Name : DISHA DAS Roll No.: CS18M509

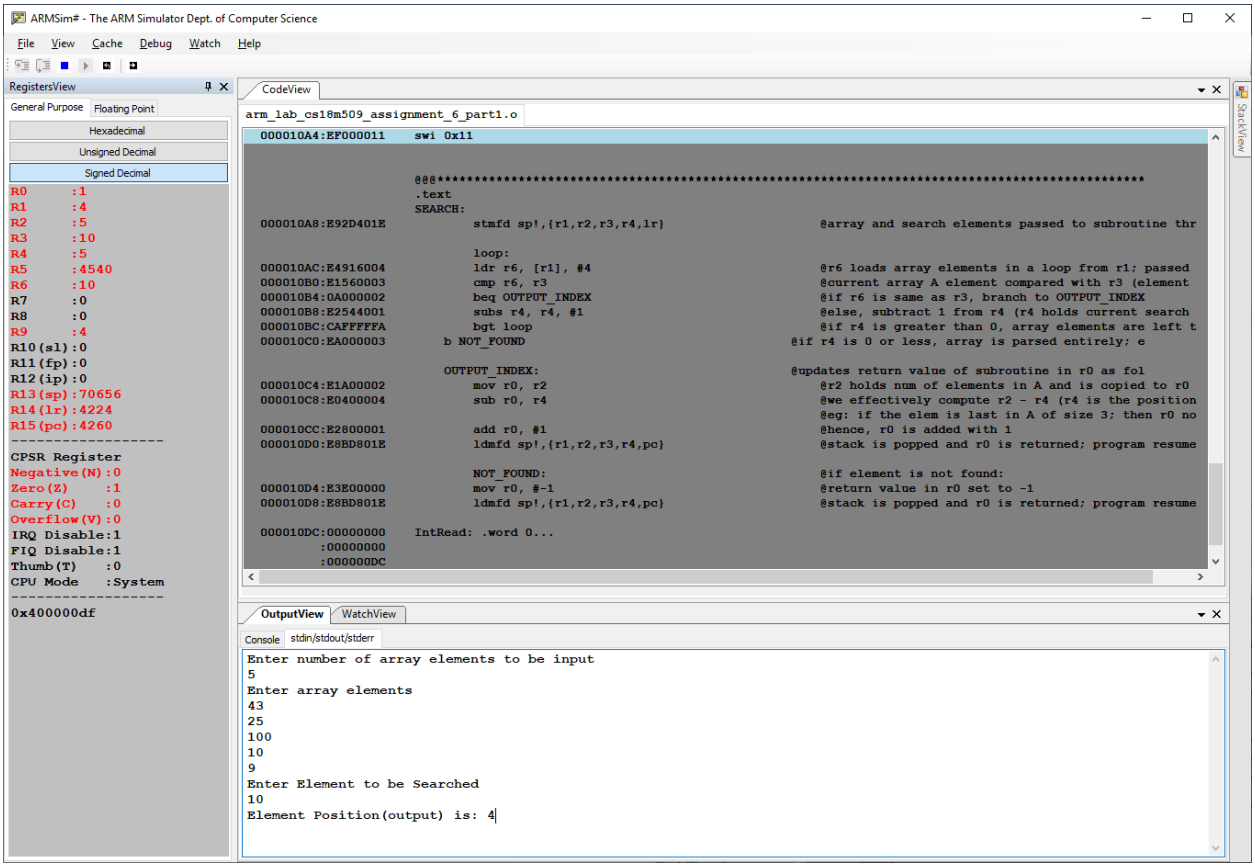
CS6620a : Advanced Computer Organization and Architecture with Lab

Assignment 6

December 3, 2019

This document contains screenshots of the ARMSim outputs to provide basic explanations on how the code was run. Any test vector apart from the one specified in problem statement can be used:

PART 1:

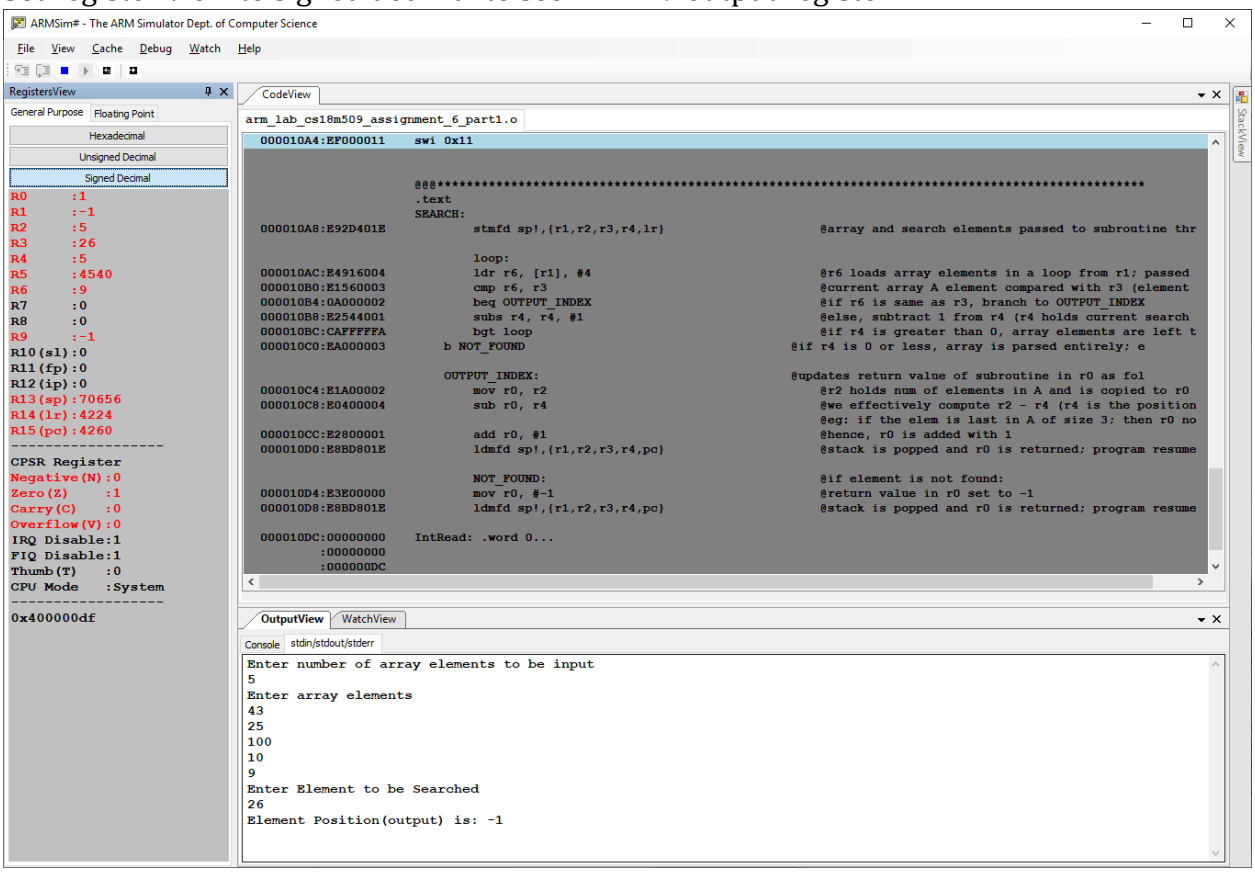


Inputs are given using swi for an integer input.
Enter input using keyboard then, press enter.

The final output is stored in the R9 register and also displayed as a print on console.

Above is a valid elem example.

Following is the testcase to case to check the invalid case (output -1)
Set register view to signed decimal to see -1 in R9 output register



The OUTPUT in R9 is also saved to the =OUTPUT addr

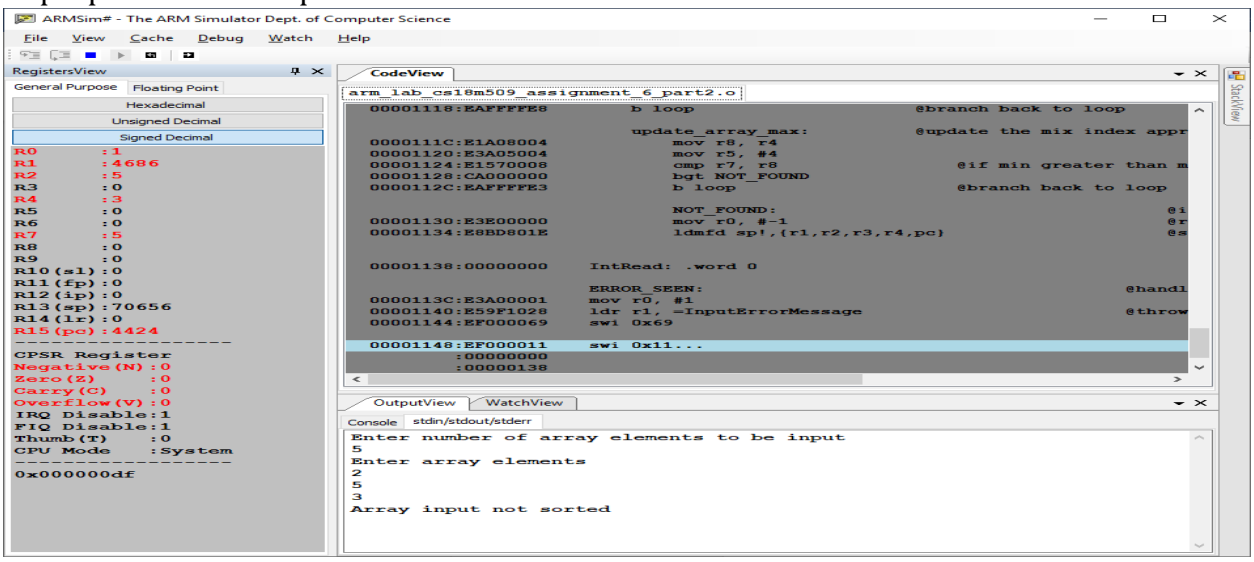
PART 2:

In part 2 for efficiency in searching based on the number of elements compared,
A binary search mechanism is used.

Also, it is assumed that the user will input an already sorted array.
If an unsorted array is being input, display message on console throws error and execution is halted.

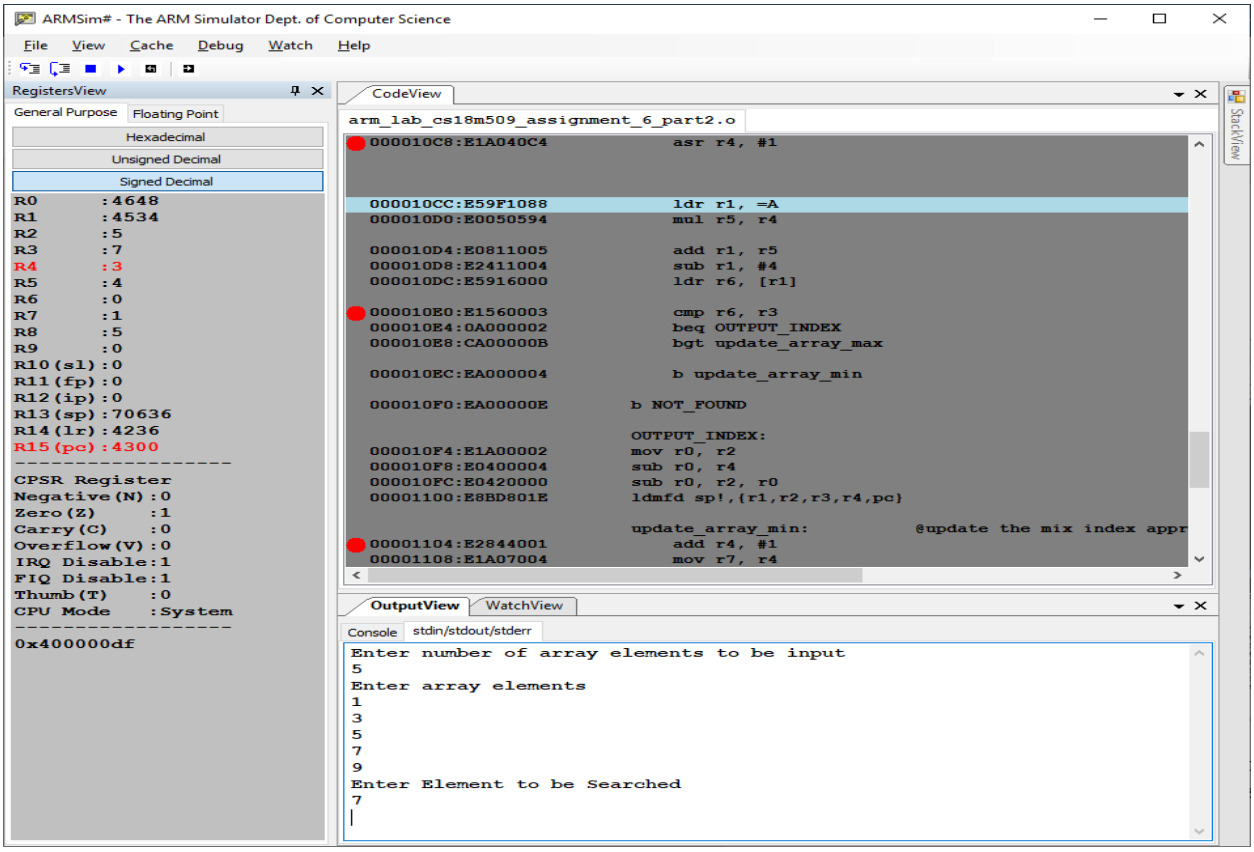
The screenshots are as follows:

Improper unsorted input:

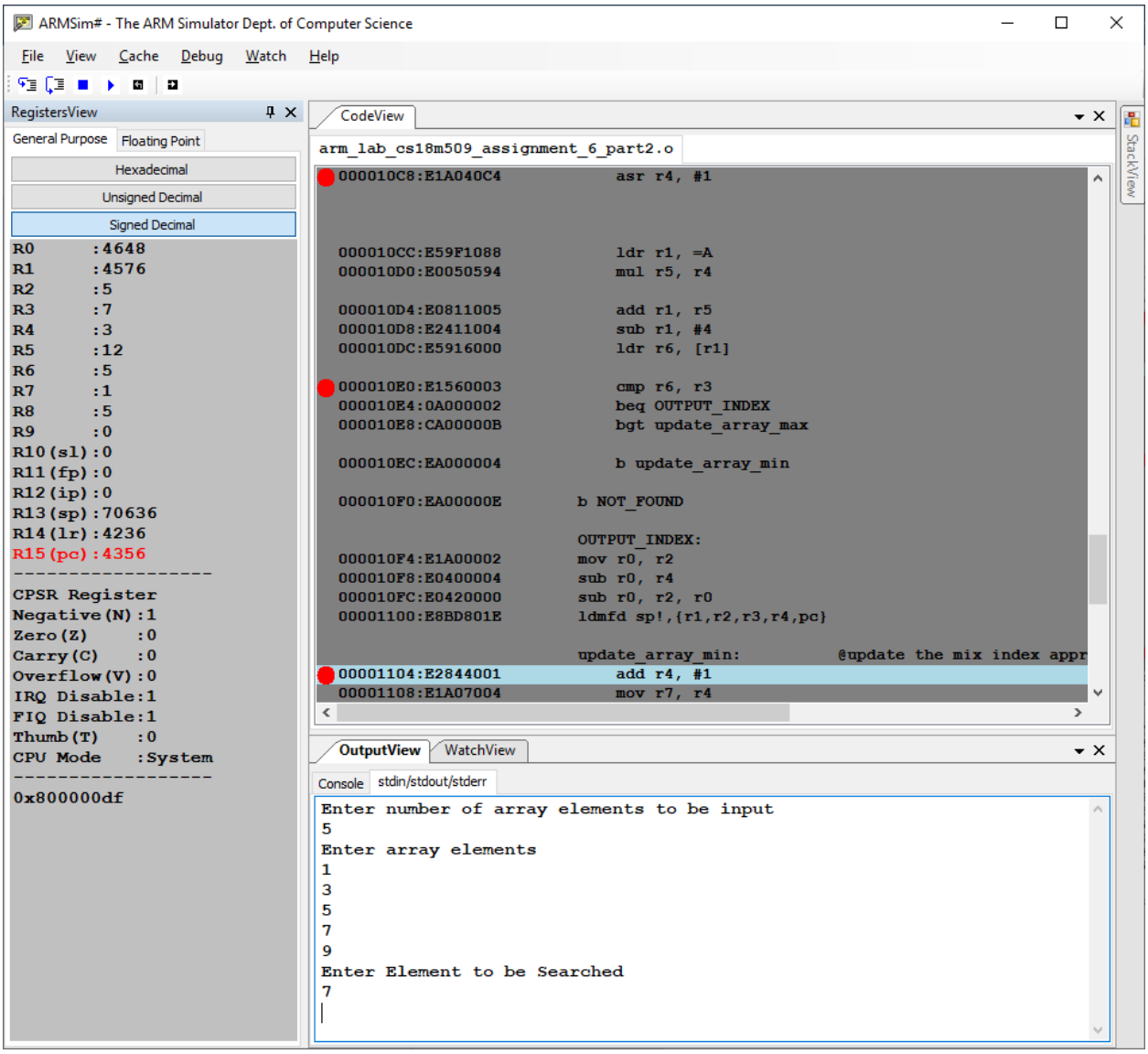


Now, for part two the outputs for part 1 test cases are the same.
The following screenshots for part 2 explain how the min max and middle element are being updated:

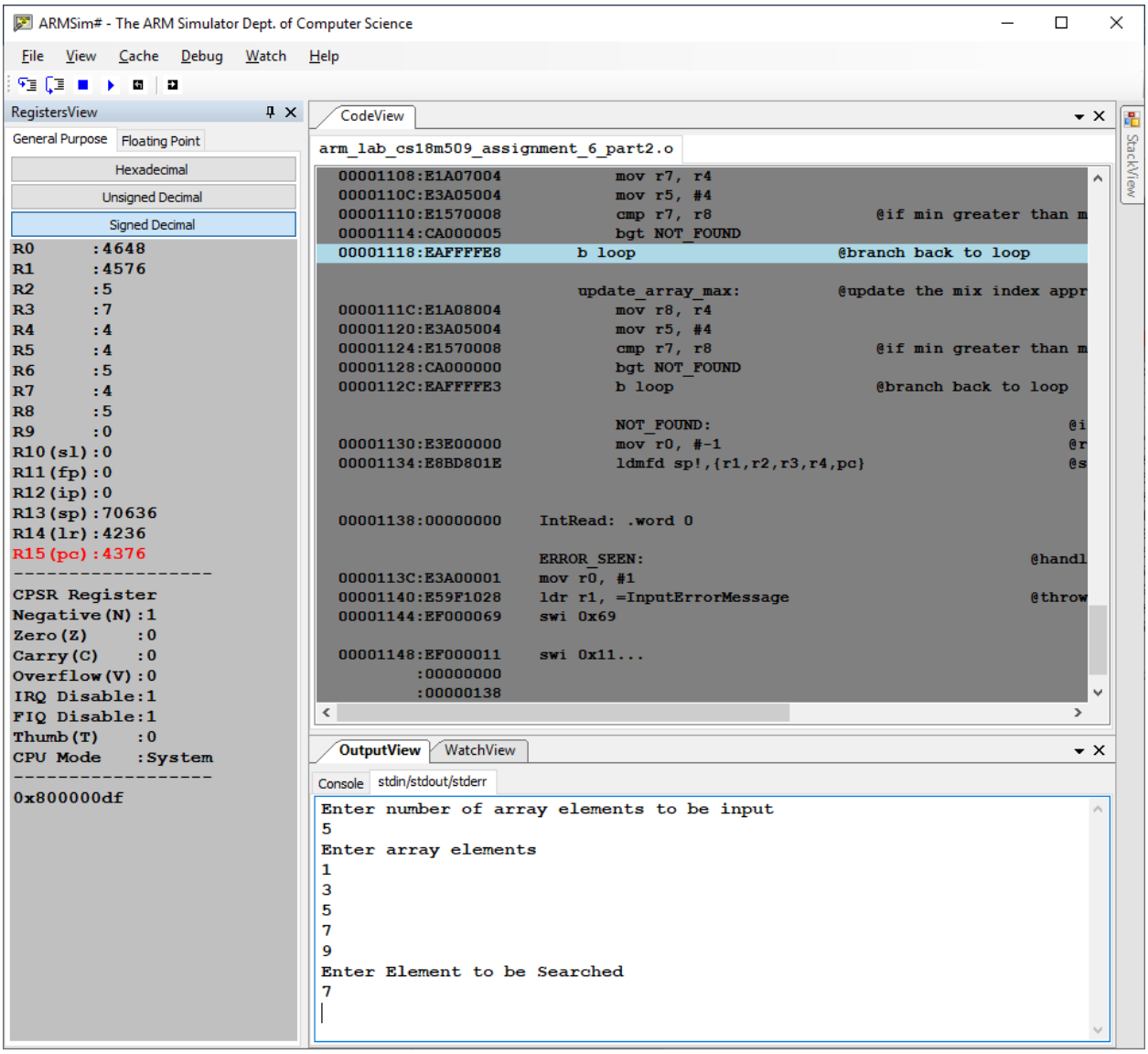
Below we see that r4 (middle index) is computed using (min(r7)+max(r8)) asr 1
Hence, for our input (5 elements) it is first equal to 3 : (1+5)/2 (r4 is 3)



Now r6 holds the middle element and on comparison, with search element 7, we see that r6 is less than seven. Therefore, we need to move up, hence next, update array min is called:



Now, after execution of update array, r7 and r8 are appropriately updated as 4 and 5 respectively. This is shown in the following screenshot:

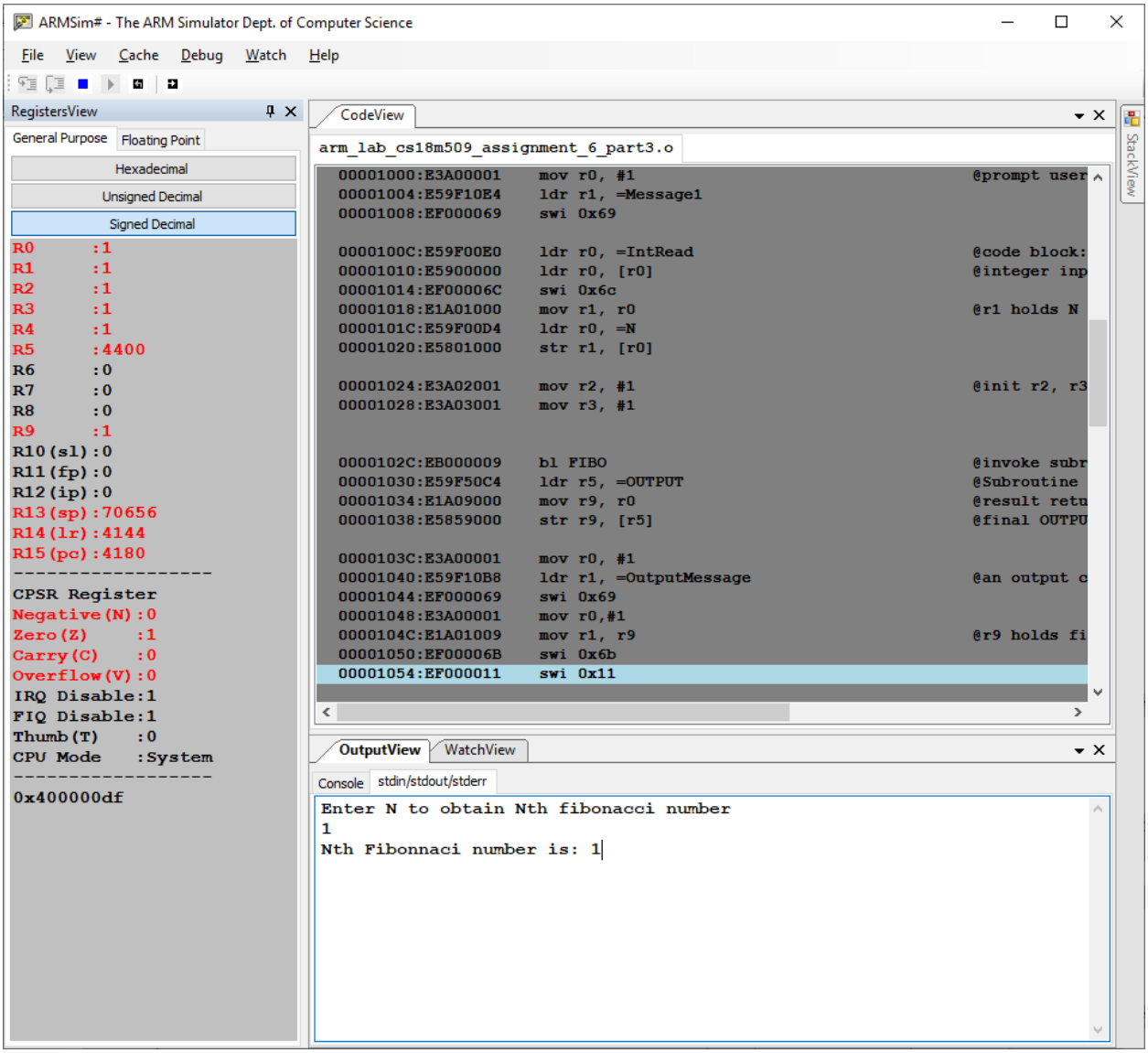


The similar can be seen for updation of min value as well by focusing on register r7 and r8

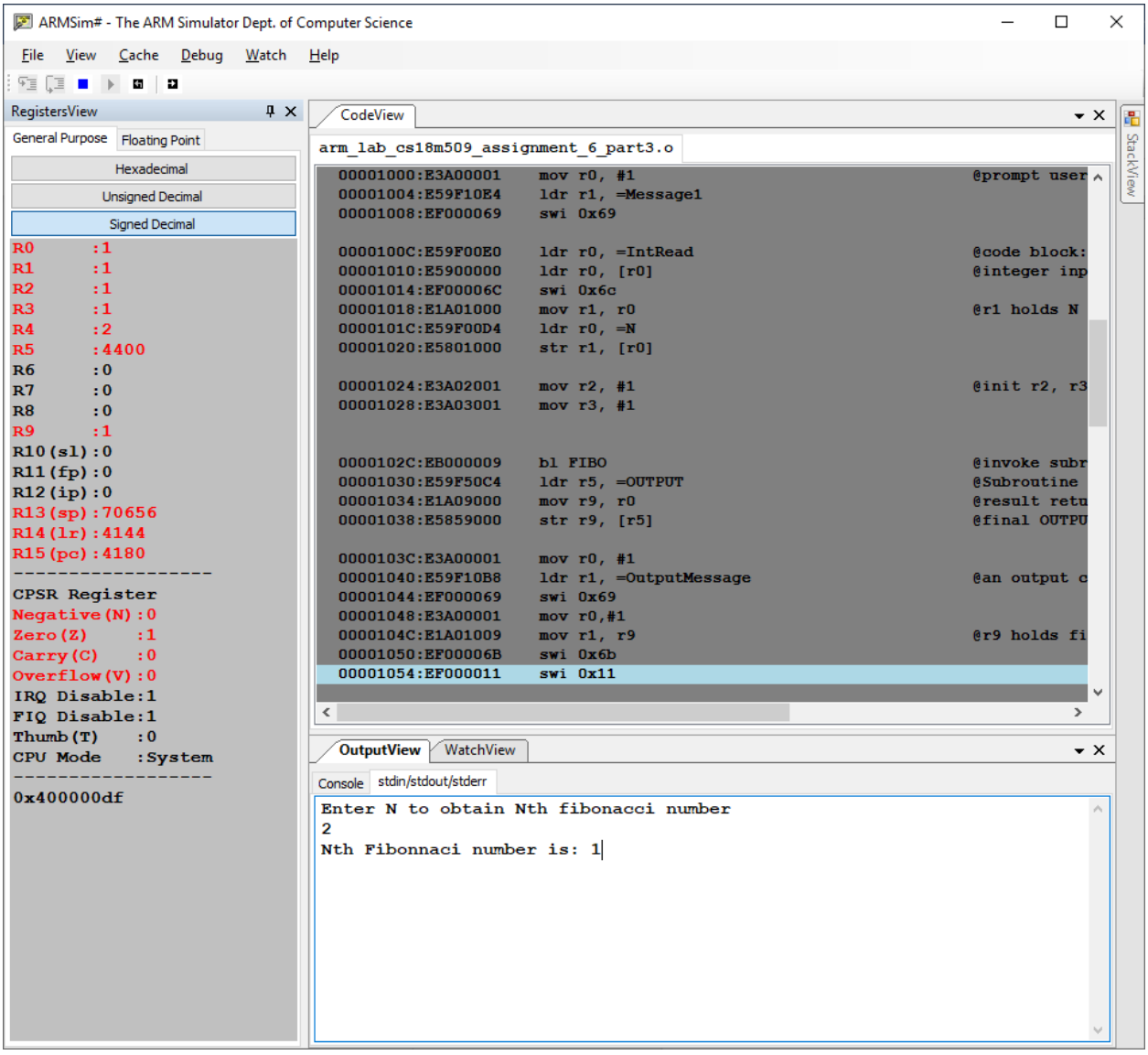
PART 3:

Recursive Fibonacci: (Set to unsigned int reg view and check register R9 for computed output) (Run step wise to see recursive stack updation)

Test1: N is 1:

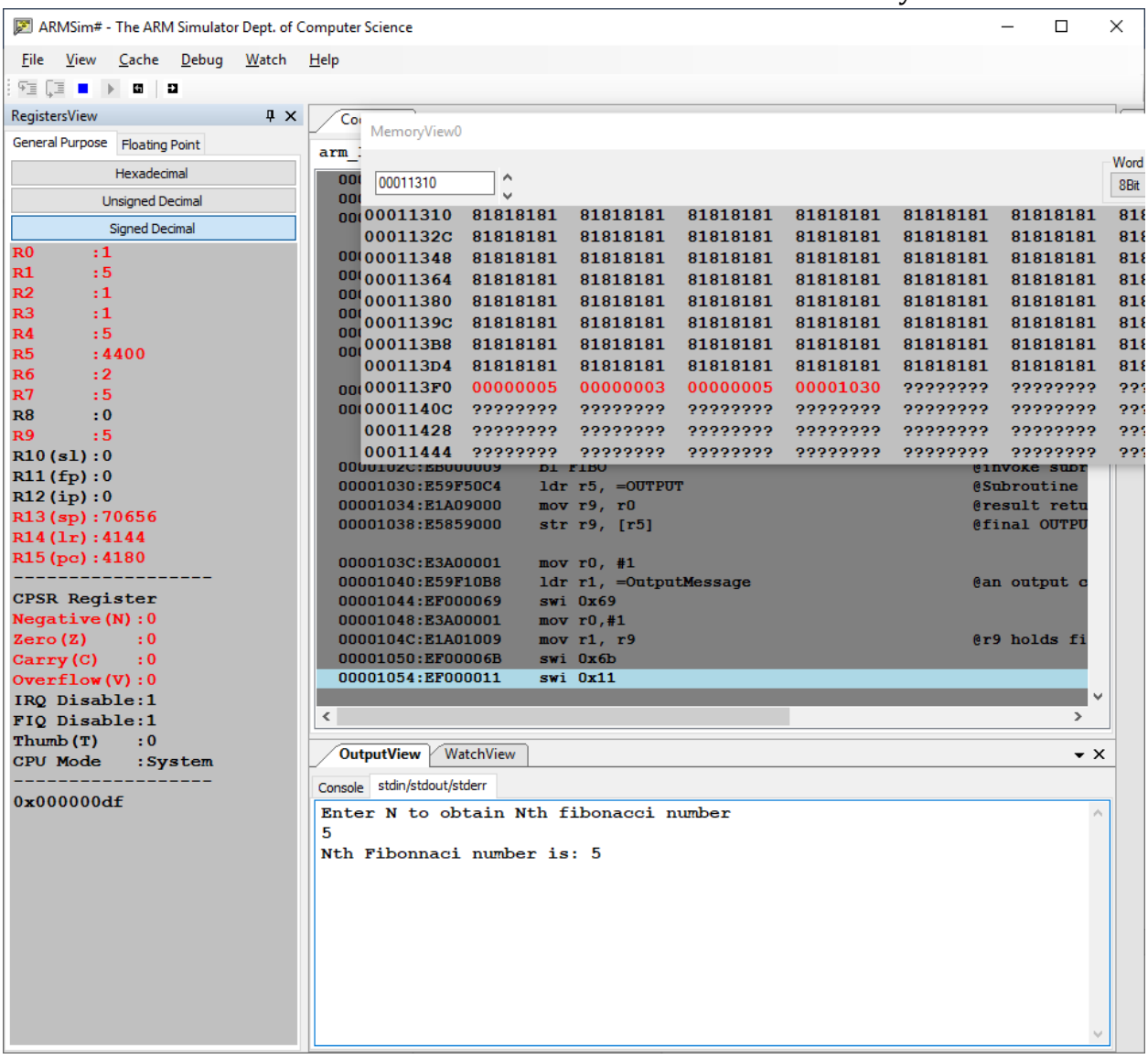


Test2: N is 2:


ARMSim# - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :1
R1 :1
R2 :1
R3 :1
R4 :2
R5 :4400
R6 :0
R7 :0
R8 :0
R9 :1
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):4144
R15(pc):4180
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
arm_lab_cs18m509_assignment_6_part3.o
00001000:E3A00001 mov r0, #1 @prompt user
00001004:E59F10E4 ldr r1, =Message1
00001008:EF000069 swi 0x69
0000100C:E59F00E0 ldr r0, =IntRead @code block:
00001010:E5900000 ldr r0, [r0] @integer inp
00001014:EF00006C swi 0x6c
00001018:E1A01000 mov r1, r0 @r1 holds N
0000101C:E59F00D4 ldr r0, =N
00001020:E5801000 str r1, [r0]
00001024:E3A02001 mov r2, #1 @init r2, r3
00001028:E3A03001 mov r3, #1
0000102C:EB000009 bl FIBO @invoke subr
00001030:E59F50C4 ldr r5, =OUTPUT @Subroutine
00001034:E1A09000 mov r9, r0 @result retu
00001038:E5859000 str r9, [r5] @final OUTPU
0000103C:E3A00001 mov r0, #1
00001040:E59F10B8 ldr r1, =OutputMessage @an output c
00001044:EF000069 swi 0x69
00001048:E3A00001 mov r0,#1
0000104C:E1A01009 mov r1, r9 @r9 holds fi
00001050:EF00006B swi 0x6b
00001054:EF000011 swi 0x11
OutputView WatchView
Console stdin/stdout/stderr
Enter N to obtain Nth fibonacci number
2
Nth Fibonnaci number is: 1
StackView

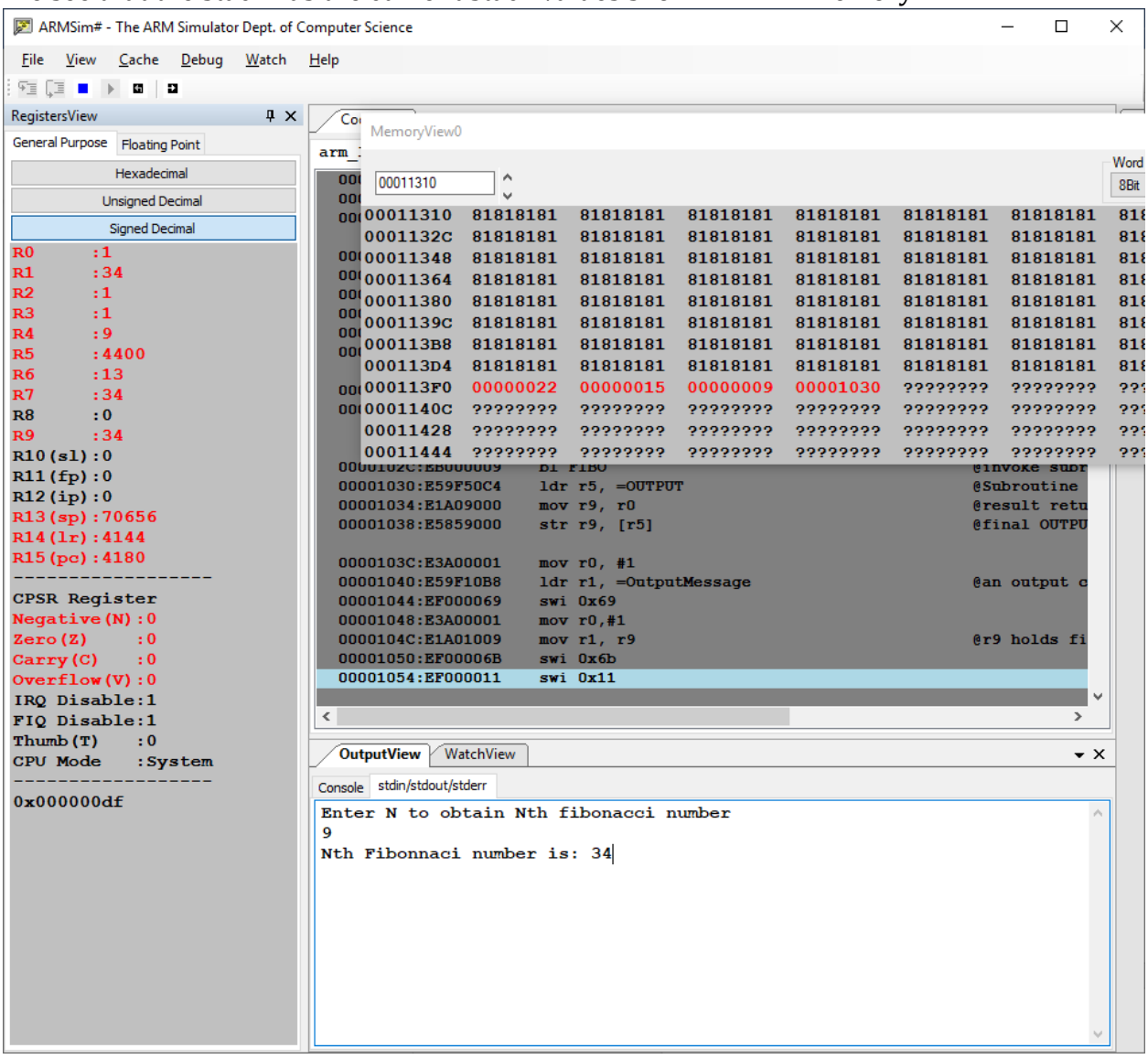
Test3: N is 5, cache mem shown:

We see that the stack has the current stack values shown in HEX memory.



Test4: N is 9, cache mem shown:

We see that the stack has the current stack values shown in HEX memory.



Test5: N is 11, cache mem shown:

We see that the stack has the current stack values shown in HEX memory.