**Introduction to SPIM Simulator for the MIPS Assembly Language**

**LAB # 01**



**Fall 2022**

**CSE304L Computer Organization & Architecture**

**Submitted by:**

Maaz Habib[20PWCSE1952]

Hamid Ur Rehman [20PWCSE1969]

Danyal Khursheed [20PWCSE1959]

**Class Section:**

C

**Submitted to:**

Dr. Amaad Khalil

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Introduction to SPIM Simulator for the MIPS Assembly Language**

**SPIM STIMULATOR**

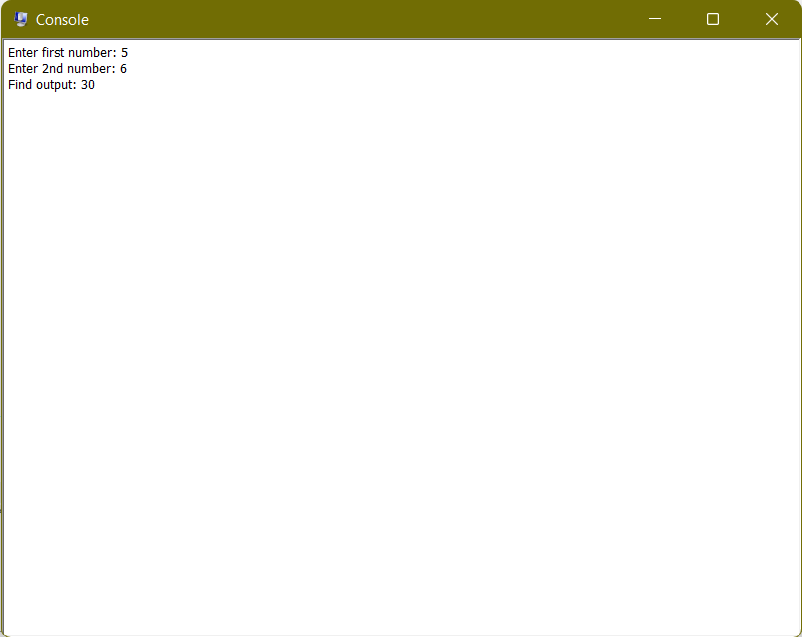
SPIM (**spam over instant messaging**) is a software simulator for the MIPS architecture, which is a Reduced Instruction Set Computing (RISC) architecture used in many computer systems. The MIPS architecture was designed to be simple, efficient, and easy to implement. Spim is a self-contained simulator that reads and executes assembly language programs written for the processor.

In SPIM, program execution begins at the location with the label main. A label is a symbolic name for an address in memory.

**THE SIMULATION:**

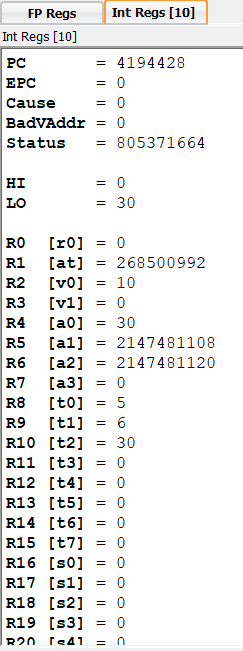
**Console:**

This window displays the input and output of the simulation. It is used for inputting commands and data.



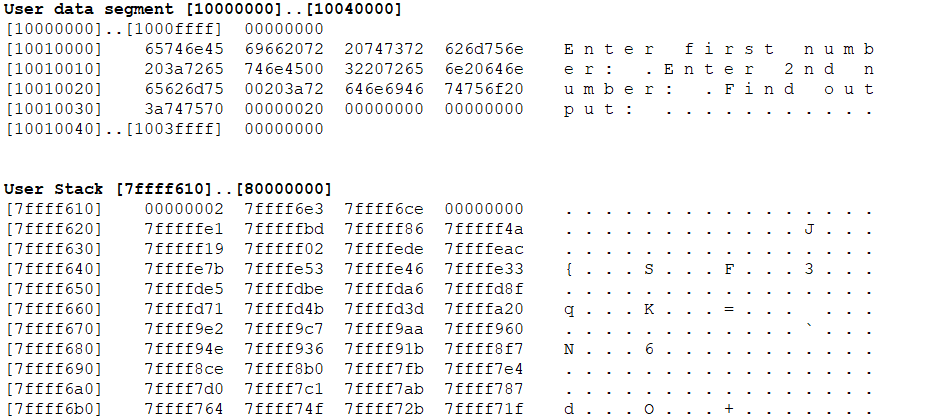
**Register File:**

This window displays the contents of the MIPS register file, including the general-purpose registers.



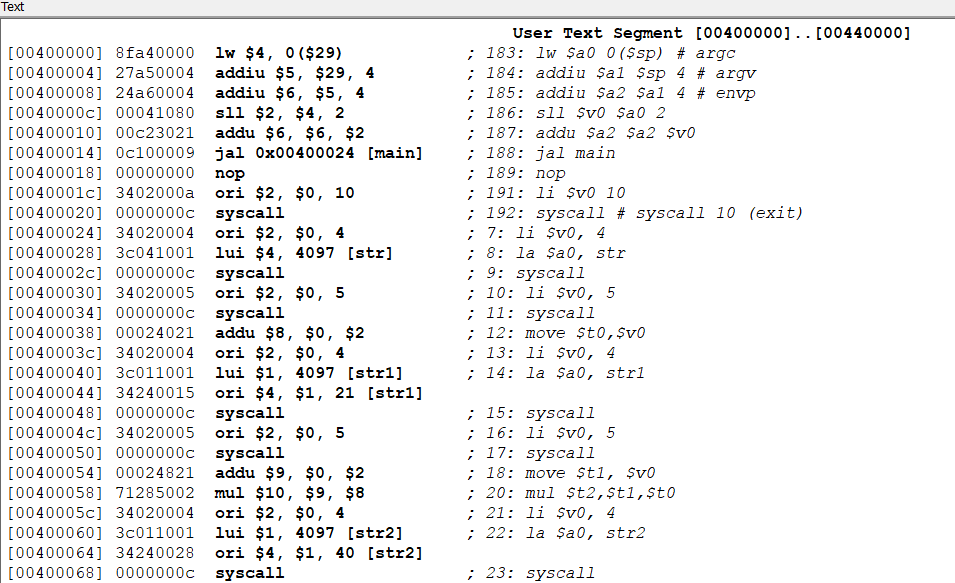
**Data Segment:**

This window displays the contents of the data segment of memory, which is used for storing variables and data for the program.



**Text Segment:**

This window displays the contents of the text segment of memory, which is used for storing the MIPS instructions for the program.



**Memory:**

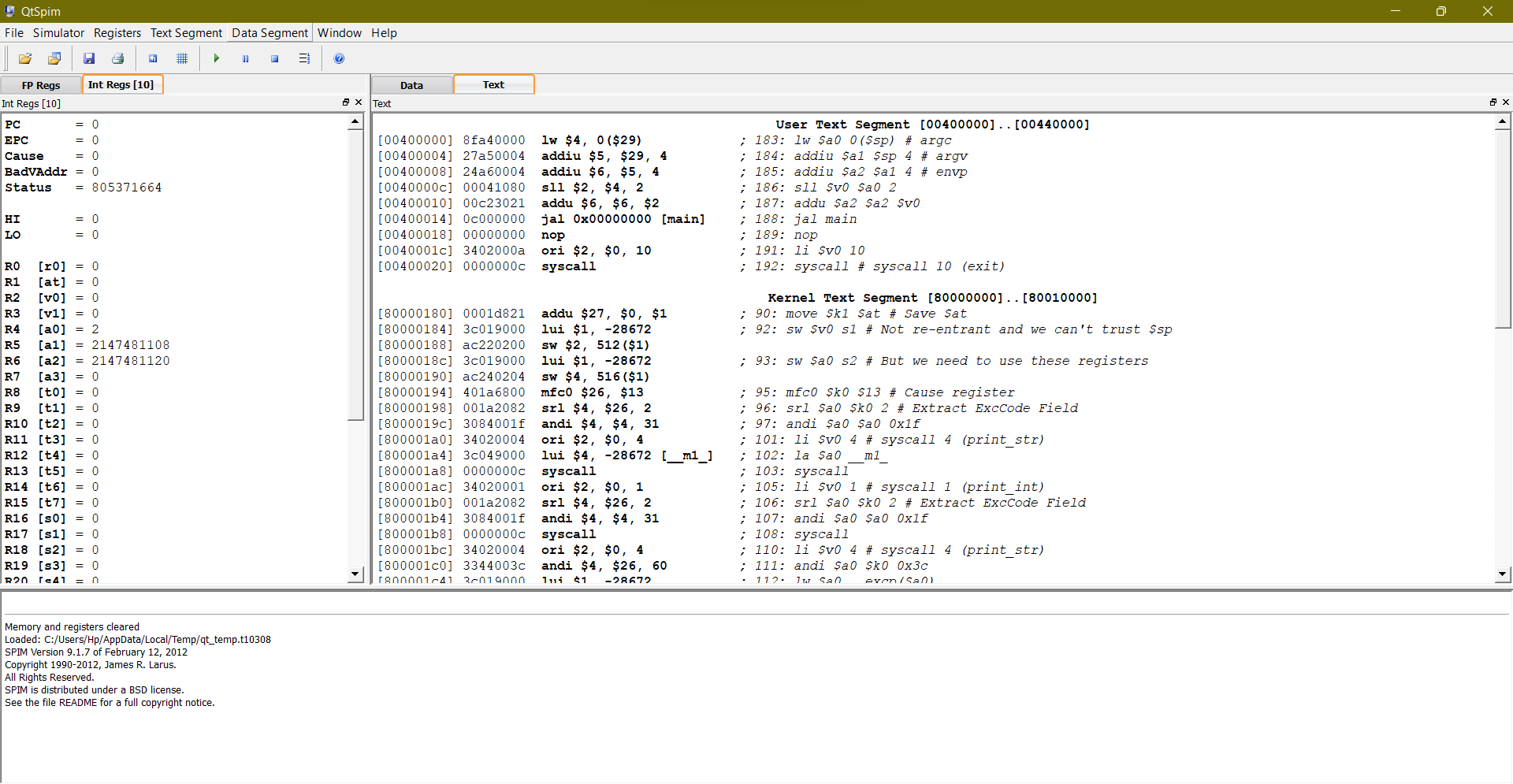
This window provides a view of the entire memory, including both the data and text segments. It allows the user to examine the contents of memory as data or instructions.

**Disassembly:**

This window displays a disassembled view of the MIPS code in memory, showing the corresponding assembly instructions for each memory address.

**Simulator Log:**

This window displays log messages generated by the SPIM simulator, including information about the execution of instructions, system calls, and other events.



**INTRO TO MIPS ASSEMBLY:**

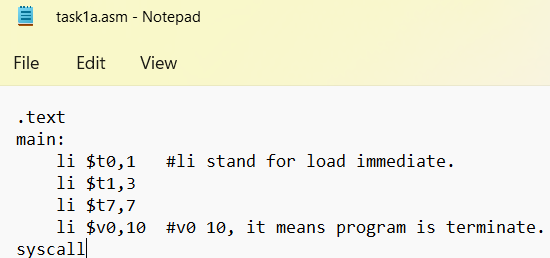
**COMMENTS:**

In MIPS assembly language, comments are used to provide human-readable annotations or explanations within the code.

**Task 1(a):**

* Create a program that simply store data in registers.

**Source code:**



**Output:**



**Task 1(b):**

* **Create a program that take two integers and then find their sum.**

**Source code:**

.text

main:

li $v0, 5

syscall

move $t0, $v0

li $v0, 5

syscall

move $t1, $v0

add $t2, $t1, $t0

move $a0,$t2

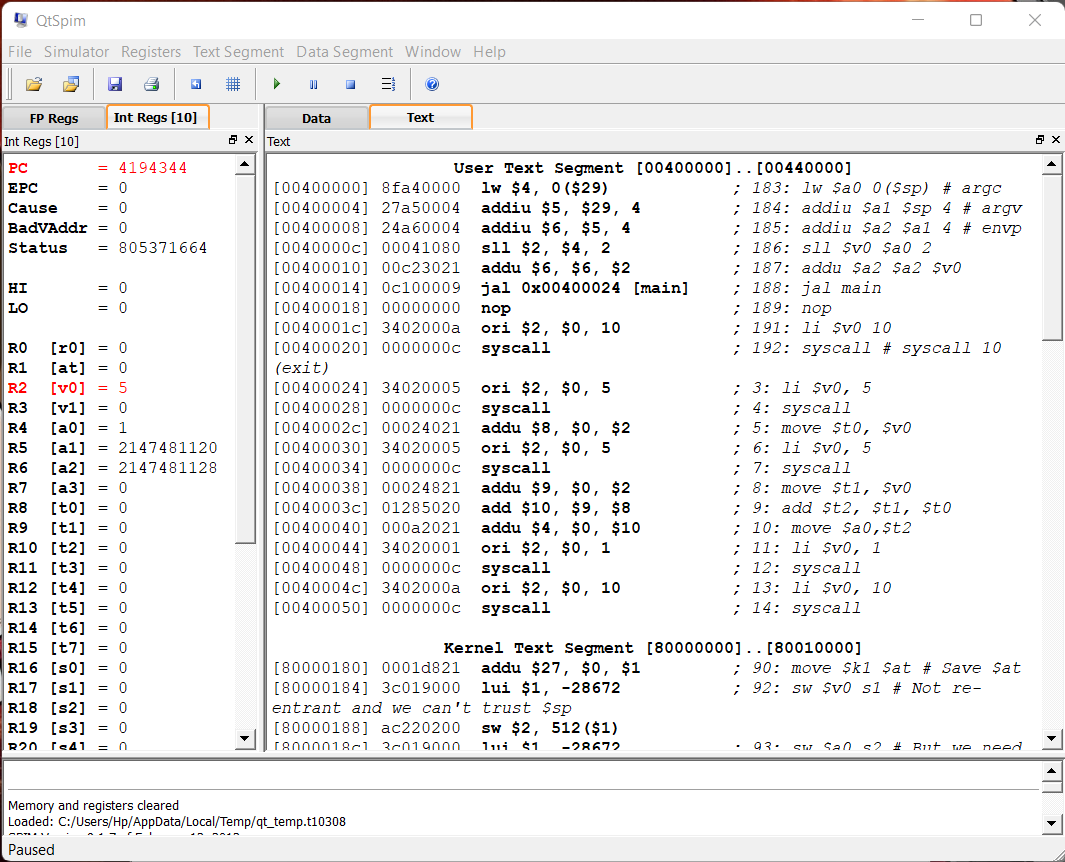
li $v0, 1

syscall

li $v0, 10

syscall

**Output:**



**Task 2:**

* **Create a program that take four integers and then find their sum.**

**Source code:**

.text

main:

    li $t6,4

    li $t5,3

    li $t4,2

    li $t3,1

    add $t8,$t6,$t5

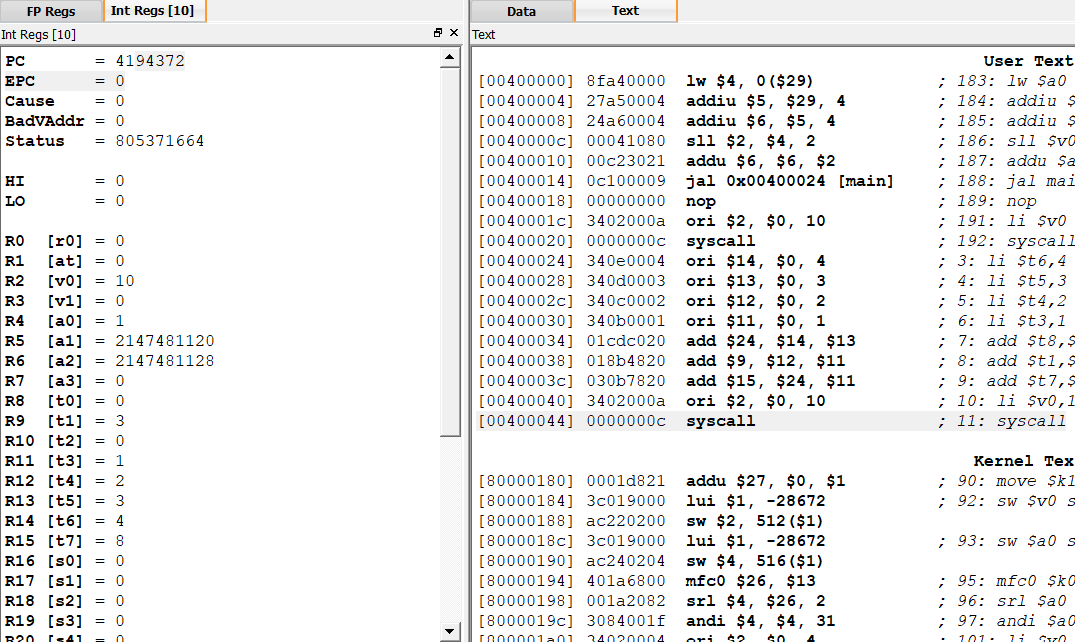
    add $t1,$t4,$t3

    add $t7,$t8,$t3

    li  $v0,10

    syscall

**Output:**



**Task 3:**

* **Create a program that take two integers from the user during run time and then find their sum.**

**Source code:**

.text

main:

li $v0, 5

syscall

move $t0, $v0

li $v0, 5

syscall

move $t1, $v0

add $t2, $t1, $t0

move $a0,$t2

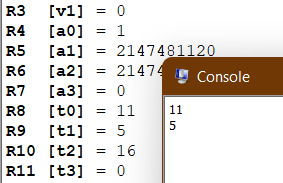
li $v0, 1

syscall

li $v0, 10

syscall

**Output:**



**Task 4:**

* **Create a program that take two integers from the user during run time and then find their difference.**

**Source code:**

.text

main:

li $v0,5

syscall

move $t0,$v0

li $v0,5

syscall

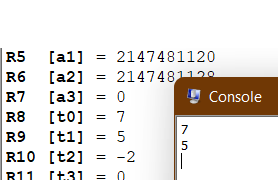
move $t1,$v0

sub $t2,$t1,$t0

li $v0,10

syscall

**Output:**



**Task 5:**

* **Create a program that take five integers from the user and perform the following operation. The output must display in console window**
* **S7=S6+S4-S5-S2+S0**

**Source code:**

.text

main:

li $v0,5

syscall

move $t0,$v0

li $v0,5

syscall

move $t2,$v0

li $v0,5

syscall

move $t5,$v0

li $v0,5

syscall

move $t4,$v0

li $v0,5

syscall

move $t6,$v0

add $s1,$t6,$t4

add $s2,$s1,$t2

add $s3,$t5,$t0

sub $s7,$s2,$s3

move $a0,$s7

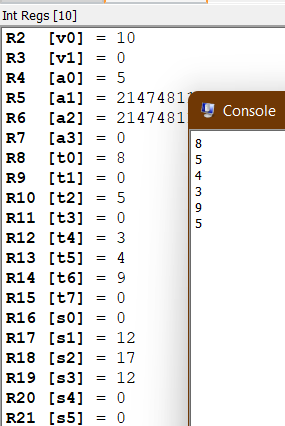
li $v0,1

syscall

li $v0,10

syscall

**Output:**

****