

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 interface with the following components:

- RegistersView:** Displays the state of various registers. R0 is 1, R1 is 4, R2 is 5, R3 is 10, R4 is 5, R5 is 4540, R6 is 10, R7 is 0, R8 is 0, R9 is 4, R10 (s1) is 0, R11 (fp) is 0, R12 (ip) is 0, R13 (sp) is 70656, R14 (lr) is 4224, and R15 (pc) is 4260. The CPSR register shows Negative (N) as 0, Zero (Z) as 1, Carry (C) as 0, and Overflow (V) as 0. IRQ and FIQ are disabled, Thumb is 0, and CPU Mode is System.
- CodeView:** Displays assembly code for 'arm_lab_cs18m509_assignment_6_part1.o'. The code includes a search routine that iterates through an array to find a specific element. Comments explain the logic, such as updating the return value in R0 and handling the 'NOT_FOUND' case.
- OutputView:** Shows the console output of the program. It prompts the user to enter the number of array elements (5), the array elements (43, 25, 100, 10, 9), and the element to be searched (10). The final output is 'Element Position(output) is: 4|'.

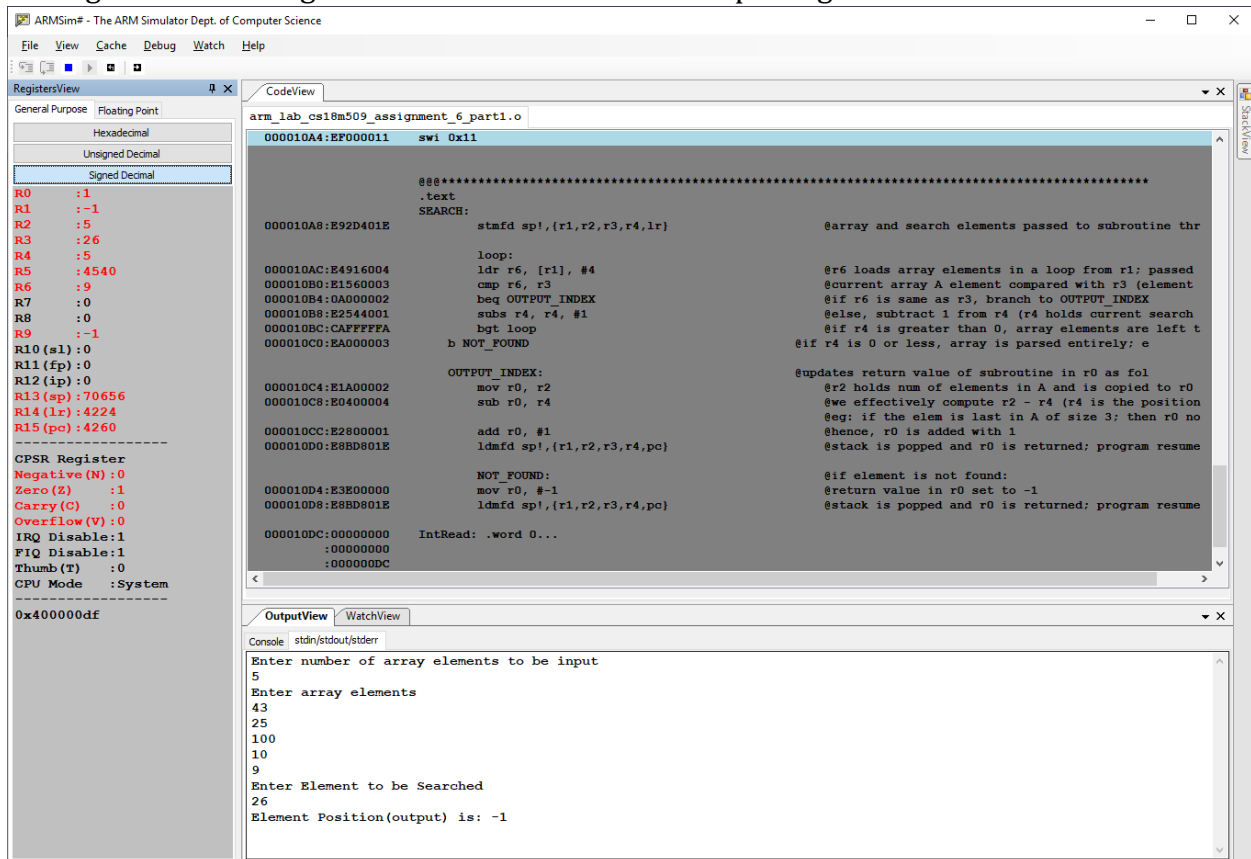
Inputs are given using swi for an integer input.
Enter input using keyboard then, press enter.

The final output is stored in the R9 register and 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:

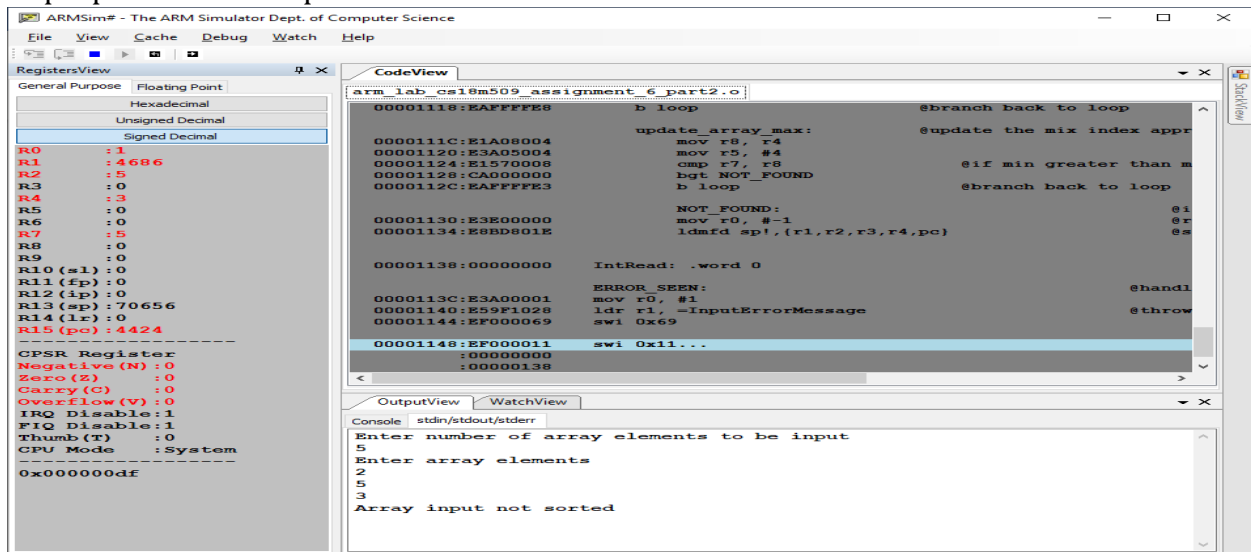
In part 2 for efficiency in searching based on the number of elements compared, A binary search mechanism is used.

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:

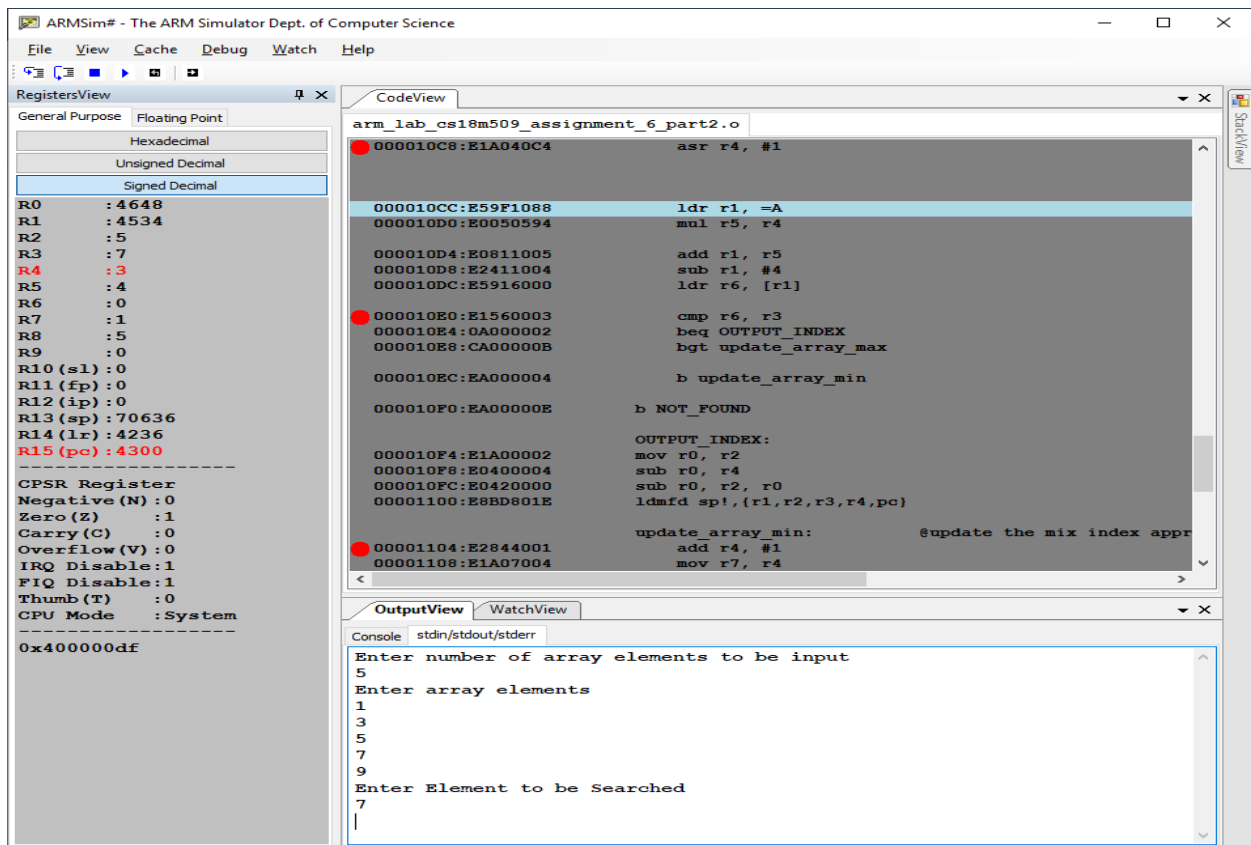
Improper unsorted input:



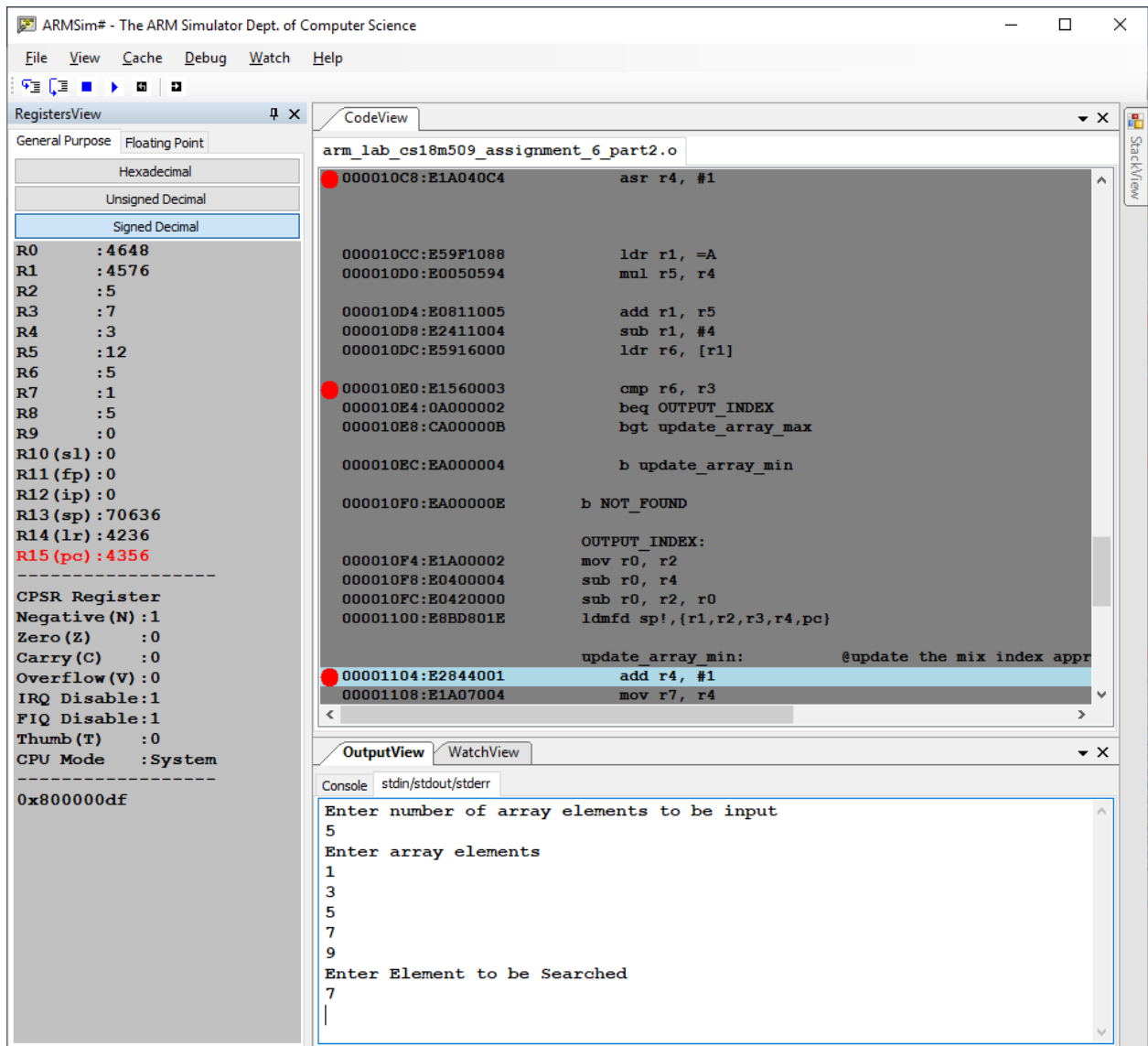
Now, for part two the outputs for part 1 test cases are the same.

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

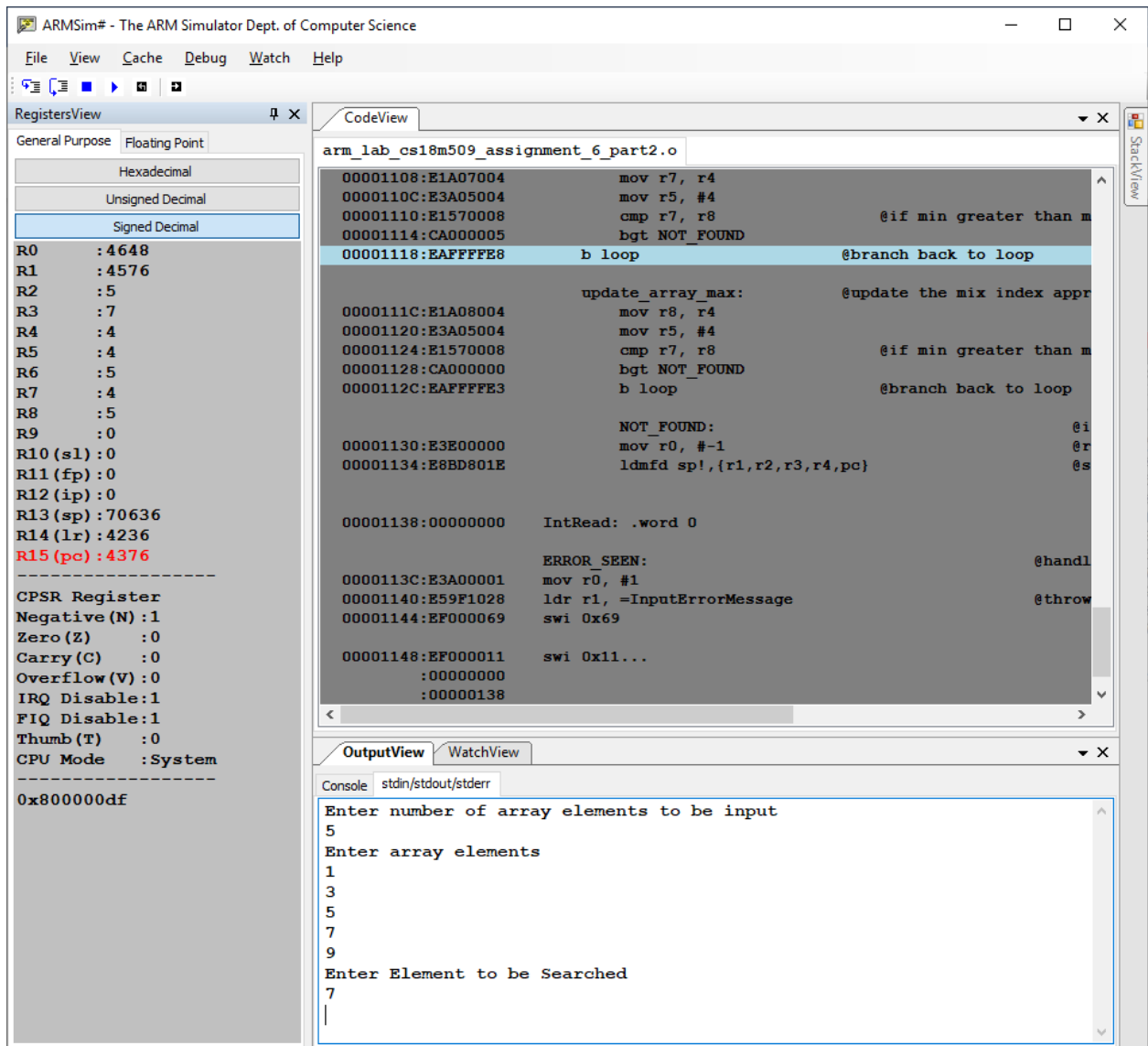
Below we see that r4 (middle index) is computed using $(\min(r7) + \max(r8)) \text{ asr } 1$. Hence, for our input (5 elements) it is first equal to 3 : $(1+5)/2$ (r4 is 3)



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, 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:

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:

Flag	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

The bottom of the RegistersView shows the memory address 0x400000df.

CodeView: This panel displays 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 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 panes:

- RegistersView** (left): Shows the state of 16 registers (R0-R15) and the CPSR register. The "Signed Decimal" format is selected. R0-R9 are shown with values, while R10-R15 and CPSR are zero. The CPSR register shows flags: Negative (N): 0, Zero (Z): 1, Carry (C): 0, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, and CPU Mode: System. The address 0x400000df is displayed at the bottom.
- CodeView** (middle): Displays assembly code for "arm_lab_cs18m509_assignment_6_part3.o". The code includes instructions like "mov r0, #1", "ldr r1, =Message1", "swi 0x69", "ldr r0, =IntRead", "ldr r0, [r0]", "swi 0x6c", "mov r1, r0", "ldr r0, =N", "str r1, [r0]", "mov r2, #1", "mov r3, #1", "bl FIBO", "ldr r5, =OUTPUT", "mov r9, r0", "str r9, [r5]", "mov r0, #1", "ldr r1, =OutputMessage", "swi 0x69", "mov r0, #1", "mov r1, r9", "swi 0x6b", and "swi 0x11". Comments like "@prompt user", "@code block:", "@integer inp", "@r1 holds N", "@init r2, r3", "@invoke subr", "@Subroutine", "@result retu", "@final OUTPU", "@an output c", and "@r9 holds fi" are present.
- OutputView** (bottom): Shows the console output. The text "Enter N to obtain Nth fibonacci number" is followed by the input "2" and the output "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. 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) : 0, Zero (Z) : 0, Carry (C) : 0, Overflow (V) : 0, IRQ Disable : 1, FIQ Disable : 1, Thumb (T) : 0, and CPU Mode : System.
- MemoryView0:** Displays memory contents in hexadecimal. The address 00011310 is selected. The memory contains a sequence of 81818181 values, followed by 00000005, 00000003, 00001030, and 00001038. The memory also contains instructions for the Fibonacci function, such as `ldr r5, =OUTPUT`, `mov r9, r0`, `str r9, [r5]`, `mov r0, #1`, `ldr r1, =OutputMessage`, `swi 0x69`, `mov r0, #1`, `mov r1, r9`, `swi 0x6b`, and `swi 0x11`.
- OutputView:** Shows the output of the program. The console output 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. The General Purpose registers are listed with their values in Signed Decimal format. 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, and 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, 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 shown at the bottom.
- MemoryView0:** Displays a memory dump starting at address 00011310. The dump shows a sequence of memory addresses and their corresponding values in hexadecimal. The values are mostly 81818181, with some variations in the later addresses.
- OutputView:** Shows the console output. The text "Enter N to obtain Nth fibonacci number" is displayed, followed by the input "9". The output "Nth Fibonnaci number is: 34" is shown.

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# - The ARM Simulator interface. The main window is divided into three panes: RegistersView, MemoryView, and OutputView.

RegistersView: Shows the state of 16 registers (R0-R15) and the CPSR register. The registers are listed in a table with their values in hexadecimal, decimal, and signed decimal. The CPSR register is also shown with its flags (Negative, Zero, Carry, Overflow, IRQ Disable, FIQ Disable, Thumb, CPU Mode).

MemoryView: Shows the current stack values in hexadecimal memory. The address 00011310 is selected, and the memory contents are displayed in a table. The stack values are shown in hexadecimal, decimal, and signed decimal. The stack is growing downwards, with the current top of the stack at 00011444.

OutputView: Shows the console output. The text "Enter N to obtain Nth fibonacci number" is displayed, followed by the user input "11". The output shows "Nth Fibonnaci number is: 89".

RegistersView Details:

Register	Hexadecimal	Unsigned Decimal	Signed Decimal
R0	:1		
R1	:89		
R2	:1		
R3	:1		
R4	:11		
R5	:4400		
R6	:34		
R7	:89		
R8	:0		
R9	:89		
R10 (s1)	:0		
R11 (fp)	:0		
R12 (ip)	:0		
R13 (sp)	:70656		
R14 (lr)	:4144		
R15 (pc)	:4180		

CPSR Register:

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

MemoryView Details:

Address	Hexadecimal	Unsigned Decimal	Signed Decimal
00011310	81818181	81818181	81818181
0001132C	81818181	81818181	81818181
00011348	81818181	81818181	81818181
00011364	81818181	81818181	81818181
00011380	81818181	81818181	81818181
0001139C	81818181	81818181	81818181
000113B8	81818181	81818181	81818181
000113D4	81818181	81818181	81818181
000113F0	00000059	00000037	0000000B
0001140C	????????	????????	????????
00011428	????????	????????	????????
00011444	????????	????????	????????

OutputView Details:

```
Enter N to obtain Nth fibonacci number
11
Nth Fibonnaci number is: 89
```