INTRODUCTION TO MIPS PROCESSOR

VLSI enables the integration of millions of transistors on a single chip, facilitating the **creation of compact, high-speed processors.** MIPS (MIPS stands for Million Instructions Per Second) processors are a widely used example in VLSI design due to their simplified instruction set and efficient architecture.

MIPS PROCESSOR:

A 32-bit single-cycle MIPS processor refers to a simplified version of the MIPS (Microprocessor without Interlocked Pipeline Stages) architecture, designed to execute all instructions within a single clock cycle. The processor operates on 16-bit data and instructions, meaning the width of the data path, registers, and instructions is 32 bits.

Key Features

- **Single-Cycle Architecture:** Each instruction (e.g., arithmetic, load/store, jump) is fetched, decoded, executed, and written back in one clock cycle, making it a simple and fast design.
- **Reduced Instruction Set Computing (RISC):** Implements a small, highly optimized set of instructions, minimizing the complexity of the control logic and improving performance.
- **Data Path:** Includes components like the ALU (Arithmetic Logic Unit), register file, instruction memory, and data memory to execute instructions efficiently.
- **Control Unit:** Generates control signals based on the current instruction to coordinate the operations of the data path elements.

APPLICATIONS

MIPS processors continue to be vital in various industries and also in real life due to their energy efficiency, real-time processing capabilities, and compact design.

EMBEDDED SYSTEMS

Used in smartphones, tablets, and IoT devices for low power and energy-efficient processing, making them ideal for battery-operated devices. excel in handling lightweight OS tasks such as sensors, network management and simple applications.

ELECTRONICS

Used in gaming consoles, digital cameras, and other devices for efficient graphics processing and image handling.

NETWORKING & COMMUNICATION

Powers routers, switches, and set-top boxes for high-speed data handling and real-time processing of media, frequently used in set-top boxes and media streaming devices, integrated into devices like smart sensors, wearables and home automation systems.

AUTOMOTIVE SYSTEMS

MIPS processors are utilized in automotive control systems, such as Advanced Driver Assistance Systems (ADAS), engine control units, and infotainment systems.

TYPES OF MIPS PROCESSOR

- **Single-Cycle MIPS Processor:** Executes every instruction in one clock cycle, offering simplicity but limited performance.
- **Multi-Cycle MIPS Processor:** Breaks instruction execution into multiple clock cycles, reducing hardware requirements.
- **Pipelined MIPS Processor:** Uses a 5-stage pipeline to execute multiple instructions simultaneously for improved throughput.
- **Superscalar MIPS Processor:** Executes multiple instructions in parallel within the same clock cycle for enhanced performance.
- **MIPS64 and MIPS32:** MIPS64 handles 64-bit data and large memory spaces, while MIPS32 is optimized for power-efficient embedded systems.

WHY Single Cycle MIPS Processor?

- Basic CPU components (data path, control unit, ALU) are understood through the single-cycle 16-bit processor.
- Instruction execution (arithmetic, memory access, branching) is efficiently explored.
- Reduced complexity allows focus on key operations: fetching, decoding, executing, memory access.
- The 32-bit design simplifies without losing essential processor functionality and complex computations also.

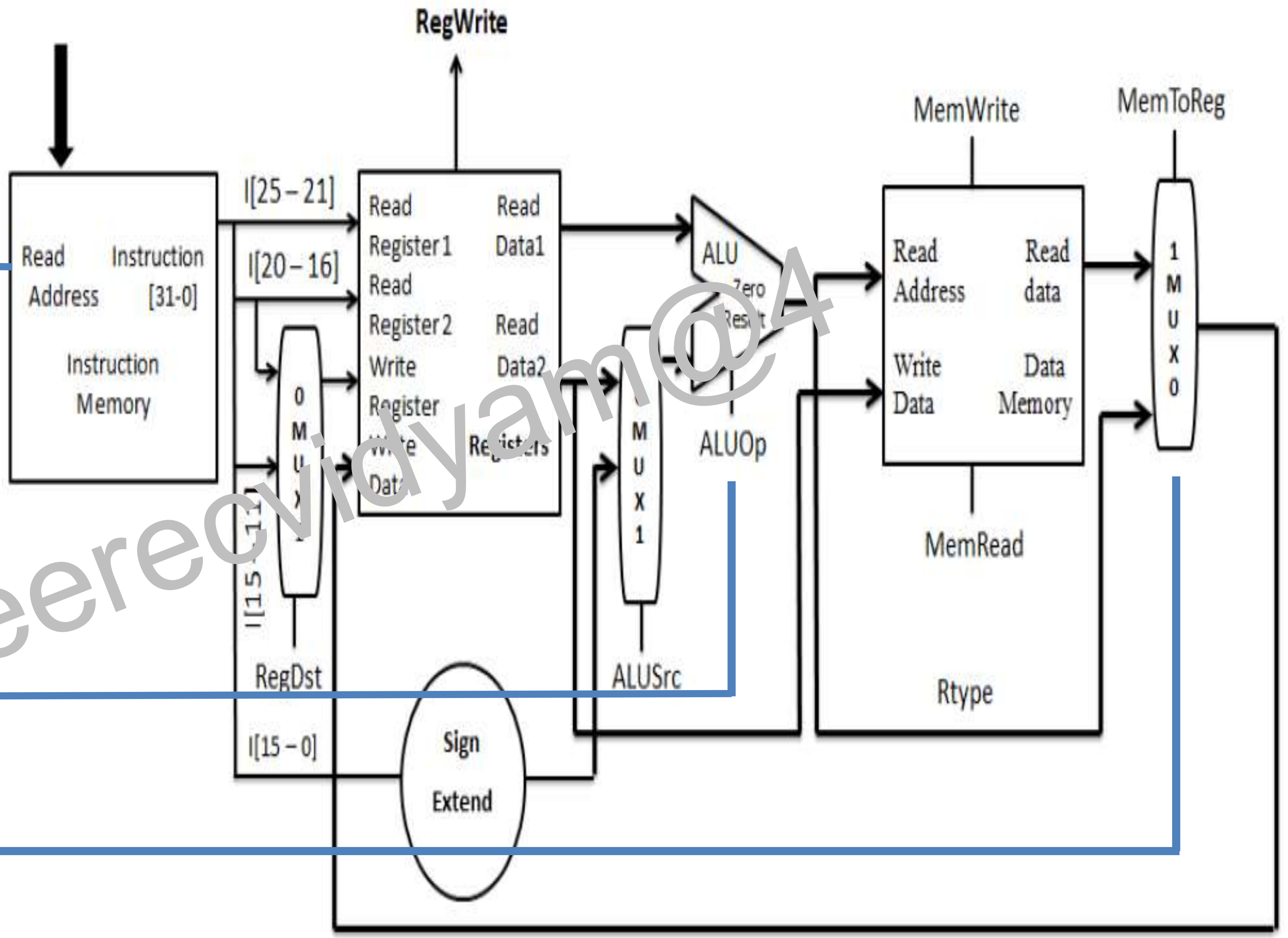
BLOCK DIAGRAM

PHASE 1: **Instruction Fetch and Decode – Implement Program Counter, Instruction Memory, and Control Unit.**

Outline to split work into three phases, integrating the coding and design work ensuring robust MIPS processor design.

PHASE 2: **Execution (ALU and Registers) – Focus on ALU and register file.**

PHASE 3: **Memory Access and Write Back – Implement data memory access and write-back logic.**



WORKFLOW

MILESTONE 1

Based on the provided Instruction Format and Instruction -Set Architecture for the 32-bit single-cycle, the data-path and control unit will be designed and implemented.

MILESTONE 2

Based on the provided instruction set, ALU Control will be designed and implemented . After completing the design for the MIPS processor, it is easy to write Verilog code for the MIPS processor.

MILESTONE 3

After implementation of 32-bit single-cycle, we have a plan of improvisation by the execution of instructions across multiple clock cycles reduces hardware requirements. Then, Moving to a pipelined MIPS processor introduces the concept of executing multiple instructions simultaneously by dividing instruction execution into stages.

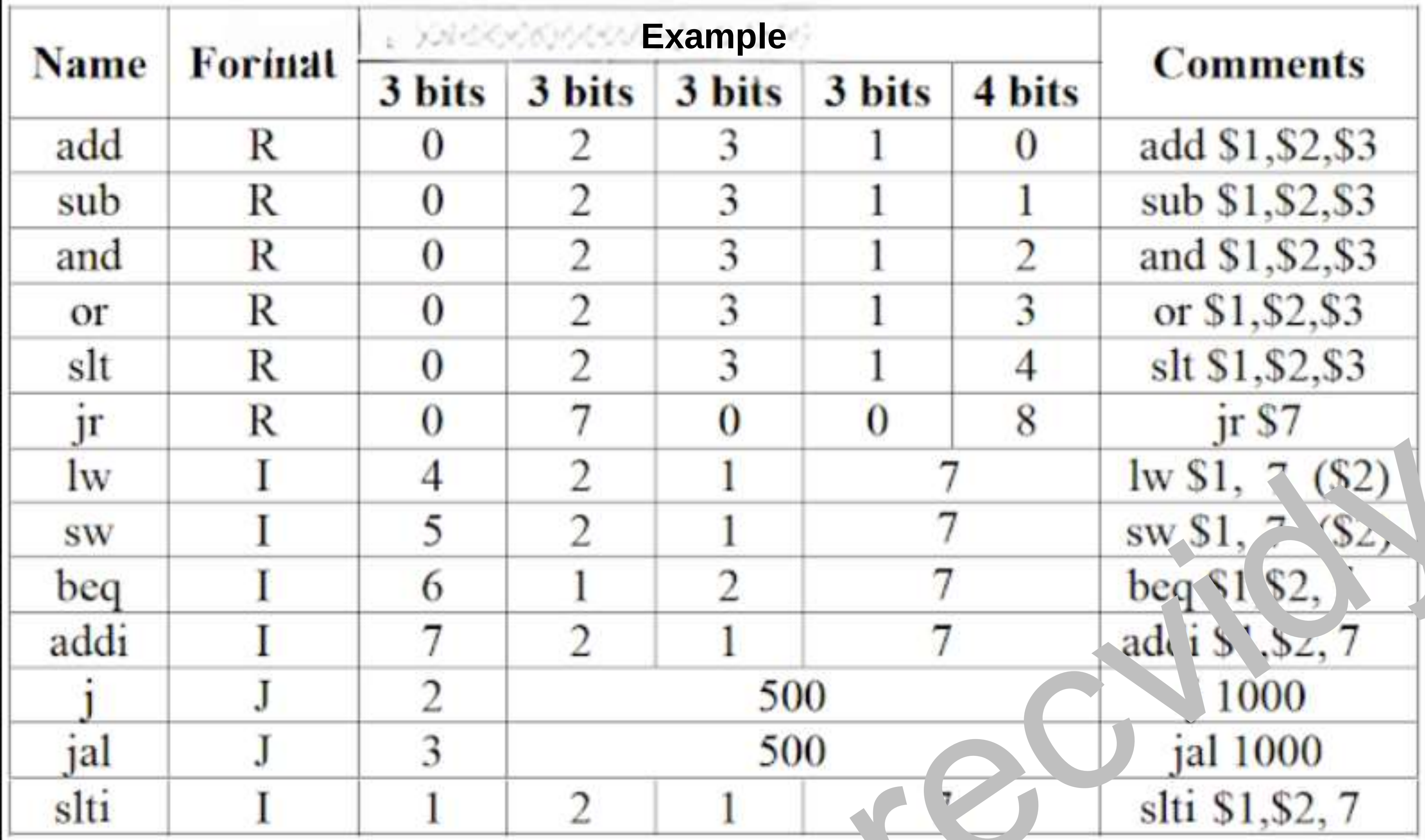
MILESTONE 4

INNOVATON IN FUTURE - Implementing Power and Clock Gating Optimization for IoT devices to enhance power efficiency in MIPS processors. Additionally, considering Security Integration to incorporate advanced hardware security features, such as encryption and secure key storage, to protect against vulnerabilities.

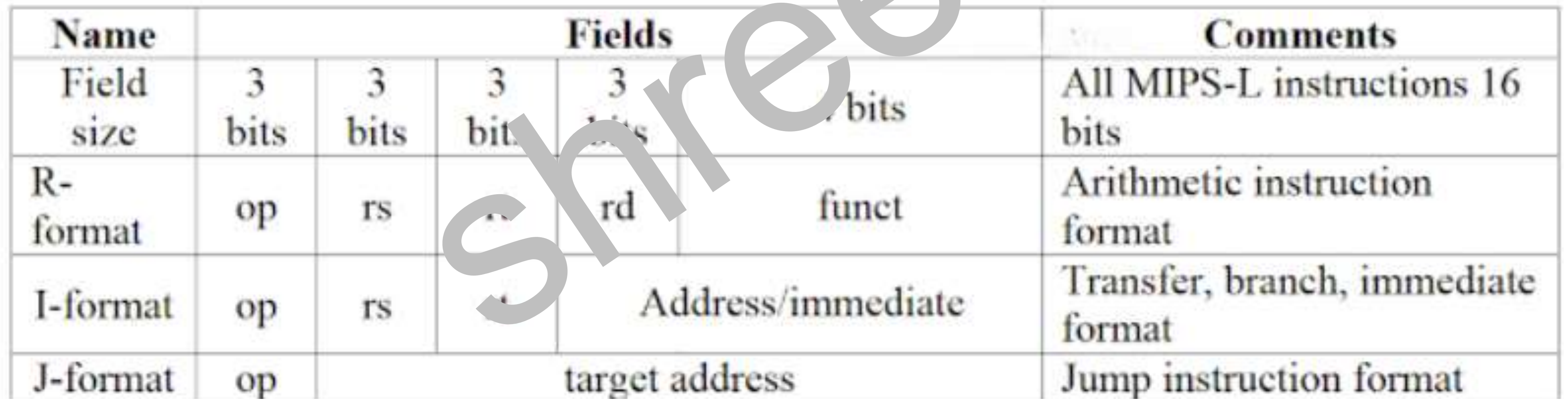
DEBUGGING AND OUTPUT

- Multiplexers allows the processor to decide between two or more sources for a particular data path and also used to integrate overall data paths.
- Sign extension is used to extend the size of an immediate value in the instruction set from 16 bits to 32 bits.
- The EP waveforms generated in each phase are to be analyzed well and should go through many debugging processes.
- After debugging, the resultant output is proved by the waveforms obtained by all data path components.

THE INSTRUCTION FORMAT AND INSTRUCTION SET ARCHITECTURE



Name	Format	Example					Comments
		3 bits	3 bits	3 bits	3 bits	4 bits	
add	R	0	2	3	1	0	add $1,$2,$3
sub	R	0	2	3	1	1	sub $1,$2,$3
and	R	0	2	3	1	2	and $1,$2,$3
or	R	0	2	3	1	3	or $1,$2,$3
slt	R	0	2	3	1	4	slt $1,$2,$3
jr	R	0	7	0	0	8	jr $7
lw	I	4	2	1	7		lw $1, 7 ($2)
sw	I	5	2	1	7		sw $1, 7 ($2)
beq	I	6	1	2	7		beq $1,$2, [illegible]
addi	I	7	2	1	7		addi $1,$2, 7
j	J	2	500				[illegible] 1000
jal	J	3	500				jal 1000
slti	I	1	2	1	7		slti $1,$2, 7



Name	Fields					Comments
Field size	3 bits	3 bits	3 bits	3 bits	[illegible] bits	All MIPS-L instructions 16 bits
R-format	op	rs	[illegible]	rd	funct	Arithmetic instruction format
I-format	op	rs	[illegible]	Address/immediate		Transfer, branch, immediate format
J-format	op	target address				Jump instruction format

- Add : R[rd] = R[rs] + R[rt]
- Subtract : R[rd] = R[rs] - R[rt]
- And: R[rd] = R[rs] & R[rt]
- Or : R[rd] = R[rs] | R[rt]
- SLT: R[rd] = 1 if R[rs] < R[rt] else 0
- Jr: PC=R[rs]
- Lw: R[rt] = M[R[rs]+SignExtImm]
- Sw : M[R[rs]+SignExtImm] = R[rt]
- Beq : if(R[rs]==R[rt]) PC=PC+1+BranchAddr
- Addi: R[rt] = R[rs] + SignExtImm
- SLTI: R[rt] = 1 if R[rs] < imm else 0
- J : PC=JumpAddr
- Jal : R[7]=PC+2;PC=JumpAddr

INSTRUCTION SET

The implementation is designed to simulate the functionality of fetching instructions from memory for a simple MIPS processor. The design includes three key modules:

InstructionMemory: Simulates a memory array containing instructions.

program_counter: Tracks the current program counter and updates it to point to the next instruction.

mips_top: Integrates the InstructionMemory and program_counter modules to form the top-level structure of the processor.

The InstructionMemory module contains a memory array memory[0:19] capable of storing 20 instructions (32-bit each).

***R-Type Instructions*:** Perform arithmetic or logical operations.

Format: opcode (6 bits) | rs (5 bits) | rt (5 bits) | rd (5 bits) | shamt (5 bits) | funct (6 bits)

Example: memory[0]=32'b000000_00001_00010_00011_00000_100000 corresponds to ADD $3, $1, $2 (Add the values in registers $1 and $2 and store the result in $3).

***I-Type Instructions*:** Use immediate values for computations or memory access.

Format: opcode (6 bits) | rs (5 bits) | rt (5 bits) | immediate (16 bits)

Example: memory[5]=32'b001000_00001_00010_0000000000000101 corresponds to ADDI $2, $1, 5 (Add the immediate value 5 to the value in register $1 and store the result in $2).

***J-Type Instructions*:** Perform unconditional jumps to specified addresses.

Format: opcode (6 bits) | address (26 bits)

Example: memory[11]=32'b000010_00000000000000000000001010 corresponds to J 10 (Jump to address 10).

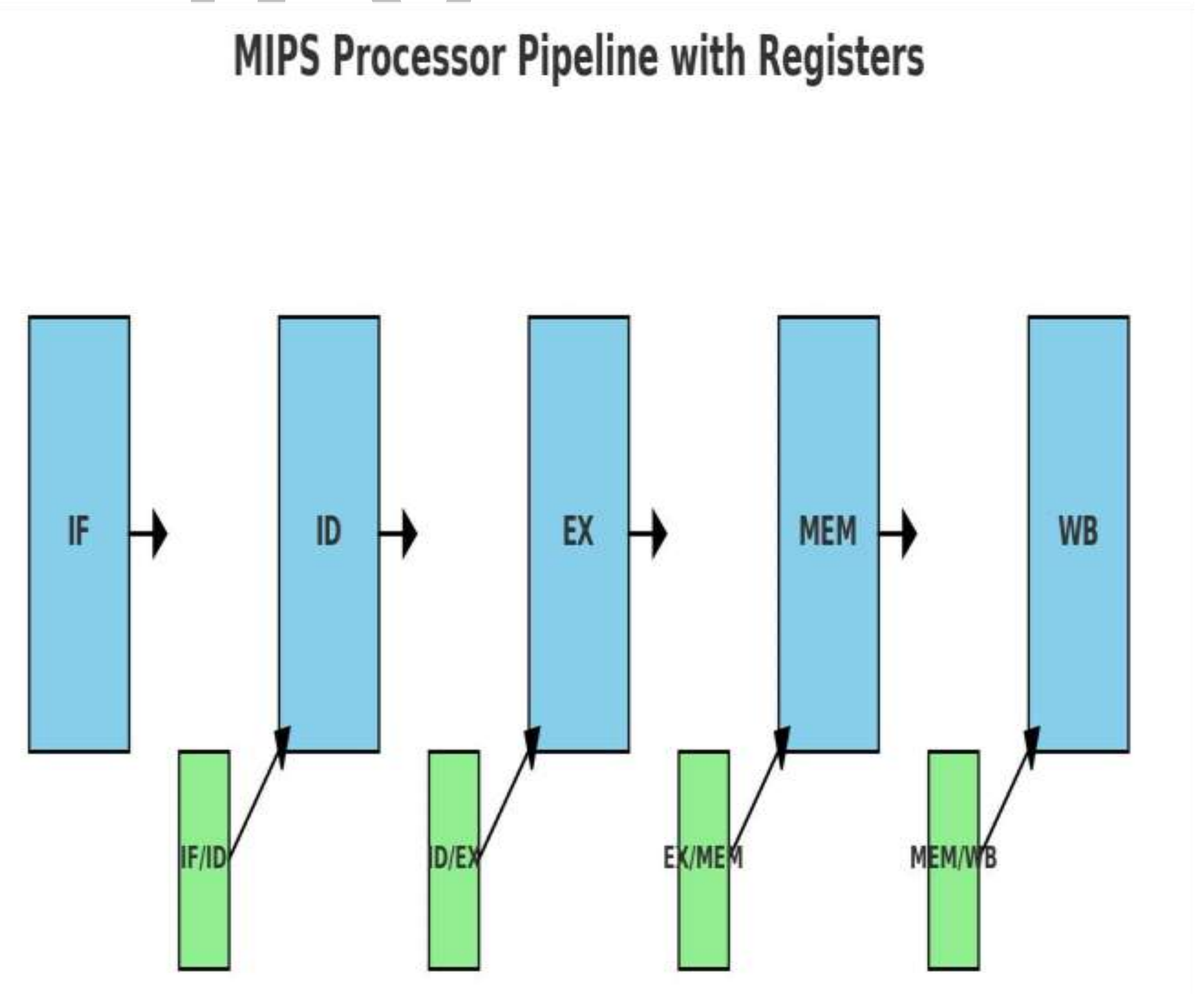
REGISTERS

- **Registers** are fast, small-sized storage elements in the MIPS processor that temporarily hold data during processing.
- They serve as the **primary interface between the processor's Arithmetic and Logic Unit (ALU) and memory**, storing operands, intermediate results, and final computation results.
- In the MIPS processor, registers play a **central role by acting as the main data source and destination for instruction execution**, minimizing the need to access slower main memory and significantly boosting performance.
- The **register file**, a dedicated hardware module, contains multiple general-purpose registers. It supports **dual-port reading and single-port writing**, allowing the processor to simultaneously fetch two operands and write back results in the same clock cycle.
- Registers collaborate closely with the **ALU** by supplying operands for arithmetic or logical operations and storing computation results back into the register file.
- In the MIPS architecture, registers are critical for supporting the **load/store operations** characteristic of RISC processors. Data is first **loaded into registers from memory**, manipulated within the registers during execution, and **written back to memory or another register** as needed.
- By facilitating **fast, local data access**, registers significantly enhance the **efficiency and speed of the MIPS processor**. They are pivotal in enabling **pipelining**, ensuring high throughput and effective utilization of the processor's resources.

REGISTERS

Specialized **pipeline registers** (e.g., IF/ID, EX/MEM, MEM/WB) are used to **isolate and transfer data between the five pipeline stages**—Instruction Fetch (IF), Instruction Decode (ID), Execution (EX), Memory Access (MEM), and Write Back (WB).These pipeline registers ensure that **data and instructions flow seamlessly through the pipeline**, enabling concurrent processing of multiple instructions without conflicts.

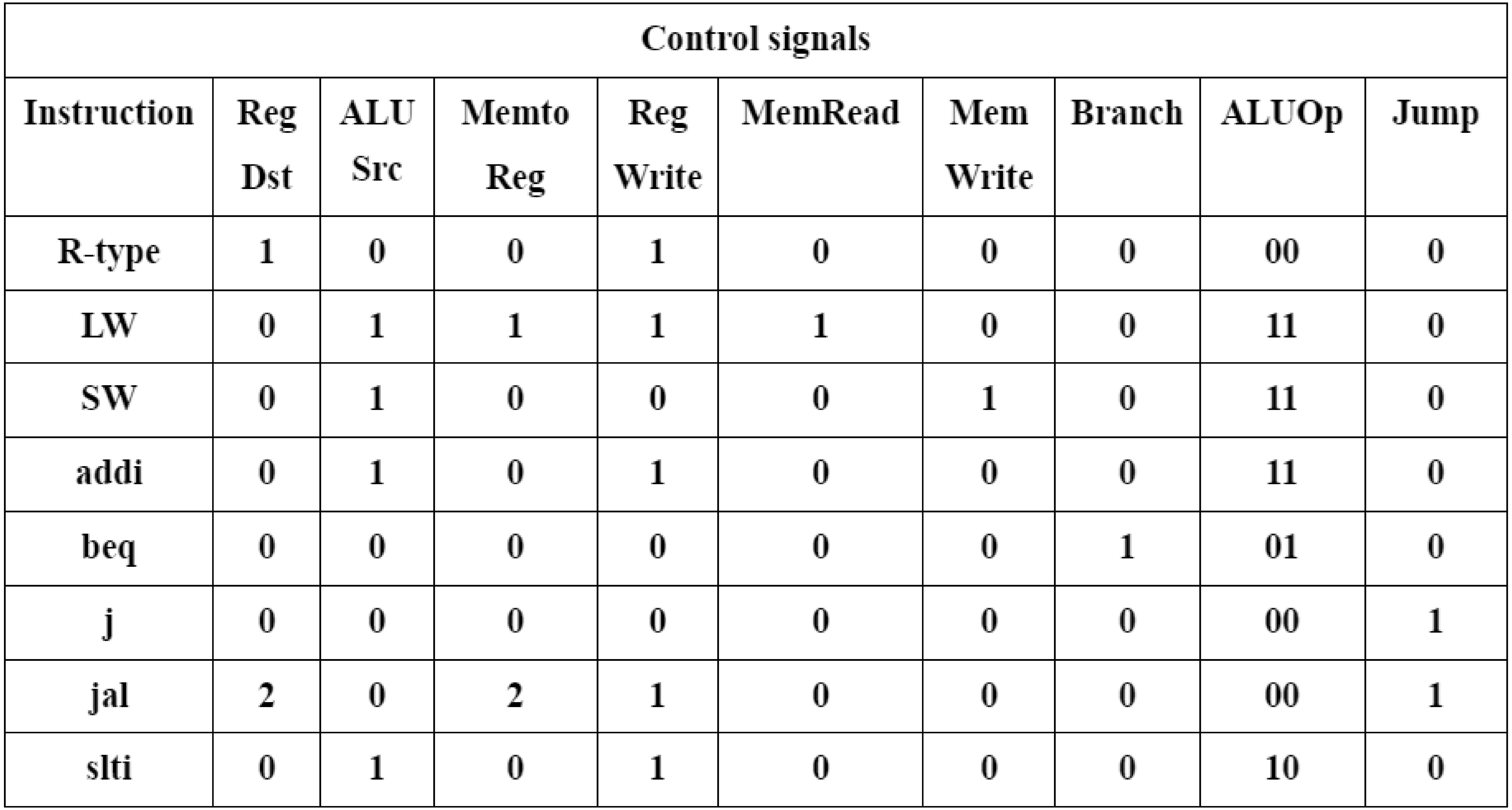
1. **IF/ID Register**
   - Stores the fetched instruction and Program Counter (PC) value after the Instruction Fetch (IF) stage.
   - Passes data to the Instruction Decode (ID) stage, ensuring no overwriting during concurrent processing.
2. **ID/EX Register**
   - Holds decoded instruction data, source operands, and control signals after the Instruction Decode (ID) stage.
   - Prepares data for the Arithmetic and Logic Unit (ALU) in the Execution (EX) stage.
3. **EX/MEM Register**
   - Stores results of ALU operations or memory addresses calculated during the Execution (EX) stage.
   - Transfers computed values and control signals to the Memory Access (MEM) stage.
4. **MEM/WB Register**
   - Holds data fetched from memory or the ALU result after the Memory Access (MEM) stage.
   - Passes final results to the Write Back (WB) stage for storage in the destination register.



ALU UNIT

ALU Control				
ALU op	Function	ALUcnt	ALU Operation	Instruction
11	xxxx	000	ADD	Addi,lw,sw
01	xxxx	001	SUB	BEQ
00	00	000	ADD	R-type: ADD
00	01	001	SUB	R-type: sub
00	02	010	AND	R-type: AND
00	03	011	OR	R-type: OR
00	04	100	slt	R-type: slt
10	xxxxxx	100	slt	i-type: slti

CONTROL UNIT AND DATA PATH



Control signals									
Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	MemRead	Mem Write	Branch	ALUOp	Jump
R-type	1	0	0	1	0	0	0	00	0
LW	0	1	1	1	1	0	0	11	0
SW	0	1	0	0	0	1	0	11	0
addi	0	1	0	1	0	0	0	11	0
beq	0	0	0	0	0	0	1	01	0
j	0	0	0	0	0	0	0	00	1
jal	2	0	2	1	0	0	0	00	1
slti	0	1	0	1	0	0	0	10	0

ALU UNIT

ALU in 32-bit Single-Cycle MIPS Processor

The Arithmetic Logic Unit (ALU) is a core component of the 32-bit single-cycle MIPS processor, responsible for performing arithmetic and logical operations. It executes operations like addition, subtraction, bitwise AND/OR, comparison (set less than), and more, depending on the instruction type.

Key Features:

- **Inputs:** Two 32-bit operands and a 4-bit control signal (ALUControl) that specifies the operation.
- **Outputs:**
  - 32-bit result of the operation.
  - Zero flag (asserted when the result is zero, used for branch decisions).

Process:

1. **Control Signal Generation:**
   - For R-type instructions, the ALUControl signal is derived from the instruction's function code (Funct) and ALUOp.
   - For I-type instructions, ALUOp directly specifies operations like addition (for load/store) or subtraction (for branches).
2. **Operation Execution:**
   - Performs the specified operation based on ALUControl (e.g., addition for load/store, subtraction for branch comparison).
   - Evaluates the Zero flag for branch instructions.

Role in the Processor:

The ALU integrates with components like the register file (providing operands), the control unit (providing ALUControl signals), and the program counter (for branch decisions). Its efficiency and correctness directly impact the overall processor performance.

ALU & CONTROL UNIT

Control Unit:

The Control Unit generates control signals to coordinate the execution of instructions. It decodes the instruction's opcode and function code to determine the required operations.

- **Main Functions:**
  - **Instruction Decode:** Interprets the opcode to determine the instruction type (R-type, I-type, J-type).
  - **Signal Generation:** Produces signals to control ALU operations, memory access, register writes, and multiplexer selections.
  - **ALU Control Signal:** Works with the ALU Control subunit to derive specific operations (e.g., addition, subtraction).

Data Path:

The Data Path represents the physical flow of data through the processor during instruction execution.

- **Main Components:**
  - **Program Counter (PC):** Holds the address of the next instruction.
  - **Instruction Memory:** Stores instructions fetched by the processor.
  - **Register File:** Provides operands to and receives results from the ALU.
  - **ALU:** Performs arithmetic and logical operations.
  - **Data Memory:** Reads/writes data for load/store instructions.
  - **Multiplexers:** Select inputs for ALU, data memory, or PC based on control signals.

Integration: The Control Unit coordinates the flow of data in the Data Path. It ensures the correct sequence of operations (fetch, decode, execute, memory access, write-back) for each instruction. Together, they implement the instruction set in a single clock cycle.

MEMORY ACCESS

The Memory Unit is a fundamental component of the 32-bit single-cycle MIPS processor, responsible for storing instructions and data required during program execution. It provides a unified memory space that accommodates both instruction and data memory, utilizing a 32-bit address bus. This enables the processor to address up to 2^32 memory locations, corresponding to a total addressable memory capacity of 4 GB. The memory unit is designed to facilitate efficient data retrieval and storage while ensuring precise synchronization with the processor’s operations.

1. Structure of the Memory Unit

- **Instruction Memory**: Stores the program's executable instructions. These instructions are fetched by the processor during each cycle.
- **Data Memory**: Holds variables and intermediate data. It is used for storing results of operations and temporary values required during program execution.
- **Unified Address Space**: The memory is accessed using a 32-bit address, allowing a total of 2^32locations, equivalent to 4 GB of addressable space.

2. Memory Access Operations

- **Load (lw)**: The memory unit retrieves data from the specified address and provides it to the processor for use in computations.
- **Store (sw)**: The memory unit saves data provided by the processor to the specified address.
- **Word Alignment**: All memory addresses are word-aligned (multiples of 4), ensuring efficient access to 32-bit words and avoiding partial-word issues.

MEMORY ACCESS

3. Control Unit and Synchronization

- The control unit is responsible for managing how the processor interacts with the memory unit. It coordinates memory operations like loading and storing data.
- **Load Operation (lw)**: The control unit sends the address to the memory, asking for data. Once the memory returns the data, the control unit provides it to the processor.
- **Store Operation (sw)**: The control unit sends the data and the address to the memory, instructing it to store the data at that location.
- Since the memory operations must complete in one clock cycle in a single-cycle architecture, synchronization is essential. The control unit ensures that each operation happens correctly and on time, allowing the processor to function smoothly without delays or errors.

4. Role of Cache Memory

In this architecture, where every operation must complete in one clock cycle, the inclusion of a cache helps improve performance without altering the simplicity of the design.

Key Roles of Cache Memory

- **Cache Integration**: A cache is a small, high-speed memory unit added to store frequently accessed data or instructions. In a single-cycle MIPS processor, it helps reduce the delay caused by accessing slower main memory.
- **Instruction Cache**: The instruction cache stores commonly used instructions from the program. This allows the processor to fetch instructions quickly during execution, reducing the time needed to access instruction memory for repetitive or frequently executed tasks.
- **Data Cache**: The data cache temporarily holds frequently accessed data values. For operations like lw (load word) and sw (store word), the processor can directly interact with the data cache instead of accessing the main memory, improving execution speed.
- **Performance Improvement**: Caches significantly reduce memory latency by decreasing the number of times the processor needs to access the main memory. This enhancement allows the processor to maintain the single-cycle execution model while handling memory-intensive tasks more efficiently.

RESULT AND DISCUSSION

The completion of the 32-bit single-cycle MIPS processor project emphasizes the simplicity and efficiency of this architecture, showcasing its capability to execute instructions like arithmetic operations, branching, and memory access within a single clock cycle. By leveraging the reduced instruction set computing (RISC) paradigm, the design achieves streamlined control logic and optimized performance, making it well-suited for tasks requiring minimal latency. The integration of core components such as the ALU, Instruction set, Registers and Memory unit ensures robustness and accuracy in execution. This foundational design serves as a stepping stone for exploring more advanced processor architectures, addressing hardware inefficiencies, and expanding functionalities to meet diverse application demands.

The MIPS Corporation was founded in 1984 by Stanford research team members to develop a commercial version of the prototype chip. The first MIPS product was the R2000 microprocessor, which was introduced in 1985. Recent applications of MIPS processors include significant advancements in automotive and embedded systems, particularly in AI and high-performance computing.

Automotive AI and ADAS: MIPS has introduced the P8700, a high-performance AI-enabled processor designed for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. It leverages RISC-V architecture to provide power efficiency and scalability while handling the intensive data demands of automotive systems. This processor supports multi-threaded and multi-core designs optimized for real-time processing of sensor data, improving the performance of AI accelerators used in autonomous vehicles.

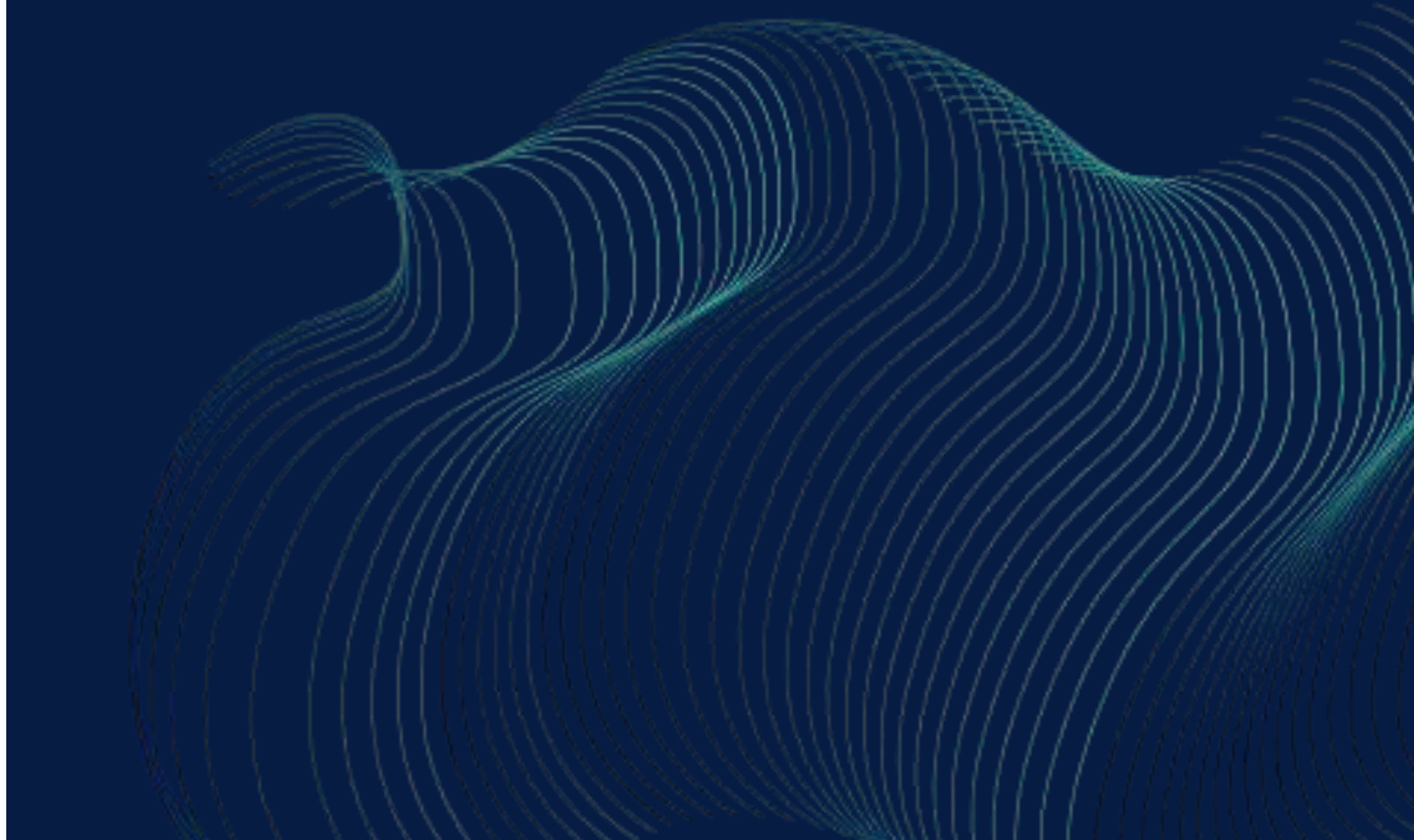
Embedded Systems and IoT: MIPS processors are widely used in embedded applications, offering efficient computation for Internet of Things (IoT) devices, industrial automation, and consumer electronics. Their designs focus on low power consumption and real-time processing, which are critical for IoT devices.

FUTURE DEVELOPMENTS

Innovating these aspects could push MIPS processors into newer markets and applications, while enhancing their competitive edge in existing embedded systems.

MIPS processors are designed especially for specific low-power, edge, or IoT applications. Here are areas with significant innovation potential:

1. **Power and Clock Gating Optimization for IoT** – Advanced power-saving techniques specific to MIPS for ultra-low power.
2. **Security Integration** – More advanced hardware security mechanisms for MIPS-based embedded systems
3. **3D Integration and Chiplets** – Applying advanced packaging techniques to MIPS designs.
4. **Custom AI Accelerators** – Integrating more sophisticated AI/ML accelerators in MIPS SoCs.
5. **AI-Driven VLSI Design** – Expanding the use of AI for optimizing MIPS design, layout, and testing.



REFERENCES

[1] Ahmad, Ahmadi., Reza, Faghih, Mirzaee. (2019). MIPS-Core Application Specific Instruction-Set Processor for IDEA Cryptography - Comparison between Single-Cycle and Multi-Cycle Architectures.

[2] Bhardwaj, Priyavrat & Murugesan, Siddharth. (2017). DESIGN & SIMULATION OF A 32-BIT RISC BASED MIPS PROCESSOR USING VERILOG. International Journal of Research in Engineering and Technology. 10.15623/ijret.2016.0511030.

[3] Design And Analysis Of 64 Bit MIPS Processor", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.5, Issue 5, pageno.1052-1057, May-2018. Available: http://www.jetir.org/papers/JETIR1805188.pdf.

[4] Gross, Thomas R., Hennessy, John L., Przybylski, Steven A., and Rowen, Christopher. (1988). Measurement and evaluation of the MIPS architecture and processor. ACM Trans. Comput. Syst. 6, 3 (Aug. 1988), 229–257. https://doi.org/10.1145/45059.45060.

[5] Hennessy, John, Jouppi, Norman, Przybylski, Steven, Rowen, Christopher, Gross, Thomas, Baskett, Forest, and Gill, John. (1982). MIPS: A microprocessor architecture. Proceedings of the 15th annual workshop on Microprogramming (MICRO 15). IEEE Press, 17–22.

[6] Marri Mounika, Aleti Shankar. "Design & Implementation Of 32-Bit Risc (MIPS) Processor". International Journal of Engineering Trends and Technology (IJETT). V4(10):4466-4474 Oct 2013. ISSN:2231-5381.www.ijettjournal.org. Published by Seventh Sense Research Group.

[7] P. M. B. G. R. (2015). “Improved RISC Processor Design Using MIPS Instruction Set Approach”. International Journal on Recent and Innovation Trends in Computing and Communication, 3(5), pp. 2599–2604. doi: 10.17762/ijritcc.v3i5.4292.

[8] Ritpurkar, S. P., Thakare, M. N., and Korde, G. D. (2015). "Design and simulation of 32-Bit RISC architecture based on MIPS using VHDL". 2015 International Conference on Advanced Computing and Communication Systems, Coimbatore, India, pp. 1-6. doi: 10.1109/ICACCS.2015.7324067.

[9] Shobhit, Shrivastav., Shubham, Kumar., Sarthak, Gupta., Bharat, Bhushan. (2020). Qualitative Analysis of 32 Bit MIPS Pipelined Processor. International Journal of Engineering Research and Technology, 9(05). doi: 10.17577/IJERTV9IS050484.

[10] Vidyashree, H. (2023). Design and Implementation of RISC MIPS Processor on FPGA. International Journal For Science Technology And Engineering, 11(4), 1406-1410. Available from: 10.22214/ijraset.2023.50352.

THANK YOU