# Full FPGA Implementation of 32-bit MIPS Processor
# Project Proposal - Team 22

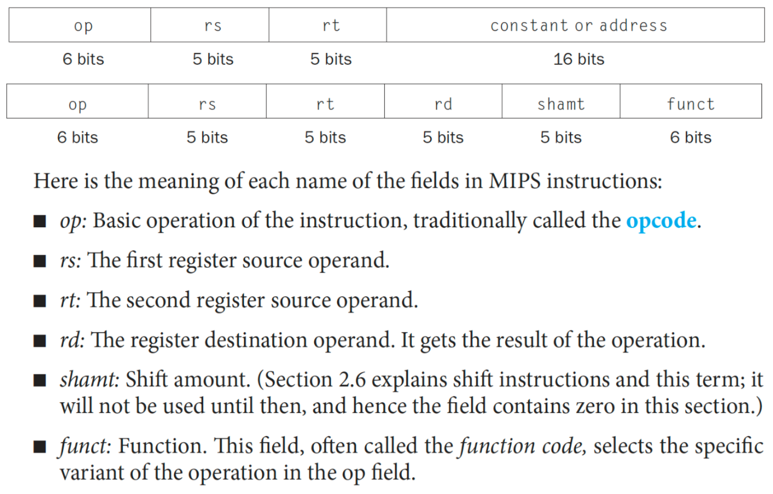
**Guntas Singh Saran** (22110089) | **Hrriday V. Ruparel** (22110099)
**Kishan Ved** (22110122) | **Pranav Patil** (22110199)
Advisor: **Prof. Sameer G Kulkarni** | Indian Institute of Technology Gandhinagar

## 1 Introduction

This project focuses on the design and implementation of a 32-bit MIPS (Microprocessor without Interlocked Pipeline Stages) processor on a Basys FPGA board. MIPS is a RISC (Reduced Instruction Set Computing) architecture known for its simplicity and efficiency, making it a suitable choice for FPGA implementation. The objective is to create a functional processor capable of executing a subset of the MIPS instruction set, with the program and data stored in FPGA's Block RAM (BRAM). The output will be displayed on a 7-segment LED display using memory-mapped I/O.

## 2 MIPS Instruction Set

The instruction set in the MIPS processor design is grouped into 3 types. They are Instruction of type Register, Instruction of type Immediate and Instruction of type Jump. In all these arrangement of instructions the Opcode is of 6-bits, which is used to choose the type of operation in the ALU block.



| op | rs | rt | constant or address |
|---|---|---|---|
| 6 bits | 5 bits | 5 bits | 16 bits |

| op | rs | rt | rd | shamt | funct |
|---|---|---|---|---|---|
| 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |

Here is the meaning of each name of the fields in MIPS instructions:

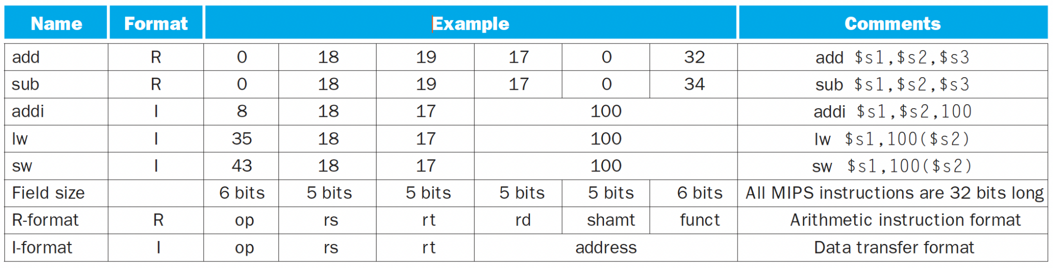
- *op:* Basic operation of the instruction, traditionally called the **opcode**.
- *rs:* The first register source operand.
- *rt:* The second register source operand.
- *rd:* The register destination operand. It gets the result of the operation.
- *shamt:* Shift amount. (Section 2.6 explains shift instructions and this term; it will not be used until then, and hence the field contains zero in this section.)
- *funct:* Function. This field, often called the *function code,* selects the specific variant of the operation in the op field.

Figure 1: R-Type and I-Type Instructions. Adapted from Patterson & Hennessy (1998)

### 2.1 R-Type (Register) Instruction Format

This type of Instruction plays out all the arithmetic and logical operations. The principle utilization of this instruction is playing out the mathematical operations. For example, addition and subtraction.

This type of instruction comprises of a instruction memory and ALU block. The instruction memory has 32 bit address and 32-bit instruction. Out of this 32 bits instruction the higher 6-bits are utilized for Opcode.



| Name | Format | Example | | | | | | Comments |
|---|---|---|---|---|---|---|---|---|
| add | R | 0 | 18 | 19 | 17 | 0 | 32 | add `$s1,$s2,$s3` |
| sub | R | 0 | 18 | 19 | 17 | 0 | 34 | sub `$s1,$s2,$s3` |
| addi | I | 8 | 18 | 17 | 100 | | | addi `$s1,$s2,100` |
| lw | I | 35 | 18 | 17 | 100 | | | lw `$s1,100($s2)` |
| sw | I | 43 | 18 | 17 | 100 | | | sw `$s1,100($s2)` |
| Field size | | 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits | All MIPS instructions are 32 bits long |
| R-format | R | op | rs | rt | rd | shamt | funct | Arithmetic instruction format |
| I-format | I | op | rs | rt | address | | | Data transfer format |

Figure 2: Examples of Instruction Set. Adapted from Patterson & Hennessy (1998)

### 2.2 I-Type (Immediate) Instruction Format

The Immediate type Instruction is utilized to play out the immediate arithmetic and logical operations. In this kind of instruction 16 bit immediate value is utilized for playing out all the operations. By utilizing this type of instruction we can plays out the load, store and branch operations.

### 2.3 J-Type (Jump) Instruction Format

Jump Type instruction is the most part used to play out the jump instruction. In this instruction the Opcode is of 6-bit is used to pick the type of instruction. The target address is of 26-bits instruction is utilized to figure out where the location must be fanned. It is utilized just when the result of the ALU block is zero.

## 3 Task List

The project will be divided into the following tasks:

1. **Instruction Set Implementation:**
   - Implement the 32-bit MIPS processor, supporting a subset of the MIPS instruction set.
   - ***Important Change***: Since we will be using BASYS 3 FPGA board, we shall restrict the memory size as per the board constraints.
2. **Pipeline Design:**
   - Design and implement a 5-stage pipeline for instruction execution: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB).
3. **Memory Management:**
   - Integrate Block RAM (BRAM) for program and data storage.
   - Develop a BRAM initialization mechanism during FPGA configuration. (Potentially use UART for initializing the program in BRAM).
4. **Output Handling:**
   - Implement memory-mapped I/O for interfacing with the 7-segment display.
   - If time permits, display the output on a monitor using VGA port.
5. **Simulation and Verification:**
   - Test and verify the processor's functionality using simulation tools.
   - Write benchmark programs using assembly code and convert them into MIPS understandable binary executable code for running on the FGPA board.

## 4 Work Split

The tasks will be divided among team members as follows:

Table 1: Work split amongst the team

| **Team Members** | **Responsibilities** |
|---|---|
| Kishan and Guntas | Instruction set implementation |
| Hrriday and Pranav | Pipeline design |
| Hrriday and Pranav | Memory management, BRAM initialization |
| Kishan and Guntas | Output handling, memory-mapped I/O |
| Hrriday | Simulation, |
| Guntas | Verification |
| Kishan | Testing |

## 5 Key Milestones / Timeline

The project is expected to achieve the following milestones:

- **Milestone 1:** Conversion of MIPS ISA to BASYS 3 compatible instruction set. ***(1 Week)***
- **Milestone 2:** Completion ALU and pipeline design. ***(4 Weeks)***
- **Milestone 3:** Integration of BRAM, memory management and Assembly code to FPGA program code pipeline. ***(2 Weeks)***
- **Milestone 4:** Successful implementation of memory-mapped I/O. ***(1 Week)***
- **Milestone 5:** Processor simulation and verification. ***(2 Weeks)***
- **Milestone 6:** Final testing and demonstration on the Basys FPGA. ***(2 Weeks)***

## 6 Tools & Technologies Used

The following tools and technologies will be employed in this project:

- **FPGA Platform:** Basys 3 FPGA board.
- **Hardware Description Language:** Verilog for designing the MIPS processor.
- **Simulation Tools:** Vivado for simulating and verifying the design. SPIM MIPS simulator.
- **Synthesis and Implementation:** Xilinx Vivado for FPGA synthesis and implementation.
- **Memory Management:** Block RAM (BRAM) within the FPGA.
- **Output Interface:** Memory-mapped I/O for controlling the 7-segment display and if time permits, VGA enabled monitor.

## References

MIPS Memory Map — wilkinsonj.people.charleston.edu. `https://wilkinsonj.people.charleston.edu/mem-map.html`.

MIPS Instruction Set &x2014; ECS Networking — ecs-network.serv.pacific.edu. `https://ecs-network.serv.pacific.edu/ecpe-170/tutorials/mips-instruction-set`.

Safaa Omran and Ali Ibada. Fpga implementation of mips risc processor for educational purposes. *Journal of Babylon University/Pure and Applied Sciences*, 24:17, 01 2016.

David A. Patterson and John L. Hennessy. *Computer Organization and Design*. Morgan Kaufmann Publishers, 2nd edition, 1998. ISBN 15-586-0428-6.