
Z	1
C	0

MOV.S

```
    AREA   Test,CODE,READONLY
           ENTRY
           MOV    R0,#0X47
           MOV    R1,#0X47
           TEQ    R0,R1

B1         B   B1
           END
```

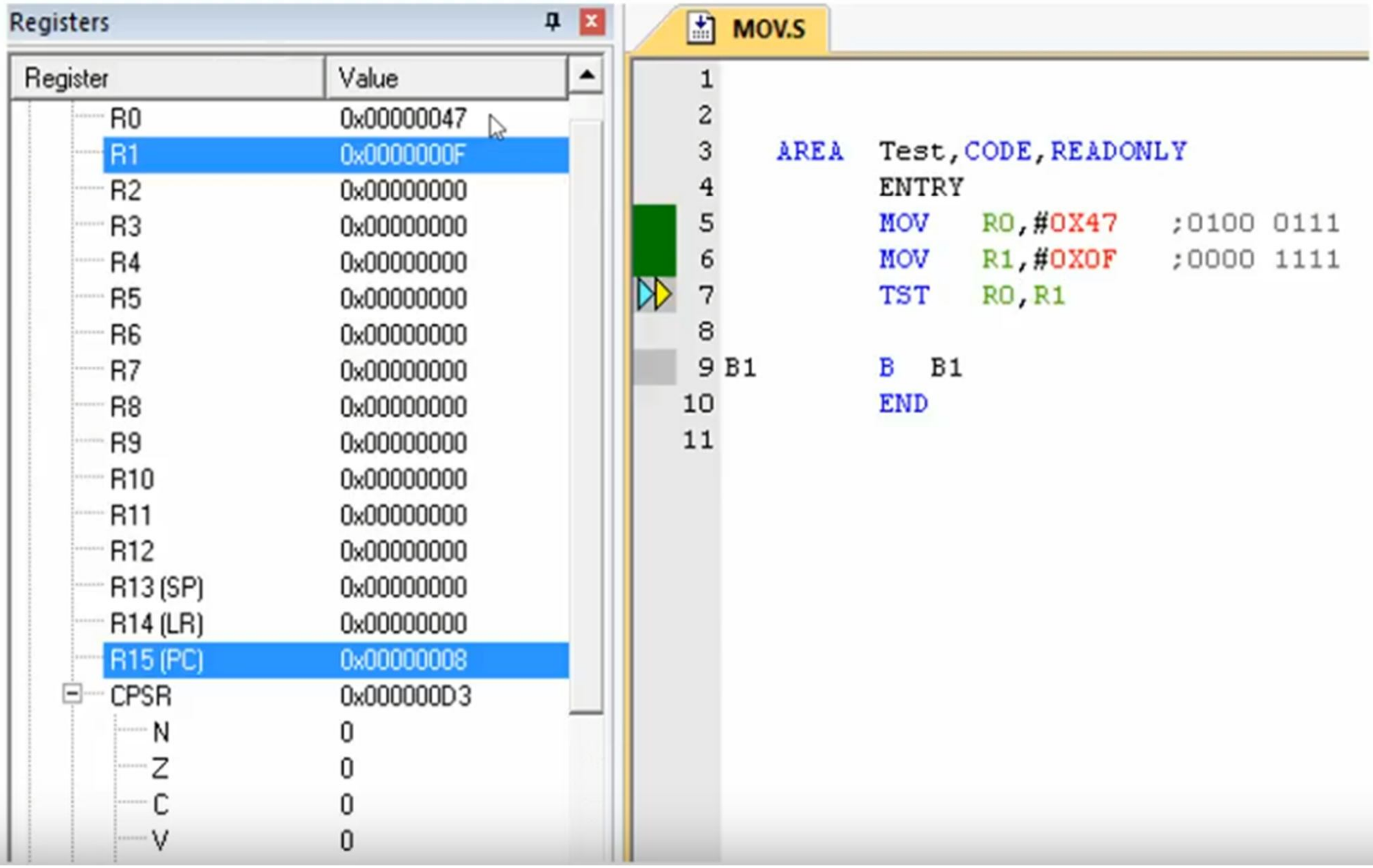
Comparison between AND - TST Instruction



Register	Value
R0	0x00000007
R1	0x0000000F
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
N	0
Z	0
C	0
V	0

```


        AREA    Test,CODE,READONLY
                ENTRY
                MOV     R0,#0X47        ;0100 0111
                MOV     R1,#0X0F        ;0000 1111
                ANDS     R0,R1           ;0000 0111

B1              B   B1
                END

```

Comparison between AND - TST Instruction



Registers

Register	Value
R0	0x00000047
R1	0x0000000F
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)	0x00000008
CPSR	0x000000D3
N	0
Z	0
C	0
V	0

MOV.S

```


    AREA   Test,CODE,READONLY
           ENTRY
           MOV    R0,#0X47    ;0100 0111
           MOV    R1,#0X0F    ;0000 1111
           TST    R0,R1

B1         B   B1
           END

```