CSE301 – Computer Organization

Lab 1: Introduction to MIPS Assembly Language

What is MIPS?

MIPS (**Microprocessor without Interlocked Pipeline Stages**) is a **RISC (Reduced Instruction Set Computer**) architecture widely used in education for teaching computer organization and assembly programming.

Unlike high-level languages, in assembly we do not have abstractions such as:

- Classes
- Functions (in the high-level sense)
- Variables with arbitrary names or types

Everything must be built manually using registers, memory, and instructions.

Our focus in this lab is on MIPS R2000, which contains:

- Registers small storage locations inside the CPU
- Memory for storing program data and instructions
- **Operations** instructions to perform arithmetic, logic, and control
- System calls requests to the operating system, mostly for I/O

MIPS Registers

MIPS has 32 general-purpose 32-bit registers. Registers are like variables, except:

- 1. Limited in number
- 2. No arbitrary names (must use the provided mnemonics)
- 3. No explicit data types

Register	Number	Usage Note
\$zero	0	Always 0 (cannot be overwritten)
\$at	1	Assembler temporary (used internally)
\$v0-\$v1	2-3	Function results / syscall values
\$a0-\$a3	4-7	Function arguments
\$t0-\$t9	8-15, 24-25	Temporary registers (caller-saved)
\$s0-\$s7	16-23	Saved registers (callee-saved)
\$k0-\$k1	26-27	Reserved for OS kernel
\$gp	28	Global pointer

\$sp	29	Stack pointer
\$fp	30	Frame pointer
\$ra	31	Return address for function calls

il Info:

\$zero cannot be overwritten; it is hardwired to 0

Register Usage Categories

Category	Registers	Usage
Generic / Temporary	\$t0-\$t9	Temporary values, not preserved across function calls
Saved / Function	\$s0-\$s7	Must preserve across function calls (callee-saved)
Function / Arguments	\$a0-\$a3	Pass arguments to functions
Function / Return Values	\$v0-\$v1	Store function results or syscall return values
Memory / Stack	\$sp, \$fp	Stack pointer and frame pointer for function calls
Reserved / Kernel	\$k0-\$k1	Reserved for OS kernel usage
Other	\$ra	Store return address after function calls

Info:

Difference between temporary and saved registers:

Temporary (\$t0-\$t9): Caller must save if needed.

Saved (\$s0-\$s7):Callee must preserve values across function calls.

QTSPIM Simulator

QTSPIM is a graphical simulator for MIPS programs. It allows:

- Writing MIPS assembly
- Running instructions step by step
- Observing registers and memory

System Calls

System calls (syscall) allow your MIPS program to **request services from the operating system**, mainly for I/O operations.

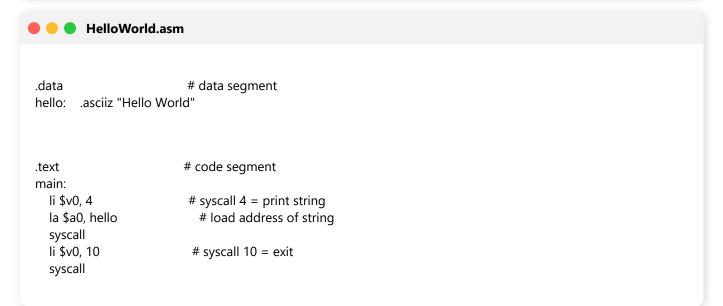
Procedure:

1. \$v0 – specify **operation number**

- 2. \$a0 specify **operation parameter** (e.g., integer or address of string)
- 3. Execute syscall
- 4. \$v0 may return result (for input operations)

Common Operation Numbers:

\$v0	Operation	Description
1	Print integer	Prints the 32-bit integer in \$a0
4	Print string	Prints the null-terminated string at address \$a0
5	Read integer	Reads a 32-bit integer from the user, returns in \$v0
10	Exit	Terminates the program



Info:

The code is divided into two segments:

- **Data segment:** stores data in memory.
- **Text segment:** contains the code.

Task:

Task 1: Create a repository to contain all your work. Name it CSE321-Computer-Organization with the following directory structure:



Name: Your Name
ID: Your ID

Course: CSE321-Computer-Organization