

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:

The screenshot shows the ARMSim simulator interface. The main window displays assembly code for a program named 'arm_lab_cs18m509_assignment_6_part1.o'. The code includes a search routine that takes an array and a search element as input. The registers view on the left shows the current state of the registers, with R9 containing the value 0x400000df. The console window at the bottom shows the program's execution, including prompts for the number of array elements, the array elements themselves, and the search element. The final output is 'Element Position(output) is: 4'.

```

arm_lab_cs18m509_assignment_6_part1.o
000010A4:E0000011 swi 0x11

@@@*****
.text
SEARCH:
000010A8:E92D401E stmfd sp!, {r1,r2,r3,r4,lr}      @array and search elements passed to subroutine thr

loop:
000010AC:E4916004 ldr r6, [r1], #4          @r6 loads array elements in a loop from r1: passed
000010B0:E1560003 cmp r6, r3          @current array A element compared with r3 (element
000010B4:0A000002 beq OUTPUT_INDEX      @if r6 is same as r3, branch to OUTPUT_INDEX
000010B8:E2544001 subs r4, r4, #1      @else, subtract 1 from r4 (r4 holds current search
000010BC:CAFFFFFFA bgt loop          @if r4 is greater than 0, array elements are left t
000010C0:EAD00003 b NOT_FOUND      @if r4 is 0 or less, array is parsed entirely; e

OUTPUT_INDEX:
000010C4:E1A00002 mov r0, r2          @updates return value of subroutine in r0 as fol
000010C8:E0400004 sub r0, r4          @r2 holds num of elements in A and is copied to r0
000010CC:E2800001 add r0, #1          @we effectively compute r2 - r4 (r4 is the position
000010D0:E6BD801E ldmfd sp!, {r1,r2,r3,r4,pc} @eg: if the elem is last in A of size 3; then r0 no
000010D4:E3E00000 NOT_FOUND:          @hence, r0 is added with 1
000010D8:E6BD801E ldmfd sp!, {r1,r2,r3,r4,pc} @stack is popped and r0 is returned; program resume

000010DC:00000000 IntRead: .word 0...
:00000000
:000000DC
  
```

RegistersView:

Register	Value
R0	:1
R1	:4
R2	:5
R3	:10
R4	:5
R5	:4540
R6	:10
R7	:0
R8	:0
R9	:4
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:70656
R14 (lr)	:4224
R15 (pc)	:4260

CPSR Register

Field	Value
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

OutputView:

```

Console  stdin/stdout/stderr
Enter number of array elements to be input
5
Enter array elements
43
25
100
10
9
Enter Element to be Searched
10
Element Position(output) is: 4
  
```

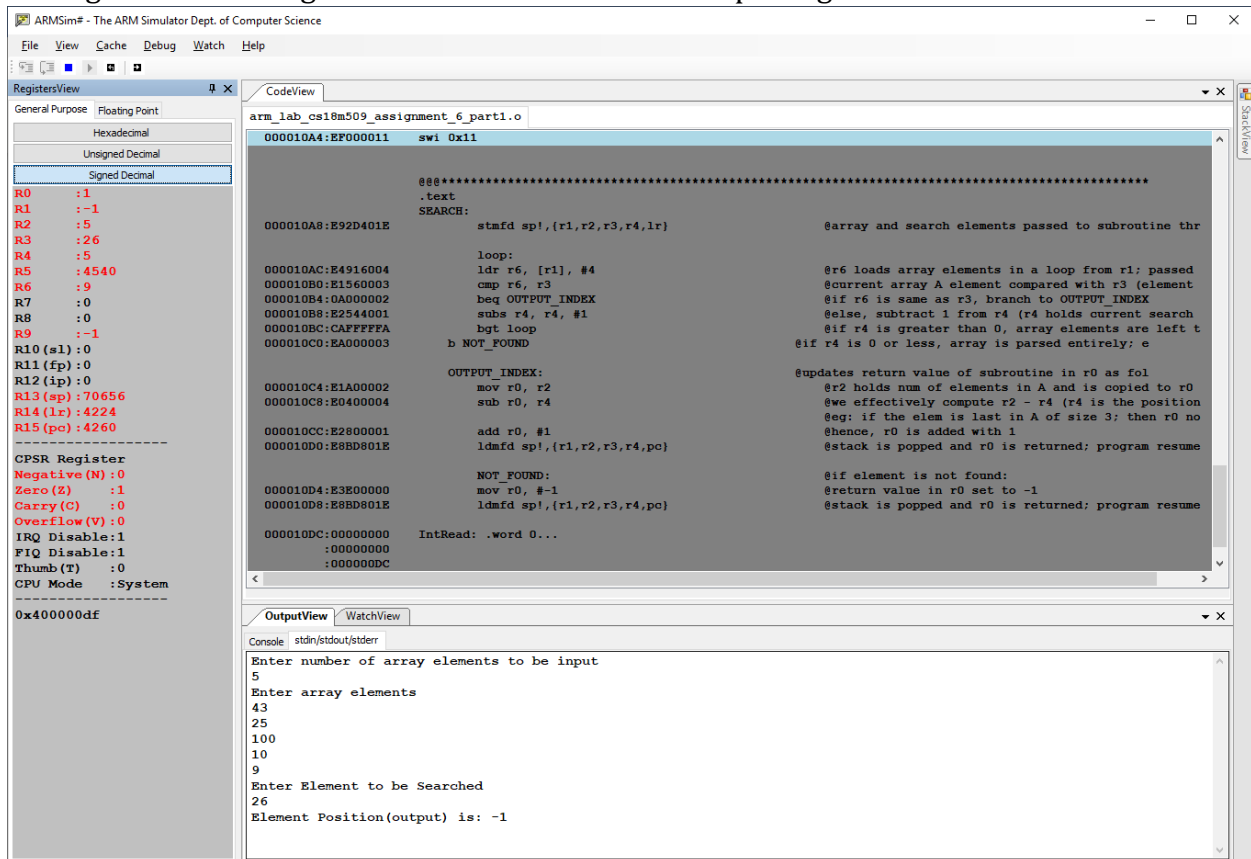
Inputs are given using swi for an integer input.
Enter input using keyboard then, press enter.

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

Above is a valid element 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:

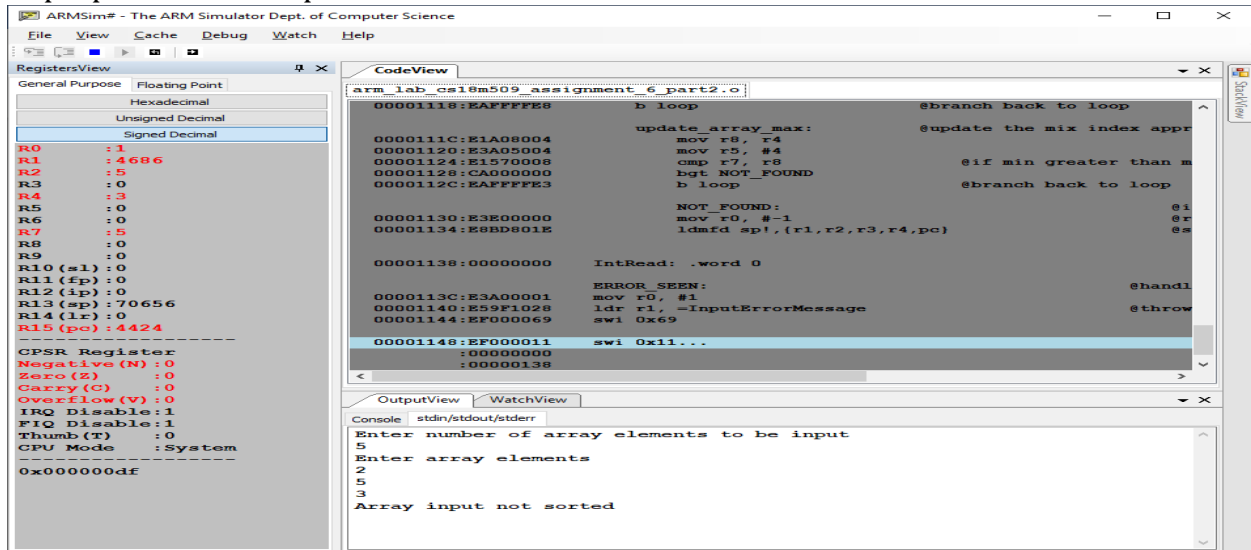
In part 2 for efficiency in searching based on the number of elements compared a binary search mechanism is used.

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:

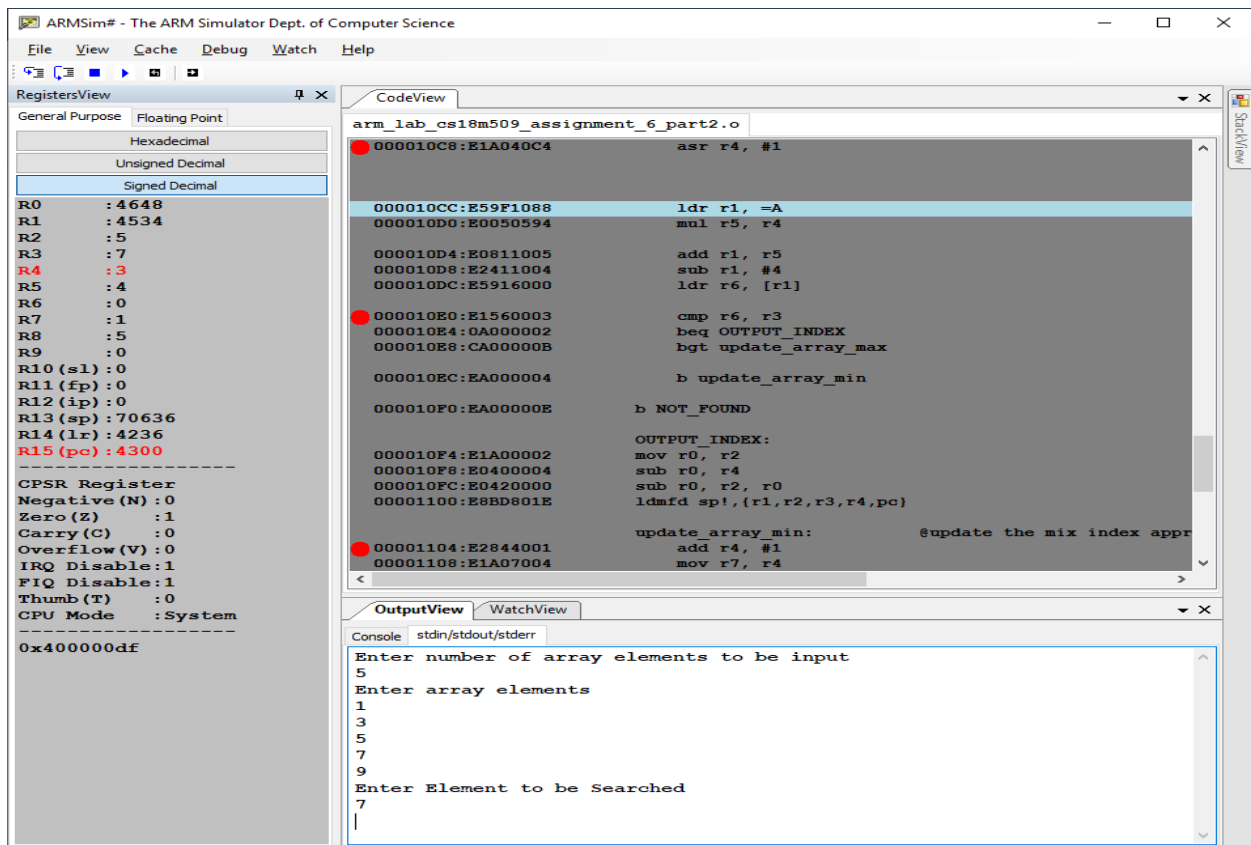
Improper unsorted input:



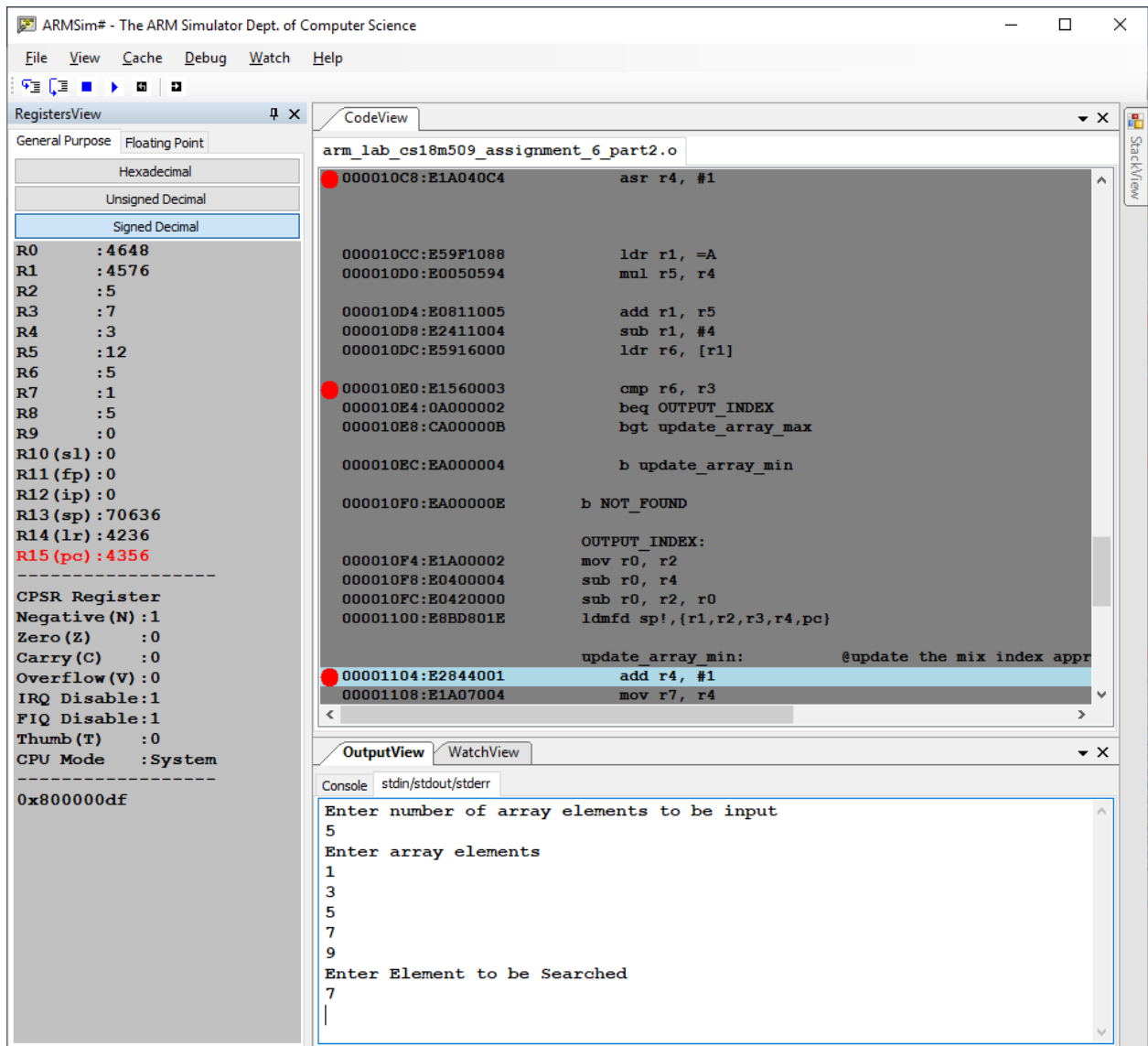
Now, for part two the outputs for part 1 test cases are the same.

The following screenshots for part 2 explain how the min max and middle element are being updated:

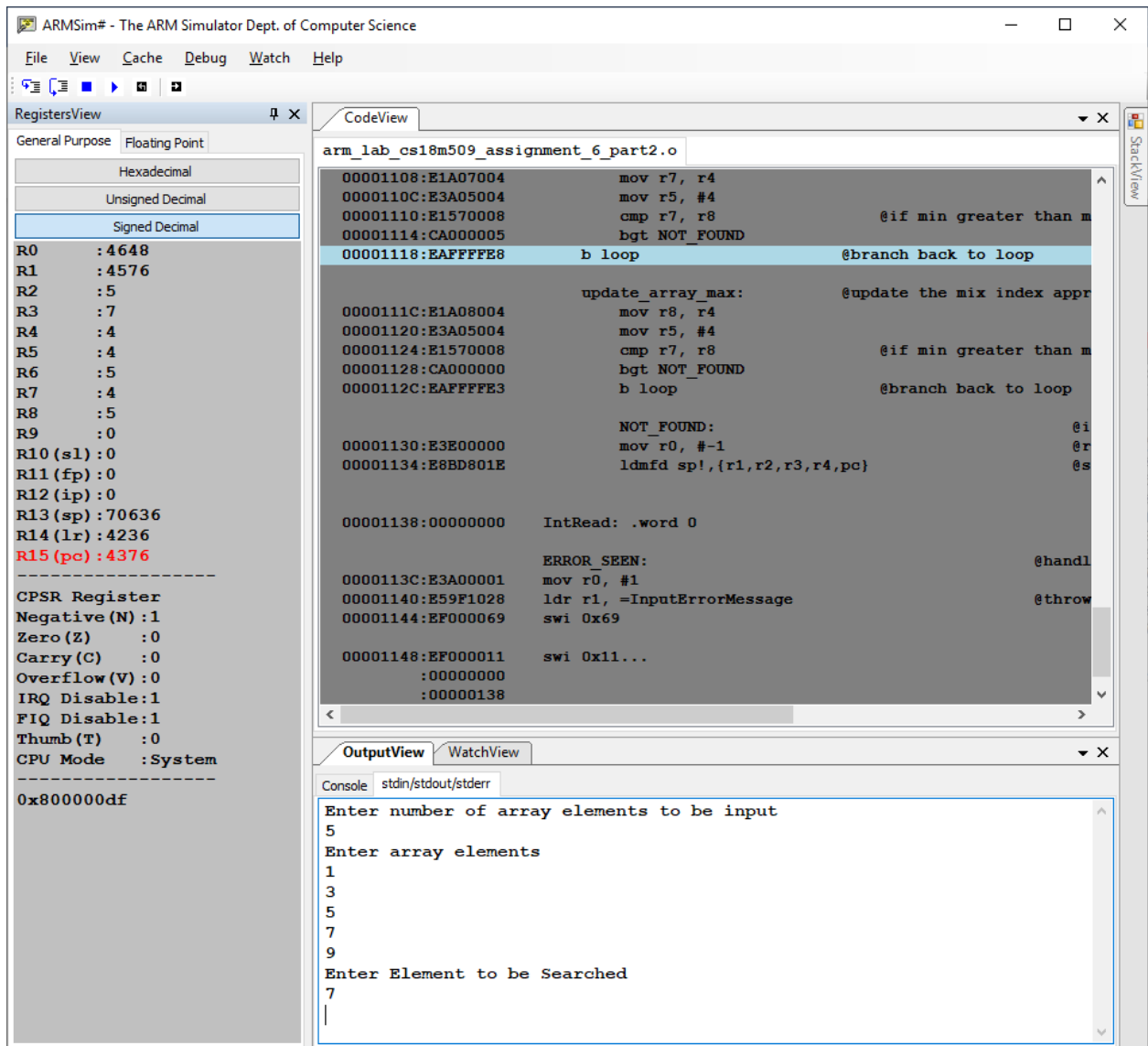
Below we see that r4 (middle index) is computed using $(\min(r7) + \max(r8)) \text{ asr } 1$. Hence, for our input (5 elements) it is first equal to 3 : $(1+5)/2$ (r4 is 3)



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 min, 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 max 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:

The screenshot displays the ARMSim# - The ARM Simulator interface. The main window is divided into three primary sections: RegistersView, CodeView, and OutputView.

RegistersView: This panel shows the state of various registers. The 'General Purpose' tab is selected, and the 'Signed Decimal' view is chosen. The registers are listed as follows:

Register	Value
R0	:1
R1	:1
R2	:1
R3	:1
R4	:1
R5	:4400
R6	:0
R7	:0
R8	:0
R9	:1
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:70656
R14 (lr)	:4144
R15 (pc)	:4180

Below the registers, the CPSR Register is shown with the following status:

- Negative (N): 0
- Zero (Z): 1
- Carry (C): 0
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System

The memory address 0x400000df is also displayed.

CodeView: This panel shows the assembly code for the file 'arm_lab_cs18m509_assignment_6_part3.o'. The code is as follows:

```
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: This panel shows the console output. The text displayed is:

```
Enter N to obtain Nth fibonacci number
1
Nth Fibonnaci number is: 1|
```

Test2: N is 2:

The screenshot displays the ARMSim# ARM Simulator interface. The title bar reads "ARMSim# - The ARM Simulator Dept. of Computer Science". The menu bar includes File, View, Cache, Debug, Watch, and Help. The interface is divided into three main sections: RegistersView, CodeView, and OutputView.

RegistersView: This section shows the state of the ARM registers. The "Signed Decimal" tab is selected. The registers are listed as follows:

Register	Value
R0	:1
R1	:1
R2	:1
R3	:1
R4	:2
R5	:4400
R6	:0
R7	:0
R8	:0
R9	:1
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:70656
R14 (lr)	:4144
R15 (pc)	:4180

Below the registers, the CPSR Register is shown with the following status:

- Negative (N): 0
- Zero (Z): 1
- Carry (C): 0
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System

The value 0x400000df is displayed at the bottom of the RegistersView.

CodeView: This section shows the assembly code for the file "arm_lab_cs18m509_assignment_6_part3.o". The code is as follows:

```
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 OUTPUT

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: This section shows the output of the program. The "Console" tab is selected, and the output is as follows:

```
Enter N to obtain Nth fibonacci number
2
Nth Fibonnaci number is: 1
```


Test3: N is 5, cache mem shown:

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

The screenshot displays the ARMSim# - The ARM Simulator interface. The main window is divided into several panes:

- RegistersView:** Shows the state of ARM registers. The General Purpose registers (R0-R15) are listed with their values in Signed Decimal format. R0 is 1, R1 is 5, R2 is 1, R3 is 1, R4 is 5, R5 is 4400, R6 is 2, R7 is 5, R8 is 0, R9 is 5, R10 (s1) is 0, R11 (fp) is 0, R12 (ip) is 0, R13 (sp) is 70656, R14 (lr) is 4144, and R15 (pc) is 4180. The CPSR Register shows Negative (N) as 0, Zero (Z) as 0, Carry (C) as 0, Overflow (V) as 0, IRQ Disable as 1, FIQ Disable as 1, Thumb (T) as 0, and CPU Mode as System. The output of the simulation is 0x000000df.
- MemoryView0:** Shows a memory dump starting at address 00011310. The dump is organized into columns of 8 bytes each. The first column shows addresses from 00011310 to 00011444. The second column shows the memory contents in hexadecimal. The third column shows the memory contents in decimal. The fourth column shows the memory contents in signed decimal. The fifth column shows the memory contents in unsigned decimal. The sixth column shows the memory contents in floating point. The seventh column shows the memory contents in string format. The eighth column shows the memory contents in ASCII format. The memory dump shows a sequence of instructions and data, including a call to a subroutine and a return from the subroutine.
- OutputView:** Shows the output of the simulation. The output is displayed in a text area with a scroll bar. The output text is: "Enter N to obtain Nth fibonacci number", "5", and "Nth Fibonnaci number is: 5".

Test4: N is 9, cache mem shown:

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

The screenshot displays the ARMSim# - The ARM Simulator interface. The main window is divided into several panes:

- RegistersView:** Shows the state of ARM registers. R0 is 1, R1 is 34, R2 is 1, R3 is 1, R4 is 9, R5 is 4400, R6 is 13, R7 is 34, R8 is 0, R9 is 34, R10 (s1) is 0, R11 (fp) is 0, R12 (ip) is 0, R13 (sp) is 70656, R14 (lr) is 4144, and R15 (pc) is 4180. The CPSR Register shows Negative (N) as 0, Zero (Z) as 0, Carry (C) as 0, and Overflow (V) as 0. IRQ and FIQ are disabled, Thumb is 0, and CPU Mode is System.
- MemoryView0:** Displays memory contents in hexadecimal. The address 00011310 is selected. The memory shows a sequence of 81818181 values, followed by 00000022, 00000015, 00000009, 00001030, and then several 00000000 values.
- OutputView:** Shows the console output. The text "Enter N to obtain Nth fibonacci number" is displayed, followed by the input "9" and the output "Nth Fibonnaci number is: 34".

Test5: N is 11, cache mem shown:

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

The screenshot displays the ARMSim# interface with the following components:

- RegistersView:** Shows general-purpose registers R0 through R15. R0 is 1, R1 is 89, R2 is 1, R3 is 1, R4 is 11, R5 is 4400, R6 is 34, R7 is 89, R8 is 0, R9 is 89, R10 (s1) is 0, R11 (fp) is 0, R12 (ip) is 0, R13 (sp) is 70656, R14 (lr) is 4144, and R15 (pc) is 4180. The CPSR register shows Negative (N) as 0, Zero (Z) as 0, Carry (C) as 0, Overflow (V) as 0, IRQ Disable as 1, FIQ Disable as 1, Thumb (T) as 0, and CPU Mode as System. The memory address 0x000000df is also shown.
- MemoryView0:** Displays a memory dump starting at address 00011310. The dump shows a sequence of 81818181 values, followed by 00000059, 00000037, 0000000B, and 00001030, and then several rows of 00000000 values. The dump is organized into columns of 8 bytes each.
- OutputView:** Shows the console output with the text "Enter N to obtain Nth fibonacci number", "11", and "Nth Fibonnaci number is: 89".