**LAB REPORT NO 1**

****

**Computer organization and Architecture (CSE 304L)**

**Fall 2024**

**Submitted By: Naveed Ahmad**

**Registration No: 22PWCSE2165**

**Section: B**

Date: 02/12/2024

**Submitted To:**

**Dr. Ammad Khalil**

**Department of Computer Systems Engineering**

**University of Engineering and Technology**

**COMPUTER ORGANIZATION AND ARCHITECTURE:**

**COMPUTER ORGANIZATION:**

Computer organisation refers to the operational unit and their interconnection that realise architectural specifications. It deals with the physical and logical arrangement of the hardware. Computer organization focus on the implementation of the hardware and physical system operations.

**EXAMPLE:**

* Timer.
* Interrupt.
* Control signal.
* Ram size.
* I/O devices.

**COMPUTER ARCHITECTURES:**

Computer architecture refers to the abstract structure and the behaviour of the computer as seen by the programmer. Computer architecture focuses on the software interaction with the hardware and what functionally the hardware provides. Computer architecture involves the design and functional behaviour of the system mainly from the programmer point of view.

**EXAMPLE:**

* Instruction set design
* Memory addressing
* Instruction execution
* CPU functionality and capability

**NOTE:**

In simple term computer arc define what computer does while computer organization explain how it does it.

**TYPES OF ARCHITECTURE:**

There are two types of architecture.

**1. CISC (Complex interaction set computing):**

Cisc provide a rich set of instructions allowing the processor to perform complex operations in a single instruction the goal is to reduce the number of instructions per programme by making each instruction more capable. Instructions are more complex often performing multiple operations in a single instruction. Variable instruction length. cisc access memory(RAM) directly.

**EXAMPLE:**

C = a\*b/d%m-g/k\*m-s/a\*b+c

**2. RISC ( Reduce instruction set computing):**

Risc simplify the interaction set of cpu making each instruction execute in a single clock cycle. Risc allow highly pipelining. Risc works on load and store process from (RAM) by the register.

**EXAMPLE:**

C = a\*b

C = C/b

C = C%m

C = C-g

**COMPARISON OF RISC AND CISC ARCHITECTURE MODEL USING TABLE:**

|  |  |
| --- | --- |
| **RICS** | **CISC** |
| 1. Fever simple instruction. | 1. Many and complex. |
| 1. Fixed instruction length. | 2. Variable instruction length. |
| 1. Each instruction exe one clock cycle. | 3. Instruction take multiple clock. |
| 1. Load and store Archi. | 4. Direct memory access with most instruction.. |
| 1. Highly optimize for pipelining. | 5. More difficult to pipelining due to complex city. |
| 6. Simple hardware with few transistor. | 6. More complex design 7. |
| 1. MIPS arm power pc. | 7. Intel. |

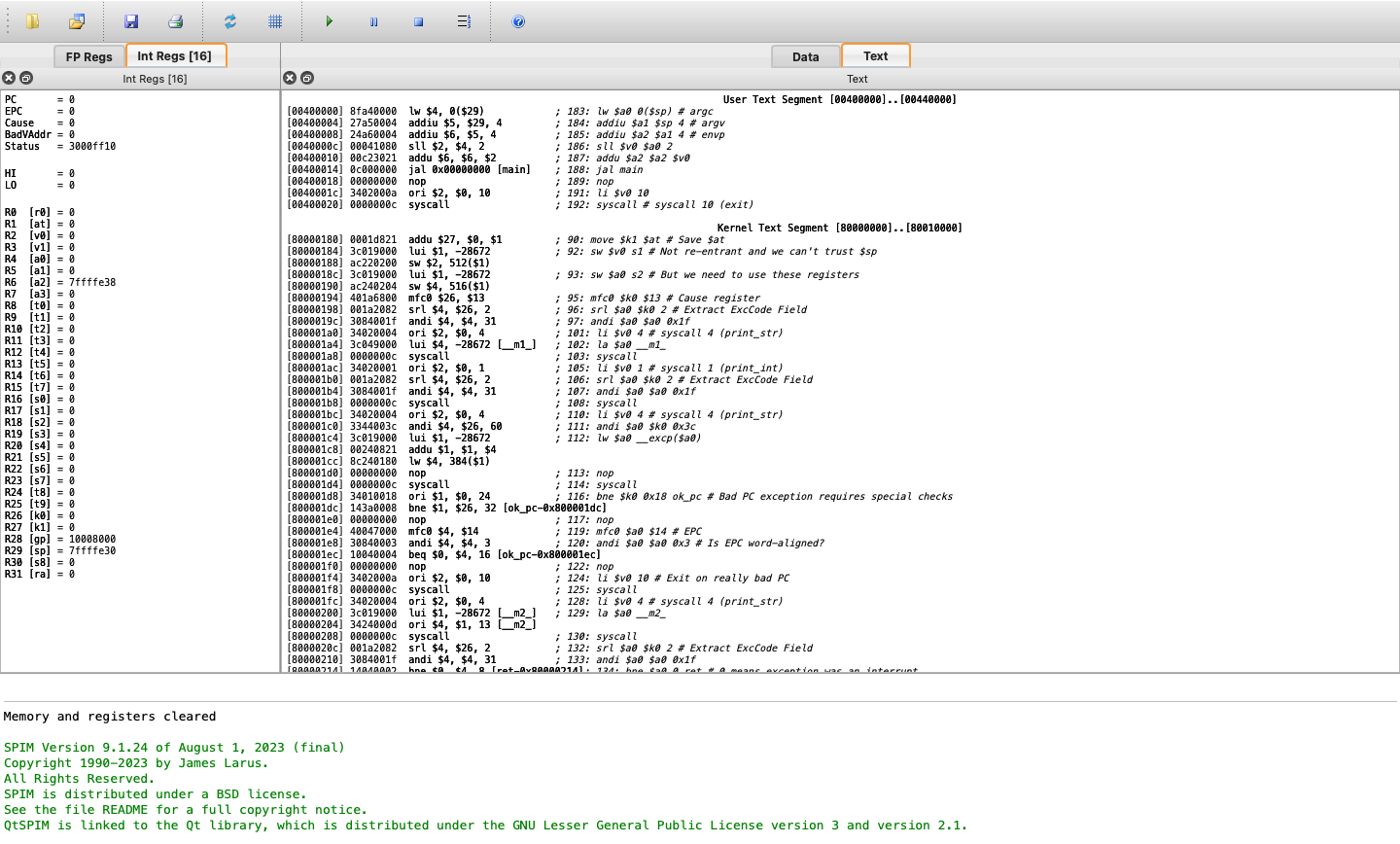
**MIPS ARCHITECTURE:**

Mips follow the risc design principle. Mips computers are developed in early 1980’s . Mips processor are famous for simplicity, efficiency and pipeline design. They are widely used in embedded systems and high-performance applications.

**QTSPIM SIMULATOR:**

Qtspim simulator is used for the mips architecture. Qt refers to the qt framework that is used for graphical user interface (GUI). Spim is the reverse version of mips.

**QTSPIM WINDOW PIC:**



**DATA SEGMENT**:

This section is not fully visible in the screenshot but typically holds the **data** the program uses, such as variables and constants. It includes static or global variables and memory used during program execution.

**TEXT SEGMENT (RIGHT SIDE):**

This section contains the **program instructions**, or the MIPS assembly code, that the CPU

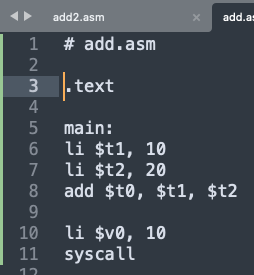
will execute. It’s the part where the logic of the program is defined (e.g., addiu, lw, syscall).

**TASKS:**

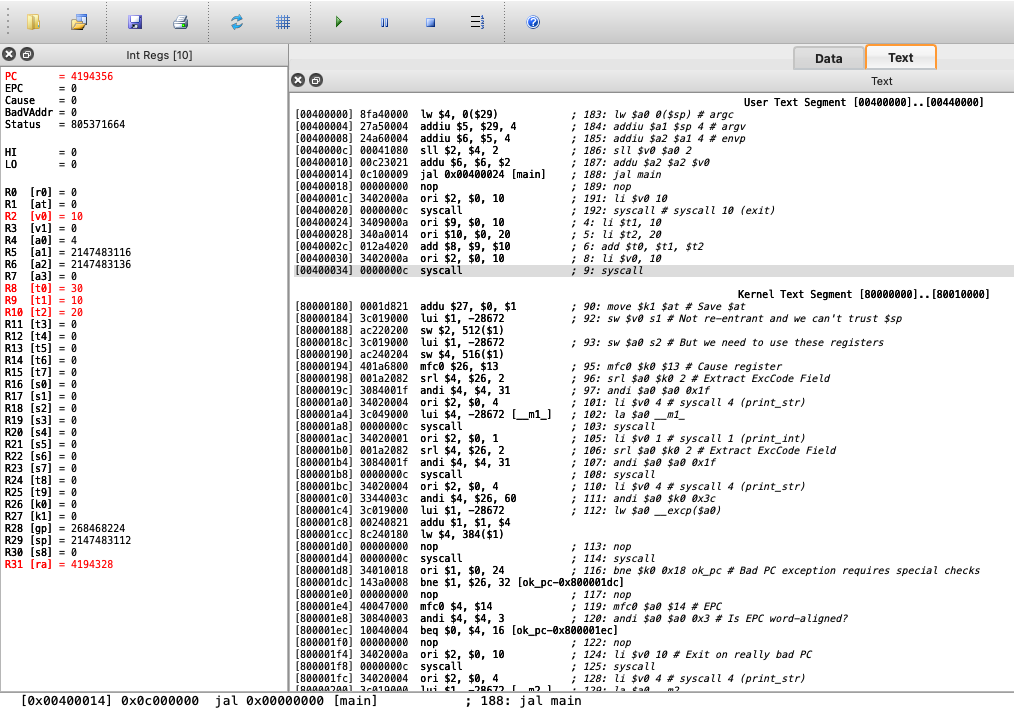
**TASK 1**:

Load two number is $t1 and $t2 and add them, store the add data in $t0.

**CODE PIC:**

****

**OUTPUT PIC:**

****

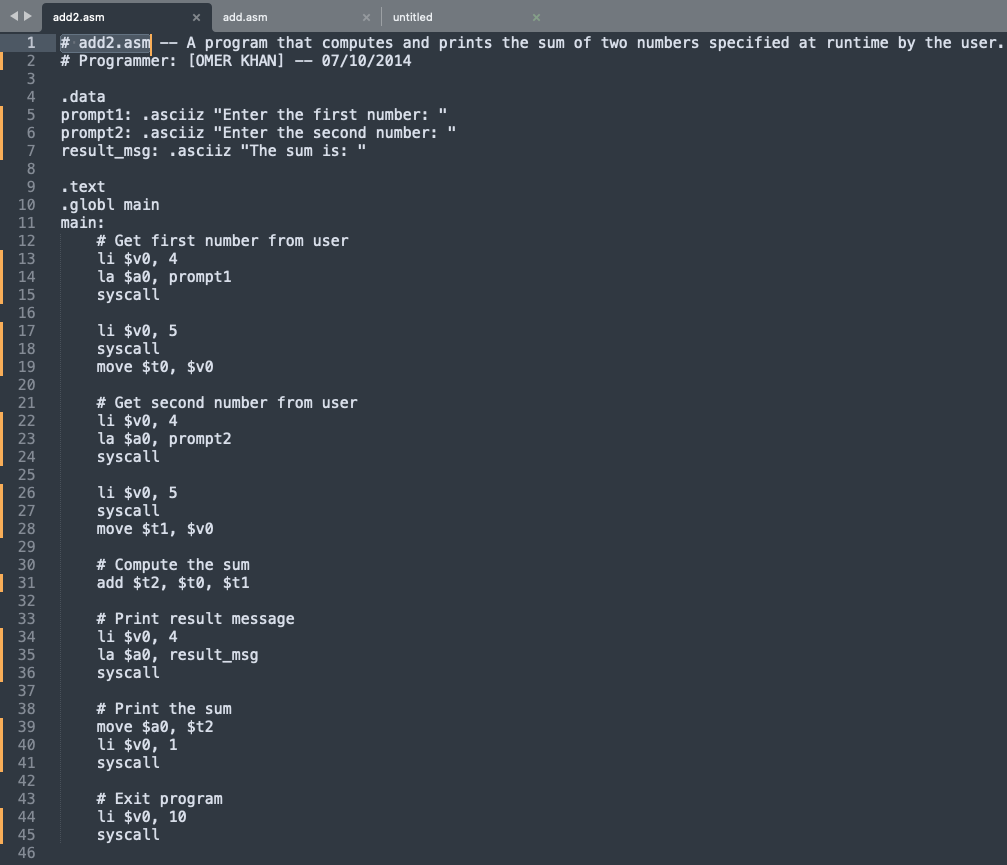
**REMARKS:**

In this code I am simply loading 10 in $t1 and 20 in $t2 and adding them. After this I am storing the add in $t0.

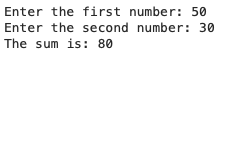
**TASK 2**:

Compute the sum of two numbers specified by the user at runtime, and displays the result on the screen.

**CODE PIC:**

****

**OUTPUT PIC:**

****

**REMARKS:**

* In this code I use string print command **(li $v0, 4).**
* Similarly I use integer input command **(li $v0, 5).**
* Add the two number and store them into **$t2 (add $t2, $t0, $t1 ).**
* I also used integer print command **(li $v0, 1).**