

Microprocessor and Computer Architecture

UE22CS251B

LAB-4

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:-PES2UG22CS641	Section:-K
------------------	--------------------	------------

Week# 4

PROGRAM :- 1(a)

(A)Write an ALP to perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)

COMMANDS:-

.data

A: .word 2, 4, 6, 8

B: .word 1, 3, 5, 7

.text

.global _start

_start:

ldr r0, =A

ldr r1, =B

ldr r2, =4

mov r5, #0

loop:

ldr r3, [r0], #4

ldr r4, [r1], #4

mul r6, r3, r4

add r5, r5, r6

subs r2, r2, #1

bne loop

bkpt #0

.end

OUTPUT SCREENSHOT:

The screenshot shows the ARMSim# - The ARM Simulator interface. The window title is "ARMSim# - The ARM Simulator Dept. of Computer Science". The menu bar includes File, View, Cache, Debug, Watch, and Help. The toolbar has icons for Stop, Run, Break, Step, and Watch.

RegistersView (Registers tab selected):

General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0 : 00001044	
R1 : 00001054	
R2 : 00000000	
R3 : 00000008	
R4 : 00000007	
R5 : 00000064	
R6 : 00000038	
R7 : 00000000	
R8 : 00000000	
R9 : 00000000	
R10(sl) : 00000000	
R11(fp) : 00000000	
R12(ip) : 00000000	
R13(sp) : 00011400	
R14(lr) : 00000000	
R15(pc) : 00001028	

CodeView (sumof32bit.o):

```
    _start:
00001000:E59F0024    ldr r0, =A          @ Load address of array
00001004:E59F1024    ldr r1, =B          @ Load address of array
00001008:E3A02004    ldr r2, =4          @ Number of elements in

0000100C:E3A05000    mov r5, #0         @ Initialize sum to 0

loop:
00001010:E4903004    ldr r3, [r0], #4   @ Load element of A into r3
00001014:E4914004    ldr r4, [r1], #4   @ Load element of B into r4
00001018:E0060493    mul r6, r3, r4     @ Multiply elements of A and B
0000101C:E0855006    add r5, r5, r6     @ Add multiplication result to sum
00001020:E2522001    subs r2, r2, #1    @ Decrement the count of elements
00001024:1AFFFFF9    bne loop          @ Repeat loop until R2 is zero
00001028:E1200070    bkpt #0           @ Breakpoint (you can re-enable it by pressing F5)
0000102C:00000000    .end...
000000010
```

StackView (Stack View):

000113B0:81818181	000113B4:81818181	000113B8:81818181	000113BC:81818181
000113C0:81818181	000113C4:81818181	000113C8:81818181	000113CC:81818181
000113D0:81818181	000113D4:81818181	000113D8:81818181	000113DC:81818181
000113E0:81818181	000113E4:81818181	000113E8:81818181	000113EC:81818181
000113F0:81818181	000113F4:81818181	000113F8:81818181	000113FC:81818181
00011400:????????	00011404:????????	00011408:????????	0001140C:????????
00011410:????????	00011414:????????	00011418:????????	0001141C:????????

OutputView (Console tab selected):

```
Loading assembly language file C:\Users\V V Mohith
Execution starting ...
Location 1028: Invalid opcode encountered
```

MemoryView0 (Memory View):

0000100C	E3A05000	E4903004	E4914004
00001018	E0060493	E0855006	E2522001
00001024	1AFFFFF9	E1200070	00001034

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:PES2UG22CS641	Section:-K
------------------	-------------------	------------

Week# _____ 4 _____

PROGRAM :- 1(b)

Write an ALP to perform Convolution using MLA instruction
(Addition of multiplication of respective numbers of loc A
and loc B).

Commands:-

_start:

ldr r0, =arrayA

ldr r1, =arrayB

ldr r2, =result

mov r3, #0

mov r4, #0

convolution_loop:

ldr r5, [r0, r4]

ldr r6, [r1, r4]

mla r3, r5, r6, r3

add r4, r4, #4

```
cmp r4, #16  
bne convolution_loop
```

```
str r3, [r2]
```

```
mov r0, #1  
ldr r1, =result  
ldr r2, =4  
mov r7, #4  
swi 0
```

```
mov r7, #1  
mov r0, #0  
swi 0
```

OUTPUT SCREENSHOT:

ARMSim# - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView X

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :0 R1 :0 R2 :4 R3 :70 R4 :16 R5 :4 R6 :8 R7 :1 R8 :0 R9 :0 R10(sl) :0 R11(fp) :0 R12(ip) :0 R13(sp) :70656 R14(lr) :0 R15(pc) :70656

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

0x6000000df

CodeView sumof32bit(b).s

```

0000107C:00000000    result:    .word 0      @ Variable to store the result
                        .section .text
                        .global _start

_start:
00001000:E59F0048    ldr r0, =arrayA      @ Load the address of array A
00001004:E59F1048    ldr r1, =arrayB      @ Load the address of array B
00001008:E59F2048    ldr r2, =result       @ Load the address of the result variable
0000100C:E3A03000    mov r3, #0          @ Initialize accumulator
00001010:E3A04000    ldr r4, #0          @ Initialize loop counter
convolution_loop:
00001014:E7905004    ldr r5, [r0, r4]    @ Load the value from array A
00001018:E7916004    ldr r6, [r1, r4]    @ Load the value from array B
0000101C:E0233695    mla r3, r5, r6, r3   @ Multiply and accumulate
00001020:E2844004    add r4, r4, #4      @ Increment loop counter
00001024:E3540010    cmp r4, #16        @ Compare loop counter with 16
00001028:1AFFFFF9    bne convolution_loop @ Branch if not equal
0000102C:E5823000    str r3, [r2]       @ Store result back to memory

```

StackView

MemoryView0

Word Size: 32Bit

0000100C	E3A03000	E3A04000
00001014	E7905004	E7916004
0000101C	E0233695	E2844004

OutputView

Console stdio/stderr

```

Loading assembly language file C:\Users\V V Mohith\Downloads\sumof32bit(b).s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400

Execution ending, Instruction Count:16658 Elapsed Time:00:00:00.1147241
Instructions per second:145200

```

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:-PES2UG22CS641	Section:-K
------------------	--------------------	------------

Week# 4

PROGRAM:-2

Write an ALP to implement $\text{Sum}[l] += a[i][j]$

Commands:-

.global _start

_start:

MOV R1, #3

MOV R2, #4

LDR R3, =ArrayA

MOV R4, #0

OuterLoop:

MOV R5, #0

InnerLoop:

LDR R6, [R3], #4

ADD R7, R7, R6

ADD R5, R5, #1

CMP R5, R2

BLT InnerLoop

STR R7, [R8], #4

MOV R7, #0

ADD R4, R4, #1

CMP R4, R1

BLT OuterLoop

MOV R7, #1

SWI 0

.DATA

ArrayA: .WORD 1, 2, 3, 4

.WORD 5, 6, 7, 8

.WORD 9, 10, 11, 12

Sum: .SPACE 12

OUTPUT SCREENSHOT:-

ARMSim# - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView CodeView MemoryView0 StackView

well2ndquestion.o

Word Size: 32Bit

Registers View (Signed Decimal):

- R0 : 0
- R1 : 3
- R2 : 4
- R3 : 4184
- R4 : 0
- R5 : 4
- R6 : 4
- R7 : 10
- R8 : 4
- R9 : 0
- R10 (s1) : 0
- R11 (fp) : 0
- R12 (ip) : 0
- R13 (sp) : 70656
- R14 (lr) : 0
- R15 (pc) : 4136**

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

CodeView (Assembly Language):

```

00001028 E4887004 E3A07000 E2844001
00001034 E1540001 BAFFFFF4 E3A07001
00001040 EF000000 00001048 00000001
_start:
    MOV R1, #3 @ Number of rows in the 2D array
    MOV R2, #4 @ Number of columns in the 2D array
    @ Initialize the 2D array a
    LDR R3, =ArrayA @ Load the address of array a into
    MOV R4, #0 @ Initialize row index to 0
OuterLoop:
    MOV R5, #0 @ Initialize column index to 0
    InnerLoop:
        LDR R6, [R3], #4 @ Load the next element from array
        ADD R7, R7, R6 @ Accumulate the element in R6 to the
        ADD R5, R5, #1 @ Increment the column index
        CMP R5, R2 @ Compare column index with the number of
        BLT InnerLoop @ Branch to InnerLoop if less than
        STR R7, [R8], #4 @ Store the sum of the row in the
        MOV R7, #0 @ Reset the sum to 0 for the next row
        ADD R4, R4, #1 @ Increment the row index
        CMP R4, R1 @ Compare row index with the number of rows
        BLT OuterLoop @ Branch to OuterLoop if less than
        @ The sums are now stored in the Sum array
        @ Terminate the program

```

MemoryView0 (Stack View):

```

000113B0:81818181
000113B4:81818181
000113B8:81818181
000113BC:81818181
000113C0:81818181
000113C4:81818181
000113C8:81818181
000113CC:81818181
000113D0:81818181
000113D4:81818181
000113D8:81818181
000113DC:81818181
000113E0:81818181
000113E4:81818181
000113E8:81818181
000113EC:81818181
000113F0:81818181
000113F4:81818181
000113F8:81818181
000113FC:81818181
00011400:?????????
00011404:?????????
00011408:?????????
0001140C:?????????
00011410:?????????
00011414:?????????
00011418:?????????
0001141C:?????????

```

OutputView (Console):

```

Loading assembly language file C:\Users\V V Mohith\Downloads\well2ndquestion.s
Execution starting ...
Location 1028: Store to invalid memory location 0x0

Execution ending, Instruction Count:26 Elapsed Time:00:00:00.0836126
Instructions per second:310

```

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:-PES2UG22CS641	Section:-K
------------------	--------------------	------------

Week# 4

PROGRAM:-3

Write an ALP to find the length of a given string

COMMAND:-

```
.data
a:.asciz "Myname"

.text
ldr r1,=a
loop:
    ldrb r0,[r1],#1
    add r10,r10,#1
    cmp r0,#0
    bne loop
    swi 0x11
    BNE LOOP
    BEQ EXIT
EXIT: SWI 0x011
.end
```

OUTPUT SCREENSHOT:-

The screenshot shows the ARMSim# - The ARM Simulator interface. The window title is "ARMSim# - The ARM Simulator Dept. of Computer Science". The menu bar includes File, View, Cache, Debug, Watch, and Help. The RegistersView pane on the left displays general purpose registers R0-R15 and CPSR register bits, with R4 highlighted in red. The CodeView pane in the center shows assembly code for "lengthofstring.o" with memory addresses and assembly instructions. The MemoryView pane at the bottom shows memory starting at address 0000100C with word sizes 8Bit and 16Bit options. The OutputView pane at the bottom shows console output for loading an assembly file.

RegistersView

General Purpose Floating Point

Hexadecimal
Unsigned Decimal
Signed Decimal

R0 : 0
R1 : 0
R2 : 0
R3 : 0
R4 : 14
R5 : 0
R6 : 0
R7 : 0
R8 : 0
R9 : 0
R10 (sl) : 0
R11 (fp) : 0
R12 (ip) : 0
R13 (sp) : 70656
R14 (lr) : 0
R15 (pc) : 70656

CPSR Register

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

CodeView

lengthofstring.o

```
.data
00001020:616E794D    a::asciz "Mynameismohith"
                     :7369656D
                     :69686F6D
                     :006874

.text
00001000:E59F1014    ldr r1,a
loop:
00001004:E4D13001    ldrb r3,[r1],#1
00001008:E3530000    cmp r3,#0
0000100C:E2844001    add r4,r4,#1
00001010:1AFFFFF8    bne loop
00001014:E2444001    sub r4,r4,#1
00001018:EF000011    swi 0x11...
                     :00000000
```

MemoryView0

Word Size: 8Bit 16Bit

	0000100C	E2844001	1AFFFFF8	E2444001
00001018	EF000011	00001020	616E794D	
00001024	7369656D	69686F6D	00006874	
00001030	01010101	01010101	01010101	01010101

OutputView

Console stdin/stdout/stderr

Loading assembly language file C:\Use

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:PES2UG22CS641	Section:-K
------------------	-------------------	------------

Week# 4

PROGRAM:-4

Write an ALP to copy string from one location to another

COMMANDS:-

.data

srcstr: .asciz "Hello, World!"

dststr: .space 50

.global _start

_start:

MOV R0, #srcstr

MOV R1, #dststr

copy_char:

LDRB R2, [R0], #1

STRB R2, [R1], #1

CMP R2, #0

BEQ display_strings

B copy_char

display_strings:

MOV R7, #4

MOV R0, #1

MOV R2, #50

MOV R0, #srcstr

SWI 0

MOV R0, #dststr

SWI 0

MOV R7, #1

MOV R0, #0

SWI 0

OUTPUTSCREENSHOT:

The screenshot shows the ARMSim# - The ARM Simulator interface. The window title is "ARMSim# - The ARM Simulator Dept. of Computer Science". The menu bar includes File, View, Cache, Debug, Watch, and Help. The main window is divided into several panes:

- RegistersView:** Shows general purpose registers R0-R15 and floating point registers F0-F15. R15 (pc) is highlighted at 4172.
- CodeView:** Displays the assembly code for the "copystring(2).o" program. The code includes data section definitions for source and destination strings, a start label, and a copy_char subroutine. The assembly code is:

```
.data
    srcstr: .asciz "Hello, World!" @ Example source string
    dststr: .space 50          @ Destination string

.global _start
_start:
    MOV R0, #srcstr           @ Initialize R0 with source address
    MOV R1, #dststr            @ Initialize R1 with destination address

copy_char:
    LDRB R2, [R0], #1          @ Load a byte from source
    STRB R2, [R1], #1          @ Store the byte to destination
    CMP R2, #0                 @ Check if the character is zero
    BEQ display_str            @ If zero, branch to display_string

    B copy_char                @ Loop back to copy the next character

    .end
```
- StackView:** Shows the stack memory starting at address 000113B0, filled with the value 81818181.
- MemoryView0:** Shows memory starting at address 00001044, containing the assembly code for the copy_string function.
- OutputView:** Shows the console output, including the loading of the assembly file, execution start, location of the store to invalid memory, and the end of execution.

The status bar at the bottom indicates the instruction count (4), elapsed time (00:00:00.0504333), and instructions per second (79).

Microprocessor and Computer Architecture

UE22CS251B

4th Semester, Academic Year 2023-2024

Date:08-02-2024

Name: V V Mohith	SRN:PES2UG22CS641	Section:-K
------------------	-------------------	------------

Week# 4

PROGRAM:-1(assignment)

Write an ALP to find whether a given character is present in a string. If present, find how many times the given character is present in a string.

COMMANDS:-

.TEXT

.global _start

_start:

LDR R0, =Str

LDR R1, =Char

LDRB R3, [R1]

MOV R5, #0

SearchLoop:

LDRB R2, [R0], #1

CMP R2, #0

BEQ EndProgram

CMP R2, R3

BEQ FoundCharacter

B SearchLoop

FoundCharacter:

ADD R5, R5, #1

B SearchLoop

EndProgram:

CMP R5, #0

BEQ CharacterNotFound

MOV R7, #4

LDR R0, =FoundMsg

SWI 0

B Exit

CharacterNotFound:

MOV R7, #4

LDR R0, =NotFoundMsg

SWI 0

Exit:

MOV R7, #1

SWI 0

.DATA

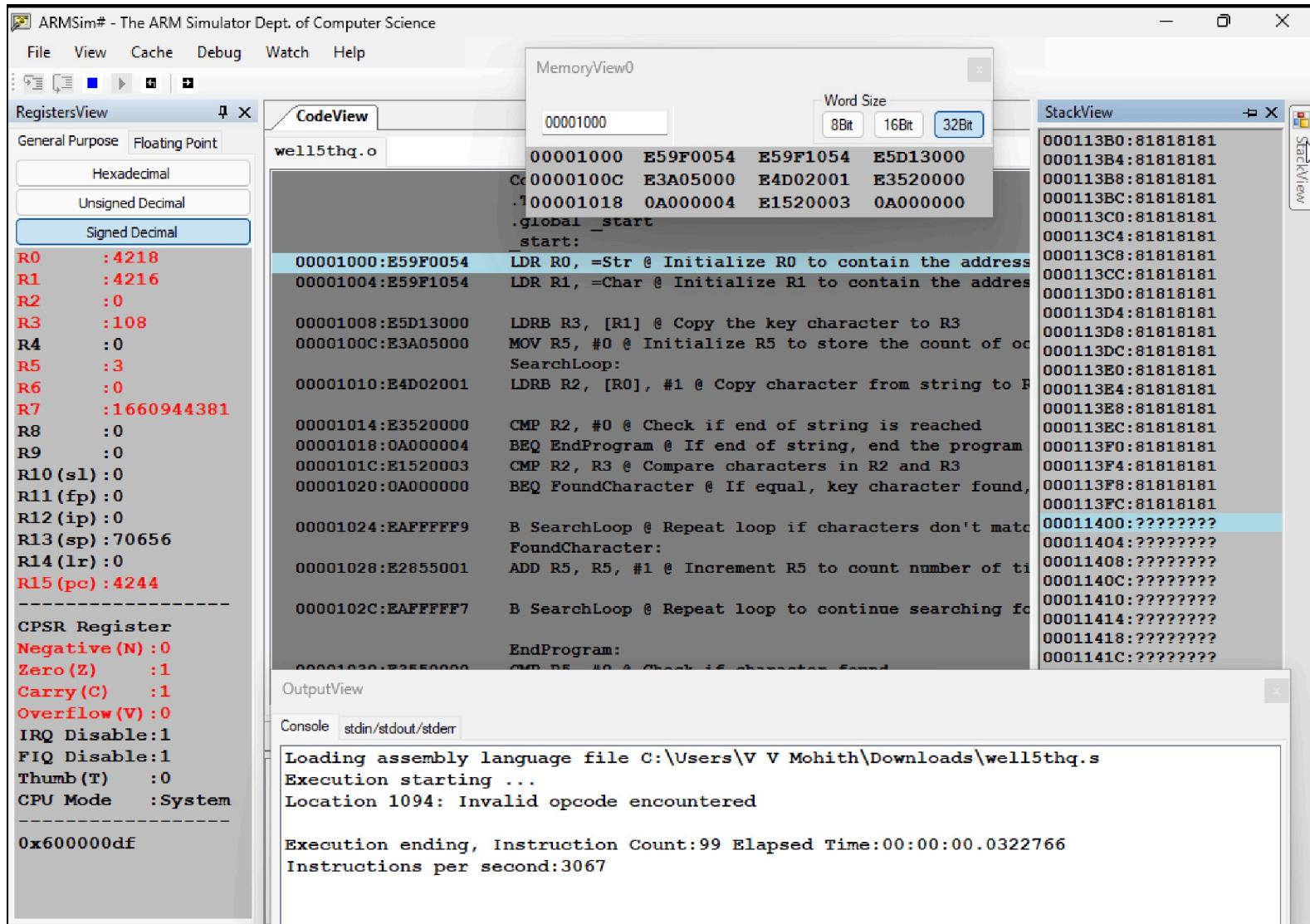
Str: .ASCIZ "hello world"

Char: .ASCIZ "l"

FoundMsg: .ASCIZ "Character Found\n"

NotFoundMsg: .ASCIZ "Character Not Found\n"

OUTPUTSCREESHOT:-



PROGRAM:-2(assignment)

Write a program in ARM7TDMI-ISA to generate a diagonal matrix.

;Note: do not read the matrix elements.

COMMANDS:-

.text

.global _start

_start:

MOV R0, #5

MOV R1, #1

LDR R2, =Matrix

GenerateDiagonal:

MOV R3, #0

RowLoop:

CMP R3, R0

BEQ EndProgram

MOV R4, R3, LSL #2

STR R1, [R2, R4]

ADD R3, R3, #1

B RowLoop

EndProgram:

MOV R7, #1

SWI 0

.data

Matrix: .space 25

OUTPUTSCREESHOT:-

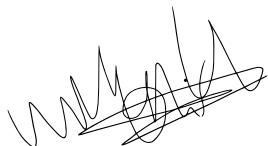
The screenshot shows the ARMSim interface with the following windows and their content:

- RegistersView:** Shows general purpose registers (R0-R15) and CPSR register values.
- CodeView:** Displays assembly code for the program. Key instructions include:
 - MOV R0, #5 @ Number of rows and columns in the matrix
 - MOV R1, #1 @ Value to set for diagonal elements
 - LDR R2, =Matrix @ Load the address of the matrix in GenerateDiagonal:
 - MOV R3, #0 @ Initialize column index to 0
 - RowLoop:
 - CMP R3, R0 @ Compare column index with number of rows
 - B EQ EndProgram @ If end of matrix, exit program
 - MOV R4, R3, LSL #2 @ Calculate the offset for the diagonal element
 - STR R1, [R2, R4] @ Set the diagonal element to the value in R1
 - ADD R3, R3, #1 @ Increment column index
 - B RowLoop @ Repeat for next row
 - EndProgram:
 - MOV R7, #1 @ SWI service command for exiting
 - SWI 0 @ Terminate program
 - .data
 - 00001034:00000000 @ Matrix: .space 25 @ Matrix of size 5x5...
- MemoryView0:** Shows memory dump starting at address 00001000.
- StackView:** Shows the stack starting at address 000113B0.
- OutputView:** Shows the console output with assembly loading information and error messages about illegal address transfers.
- Console:** Shows the command stdin/stdout/stderr.

Disclaimer:

- The programs and output submitted is duly written, verified and executed by me.
- I have not copied from any of my peers nor from the external resource such as internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature:



Name:V V Mohith

SRN:PES2UG22CS641

Section: K

Date:18-01-2024