Computer Organization CSI 504 – Fall 2023
Programming Assignment-2

Amulya

November 1, 2023

Contents

1 How the Program Was Tested

In order to make sure that the program works properly and gives results we conducted tests. These tests included scenarios, like expressions and single digit expressions as well, as more complicated cases that involved nested parentheses and various arithmetic operations.

2 Testing Outputs

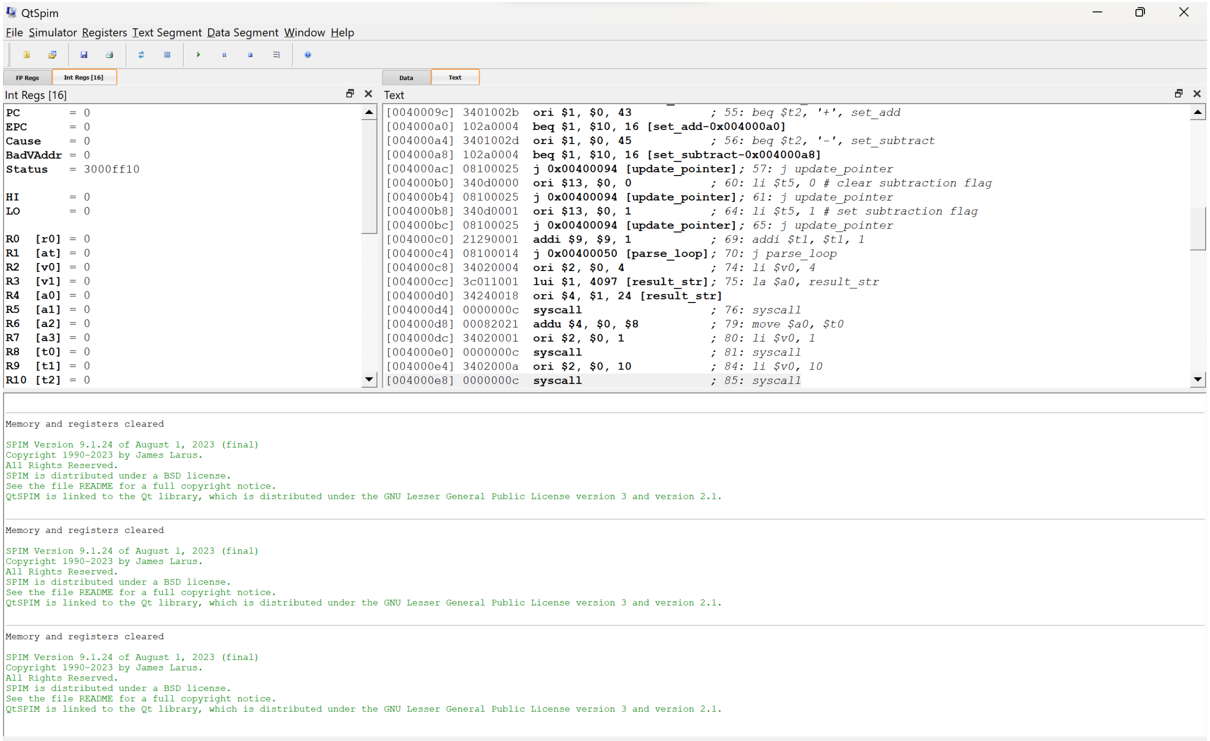
I ran the program through the MIPS simulator. Observed that the outputs matched the expected results, for the provided test cases. The output format included both the expression, in postfix notation and the calculated result.

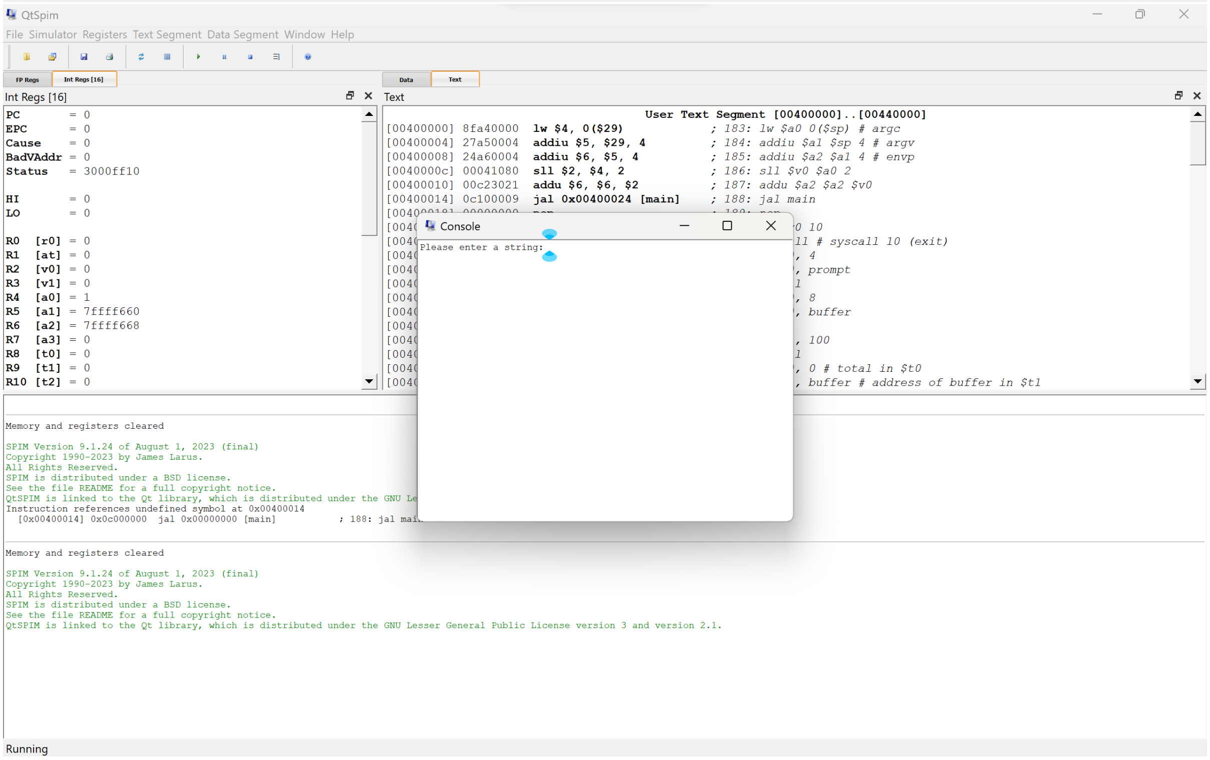
Listing 1: Example Testing Output

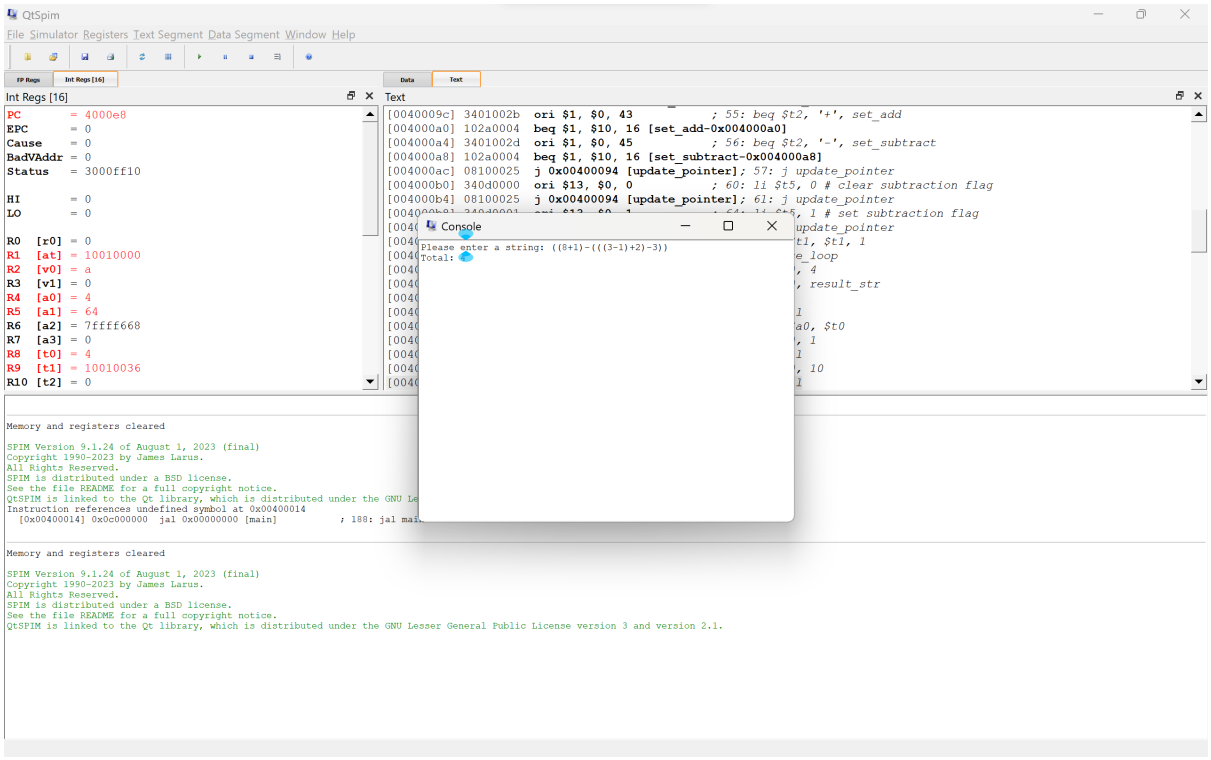
```
Input Expression: ((1-3)+5)
Postfix Notation: 13-5+
Result: 3

Input Expression: (1-(3+5))
Postfix Notation: 135+-
Result: -7
```

Below are the screenshots attached:


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 0
R6 [a2] = 0
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
Text
[0040009c] 3401002b ori $1, $0, 43 ; 55: beq $t2, '+', set_add
[004000a0] 102a0004 beq $1, $10, 16 [set_add-0x004000a0]
[004000a4] 3401002d ori $1, $0, 45 ; 56: beq $t2, '-', set_subtract
[004000a8] 102a0004 beq $1, $10, 16 [set_subtract-0x004000a8]
[004000ac] 08100025 j 0x00400094 [update_pointer]; 57: j update_pointer
[004000b0] 340d0000 ori $13, $0, 0 ; 60: li $t5, 0 # clear subtraction flag
[004000b4] 08100025 j 0x00400094 [update_pointer]; 61: j update_pointer
[004000b8] 340d0001 ori $13, $0, 1 ; 64: li $t5, 1 # set subtraction flag
[004000bc] 08100025 j 0x00400094 [update_pointer]; 65: j update_pointer
[004000c0] 21290001 addi $9, $9, 1 ; 69: addi $t1, $t1, 1
[004000c4] 08100014 j 0x00400050 [parse_loop]; 70: j parse_loop
[004000c8] 34020004 ori $2, $0, 4 ; 74: li $v0, 4
[004000cc] 3c011001 lui $1, 4097 [result_str]; 75: la $a0, result_str
[004000d0] 34240018 ori $4, $1, 24 [result_str]
[004000d4] 0000000c syscall ; 76: syscall
[004000d8] 00082021 addu $4, $0, $8 ; 79: move $a0, $t0
[004000dc] 34020001 ori $2, $0, 1 ; 80: li $v0, 1
[004000e0] 0000000c syscall ; 81: syscall
[004000e4] 3402000a ori $2, $0, 10 ; 84: li $v0, 10
[004000e8] 0000000c syscall ; 85: syscall
Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffff660
R6 [a2] = 7ffff668
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
Text
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
ll # syscall 10 (exit)
, prompt
, buffer
, 100
, 0 # total in $t0
, buffer # address of buffer in $t1
Console
Please enter a string:
Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
Instruction references undefined symbol at 0x00400014
[0x00400014] 0x0c000000 jal 0x00000000 [main] ; 188: jal mai
Memory and registers cleared
SPIM Version 9.1.24 of August 1, 2023 (final)
Copyright 1990-2023 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
Running



3 How to Run the Program

To run the program, follow these steps:

1. Open the MIPS simulator.
2. Load the assembly file into the simulator.
3. Execute the program.

4 Parameters

The program does not require any parameters to run. All inputs are provided via the keyboard during execution.

5 Program Code

```
    .data
prompt: .asciiz "Please enter a string: "
result_str: .asciiz "Total: "

.text
.globl main

main:
    # Print prompt
    li $v0, 4
    la $a0, prompt
    syscall

    # Read input string
    li $v0, 8
```

```
    la $a0, buffer
    li $a1, 100
    syscall

    # Initialize total to 0
    li $t0, 0 # total in $t0
    la $t1, buffer # address of buffer in $t1

parse_loop:
    lb $t2, 0($t1) # load next byte from buffer to $t2
    beqz $t2, print_result # if zero (end of string), go to print_result

    # Check if character is '(' or ')'
    beq $t2, '(', skip_char
    beq $t2, ')', skip_char

    # Check if character is digit
    li $t3, '0'
    li $t4, '9'
    blt $t2, $t3, check_operator
    bgt $t2, $t4, check_operator

    # If we are here, character is a digit.
    # Add or subtract it to/from total.
    sub $t2, $t2, '0' # convert ASCII to integer
    bnez $t5, subtract
    add $t0, $t0, $t2
    j update_pointer

subtract:
    sub $t0, $t0, $t2

update_pointer:
    # Move to the next character
    addi $t1, $t1, 1
    j parse_loop

check_operator:
    # Check if character is '+' or '-'
    beq $t2, '+', set_add
    beq $t2, '-', set_subtract
    j update_pointer

set_add:
    li $t5, 0 # clear subtraction flag
    j update_pointer

set_subtract:
    li $t5, 1 # set subtraction flag
    j update_pointer

skip_char:
    # Skip this character
    addi $t1, $t1, 1
    j parse_loop

print_result:
    # Print the result string
```

```
li $v0, 4
la $a0, result_str
syscall

# Print the total
move $a0, $t0
li $v0, 1
syscall

# Exit the program
li $v0, 10
syscall

.data
buffer: .space 100
```