Question 2.1.1

PRITISH WADHWA

- 1)
  $(1010\ 1011\ 1100\ 1101)_2 = (1 \times 2^{15}) + (0 \times 2^{14}) + (1 \times 2^{13}) + (0 \times 2^{12}) + (1 \times 2^{11}) + (0 \times 2^{10}) + (1 \times 2^{9}) + (1 \times 2^{8}) + (1 \times 2^{7}) + (1 \times 2^{6}) + (0 \times 2^{5}) + (0 \times 2^{4}) + (1 \times 2^{3}) + (1 \times 2^{2}) + (0 \times 2^{1}) + (1 \times 2^{0}) = (43981)_{10}$

- 2)
  $(1111\ 1110\ 1101\ 1100)_2 = (1 \times 2^{15}) + (1 \times 2^{14}) + (1 \times 2^{13}) + (1 \times 2^{12}) + (1 \times 2^{11}) + (1 \times 2^{10}) + (1 \times 2^{9}) + (0 \times 2^{8}) + (1 \times 2^{7}) + (1 \times 2^{6}) + (0 \times 2^{5}) + (1 \times 2^{4}) + (1 \times 2^{3}) + (1 \times 2^{2}) + (0 \times 2^{1}) + (0 \times 2^{0}) = (65244)_{10}$

- 3)
  $(0111\ 1101\ 1111\ 1000)_2 = (0 \times 2^{15}) + (1 \times 2^{14}) + (1 \times 2^{13}) + (1 \times 2^{12}) + (1 \times 2^{11}) + (1 \times 2^{10}) + (0 \times 2^{9}) + (1 \times 2^{8}) + (1 \times 2^{7}) + (1 \times 2^{6}) + (1 \times 2^{5}) + (1 \times 2^{4}) + (1 \times 2^{3}) + (0 \times 2^{2}) + (0 \times 2^{1}) + (0 \times 2^{0}) = (32248)_{10}$

- 4)
  $(0011\ 0000\ 0011\ 1001)_2 = (0 \times 2^{15}) + (0 \times 2^{14}) + (1 \times 2^{13}) + (1 \times 2^{12}) + (0 \times 2^{11}) + (0 \times 2^{10}) + (0 \times 2^{9}) + (0 \times 2^{8}) + (0 \times 2^{7}) + (0 \times 2^{6}) + (1 \times 2^{5}) + (1 \times 2^{4}) + (1 \times 2^{3}) + (0 \times 2^{2}) + (0 \times 2^{1}) + (1 \times 2^{0}) = (12345)_{10}$

Question 2.1.2

PRITISH WADHWA

- 1)
  $(A000)_{16} = (10 \times 16^3) + (0 \times 16^2) + (0 \times 16^1) + (0 \times 16^0) = (40960)_{10}$

- 2)
  $(8A89)_{16} = (8 \times 16^3) + (10 \times 16^2) + (8 \times 16^1) + (9 \times 16^0) = (35465)_{10}$

- 3)
  $(0190)_{16} = (0 \times 16^3) + (1 \times 16^2) + (9 \times 16^1) + (0 \times 16^0) = (400)_{10}$

- 4)
  $(AFCD)_{16} = (10 \times 16^3) + (15 \times 16^2) + (12 \times 16^1) + (13 \times 16^0)$
  $= (45005)_{10}$

Question 2.1.3

PRITISH WADHWA

- To convert Hex to Octal, I will first convert hex to binary and then I will convert the corresponding binary sequence to octal.

- 1)

A	0	0	0		
1010	0000	0000	0000		
001	010	000	000	000	000
1	2	0	0	0	0

$(A000)_{16} = (120000)_8$

- 2)

8	A	8	9		
1000	1010	1000	1001		
001	000	101	010	001	001
1	0	5	2	1	1

$(8A89)_{16} = (105211)_8$

- 3)

0	1	9	0		
0000	0001	1001	0000		
000	000	000	110	010	000
0	0	0	6	2	0

$(0190)_{16} = (000620)_8$

- 4)

A	F	C	D		
1010	1111	1100	1101		
001	010	111	111	001	101
1	2	7	7	1	5

$(AFCD)_{16} = (127715)_8$

Question 2.1.4

PRITISH WADHWA

- 1)

001	010	101	111	001	101
1	2	5	7	1	5

$(1010\ 1011\ 1100\ 1101)_2 = (125715)_8$

- 2)

001	111	111	011	011	100
1	7	7	3	3	4

$(1111\ 1110\ 1101\ 1100)_2 = (177334)_8$

- 3)

000	111	110	111	111	000
0	7	6	7	7	0

$(0111\ 1101\ 1111\ 1000)_2 = (076770)_8$

- 4)

000	011	000	000	111	001
0	3	0	0	7	1

$(0011\ 0000\ 0011\ 1001)_2 = (030071)_8$

Question 2.2.2 - ARMsim

PRITISH WADHWA

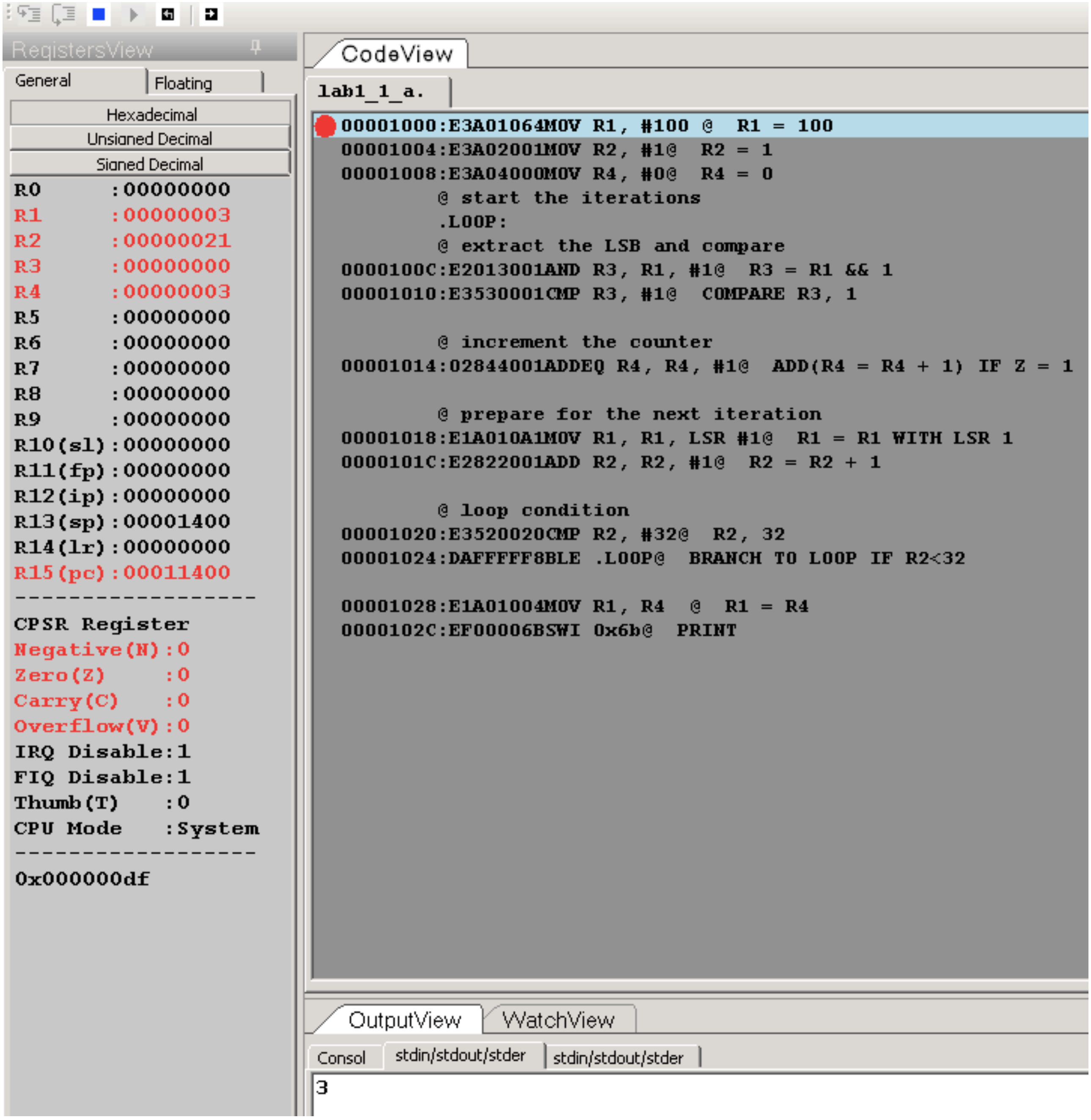
ARMsim code :

```
MOV  R1, #100                    @  R1 = 100
MOV  R2, #1                      @  R2 = 1
MOV  R4, #0                      @  R4 = 0
@ start the iterations
.LOOP:
        @ extract the LSB and compare
                AND  R3, R1, #1          @  R3 = R1 && 1
                CMP  R3, #1              @  COMPARE R3, 1

        @ increment the counter
                ADDEQ      R4, R4, #1    @  ADD(R4 = R4 + 1) IF Z = 1

        @ prepare for the next iteration
                MOV  R1, R1, LSR #1      @  R1 = R1 WITH LSR 1
                ADD  R2, R2, #1          @  R2 = R2 + 1

        @ loop condition
                CMP  R2, #32             @  R2, 32
                BLE  .LOOP               @  BRANCH TO LOOP IF R2<32


MOV  R1, R4                      @  R1 = R4
SWI  0x6b                        @  PRINT
```


```
MOV      R1, #100                          @  R1 = 100
MOV      R2, #1                            @  R2 = 1
MOV      R4, #0                            @  R4 = 0
@ start the iterations
.LOOP:
    @ extract the LSB and compare
        AND      R3, R1, #1                @  R3 = R1 && 1
        CMP      R3, #1                    @  COMPARE R3, 1

    @ increment the counter
        ADDEQ    R4, R4, #1                @  ADD(R4 = R4 + 1) IF Z = 1

    @ prepare for the next iteration
        MOV      R1, R1, LSR #1            @  R1 = R1 WITH LSR 1
        ADD      R2, R2, #1                @  R2 = R2 + 1

    @ loop condition
        CMP      R2, #32                   @  R2, 32
        BLE      .LOOP                     @  BRANCH TO LOOP IF R2<32

MOV      R1, R4                            @  R1 = R4
SWI      0x6b                              @  PRINT
```

Question 2.2.2 - ARMsim

PRITISH WADHWA

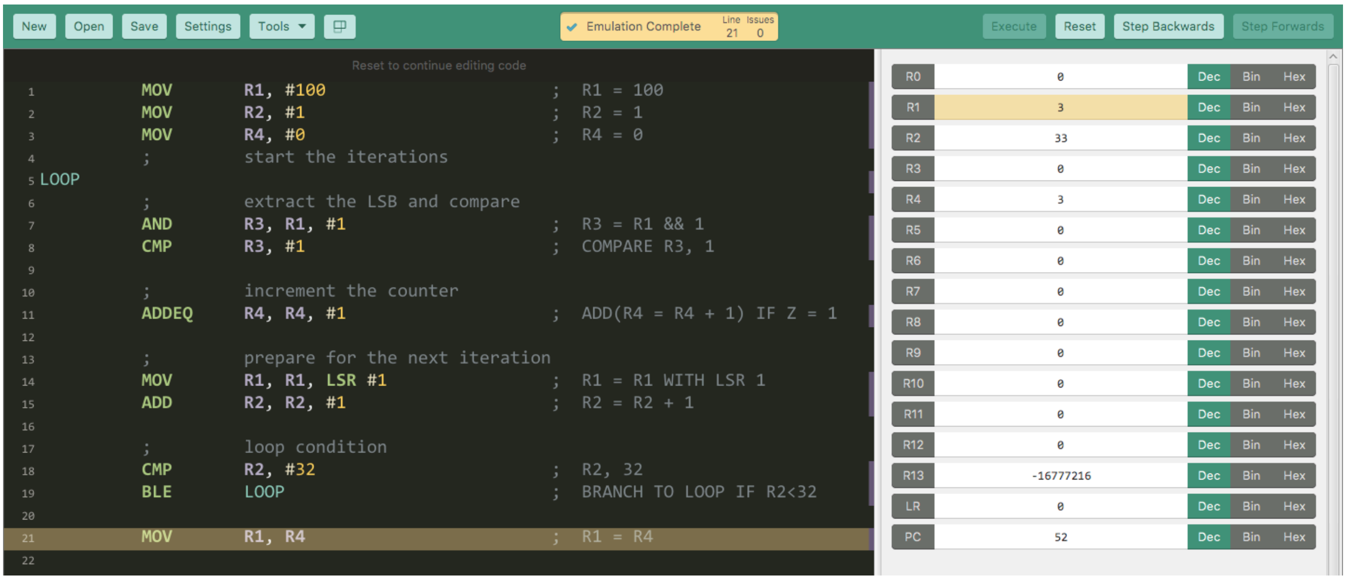


Question 2.2.2 - visUAL

PRITISH WADHWA

visUAL code :

```
            MOV      R1, #100         ;  R1 = 100
            MOV      R2, #1           ;  R2 = 1
            MOV      R4, #0           ;  R4 = 0
            ;        start the iterations
LOOP
            ;        extract the LSB and compare
            AND      R3, R1, #1       ;  R3 = R1 && 1
            CMP      R3, #1           ;  COMPARE R3, 1

            ;        increment the counter
            ADDEQ    R4, R4, #1       ;  ADD(R4 = R4 + 1) IF Z = 1

            ;        prepare for the next iteration
            MOV      R1, R1, LSR #1   ;  R1 = R1 WITH LSR 1
            ADD      R2, R2, #1       ;  R2 = R2 + 1

            ;        loop condition
            CMP      R2, #32          ;  R2, 32
            BLE      LOOP             ;  BRANCH TO LOOP IF R2<32

            MOV      R1, R4           ;  R1 = R4
```



Question 2.2.3 – Code Explanation

PRITISH WADHWA

- The code is well explained using comments, still I will give a brief description about the same.

- I would be using the code written in Sublime Text(Image attached previously) to explain my code.

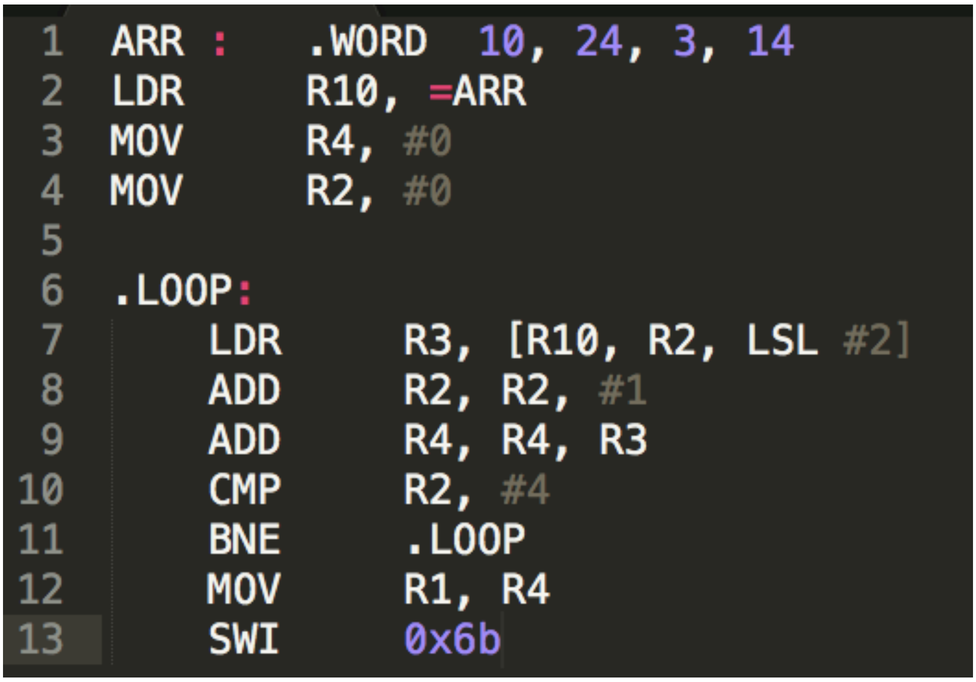
- The code starts at line 1 when '100' is transferred to R1.
- In line 2, I am transferring '1' to R2. R2 would behave as the index for my program.
- In line 3, I am transferring '0' to R4. R4 would function as my counter.
- The main loop starts at line 5.
- In line 7, I Bitwise AND my number(100) stored in R1 with '1'. This generates the LSB of my number(100) and stores it in R3.
- Line 8 compares the value stored in R3 with '1' and updates the flag 'Z' accordingly.
- The value of the flag 'Z' is '1', that is the value in R3 is equal to '1' else the value of the flag 'Z' is '0'(When the value in R3 is not equal to '1').
- Line 11 checks the value of flag 'Z', if it is equal, '1' is added to R4(counter) and the value is stored back in R4.
- In line 14, using the LSR(Logical Shift Right) command, the LSB of R1 is removed and the updated R1 is stored in R1 itself.
- In line 15, the value of R2(index of the loop) is incremented by '1'.
- Line 18 compares the value stored in R2 with the number '32'.
- If the value in R2 is less than '32', the BLE command in the line 19 transfers back the control to line 5.
- The above series of steps is repeated until R2 becomes equal to '32'.
- Following the above statement, the control of the program moves to line 21 where the value stored in R4(the final answer) is finally transferred to R1.
- Line 22 completes the program by printing the value stored in R1

Question 2.3.1 - ARMsim

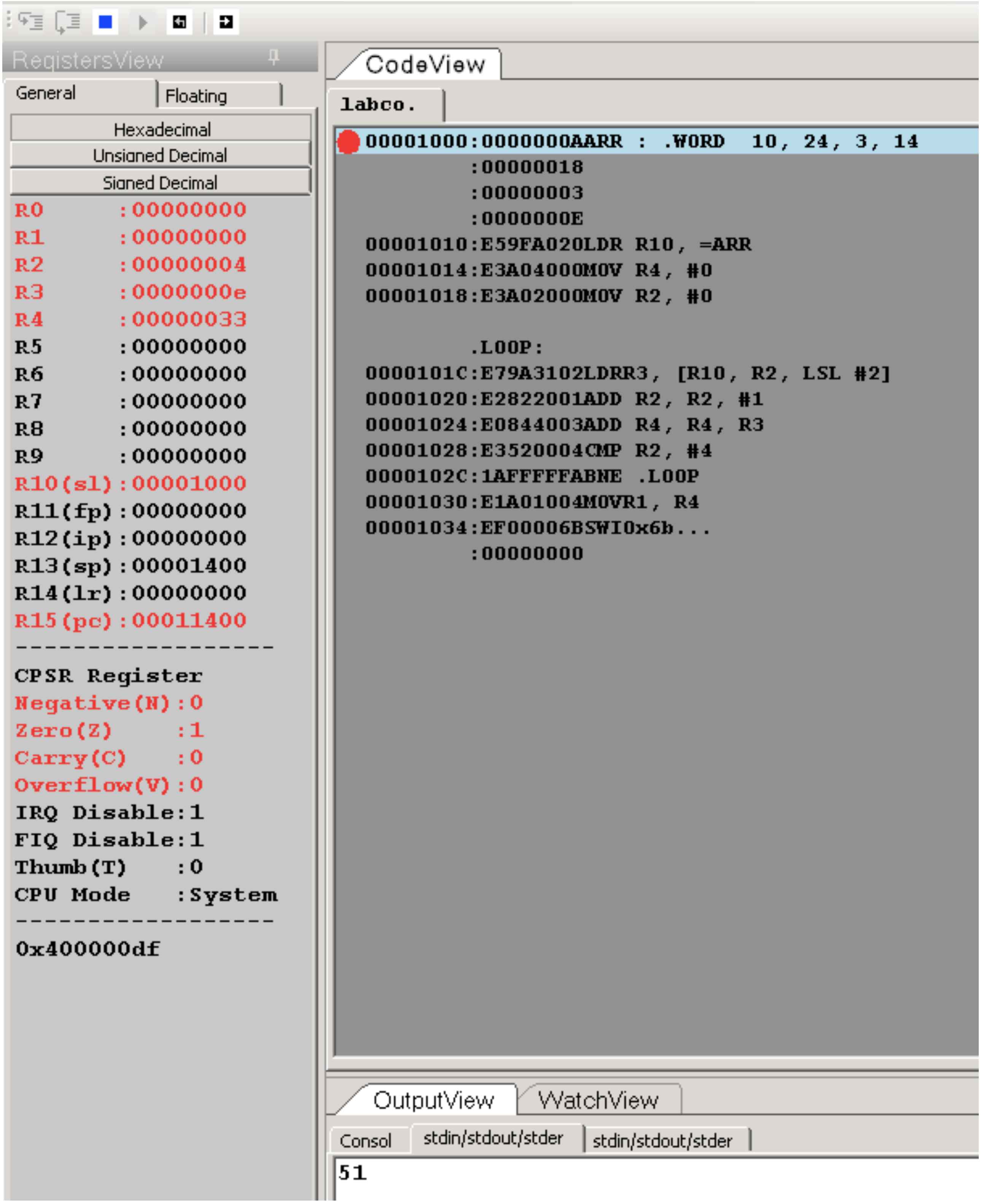
PRITISH WADHWA

ARMsim :

```
ARR :       .WORD  10, 24, 3, 14
LDR         R10, =ARR
MOV         R4, #0
MOV         R2, #0

.LOOP:
        LDR         R3, [R10, R2, LSL #2]
        ADD         R2, R2, #1
        ADD         R4, R4, R3
        CMP         R2, #4
        BNE         .LOOP
        MOV         R1, R4
        SWI         0x6b
```



```
ARR :    .WORD  10, 24, 3, 14
LDR      R10, =ARR
MOV      R4, #0
MOV      R2, #0

.LOOP:
    LDR      R3, [R10, R2, LSL #2]
    ADD      R2, R2, #1
    ADD      R4, R4, R3
    CMP      R2, #4
    BNE      .LOOP
    MOV      R1, R4
    SWI      0x6b
```

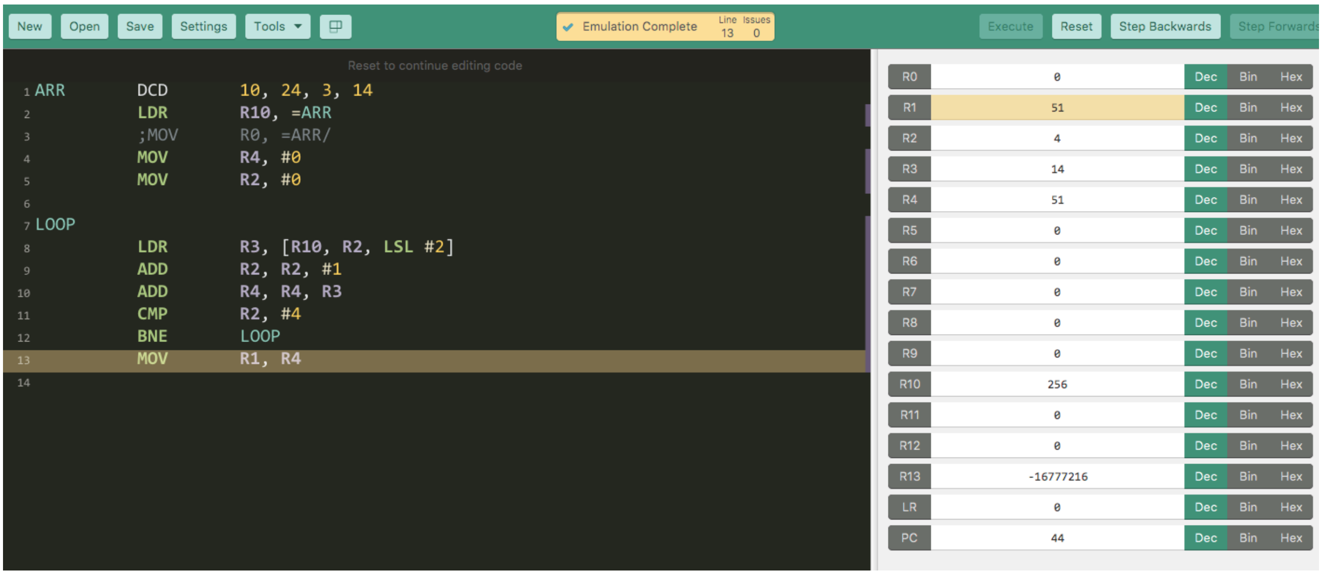

RegistersView
General
Floating
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00000000
R1 :00000000
R2 :00000004
R3 :0000000e
R4 :00000033
R5 :00000000
R6 :00000000
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00001000
R11(fp):00000000
R12(ip):00000000
R13(sp):00001400
R14(lr):00000000
R15(pc):00011400
CPSR Register
Negative(N):0
Zero(Z) :1
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x400000df
CodeView
labco.
00001000:0000000AARR : .WORD 10, 24, 3, 14
:00000018
:00000003
:0000000E
00001010:E59FA020LDR R10, =ARR
00001014:E3A04000MOV R4, #0
00001018:E3A02000MOV R2, #0
.LOOP:
0000101C:E79A3102LDRR3, [R10, R2, LSL #2]
00001020:E2822001ADD R2, R2, #1
00001024:E0844003ADD R4, R4, R3
00001028:E3520004CMP R2, #4
0000102C:1AFFFFFABNE .LOOP
00001030:E1A01004MOVR1, R4
00001034:EF00006BSWI0x6b...
:00000000
OutputView
WatchView
Consol
stdin/stdout/stder
stdin/stdout/stder
51

Question 2.3.1 - visUAL

PRITISH WADHWA

visUAL code :

```
ARR     DCD     10, 24, 3, 14
        LDR     R10, =ARR
        MOV     R4, #0
        MOV     R2, #0

LOOP
        LDR     R3, [R10, R2, LSL #2]
        ADD     R2, R2, #1
        ADD     R4, R4, R3
        CMP     R2, #4
        BNE     LOOP
        MOV     R1, R4
```

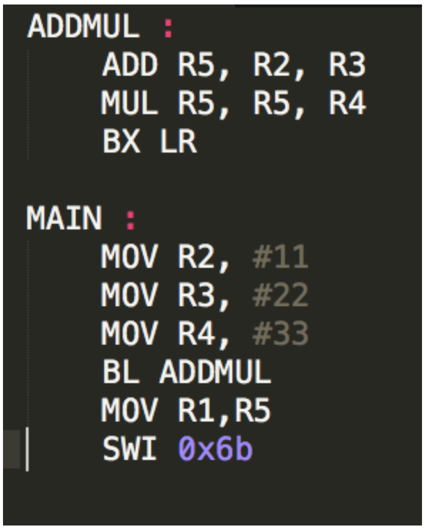


Question 2.3.2 - ARMsim

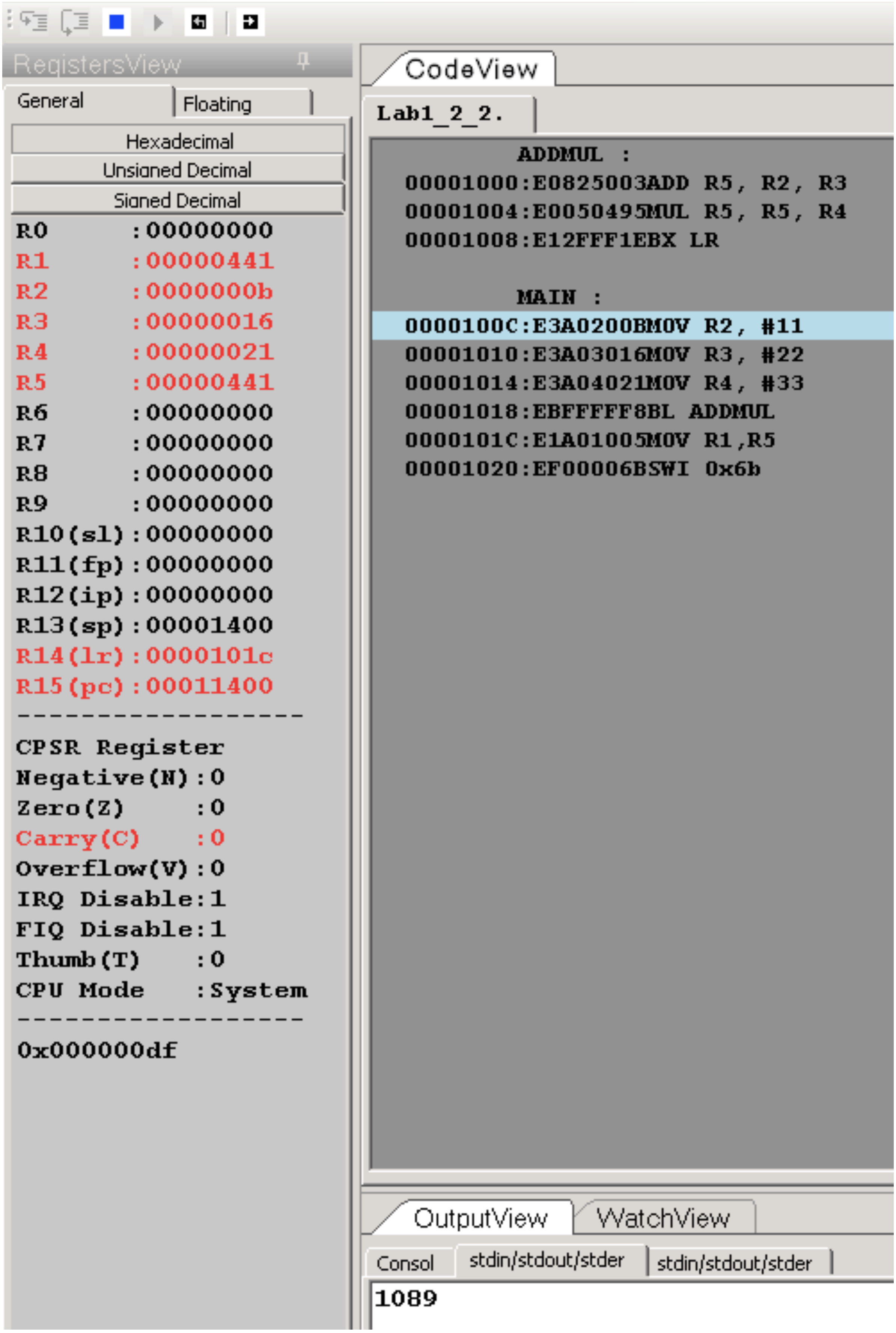
PRITISH WADHWA

ARMsim code:

```
ADDMUL :
        ADD R5, R2, R3
        MUL R5, R5, R4
        BX LR

MAIN :
        MOV R2, #11
        MOV R3, #22
        MOV R4, #33
        BL ADDMUL
        MOV R1,R5
        SWI 0x6b
```



```
ADDMUL :
    ADD R5, R2, R3
    MUL R5, R5, R4
    BX LR

MAIN :
    MOV R2, #11
    MOV R3, #22
    MOV R4, #33
    BL ADDMUL
    MOV R1,R5
    SWI 0x6b
```


RegistersView
General
Floating
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00000000
R1 :00000441
R2 :0000000b
R3 :00000016
R4 :00000021
R5 :00000441
R6 :00000000
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00001400
R14(lr):0000101c
R15(pc):00011400
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
CodeView
Lab1_2_2.
ADDMUL :
00001000:E0825003ADD R5, R2, R3
00001004:E0050495MUL R5, R5, R4
00001008:E12FFF1EBX LR
MAIN :
0000100C:E3A0200BMOV R2, #11
00001010:E3A03016MOV R3, #22
00001014:E3A04021MOV R4, #33
00001018:EBFFFFF8BL ADDMUL
0000101C:E1A01005MOV R1,R5
00001020:EF00006BSWI 0x6b
OutputView
WatchView
Consol
stdin/stdout/stder
stdin/stdout/stder
1089

Question 2.3.2 - visUAL

PRITISH WADHWA

visUAL code:

```
        MOV     R1, #11
        MOV     R2, #22
        MOV     R3, #33
        BL      ADDMUL
        END

ADDMUL
        ADD     R4, R1, R2
        MOV     R5, #0
LOOP
        ADD     R5, R5, R4
        SUB     R3, R3, #1
        CMP     R3, #0
        BNE     LOOP
        MOV     R10, R5
        END
```