Experiment No 5

RISC-V assembly programming using Logic and Jump instructions in venus

5.1 Objectives:

After completing this experiment, student will be able:

- To know the RISC-V architecture and its significance in contemporary computer systems by providing an overview of its simple yet powerful design principles.
- To know fundamental concepts in RISC-V assembly programming, focusing specifically on logic and jump instructions, which are essential for controlling program flow and performing bitwise operations.
- To gain practical experience in using logic instructions such as AND, OR, XOR, and NOT to manipulate data at the bitwise level, thereby enhancing their understanding of low-level programming concepts.
- Hands-on learning through the utilization of Venus, a RISC-V simulator, allowing students to experiment with logic and jump instructions in a simulated environment, reinforcing theoretical knowledge with practical application and promoting deeper comprehension of computer architecture principles.

5.2 Background Theory

Understanding RISC-V assembly programming is foundational for comprehending computer architecture and low-level system programming. Logic and jump instructions play crucial roles in controlling program flow and performing bitwise operations in RISC-V assembly. Logic instructions such as AND, OR, XOR, and NOT facilitate bitwise manipulation of data, while jump instructions like unconditional jumps and conditional branches enable the execution of code

Conditional branch	Branch if equal	`beq x5, x6, 100`	`if (x5 == x6) go to PC+100`	PC-relative branch if registers equal
	Branch if not equal	`bne x5, x6, 100`	`if (x5 != x6) go to PC+100`	PC-relative branch if registers not equal
	Branch if less than	`blt x5, x6, 100`	`if (x5 < x6) go to PC+100`	PC-relative branch if registers less
	Branch if greater or equal	`bge x5, x6, 100`	`if (x5 >= x6) go to PC+100`	PC-relative branch if registers greater or equal
	Branch if less, unsigned	`bltu x5, x6, 100`	`if (x5 < x6) go to PC+100`	PC-relative branch if registers less, unsigned
	Branch if greater or equal, unsigned	`bgeu x5, x6, 100`	`if (x5 >= x6) go to PC+100`	PC-relative branch if registers greater or equal, unsigned
Unconditional branch	Jump and link	`jal x1, 100`	`x1 = PC+4; go to PC+100`	PC-relative procedure call
	Jump and link register	`jalr x1, 100(x5)`	`x1 = PC+4; go to x5+100`	Procedure return; indirect call

Figure 5.1: Conditional and Unconditional jumps/branch

segments based on specific conditions. Venus, a popular RISC-V simulator, provides a conducive environment for learning and experimenting with RISC-V assembly programming concepts,

allowing students and researchers to gain practical insights into computer organization and programming paradigms. This lab aims to provide hands-on experience with logic and jump instructions in RISC-V assembly using Venus, thereby fostering a deeper understanding of fundamental concepts in computer architecture and assembly programming.

5.2.1 Overview of Logical Instructions

In RISC-V assembly programming, logical instructions perform bitwise operations on binary data. These instructions include AND, OR, XOR, and NOT. Below, I'll explain each type of logical instruction with proper examples in RISC-V assembly programming using the Venus simulator:

5.2.1.1 AND (Bitwise AND)

The AND instruction performs a bitwise AND operation between corresponding bits of two operands.

```
# RISC-V Assembly code for bitwise AND operation

# Initialize registers
li a0, 5     # Load immediate value 5 into register a0 (101 in binary)
li a1, 3     # Load immediate value 3 into register a1 (011 in binary)

# Perform AND operation
and a2, a0, a1  # Perform bitwise AND of a0 and a1, store result in a2

# a2 will contain the result: 101 AND 011 = 001 (decimal 1)
```

Figure 5.2: AND Logic

5.2.1.2 OR (Bitwise OR)

The OR instruction performs a bitwise OR operation between corresponding bits of two operands.

```
# RISC-V Assembly code for bitwise OR operation

# Initialize registers
li a0, 5     # Load immediate value 5 into register a0 (101 in binary)
li a1, 3     # Load immediate value 3 into register a1 (011 in binary)

# Perform OR operation
or a2, a0, a1  # Perform bitwise OR of a0 and a1, store result in a2

# a2 will contain the result: 101 OR 011 = 111 (decimal 7)
```

Figure 5.3: OR Logic

5.2.1.3 XOR (Bitwise XOR)

The XOR instruction performs a bitwise XOR (exclusive OR) operation between corresponding bits of two operands.

```
# RISC-V Assembly code for bitwise XOR operation

# Initialize registers
li a0, 5     # Load immediate value 5 into register a0 (101 in binary)
li a1, 3     # Load immediate value 3 into register a1 (011 in binary)

# Perform XOR operation
xor a2, a0, a1  # Perform bitwise XOR of a0 and a1, store result in a2

# a2 will contain the result: 101 XOR 011 = 110 (decimal 6)
```

Figure 5.4: XOR Logic

5.2.1.4 NOT (Bitwise NOT)

The NOT instruction performs a bitwise negation (complement) operation on a single operand.

```
# RISC-V Assembly code for bitwise NOT operation

# Initialize registers
li a0, 5     # Load immediate value 5 into register a0 (101 in binary)

# Perform NOT operation
not a1, a0  # Perform bitwise NOT of a0, store result in a1

# a1 will contain the result: NOT 101 = 010 (decimal 2)
```

Figure 5.5: NOT Logic

5.2.2 Overview of Jump and Branch Instructions

In RISC-V assembly programming, jump and branch instructions are used to control the flow of execution within a program. Let's explain each type of jump and branch instruction separately with examples in RISC-V assembly programming using Venus.

5.2.2.1 Unconditional Jump

Unconditional jump instructions cause the control flow to transfer to a specific target address unconditionally.

Example:

```
# RISC-V Assembly code for unconditional jump

# Jump to label 'target'
j target

# Some code that will be skipped due to the jump
li a1, 42

target:
# Code continues here
li a0, 10     # Load immediate value 10 into register a0

# Exit
li a7, 10     # Load immediate value 10 (exit syscall number)
ecall         # Perform exit syscall
```

Figure 5.6: Unconditional Jump

5.2.2.2 Conditional Branch

Conditional branch instructions allow the program to change the flow of execution based on specified conditions.

Example:

```
# Initialize registers
li a0, 5     # Load immediate value 5 into register a0
li a1, 3     # Load immediate value 3 into register a1

# Compare values
bge a0, a1, greater_equal  # Branch if a0 >= a1

# If a0 < a1
li a0, 1     # Load immediate value 1 into register a0
j end        # Jump to end

greater_equal:
li a0, 0     # Load immediate value 0 into register a0

end:
# Code continues here

# Exit
li a7, 10    # Load immediate value 10 (exit syscall number)
ecall        # Perform exit syscall
```

Figure 5.7: Conditional Branch

5.2.2.3 Jump and Link (jal)

The **jal** instruction jumps to a target address and saves the address of the instruction following the jump (return address) into a specified register.

Example:

```
# RISC-V Assembly code for jump and link

# Jump to label 'target' and save return address in ra
jal ra, target

# Some code that will be skipped due to the jump
li a1, 42

target:
# Code continues here
li a0, 10     # Load immediate value 10 into register a0

# Exit
li a7, 10     # Load immediate value 10 (exit syscall number)
ecall         # Perform exit syscall
```

Figure 5.8: Jump and Link (jal)

5.2.2.4 Jump Register (jalr)

The **jalr** instruction jumps to an address stored in a register and saves the address of the instruction following the jump (return address) into a specified register.

Example:

```
# RISC-V Assembly code for jump register

# Load target address into t0
lui t0, %hi(target)
addi t0, t0, %lo(target)

# Jump to address in t0 and save return address in ra
jalr ra, t0, 0

# Some code that will be skipped due to the jump
li a1, 42

target:
# Code continues here
li a0, 10     # Load immediate value 10 into register a0

# Exit
li a7, 10     # Load immediate value 10 (exit syscall number)
ecall         # Perform exit syscall
```

Figure 5.9: Jump Register

These examples demonstrate the usage of various jump and branch instructions in RISC-V assembly programming using the Venus simulator. Each instruction type serves specific purposes in controlling program flow and enabling conditional execution based on runtime conditions.

5.3 Lab Activities

5.3.2 Task-1: Write a RISC V Assembly code to implement and verify the logical operations of OR, AND, XOR and NOT:

5.3.3 Task-2: Write a RISC-V assembly code that incorporates unconditional jump instructions and observe their effects on program execution.

1) Write a RISC-V assembly program that demonstrates the usage of the unconditional jump instruction (**j**).

2) Include at least two sections of code separated by an unconditional jump instruction. The first section should contain some initialization or computation, and the second section should be the target of the jump instruction.

3) Execute the program in the simulator and observe the flow of execution. Pay attention to how the control flow transfers to the target section upon encountering the unconditional jump instruction.

4) Modify the program to include multiple unconditional jump instructions and observe how each jump affects the flow of execution.

5.3.4 Task-3: Write a RISC-V assembly code that implements conditional instructions to achieve specific tasks. For example:

1) Implement a program that compares two numbers and prints a message indicating which number is greater.
2) Write a program that checks if a given value is positive, negative, or zero and outputs the result accordingly.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature: _____ **Date: _____**

Cognitive Activity

Q1. Trace this snippet by hand, given t2 starts at 0:

```
li t0, 4
li t1, 4
beq t0, t1, label
addi t2, t2, 1
label:
addi t2, t2, 5
```

What is the final value of t2, and why is the 'addi t2,t2,1' line skipped or not skipped? **(5)**

Ans: _____

_____

_____

_____

_____

Q2. Using AND, OR/branch-type instructions, describe (in RISC-V pseudo-code or plain steps) how you would check whether the value in register t0 is even or odd, and store 1 in t1 if even or 0 if odd. Explain your logic. **(5)**

Ans: _____

_____

_____

_____

_____

Obtained Marks	**/10**

Psychomotor Activity 1 — Experimental Setup

S. No	Criteria	Performance	Description	Marks
1	Tool Setup (4)	Level 4	Opens and configures Venus and selects the required environment correctly and independently.	4
		Level 3	Configures the required tool with minor errors or assistance.	3
		Level 2	Completes the setup with considerable assistance.	2
		Level 1	Setup is incomplete and requires substantial assistance.	1
2	File/Design Setup (4)	Level 4	Correctly creates/selects the required assembly file, Logisim circuit/design and required components for the RISC-V assembly program.	4
		Level 3	File/design setup is mostly correct with minor errors or omissions.	3
		Level 2	Setup is partially completed and requires assistance.	2
		Level 1	Major errors or omissions are present in the setup.	1
3	Parameter Setup (2)	Level 2	Correctly defines required data, registers, parameters, components and initial conditions.	2
		Level 1	Most required inputs/components are correctly configured with minor errors.	1
			Total Obtained	/10

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	0

Psychomotor Activity 2 — Implementation + Results

S. No	Criteria	Performance	Description	Marks
1	**Implementation (4)**	Level 4	Implements the required RISC-V assembly program correctly and independently.	**4**
		Level 3	Implements the required program/design with minor errors or assistance.	**3**
		Level 2	Partially implements the program/design and requires considerable assistance.	**2**
		Level 1	Implementation is largely incomplete and requires substantial assistance.	**1**
2	**Execution/Simulation (2)**	Level 2	Executes the RISC-V program or simulates the Logisim design correctly and handles basic errors appropriately.	**2**
		Level 1	Executes/simulates the work with minor errors or assistance.	**1**
3	**Results/Output (4)**	Level 4	Obtains correct results and clearly demonstrates the required program output or circuit behavior.	**4**
		Level 3	Obtains mostly correct results with minor errors.	**3**
		Level 2	Obtains partially correct results with noticeable errors or omissions.	**2**
		Level 1	Results are largely incorrect or incomplete.	**1**
			Total Obtained	**/10**

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	**0**

Affective Activity — Lab Report + Viva Voce

S. No	Criteria	Performance	Description	Marks
1	Lab Report Completeness & Organization (4)	Level 4	Report is complete, well organized and demonstrates full responsibility for the assigned work.	4
		Level 3	Report is complete with minor omissions or organizational issues and demonstrates satisfactory responsibility for the assigned work.	3
		Level 2	Report is partially complete with several omissions and demonstrates limited responsibility for the assigned work.	2
		Level 1	Report has major omissions or poor organization and demonstrates minimal responsibility for the assigned work.	1
2	Code/Design, Results & Conclusion (2)	Level 2	RISC-V assembly program, results, and conclusion are correctly presented and discussed, demonstrating ownership of the assigned work.	2
		Level 1	RISC-V assembly program, results, or conclusion are presented with minor errors or omissions.	1
3	Viva Voce (4)	Level 4	Clearly explains the assigned RISC-V assembly program, working, and results, demonstrating individual responsibility, participation, and effective communication.	4
		Level 3	Explains most of the assigned program/design and results correctly, with minor difficulty, and demonstrates good individual participation.	3
		Level 2	Explains basic aspects of the assigned program/design with some prompting and demonstrates basic participation.	2
		Level 1	Demonstrates limited understanding or involvement and requires substantial prompting to explain the assigned work.	1
			Total Obtained	/10

The student has not submitted the report and has not appeared in the Viva Voce.

Criteria	Description	Marks
Lab Report and Results Not Submitted	Student has not submitted the report and results file.	**0**
Absent in Viva Voce	Student is absent from the viva voce.	**0**

Instructor's Signature: _____ **Date:** _____