Microprocessor and Computer Architecture

UE23CS251B

4th Semester, Academic Year 2024-25

BANANA PROBLEM

Date:14/02/2025

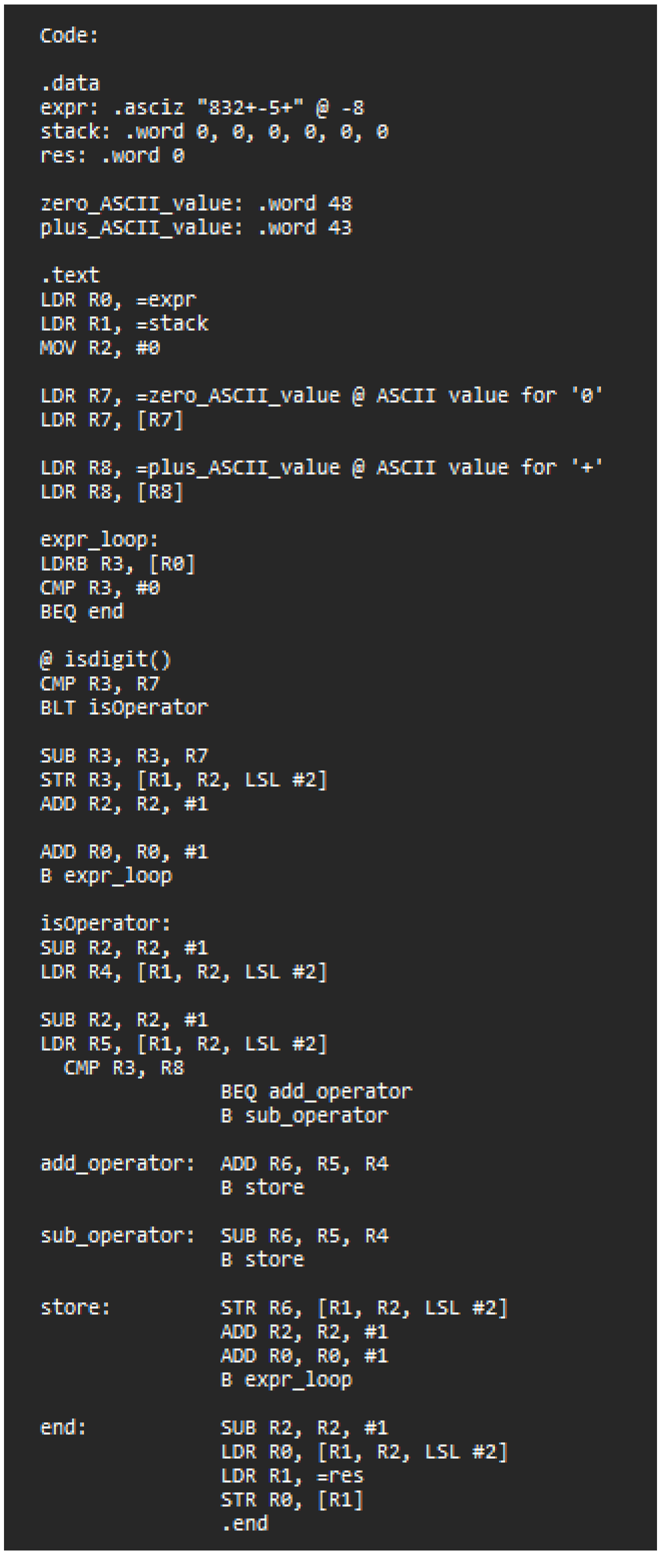
Name: KEERTHAN P.V	SRN: PES2UG23CS272	Section: 4E

Title of the Program

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).

I. ARM Assembly Code(1)

Typed Code to be Included. Screenshot of Code not permitted



```
Code:

.data
expr: .asciz "832+-5+" @ -8
stack: .word 0, 0, 0, 0, 0, 0
res: .word 0

zero_ASCII_value: .word 48
plus_ASCII_value: .word 43

.text
LDR R0, =expr
LDR R1, =stack
MOV R2, #0

LDR R7, =zero_ASCII_value @ ASCII value for '0'
LDR R7, [R7]

LDR R8, =plus_ASCII_value @ ASCII value for '+'
LDR R8, [R8]

expr_loop:
LDRB R3, [R0]
CMP R3, #0
BEQ end

@ isdigit()
CMP R3, R7
BLT isOperator

SUB R3, R3, R7
STR R3, [R1, R2, LSL #2]
ADD R2, R2, #1

ADD R0, R0, #1
B expr_loop

isOperator:
SUB R2, R2, #1
LDR R4, [R1, R2, LSL #2]

SUB R2, R2, #1
LDR R5, [R1, R2, LSL #2]
  CMP R3, R8
               BEQ add_operator
               B sub_operator

add_operator:  ADD R6, R5, R4
               B store

sub_operator:  SUB R6, R5, R4
               B store

store:         STR R6, [R1, R2, LSL #2]
               ADD R2, R2, #1
               ADD R0, R0, #1
               B expr_loop

end:           SUB R2, R2, #1
               LDR R0, [R1, R2, LSL #2]
               LDR R1, =res
               STR R0, [R1]
               .end
```

Code:

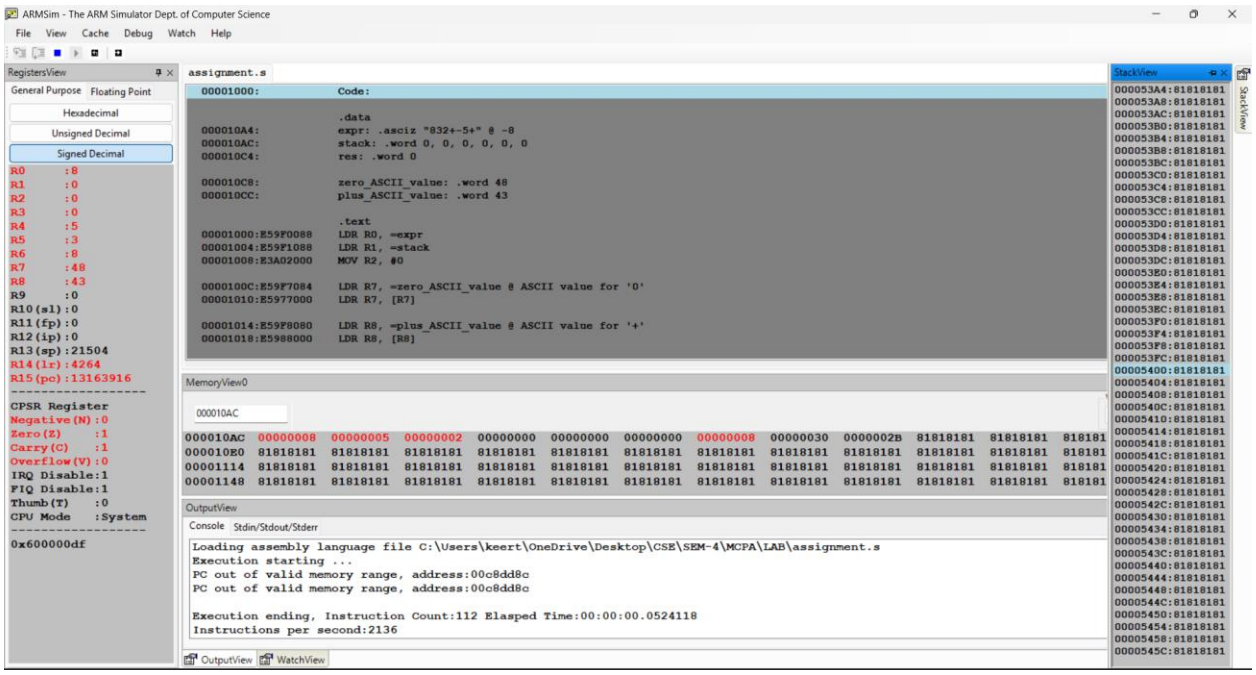
```
.data
expr: .asciz "832+-5+" @ -8
stack: .word 0, 0, 0, 0, 0, 0
res: .word 0

zero_ASCII_value: .word 48
plus_ASCII_value: .word 43

.text
LDR R0, =expr
LDR R1, =stack
MOV R2, #0

LDR R7, =zero_ASCII_value
LDR R7, [R7]

LDR R8, =plus_ASCII_value
LDR R8, [R8]

expr_loop:
LDRB R3, [R0]
CMP R3, #0
BEQ end

@ isdigit()
CMP R3, R7
BLT isOperator

SUB R3, R3, R7
STR R3, [R1, R2, LSL #2]
```

```
ADD R2, R2, #1

ADD R0, R0, #1
B expr_loop

isOperator:
SUB R2, R2, #1
LDR R4, [R1, R2, LSL #2]

SUB R2, R2, #1
LDR R5, [R1, R2, LSL #2]
  CMP R3, R8
            BEQ add_operator
            B sub_operator

add_operator:  ADD R6, R5, R4
            B store

sub_operator:  SUB R6, R5, R4
            B store

store:        STR R6, [R1, R2, LSL #2]
            ADD R2, R2, #1
            ADD R0, R0, #1
            B expr_loop

end:          SUB R2, R2, #1
            LDR R0, [R1, R2, LSL #2]
            LDR R1, =res
            STR R0, [R1]
```

.end

II. Output Screen Shots (1)

The result should be clearly visible in the screenshots. The screenshot should include the code window, register window, memory window and console window



```
00001000:              Code:

                       .data
000010A4:              expr: .asciz "832+-5+" @ -8
000010AC:              stack: .word 0, 0, 0, 0, 0, 0
000010C4:              res: .word 0

000010C8:              zero_ASCII_value: .word 48
000010CC:              plus_ASCII_value: .word 43

                       .text
00001000:E59F0088      LDR R0, =expr
00001004:E59F1088      LDR R1, =stack
00001008:E3A02000      MOV R2, #0

0000100C:E59F7084      LDR R7, =zero_ASCII_value @ ASCII value for '0'
00001010:E5977000      LDR R7, [R7]

00001014:E59F8080      LDR R8, =plus_ASCII_value @ ASCII value for '+'
00001018:E5988000      LDR R8, [R8]
```

```
R0       :8
R1       :0
R2       :0
R3       :0
R4       :5
R5       :3
R6       :8
R7       :48
R8       :43
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):4264
R15(pc):13163916
-----------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
-----------------
0x600000df
```

```
000010AC

000010AC  00000008  00000005  00000002  00000000  00000000  00000000  00000008  00000030  0000002B  81818181  81818181  818181
000010E0  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
00001114  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
00001148  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
```

```
Loading assembly language file C:\Users\keert\OneDrive\Desktop\CSE\SEM-4\MCPA\LAB\assignment.s
Execution starting ...
PC out of valid memory range, address:00c8dd8c
PC out of valid memory range, address:00c8dd8c

Execution ending, Instruction Count:112 Elasped Time:00:00:00.0524118
Instructions per second:2136
```

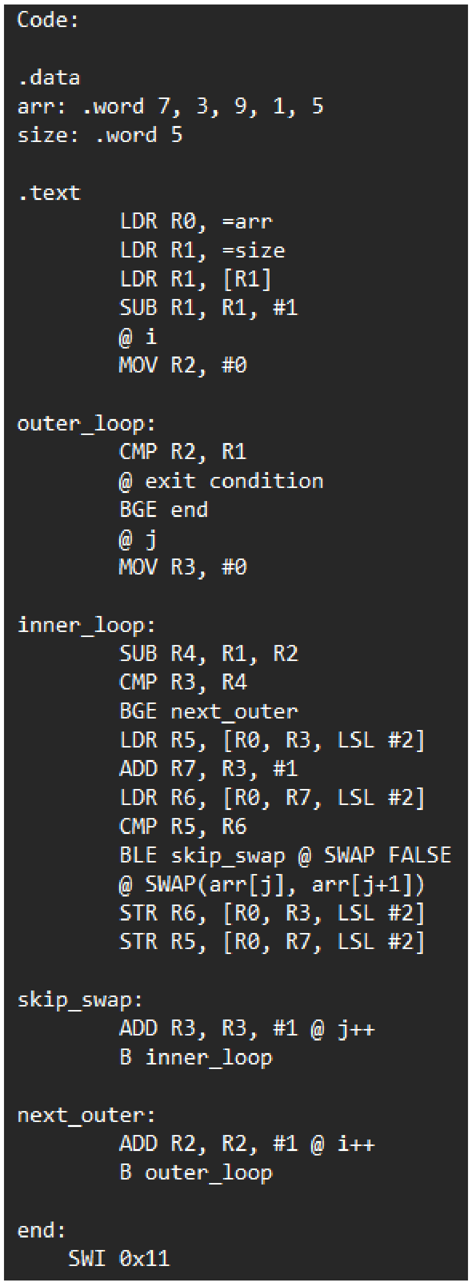
```
000053A4:81818181
000053A8:81818181
000053AC:81818181
000053B0:81818181
000053B4:81818181
000053B8:81818181
000053BC:81818181
000053C0:81818181
000053C4:81818181
000053C8:81818181
000053CC:81818181
000053D0:81818181
000053D4:81818181
000053D8:81818181
000053DC:81818181
000053E0:81818181
000053E4:81818181
000053E8:81818181
000053EC:81818181
000053F0:81818181
000053F4:81818181
000053F8:81818181
000053FC:81818181
00005400:81818181
00005404:81818181
00005408:81818181
0000540C:81818181
00005410:81818181
00005414:81818181
00005418:81818181
0000541C:81818181
00005420:81818181
00005424:81818181
00005428:81818181
0000542C:81818181
00005430:81818181
00005434:81818181
00005438:81818181
0000543C:81818181
00005440:81818181
00005444:81818181
00005448:81818181
0000544C:81818181
00005450:81818181
00005454:81818181
00005458:81818181
0000545C:81818181
```

Title of the Program

2. 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.

I. ARM Assembly Code(1)

Typed Code to be Included. Screenshot of Code not permitted



```
Code:

.data
arr: .word 7, 3, 9, 1, 5
size: .word 5

.text
        LDR R0, =arr
        LDR R1, =size
        LDR R1, [R1]
        SUB R1, R1, #1
        @ i
        MOV R2, #0

outer_loop:
        CMP R2, R1
        @ exit condition
        BGE end
        @ j
        MOV R3, #0

inner_loop:
        SUB R4, R1, R2
        CMP R3, R4
        BGE next_outer
        LDR R5, [R0, R3, LSL #2]
        ADD R7, R3, #1
        LDR R6, [R0, R7, LSL #2]
        CMP R5, R6
        BLE skip_swap @ SWAP FALSE
        @ SWAP(arr[j], arr[j+1])
        STR R6, [R0, R3, LSL #2]
        STR R5, [R0, R7, LSL #2]

skip_swap:
        ADD R3, R3, #1 @ j++
        B inner_loop

next_outer:
        ADD R2, R2, #1 @ i++
        B outer_loop

end:
    SWI 0x11
```

Code:

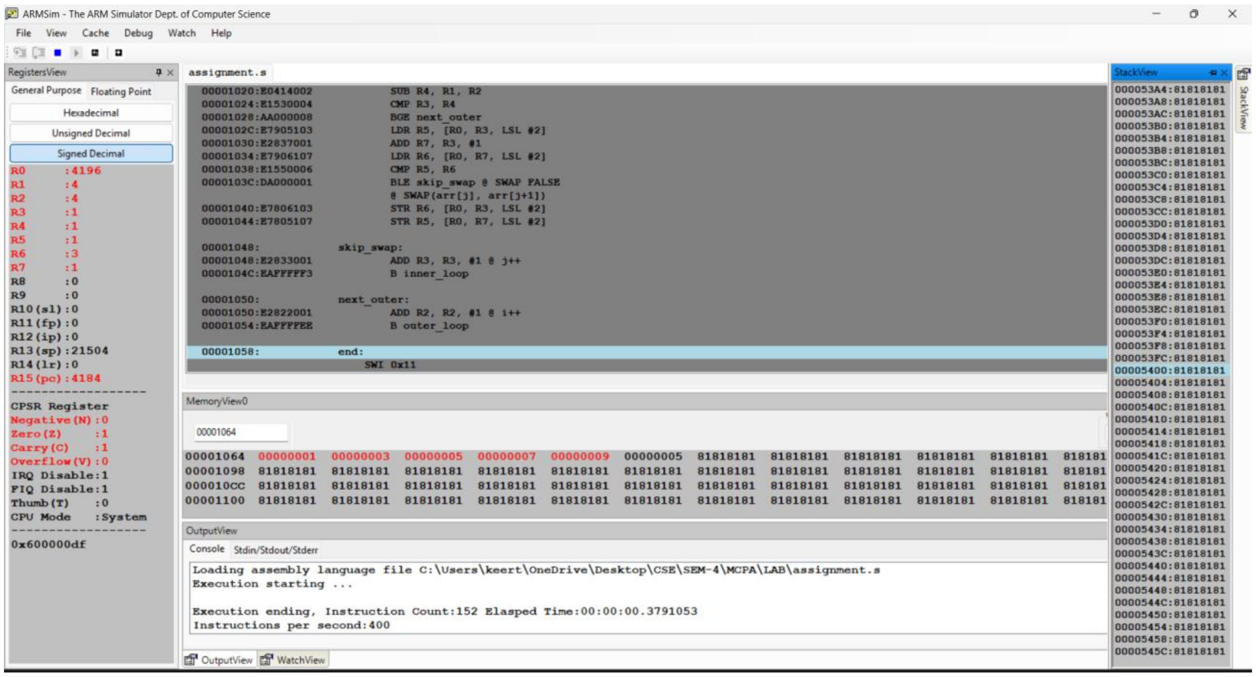
.data
arr: .word 7, 3, 9, 1, 5
size: .word 5

.text

```
        LDR R0, =arr
        LDR R1, =size
        LDR R1, [R1]
        SUB R1, R1, #1
        @ i
        MOV R2, #0

outer_loop:
        CMP R2, R1
        @ exit condition
        BGE end
        @ j
        MOV R3, #0

inner_loop:
        SUB R4, R1, R2
        CMP R3, R4
        BGE next_outer
        LDR R5, [R0, R3, LSL #2]
        ADD R7, R3, #1
        LDR R6, [R0, R7, LSL #2]
        CMP R5, R6
        BLE skip_swap @ SWAP FALSE
        @ SWAP(arr[j], arr[j+1])
        STR R6, [R0, R3, LSL #2]
        STR R5, [R0, R7, LSL #2]

skip_swap:
        ADD R3, R3, #1 @ j++
        B inner_loop
```

```
next_outer:
        ADD R2, R2, #1 @ i++
        B outer_loop

end:
   SWI 0x11
```

II. Output Screen Shots



ARMSim - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
assignment.s
MemoryView0
OutputView
Console Stdin/Stdout/Stderr
OutputView WatchView
StackView

```
R0        :4196
R1        :4
R2        :4
R3        :1
R4        :1
R5        :1
R6        :3
R7        :1
R8        :0
R9        :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4184
--------------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
--------------------
0x600000df
```

```
00001020:E0414002               SUB R4, R1, R2
00001024:E1530004               CMP R3, R4
00001028:AA000008               BGE next_outer
0000102C:E7905103               LDR R5, [R0, R3, LSL #2]
00001030:E2837001               ADD R7, R3, #1
00001034:E7906107               LDR R6, [R0, R7, LSL #2]
00001038:E1550006               CMP R5, R6
0000103C:DA000001               BLE skip_swap @ SWAP FALSE
                                @ SWAP(arr[j], arr[j+1])
00001040:E7806103               STR R6, [R0, R3, LSL #2]
00001044:E7805107               STR R5, [R0, R7, LSL #2]

00001048:               skip_swap:
00001048:E2833001               ADD R3, R3, #1 @ j++
0000104C:EAFFFFF3               B inner_loop

00001050:               next_outer:
00001050:E2822001               ADD R2, R2, #1 @ i++
00001054:EAFFFFEE               B outer_loop

00001058:               end:
                            SWI 0x11
```

```
00001064
00001064  00000001  00000003  00000005  00000007  00000009  00000005  81818181  81818181  81818181  81818181  81818181  818181
00001098  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
000010CC  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
00001100  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  818181
```

```
Loading assembly language file C:\Users\keert\OneDrive\Desktop\CSE\SEM-4\MCPA\LAB\assignment.s
Execution starting ...

Execution ending, Instruction Count:152 Elasped Time:00:00:00.3791053
Instructions per second:400
```

```
000053A4:81818181
000053A8:81818181
000053AC:81818181
000053B0:81818181
000053B4:81818181
000053B8:81818181
000053BC:81818181
000053C0:81818181
000053C4:81818181
000053C8:81818181
000053CC:81818181
000053D0:81818181
000053D4:81818181
000053D8:81818181
000053DC:81818181
000053E0:81818181
000053E4:81818181
000053E8:81818181
000053EC:81818181
000053F0:81818181
000053F4:81818181
000053F8:81818181
000053FC:81818181
00005400:81818181
00005404:81818181
00005408:81818181
0000540C:81818181
00005410:81818181
00005414:81818181
00005418:81818181
0000541C:81818181
00005420:81818181
00005424:81818181
00005428:81818181
0000542C:81818181
00005430:81818181
00005434:81818181
00005438:81818181
0000543C:81818181
00005440:81818181
00005444:81818181
00005448:81818181
0000544C:81818181
00005450:81818181
00005454:81818181
00005458:81818181
0000545C:81818181
```