The Multiverse of MIPS: Reviewing Across the Implementations

Nandi Sree Harshitha[1], Krishna Teja Murikipudi[1], S. Gopikrishna[2], A.V. Ananthalakshmi[3]
[1, 1] *B.Tech. student, Dept. of E.C.E, Puducherry Technological University, Puducherry, India*
[2] *Scientist 'F', Research Centre Imarat, D.R.D.O., Vignyanakancha, Hyderabad, India*
[3] *Associate professor, Dept. of E.C.E, Puducherry Technological University, Puducherry, India*

Abstract**—***Originally developed in the early 1980s, the MIPS architecture continues to have a significant impact on modern computing. By examining different implementations of MIPS such as single-cycle, multi-cycle, pipeline, MIPS32, and MIPS64, this literature review provides a comprehensive overview of MIPS processors. A thorough discussion of pipelines, parallel processing, instruction formats, register conventions, instruction types, and various stages in the pipeline is presented in this paper, highlighting the role of hazard handling in a pipelined processor.*

Keywords**—***Hazard Handling, Instruction Formats, MIPS, Multi-Cycle, Pipeline, Processor, Single-Cycle*

1. Introduction

MIPS (Microprocessor without Interlocked Pipeline Stages) is a RISC-based processor with a load store architecture and 32 general-purpose registers. Register $0 is hardwired to zero. MIPS has four instruction sets (MIPS I through IV). Compared to their predecessors, the latest MIPS instruction sets have improved functionality and complexity. These instructions can be executed using single-cycle, multi-cycle, or pipeline methods. It is the goal of pipelining to maximize resource utilization, but it also produces hazards, which will be discussed in greater detail in the upcoming sections of this paper. There is dedicated hardware between pipeline stages in the "Interlocked pipeline stages" model to detect hazards. MIPS, as its name suggests, lacks such hardware for simplicity and faster execution. Thus, hazard handling is the responsibility of the program. Failing to do so might result in inaccurate outputs or undesired behaviour.

2. Pipelining

It is essential to examine the single-cycle and multi-cycle implementations of an MIPS processor to appreciate the impact of pipelining. In a single-cycle implementation, any instruction executes in one clock cycle, the CPI being 1. Although it sounds like a great idea, it is inefficient because the clock cycle length is fixed for all kinds of instructions and is determined by the longest possible path in the processor. The overall performance of such implementation might be poor, given its long clock cycle. Pipelining is a technique of overlapping stages of multiple instructions to achieve parallel processing. Fetch, Decode, Execute, Memory Access, and Write Back are the stages of a five-stage pipeline model. When Instruction0 is in the decode stage, Instruction1 is fetched simultaneously, so the fetch stage of Instruction1 overlaps with the decode stage of Instruction0. Similarly, the stages of the pipeline overlap for multiple instructions, as illustrated in Figure 1. The pipeline is faster because it executes multiple instructions in parallel, but the time taken to process a single instruction is still the same. The effects of pipeline are better appreciated when there are multitudinous instructions to be executed, under ideal conditions a five-stage pipeline processor is five times faster than a single-cycle processor.

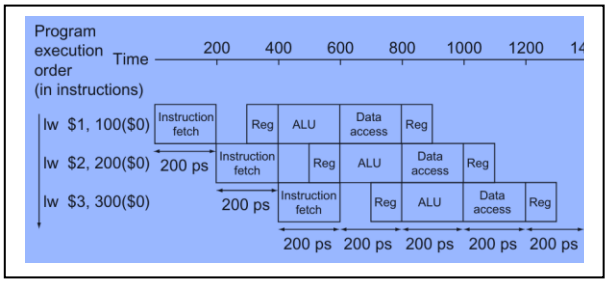

Figure 1. Five-Stage Pipeline in MIPS [1]

In a multi-cycle implementation, each instruction takes multiple clock cycles, executing one instruction at any time. It is faster than the single-cycle model because of the variable execution time of the multi-cycle model, which only goes through

the necessary stages. The pipeline is faster and ensures maximum resource utilization, but it poses the constant risk of encountering hazards; the details of it are in **Section 4.** [1]

3. Overview of MIPS

The MIPS architecture has 32 general-purpose registers and a load-store architecture, register conventions of the MIPS improve the efficiency of execution. The registers are divided into groups to suit various functionalities of MIPS, as shown in Figure 2.

Name	Register number	Usage
$zero	0	The constant value 0
$v0–$v1	2–3	Values for results and expression evaluation
$a0–$a3	4–7	Arguments
$t0–$t7	8–15	Temporaries
$s0–$s7	16–23	Saved
$t8–$t9	24–25	More temporaries
$gp	28	Global pointer
$sp	29	Stack pointer
$fp	30	Frame pointer
$ra	31	Return address

Figure 2. MIPS Register Conventions [1]

Register 1, called $at, is reserved for the assembler and registers 26–27, called $k0–$k1, are reserved for the operating system. The original version of MIPS is of 32-bit (MIPS32), five-stage pipeline and later it is extended to 64-bit (MIPS64), there are studies that suggest the usage of six-stage pipeline, 16-bits and other variants to gain added advantages for specific applications. A descriptive analysis of the same is discussed in this paper. [1]

3.1. 16-bit MIPS

The term "16-bit MIPS" refers to a MIPS processor with a 16-bit data bus and a processor that primarily operates on 16-bit data. In MIPS16, each instruction is 16 bits wide. This compact instruction format allows for efficient use of memory and reduced instruction fetch and decode overhead. Input and output operations are performed by accessing memory-mapped I/O devices. This means that special memory addresses are assigned to I/O devices, and reading from or writing to these addresses' triggers communication with the respective devices. MIPS16 instructions primarily include LOAD and STORE operations for accessing memory. LOAD instructions transfer data from memory to registers, while STORE instructions transfer data from registers to memory. Instruction sets are divided into 4 types:

- **Arithmetic:** These arithmetic instructions manipulate data stored in registers.
- **Logical operations:** Logical operations such as AND, OR, and XOR are provided to manipulate binary data at the bit level.
- **Data Transfer:** Load and Store instructions facilitate data transfer between memory and registers. These operations are essential for accessing variables, arrays, and other data structures.
- **Branch and Control:** MIPS16 instructions include branching operations for altering the program flow based on conditions. This includes unconditional jumps, conditional branches, subroutine calls, and returns.

3.2. Instruction Word Format

3.2.1. Register Type (R TYPE): The MIPS16 instruction R format comprises a 5-bit opcode field followed by three 3-bit fields representing source operand registers (Rs1 and Rs2) and a destination register (Rd). This structure allows for efficient encoding of instructions while providing flexibility in specifying source and destination operands for arithmetic and logical operations.

Op-code	Rs	Rt	Rd	Reserved
(5)	(3)	(3)	(3)	(2)

Figure 3. R Type Instruction Format [2]

Let the ADD instruction be, **ADD** $t0 $t1 $t2, the operands are:

- Rs1 = $t1
- Rs2 = $t2
- Rd = $t0

3.2.2. Immediate Type (I TYPE): The MIPS16 immediate instruction format comprises a 5-bit opcode field followed by two 3-bit fields representing source operand register (Rs) and destination register (Rt), and a 5-bit immediate field containing the constant value. This structure

allows for efficient encoding of instructions involving immediate values, providing flexibility in specifying operands for arithmetic and logical operations.

Op-code	Rs	Rt	Immediate
(5)	(3)	(3)	(5)

Figure 4. I Type Instruction Format [2]

Suppose we have an ADDI instruction, **ADDI** $t0 $t1 1010, the operands are:

- Rs = $t0
- Rt = $t1
- Immediate value = 10

3.2.3. Jump Type (J TYPE): The MIPS16 jump instruction format comprises a 5-bit opcode field followed by an 11-bit field representing the jump target address. This structure allows for efficient encoding of instructions involving control transfer operations, providing flexibility in altering the program flow within the limited 16-bit instruction format of MIPS16.

Op-code	Address
(5)	(11)

Figure 5. J Type Instruction Format [2]

3.3. MIPS16 Architecture

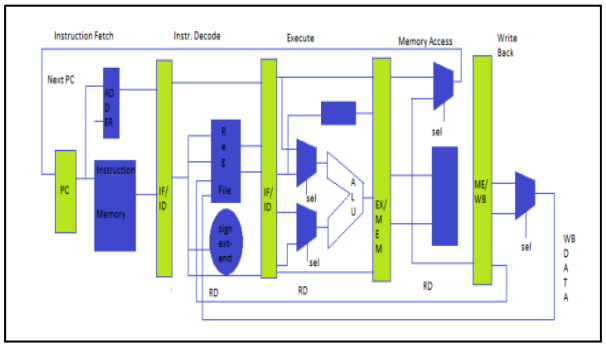

Figure 6. MIPS16 Architecture [4]

Jump instructions, which alter the program flow by transferring control to a different part of the code, are typically handled separately. When a jump instruction is encountered, the pipeline may need to be flushed or stalled to ensure that the correct instruction sequence is followed.

The pipeline allows multiple instructions to be processed simultaneously, with different instructions at different stages of execution. As each instruction progresses through the pipeline stages, subsequent instructions can enter the pipeline, overlapping their execution with previous instructions. This overlapping of instructions helps shorten the overall execution time of the program by maximizing processor utilization and throughput. Each instruction is executed in a single cycle. This means that each phase of the pipeline takes one clock cycle to complete before the next phase begins. [2][3][4]

3.4. 32-bit MIPS

In a 32-bit MIPS processor, instructions are typically encoded as 32-bit binary words. The processor operates on 32-bit data at a time and uses a set of 32-bit general-purpose registers for temporary storage and computation. In a 3-stage pipeline, instructions are executed in three stages: Fetch, Decode, and Execute.

1) Fetch Stage: In this stage, the processor fetches the instruction from memory. The program counter (PC) holds the address of the next instruction to be fetched. The instruction is fetched from memory using this address and loaded into the instruction register (IR).

2) Decode Stage: The fetched instruction is decoded in this stage. The opcode (operation code) and operands are identified. The control signals for the execution unit are generated based on the opcode.

3) Execute Stage: In this stage, the decoded instruction is executed. This may involve arithmetic or logical operations, data transfers between registers, or branching to a different instruction address based on a conditional branch instruction. [6]

Husainali S. Bhimani et al., (2016) simulated and synthesized using Xilinx ISE, followed by static timing and power analysis. The project was developed for the FFG1157 package, utilizing the Virtex 7 FPGA with the XC7VX330T device. With frequency and clock period of 310.878 MHz and 3.217 ns. [6]

Priyavrat Bhardwaj et al., 2016, simulated a 32-bit model that has the architecture and operation of

a 5-stage pipelined processor with separate phases for instruction fetch (IF), decode (ID), execution (EX), data memory (MEM), and write back (WB).

1) Instruction Fetch (IF): In this phase, instructions are fetched from the program memory using the program counter (PC) as a pointer. The fetched instruction is typically stored in an instruction buffer or pipeline register to await decoding in the next stage.

2) Decode (ID): In this phase, the fetched instruction is decoded to determine the operation to be performed and the operands involved. Operand values are typically read from registers or immediate fields in the instruction, preparing them for execution in the next stage.

3) Execution (EX): This phase executes the decoded instruction. For arithmetic or logical instructions, this involves performing the specified operation on the operand values.
For control transfer instructions like jumps or branches, the target address calculation may occur in this phase.

4) Data Memory (MEM): In this phase, memory access operations such as loads and stores are performed. If the instruction involves accessing data memory, the address calculation is completed, and data is either read from or written to memory.

5) Write Back (WB): The final phase involves writing back the results of the executed instruction to the appropriate register file. This phase updates the register file with the result of arithmetic or logical operations or the data read from memory. [8]

P.Indira et al., 2019, developed the design that minimizes power consumption while optimizing layout area and Power-Delay Product (PDP), Low Power Unit in this design employs 28nm high-performance low-power process technology, prioritizing performance while balancing speed, delay, area, and power. It integrates intelligent clock gating for further power reduction without compromising bandwidth. [5]

Pipeline Registers feature a novel implicit pulse-triggered flip-flop design, addressing transistor stacking issues with gated Pull-up control in this model, enhancing power efficiency in pipelining operations. These auxiliary components are used in this processor. This project utilizes the Virtex-7 FPGA for implementing pipelined operations, offering superior performance compared to other 7-series devices like Artix7 and Kintex7. It also supports compatibility with the Virtex6 FPGA series. The pipelining operates at a frequency of 420.028 MHz, with a minimum operation time of 1.385ns and a maximum path delay of 1.143ns. Comparative analysis is developed. [5]

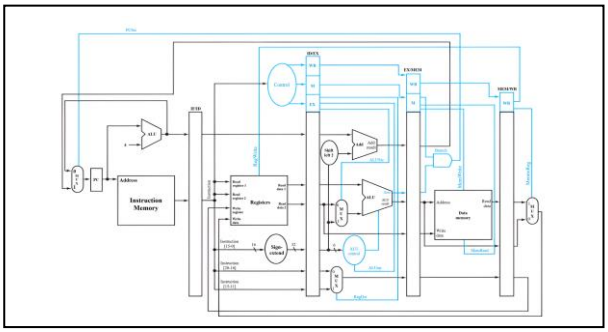

Figure 7. Five-Stage MIPS32 Architecture [1]

3.5. 32-bit Instruction Format

3.5.1. Register Type: The instruction format depicted in Figure 8 is Register (R) Type. It utilizes the MSB 6 bits for the Opcode. Following the Opcode, there are 15 bits allocated for the three registers: Rs, Rt, and Rd, where Rs and Rt are the source registers, and Rd is the target register. The subsequent 5 bits indicate the shift amount for moving bits. The remaining 6 bits denote the function field, specifying the operation to be executed on the registers.

Op-code	Rs	Rt	Rd	Shamt	Funct
(6)	(5)	(5)	(5)	(5)	(6)

Figure 8. R Type 32-bit Instruction Format

Its primary application lies in executing mathematical operations like addition and subtraction, exemplified by "add Rd, Rs, Rt", where Rd receives the signed addition of Rs and Rt

3.5.2. Immediate Type: Immediate (I) type instructions, feature four fields: a 6-bit Opcode for selecting the instruction type, followed by the storage of data in the Source Register (Rs) and

Target Register (Rt), each occupying 5 bits. The final field, spanning 16 bits, represents the Address/Immediate Value, containing immediate data.

Op-code	Rs	Rt	Immediate
(6)	(5)	(5)	(16)

Figure 9. I Type 32-bit Instruction Format

It demonstrates the dichotomy of Rt register which can be used both as a source and a destination. The immediate value received by sign extend is represented by the last 16-bits which is then sent to the Arithmetic and Logical Unit for playing out the desired function.

3.5.3. Jump Type: In this arrangement type, there are two fields: a 6-bit Opcode used to select the instruction format, and a 26-bit Ending or Target address determining the branch destination. the 32-bit jump (J) address is derived by appending the last 4 bits of PC + 4 to the left-shifted 26-bit instruction value fetched from MEM. Furthermore, it facilitates jumping to the destination while disregarding any alternate instructions.

Op-code	Address
(6)	(26)

Figure 10. J Type 32-bit Instruction Format

Soumya Murthy et.al., 2015 proposed DLX architecture with pipelined control in a RISC core in this research paper. RISC processors typically exhibit high power consumption, which can be mitigated through complex fabrication processes. However, implementing low-power versions is less complex if addressed at the front end. This paper employs HDL modification techniques to reduce dynamic power, particularly targeting IO power consumption for optimization. The proposed power reduction focuses on algorithm-level modifications, acknowledging the limitation in reducing leakage power. The overall power optimization achieved from HDL technique is 13.33%. [7]

Rohit J et.al., 2017 designed MIPS 32 BIT 5 stage pipelined model which resulted in achieving an on-chip power consumption of 0.09W. The Opcode (instructions) were generated using the MARS compiler and then extracted for use in the instruction memory. Subsequently, this design was implemented on the Artix-7 FPGA using the Xilinx Vivado platform. [11]

TriptiMahajan et.al., 2019 developed a Low Power 32-bit multicycle MIPS processor using Xilinx ISE and Altera Quartus EDA tools. [9]

Takahito Hayashi et.al., 2019 proposed an EPIC-type processor based on the MIPS architecture, featuring instruction code compression, branch prediction, immediate expansion, and multi-core capabilities. The effective utilization of these features enhances instruction-level parallelism. The proposed processor outperforms the basic pipeline processor by 1.9 times in speed, despite a slight increase in program size. Additionally, it reduces NOP instructions compared to VLIW processors. The paper evaluates the effectiveness of the "parallel bit" and suggests further examination of other processor bits in future research. [10]

3.6. 64-bit MIPS

A 64-bit MIPS processor refers to MIPS architecture that utilizes 64-bit data paths and registers. This architecture supports processing and addressing data in 64-bit chunks, enabling higher performance and larger memory addressing capabilities compared to processors with narrower data paths, such as 32-bit or 16-bit processors.

Macha Ashok Kumar et.al., 2018 designed 64-bit Microprocessor without Interlocked Pipeline Stages (MIPS) based RISC processor and executed successfully with 5 stage pipelining in Xilinx kintex7 14.3 Tool. Execution of multiple instructions at a time in a single clock cycle is achieved in this paper. [12]

3.7. 64-bit 6-Stage Pipeline MIPS

This design has 6 pipeline stages which enhances its performance compared to 5-stage pipeline architecture. The 6 stages are:

1) Instruction Fetch: Instructions are fetched from the instruction cache (I-Cache) based on the program counter (PC) address.

2) Instruction Decode: The fetched instruction code is decoded by a Decoder Unit, dividing the

64-bit instruction into multiple parts based on the operation.

3) Operand Fetch: Operands are retrieved from registers or memory, with registers being preferred for their efficiency. Indirect addressing modes benefit from registers, enhancing speed and ease of locating data addresses.

4) Data Cache: Data retrieval utilizes the data cache memory unit. In MIPS architecture, only load and store instructions are used for read and write operations, allowing ALU results to be stored directly in data memory.

5) Execute: Arithmetic and logic operations are performed on the operands using the Arithmetic and Logic Unit (ALU).

6) Write Back: The execution result is written back to registers and memory. [13]

4. Hazard Handling Unit

The Hazard Handling Unit is responsible for the efficient execution of the instructions by detecting and mitigating hazards. Hazards are the problems in the pipeline that might lead to incorrect computational outputs or imprecise instruction flow, jeopardizing the accuracy and the efficiency of the pipeline and the processor.

The MIPS architecture is prone to three hazards, namely data, control, and structural hazards. Data hazards occur when there is a dependency between two instructions. Dependency is a situation where the register read by an instruction contains an outdated value due to the execution of previous instructions. Hence, the operation is carried out with an outdated value, leading to incorrect results. Dependency resolution and data forwarding is used to take care of this hazard. Control hazards occur when an instruction leads to a jump or branching to an effective address. The Program Counter (PC) is updated with the effective address only after the execution stage of the current instruction, fetching two instructions that are not supposed to execute, leading to incorrect results and might cause deadlock situations. Flushing of instructions, bubble and stall are used to take care of this hazard. Structural hazards are caused by resource conflict when two operations require the usage of the same limited resource, which are handled by using techniques like renaming, and resource duplication. The memory is subdivided into instruction memory and data memory to address this. The memory is written in the first half and read in the second half of the clock cycle, as illustrated in Figure 1.

5. Issues In Pipelining

In pipelining, overall completion time decreases, but individual stage delays can increase due to issues like pipeline latency, uneven stage balancing, and varying device delays. These challenges impact pipeline efficiency and necessitate careful management for optimal performance. [5]

6. Future Scope and Conclusion

The MIPS architecture possesses enormous potential for development. Future advancements may involve techniques that enhance branch prediction mechanisms, cache optimization, enhanced pipeline efficiency, and dynamic power management. Power efficiency improves by employing techniques like clock gating. Recently, there has been a transition towards dual pipeline architectures that can double the throughput, optimize instruction scheduling, allocate resources, and handle hazards to unleash the full potential of parallel execution. Hence, we explored various implementations of the MIPS architecture, the intricacies of pipelining, and the instruction sets. The journey through the multiverse of MIPS is incomplete without addressing hazards, they are the black holes of the processor world. They spread throughout the multiverse and when you pass too close by one, the gravity of it might trap you in an eternal loop. Likewise, hazards can jeopardize a pipelined processor in no time.

REFERENCES

[1] Patterson, David A., and John L. Hennessy. "*Computer Organization and Design ARM Edition: The Hardware Software Interface*". Morgan Kaufmann, 2016.

[2] Sagar Bhavsar, Akhil Rao, Abhishek Sen, Rohan Joshi, "*A 16-bit MIPS Based Instruction Set Architecture for RISC Processor*", International Journal of Scientific and Research Publications, Volume 3, Issue 4, April 2013 | ISSN 2250-3153.

[3] Kanaka Sai Hemanth Dogga 1, Chand Basha Shaik 2, Sreemukhi Muddusetty, "*Design of Instruction Set Architecture Based 16-bit MIPS Architecture with Pipeline Stages*", International Research Journal of Engineering and Technology (IRJET), Volume: 08 Issue: 07, July 2021.

[4] Rakesh C R, Chetan S, J S Baligar, "*Design of 16 Bit RISC Processor and Implementation Using MIPS Technique*", Scholars Journal of Engineering and Technology, ISSN 2347-9523 (Print) | ISSN 2321-435X.

[5] P. Indira, Dr. Ved Vyas Dwivedi, Dr. M. Kamaraju, *"Verilog Implementation of a MIPS RISC 32-bit Pipelined Processor Architecture"*, IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834, p- ISSN: 2278-8735.Volume 14, Issue 1, Ver. I (Jan.-Feb. 2019), PP 31-40.

[6] Husainali S. Bhimani, Changa Hitesh N. Patel, Changa Abhishek A. Davda, "*Design of 32-bit 3-Stage Pipelined Processor Based on MIPS in Verilog HDL and Implementation on FPGA Virtex7*", International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 10 – No.9, May 2016.

[7] Soumya Murthy, Usha Verma, "*FPGA Based Implementation of Power Optimization of 32-bit RISC Core using DLX Architecture*", International Conference on Computing Communication Control and Automation, DOI 10.1109/ICCUBEA.2015.191, 2015.

[8] Priyavrat Bhardwaj, Siddharth Murugesan, "*Design & Simulation of A 32-bit RISC Based MIPS Processor Using Verilog*", IJRET: International Journal of Research in Engineering and Technology, volume: 05 Issue: 11 | Nov-2016.

[9] TriptiMahajan, Prof. Nikhil P. Wyawahare, "*Review of Low Power Multicycle MIPS Processor Using HDL*", Journal of Emerging Technologies and Innovative Research (JETIR), April 2019, Volume 6, Issue 4.

[10] Takahito Hayashi, Akinori Kanasug," *A Design of EPIC Type Processor Based on MIPS Architecture*", International Society of Artificial Life and Robotics (ISAROB), 2019.

[11] Rohit J, Raghavendra M, "*Implementation of 32-bit RISC processors without interlocked Pipelining on Artix-7 FPGA Board*", Proceedings of Second International Conference on Circuits, Controls and Communications, 2017.

[12] Macha Ashok Kumar, Mr. T. Krishna Moorthy, "*Design and Analysis Of 64-bit MIPS Processor*", Journal of Emerging Technologies and Innovative Research (JETIR) May 2018, Volume 5, Issue 5.

[13] P. Indira, M. Kamaraju, "*Design and Implementation of 6-Stage 64-bit MIPS Pipelined Architecture*", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-8 Issue-6S2, August 2019.