PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

Assignment 1 submission
Level 1- Banana Problem

Name of the Student	**Pranav Hemanth**
SRN	**PES1UG23CS433**
Section	**G**
Department	**CSE**
Campus	**RR**

Jan -May 2025 ASSIGNMENT SUBMISSION_**UE23CS251B**

Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

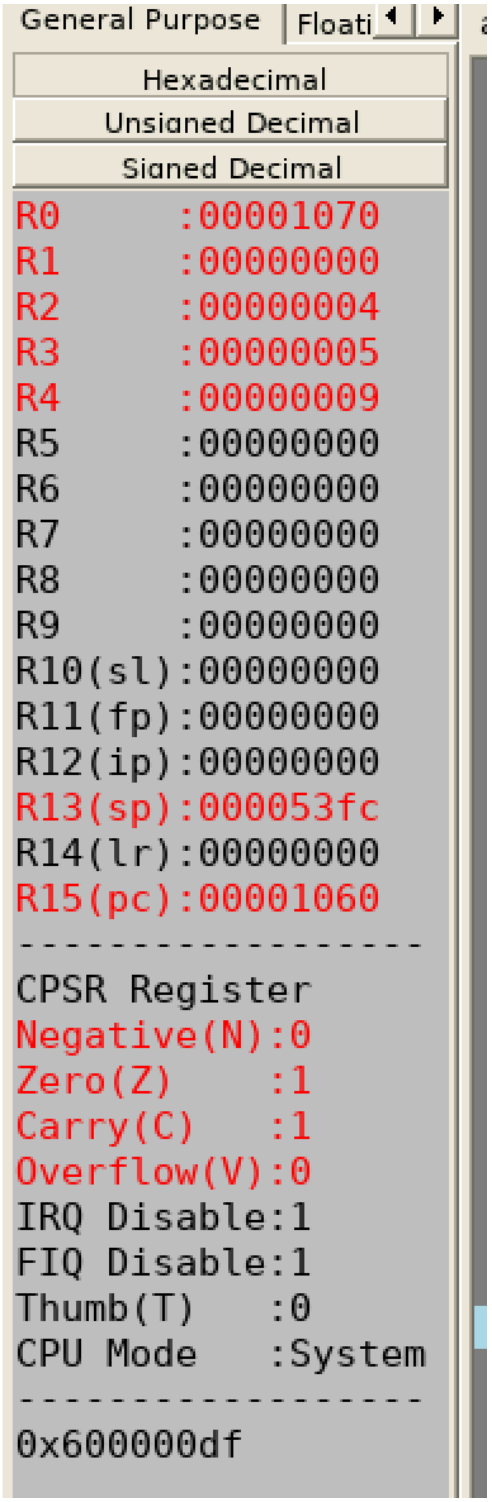
UE23CS251B

1	1. Problem statement: write an ARM assembly program to evaluate a postfix expression using stack. The program should support the basic arithmetic operations: addition (+), subtraction (-).The operands will be single-digit integers (1-9). Input Format: The postfix expression will be represented as an array of characters (i.e., a string). The expression should terminate with a null character ('\0'), indicating the end of the string. Output Format: The program will output the result of the postfix evaluation. Assumptions: • The operands are non-negative single-digit integers. • The expression contains no spaces. • The expression is well-formed, meaning there are no errors in the input (e.g., mismatched operators or operands). Program screen shot:

ARMSim_files > ASM assignment1_q1a.s

```
@ 1. Problem statement: write an ARM assembly program to evaluate a postfix expressionession using
@ stack. The program should support the basic arithmetic operations: additionition (+), subtractiontraction
@ (-).The operands will be single-digit integers (1-9).

.data
expression: .asciz "23+51-+"

.text

init:
    LDR R0, =expression

link:
    LDRB R1, [R0], #1
    CMP R1, #0
    BEQ end
    CMP R1, #'0'
    BLT operator
    CMP R1, #'9'
    BGT operator

    SUB R1, R1, #'0'            @ Convert ascii digit to numeric value
    STMFD R13!, {R1}
    B link

operator:
    CMP R1, #'+'
    BEQ addition
    CMP R1, #'-'
    BEQ subtraction
    B end

addition:
    LDMFD R13!, {R2, R3}        @ Pop the top two values from stack
    ADD R4, R2, R3
    STMFD R13!, {R4}            @ Push the result to stack
    B link

subtraction:
    LDMFD R13!, {R2, R3}        @ Pop the top two values of stack
    SUB R4, R3, R2
    STMFD R13!, {R4}            @ Push the result to stack
    B link

end:
    SWI 0x011
```

Screen shot of Register set output:



2

2. Problem statement : implement the Bubble Sort algorithm in ARM Assembly language to sort an array of integers in ascending order. The Bubble Sort algorithm repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process continues until the array is sorted.

Input:

• An array of integers stored in memory. The array will have N elements, where N is the size of the array. Each element is a single integer.

• The size of the array N will be provided as part of the input.

Output:

• The array will be sorted in ascending order. The program will output the sorted array.

Constraints:

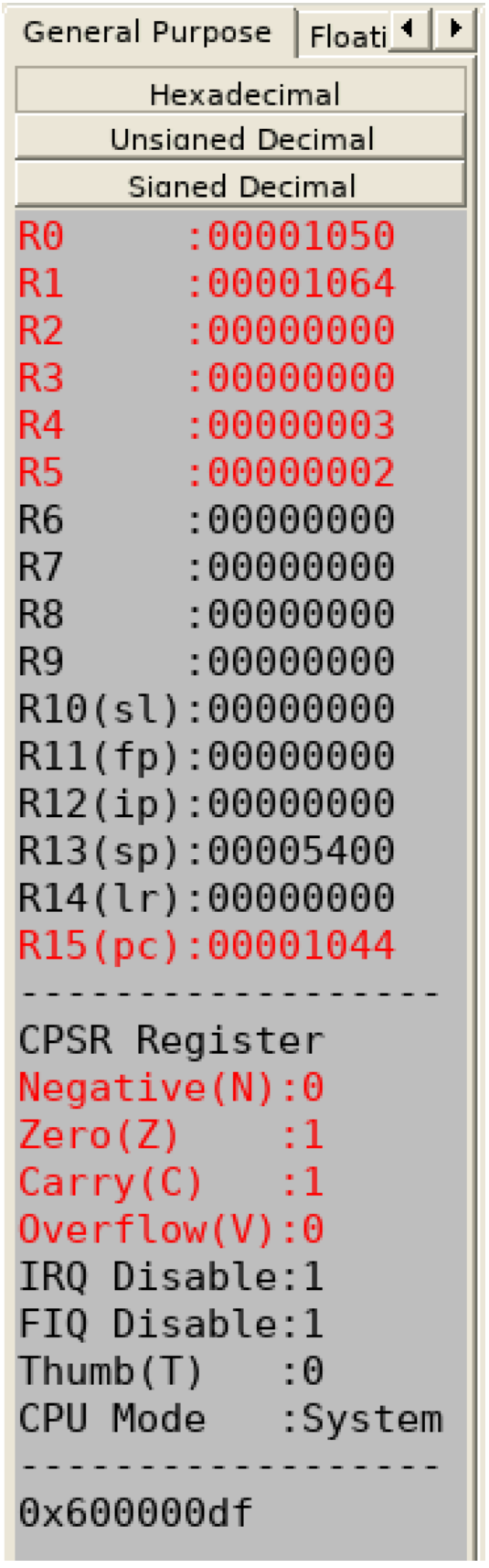
• The elements in the array are integers.• The program will handle sorting a small array. Student can define the array and array size (5 to 10 elements)

Program screen shot:

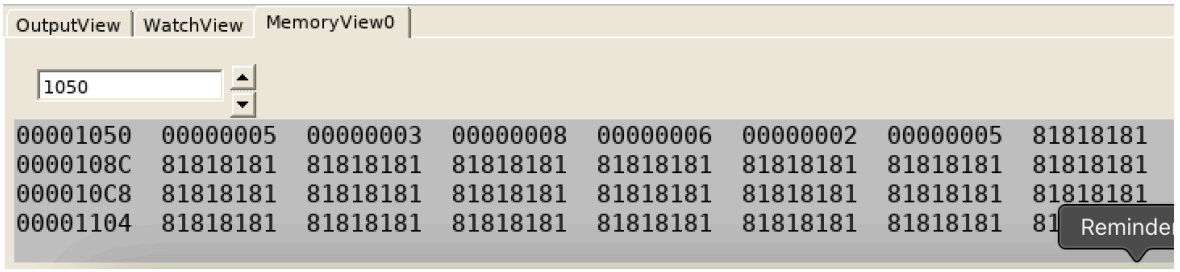
ARMSim_files > ASM assignment1_q2.s

```
1   @ Problem statement : implement the Bubble Sort algorithm in ARM Assembly language to
2   @ sort an array of integers in ascending order. The Bubble Sort algorithm repeatedly steps
3   @ through the list, compares adjacent elements, and swaps them if they are in the wrong
4   @ order. This process continues until the array is sorted.
5
6   .data
7   array:  .word 5, 3, 8, 6, 2  @ Input array
8   size:   .word 5              @ Size of the array
9
10  .text
11
12  start:
13      LDR R0, =array
14      LDR R1, =size
15      LDR R2, [R1]
16      SUB R2, R2, #1
17
18  outer_loop:
19      MOV R3, R2               @ Set inner loop counter
20
21  inner_loop:
22      LDR R4, [R0]             @ Load the current element into R4
23      LDR R5, [R0, #4]         @ Load the next element into R5
24      CMP R4, R5
25      BLE no_swap              @ If R4 <= R5 no swap
26
27      @ Swap R4 and R5
28      STR R5, [R0]             @ Store R5 in current position
29      STR R4, [R0, #4]         @ Store R4 in next position
30
31  no_swap:
32      ADD R0, R0, #4
33      SUBS R3, R3, #1          @ Decrement inner loop
34      BNE inner_loop
35
36
37      LDR R0, =array
38      SUBS R2, R2, #1          @ Decrement outer loop counter
39      BNE outer_loop
40
41  end:
42      SWI 0x11
43
```

Screen shot of Register set output and memory location:



Before:



After: