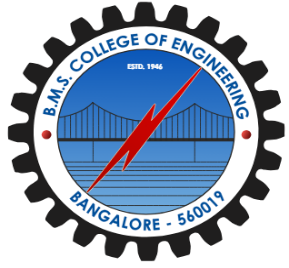
**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institute, Affiliated to VTU)**

**Bull Temple Road, Basavanagudi, Bangalore – 560019**



An AAT report on

VLIW MIPS

PROCESSOR

(Very Long Instruction Word

Microprocessor without Interlocked Pipeline Stages)

Submitted in partial fulfillment of the requirements for the award of degree

Bachelor of Engineering (BE)

in

**Information Science and Engineering**

BY

|  |  |
| --- | --- |
| Divyang Sharma | 1BM22IS071 |
| G Aditya Raman | 1BM22IS073 |
| Gunanka D | 1BM22IS081 |
| Karan Kulkarni | 1BM22IS097 |
| Mohammed R Umar Farooq | 1BM22IS116 |

Under the guidance of

**Dr. Srivinay**

Assistant Professor

Department of Information Science and Engineering

A.Y 2022-23

**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institute, Affiliate to VTU)**

**Bull Temple Road, Basavanagudi, Bangalore**

**Department of Information Science and Engineering**

**CERTIFICATE**

This is to certify that the individual project entitled “VlLIW MIPS Processor” is a bonafide work carried out by **Divyang Sharma, G Aditya Raman, Gunanka D, Mohammed R Umar Farooq, Karan Kulkarni** in partial fulfillment for the award of degree of B.E in Information Science and Engineering from **BMS College of Engineering, Bengaluru** during the year **2022-23**. The individual project report has been approved as itsatisfies the academic requirements in respect of individual project prescribed for the Degree.

**Signature of the Guide Signature of the HOD**

**Dr.Srivinay**

**Assistant Professor**

**TABLE OF CONTENTS:**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **CONTENT** | **PAGE NO.** |
| 1. | ABSTRACT | 4 |
| 2. | INTRODUCTION | 5 |
| 3. | OVERVIEW :  **MIPS32 PROCESSOR** | 6 |
| 4. | WORKING :  **VLIW MIPS PROCESSOR** | 8 |
| 5. | COMPARISION:  **VLIW MIPS v/s MIPS32** | 13 |
| 6. | CONCLUSION | 16 |
| 7. | REFERENCES | 17 |

1. **ABSTRACT**

The VLIW-MIPS processor, an evolution of the MIPS32 architecture, introduces a dual-issue paradigm with two parallel issue slots within its 64-bit instruction word. Unlike MIPS32, where each instruction word corresponds to a single operation, VLIW-MIPS utilizes two separate instructions concurrently. Notably, the second issue slot primarily handles memory accesses, optimizing its capability for operations related to base-offset memory addresses.

To achieve an efficient hardware implementation and streamline the pipeline structure, the VLIW-MIPS architecture merges the execution (EX) and memory access (MEM) stages into a consolidated phase, reducing the number of pipeline stages to 4. This consolidation is facilitated by simplifying the arithmetic logic unit (ALU) in the MEM stage, which focuses on basic address computation, supporting simple arithmetic operations. This design choice aims to exploit instruction-level parallelism, particularly valuable in applications where memory instructions constitute a substantial portion, approximately 35%, based on SPEC CPU2006 benchmark data.

Despite the advantages of dual-issue and streamlined memory access, the VLIW-MIPS architecture faces challenges related to data dependencies within applications, limiting the full utilization of both issue slots. To accommodate the dual-issue nature, the number of ports in the 32-entry register file is doubled, enabling efficient handling of dual-issue data accesses.

However, a notable drawback lies in the VLIW-MIPS's implementation of two branch delay slots. Unlike MIPS32, where the branch condition is assessed during the instruction decode (ID) stage, VLIW-MIPS evaluates the branch condition during the execution (EX) stage. This design choice leads to fetching two address-consecutive very long instruction words before determining whether to take the branch, potentially resulting in the flushing of four instructions (two very long instruction words) upon branch resolution. Such flushing incurs a performance penalty.

A suggestion for improvement is to reconsider the branch condition evaluation timing, proposing a shift back to the ID stage similar to the MIPS32 microarchitecture.

1. **INTRODUCTION**

Research in the areas of automotive, deep-drilling, and aerospace applications utilizing on-site microcontroller-like devices is still crucial. These systems ought to offer adequate adaptability to enable ongoing maintenance. Furthermore, the electronic parts themselves must provide dependable performance even in challenging environmental circumstances, including elevated temperatures or elevated radiation levels.

These days, silicon on insulator (SOI) stacks and relatively big technology nodes are used in the production of integrated circuits for hostile environments in order to lower the probability of latch-up effects and leakage current. The system's overall performance is decreased since the massive technology nodes only offer a limited operating frequency. Moreover, there is a limit to the quantity of transistors and, hence, the circuit complexity on a die. Given that embedded applications typically have limited power consumption, the processor architecture organization of the corresponding system greatly affects overall efficiency.

The performance of a processor architecture, which is limited by the previously listed technology node limits, is determined by the instruction set architecture (ISA) and how this ISA is implemented in hardware (i.e., processor architecture organization or microarchitecture). On the hardware implementation side, two of the most critical design characteristics are the data path width and the instruction execution paradigm. There are three main categories for prevalent microcontroller data path widths: 8-, 16-, and 32-bit designs. The three most popular methods for executing instructions are pipelined, multi-cycle, and single-cycle, or RISC- or CISC-like design concepts. The performance of the CPU, as well as the needs for silicon area and energy, are all greatly influenced by these factors.

Powerline communication is one excellent use case for harsh environment systems. By using fewer communication lines for data transfers, this technique has the advantage of decreasing the hardware points of failure in the application systems.

This report includes the architecture of the **VLIW MIPS32** processor series and how they can be used to handle all the above drawbacks and problems

1. **OVERVIEW**

Before diving into the VLIW MIPS processor, let’s first understand what makes this processor, the MIPS32 processor:

* 1. MIPS32 Processor:

The implemented MIPS32, is a 32-bit pipelined architecture with in order instruction execution and has 5 pipeline stages that support hazard resolution. Data and instruction memory use independent 32-bit address spaces with byte-wise alignment. The 32-bit wide instructions of the ISA form a RISC architecture and support register-to-register operations using a register file with 32 entries and the encoding of 16-bit short immediate values within the instruction. Each instruction can have up to three operands, defining two sources and one destination, making the MIPS32 a three-operand machine (e.g. ADD R3, R1, R2). In case of executing branch instructions (and also jumps), the instruction following in the code is always executed independently of if the branch is taken or not. This is known as branch delay slot. If the compiler cannot allocate any instruction after a branch/jump instruction, a NOP instruction will be used, which will decrease the performance. However, branch instructions can be avoided by using partial conditional execution, which is implemented using conditional move operations. The MIPS32 processor also includes a Multiply-and-Accumulate (MAC) unit as well as a divider unit in the execution stage.

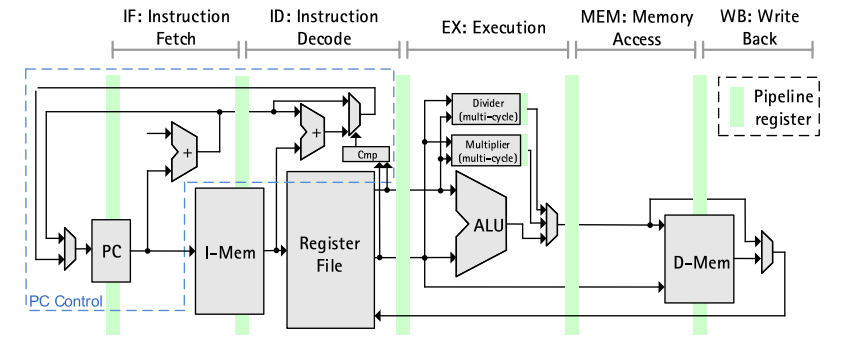


Fig: Five stage pipeline structure of the MIPS processor

Those hardware units are implemented using a configurable processing latency of 1 to 32 cycles at design time. In case of accessing the 64-bit wide result register, the execution stage may cause a stall for instruction fetch and decode until the multiplication (or division) operation is finished.

* 1. VLIW MIPS PROCESSOR:

The VLIW (Very Long Instruction Word) principles used in the VLIW-MIPS processor include:

* **Parallelism Exploitation:** VLIW processors aim to exploit instruction-level parallelism (ILP) by executing multiple operations simultaneously within each instruction. In the case of the VLIW-MIPS processor, each 64-bit instruction word contains two separate instructions, allowing for parallel execution of operations from different functional units.
* **Static Instruction Scheduling**: Instructions in VLIW architectures are statically scheduled at compile time rather than dynamically scheduled at runtime. This means that the compiler determines which operations can execute concurrently and bundles them together into instruction words. In the VLIW-MIPS processor, the instructions are scheduled to use two parallel issue slots within each instruction word.
* **Resource Allocation**: VLIW architectures typically have multiple functional units, and instructions are scheduled to utilize these resources efficiently. In the VLIW-MIPS processor, one issue slot is dedicated to memory access operations, while the other slot handles arithmetic/logic and control flow operations. This allocation of resources helps balance the workload and optimize performance.
* **Instruction Format**: VLIW instruction formats are designed to accommodate multiple operations within a single instruction word. In the case of the VLIW-MIPS processor, each instruction word is 64 bits long and contains two separate instructions.

By adhering to these VLIW principles, the VLIW-MIPS processor aims to achieve high performance by effectively exploiting instruction-level parallelism while

maintaining compatibility with the MIPS architecture.

1. **WORKING:**
   1. VLIW MIPS PROCESSOR

The VLIW-MIPS processor is derived from the aforementioned MIPS32. In contrast to the MIPS32, the VLIW-MIPS uses two parallel issue slots. Thus, the 64-bit instruction word contains two separate instructions based on the ISA of the MIPS32. Instructions allocated in the second issue slot are mainly used for performing memory accesses, while the other issue slot is dedicated to arithmetic/logic and control flow operations. However, the second issue slot can still be used to execute add or sub instructions (without any flag computation) by using the second small adder of the execution stage, which is required for calculation of base-offset memory addresses initially. The use of a specialized issue slot for memory access allows to reduce the number of pipeline stages to 4. Now, the execution (EX) and the memory access (MEM) stage of the MIPS32 are combined (see below figure).

In order to avoid increasing the critical path of the VLIW-MIPS hardware architecture implementation, the complexity of the ALU in the MEM stage is reduced for address computation and therefore supports simple arithmetic only. The parallel execution of memory accesses along with arithmetic/logic instructions allows the use of instruction-level parallelism in applications, due to the high amount of load/store instructions in contrast to pure computational instructions. A typical amount of parallelizable memory instructions is about 35% (based on SPEC CPU2006 benchmark ). However, due to data dependencies within an application, the two issue slots cannot always be fully utilized. The number of ports of the 32-entry register file of the VLIW-MIPS is doubled in comparison to the MIPS32, due to the dual-issue data accesses. Another drawback is the implementation of two branch delay slots. In contrast to the MIPS32 microarchitecture, the VLIW-MIPS microarchitecture evaluates the branch condition during the EX stage, fetching always the two address-consecutive very long instruction words before taking or not taking the branch. In case of taking the branch, 4 instructions (i.e., 2 very long instruction words) will be flushed with the corresponding loss of performance. This part of the microarchitecture could be improved by evaluating the branch condition on the ID stage (like in MIPS32 microarchitecture), which introduces a trade-off between the required silicon area, the performance and the code density.

* 1. ARCHITECTURE

Let's elaborate further on each component and stage depicted in the image of the VLIW MIPS processor architecture:

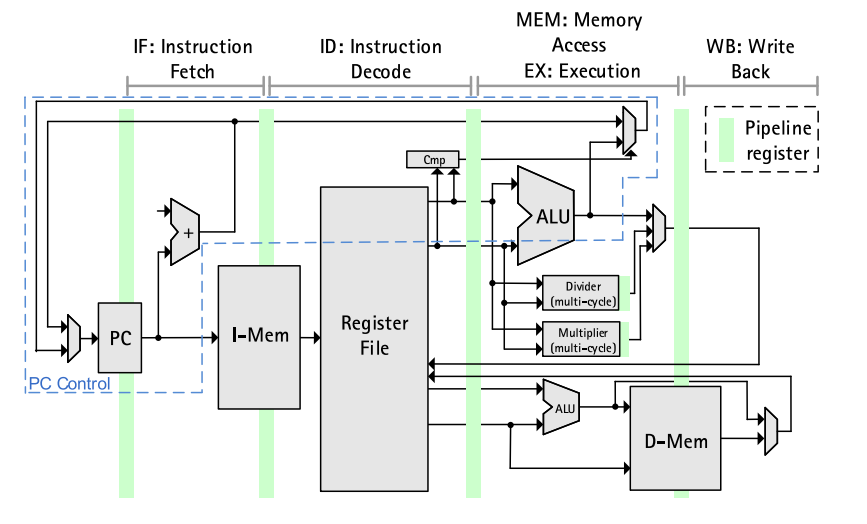


Fig: Four stage pipeline structure of the VLIW-MIPS processor

**1. Program Counter (PC):**

* The PC keeps track of the address of the next instruction to be fetched from memory.
* It is updated at the end of each instruction fetch cycle to point to the next instruction in sequence.

**2. Instruction Memory (I-Mem):**

* Instruction Memory stores the program instructions.
* During the instruction fetch stage (IF), instructions are fetched from the instruction memory based on the value of the program counter (PC).

**3. Register File:**

* The Register File contains a set of general-purpose registers that hold data operands.
* Instructions typically read data from the register file as input operands and write back results to the register file.

**4. ALU (Arithmetic Logic Unit):**

* The ALU performs arithmetic and logical operations such as addition, subtraction, AND, OR, etc.
* It receives operands from the register file and generates results based on the instruction opcode.

**5. Data Memory (D-Mem):**

* Data Memory stores data that may be accessed by load and store instructions.
* During the memory access stage (MEM), data is read from or written to the data memory based on the memory address calculated in the execution stage (EX).

**6. Pipeline Stages:**

1. **Instruction Fetch (IF):** - Fetches the next instruction from the instruction memory based on the value of the program counter.
2. **Instruction Decode (ID):** - Decodes the fetched instruction, determining the operation to be performed and the operands involved.
3. **Execution (EX):** - Executes the instruction, which may involve arithmetic or logical operations using the ALU. - For memory-related operations, it calculates memory addresses or performs address calculations.
4. **Memory Access (MEM):** - Handles memory-related operations, such as loading data from or storing data to memory. - Data is read from or written to the data memory during this stage.
5. **Write Back (WB):** - Writes the results of the executed instruction back to the register file.

**7. Pipeline Registers:**

* Pipeline registers are used to store intermediate results and control signals between pipeline stages.
* They facilitate the smooth flow of data and control signals through the pipeline stages.

**8. ALU Inputs:**

* The ALU receives two inputs: "Data" and "Multiplier result."
* These inputs are typically operands fetched from the register file or intermediate results generated during the execution stage.

Overall, the VLIW MIPS processor architecture depicted in the image follows a five-stage pipeline structure, where each stage performs specific tasks such as instruction fetch, decode, execution, memory access, and write back. The components and stages work together to fetch, decode, execute, and store program instructions and data, facilitating efficient instruction execution and data processing.

* 1. HOW PARALLEISM WORKS?
* The processors in this architecture have multiple functional units, fetch from the Instruction cache that have the Very Long Instruction Word.
* Multiple independent operations are grouped together in a single VLIW Instruction. They are initialized in the same clock cycle.
* Each operation is assigned an independent functional unit.
* All the functional units share a common register file.
* Instruction words are typically of the length 64-1024 bits depending on the number of execution unit and the code length required to control each unit.
* Instruction scheduling and parallel dispatch of the word is done statically by the compiler.
* The compiler checks for dependencies before scheduling parallel execution of the instructions.

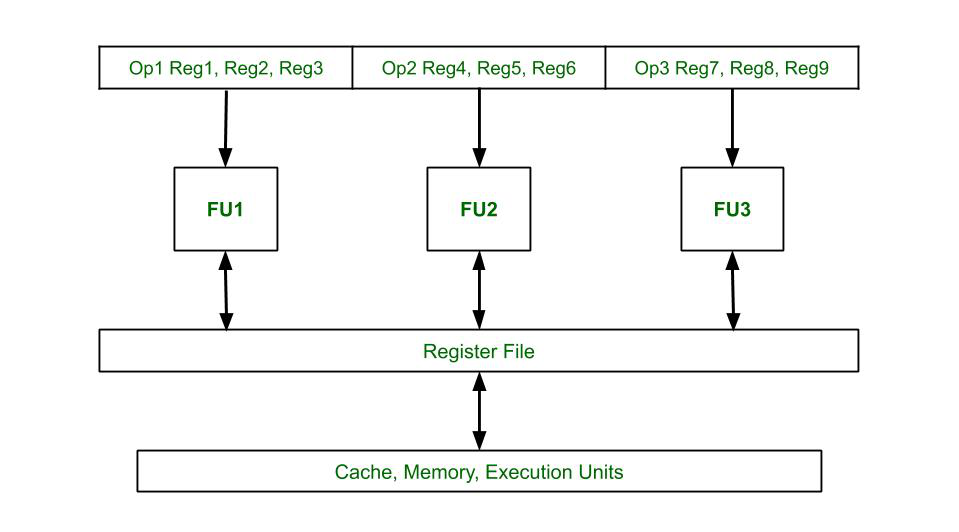


Fig: Block Diagram of VLIW Architecture

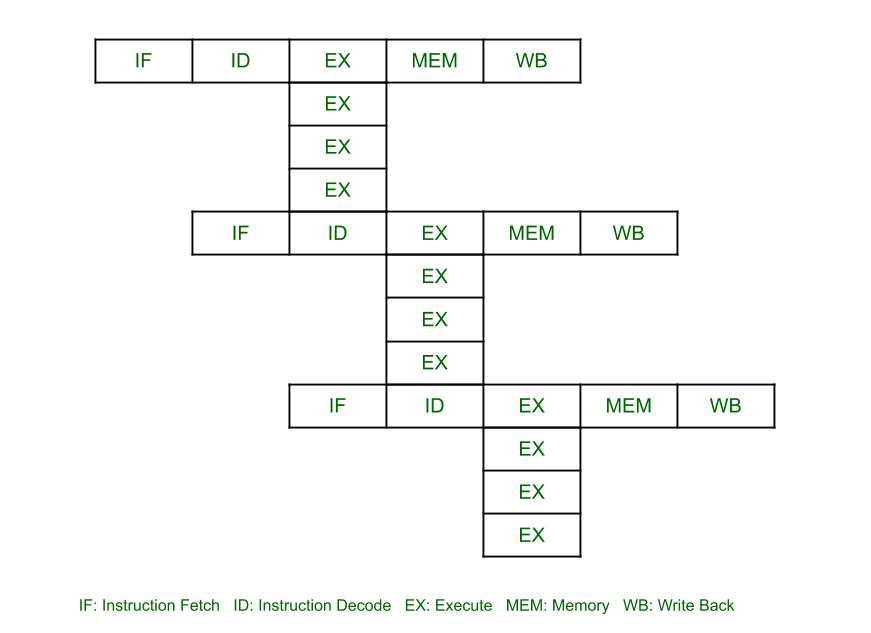


Fig: Time Space Diagram of VLIW Processor where 4 instructions are executed in parallel in a single instruction word

1. **COMPARISION**

Everything has pros and cons, and lets discuss VLIW MIPS processor’s limitations by comparison with its predecessor, MIPS32

Determining which type of processor is "better" depends on the specific use case and requirements. Both VLIW-MIPS processors and MIPS32 processors have their own strengths and weaknesses, and the choice between them depends on factors such as performance needs, power efficiency, and the nature of the applications they will be used for.

* 1. VLIW-MIPS Processor:

Pros:

1. **Parallelism:** VLIW architectures can potentially achieve high levels of parallelism by executing multiple instructions simultaneously, leading to increased performance.
2. **Flexibility:** The ability to exploit parallelism is largely dependent on the compiler, offering some flexibility in optimizing code for specific applications.
3. **Customization:** Depending on the design and compiler support, VLIW processors can be customized for specific workloads.

Cons:

1. **Compiler Dependency:** The effectiveness of VLIW processors heavily relies on the compiler's ability to identify and schedule parallel instructions, making them more sensitive to compiler quality.
2. **Complexity:** The design and implementation of VLIW processors can be more complex compared to traditional architectures, potentially affecting power consumption and chip area.
   1. MIPS32 Processor:

Pros:

1. **Simplicity:** MIPS architectures, including MIPS32, are known for their simplicity and efficiency. The reduced instruction set can lead to straightforward hardware implementations and efficient pipeline designs.
2. **Widespread Usage:** MIPS processors are used in a variety of applications, demonstrating their versatility and reliability.
3. **Energy Efficiency:** The simplicity of the architecture often translates into energy-efficient designs, making MIPS processors suitable for embedded systems and other power-sensitive applications.

Cons:

1. **Limited Parallelism:** Compared to VLIW architectures, traditional MIPS processors may have limitations in exploiting parallelism at the instruction level.

**Considerations:**

1. **Application Requirements:** Consider the specific requirements of the applications you intend to run. Some applications may benefit more from parallelism, while others may prioritize simplicity and energy efficiency.
2. **Development Ecosystem:** The availability and maturity of development tools, compilers, and support for a particular architecture can influence the ease of programming and debugging.
3. **Power Efficiency:** If power efficiency is a critical factor, a MIPS32 processor might be more suitable, as the architecture is often designed with energy efficiency in mind.

Ultimately, the "better" choice depends on your specific needs and the characteristics that align with your application requirements.

* 1. SO… FOR WHO *IS*  THIS FOR?

The suitability of VLIW-MIPS and MIPS32 processors depends on the specific requirements and use cases of different populations. Here's a brief overview of the potential users who can make the best use of these processors:

VLIW-MIPS Processors:

**High-Performance Computing (HPC) Professionals:** VLIW architectures, with their emphasis on parallelism, can be beneficial for applications in the field of high-performance computing. Tasks that can be parallelized effectively, such as scientific simulations, complex mathematical computations, and certain types of simulations, may benefit from VLIW architectures.

**Signal Processing and Multimedia Professionals:** Applications in signal processing, multimedia, and graphics processing may leverage the parallel execution capabilities of VLIW architectures. Video encoding/decoding, image processing, and other media-related tasks could potentially benefit from the parallel nature of VLIW processors.

**Custom Hardware Acceleration:** In scenarios where custom hardware acceleration is required, and developers have control over the compilation process, VLIW architectures can be tailored to specific workloads. This might be relevant in certain industrial or specialized computing applications.

MIPS32 Processors:

**Embedded Systems Developers:** MIPS32 processors are commonly used in embedded systems due to their efficiency, simplicity, and versatility. Developers working on devices such as routers, smart home devices, and other embedded applications can benefit from MIPS32's balance of performance and energy efficiency.

**Networking Equipment Manufacturers**: MIPS processors have historically been used in networking equipment, such as routers and switches. Networking professionals and manufacturers might find MIPS32 processors suitable for their devices, offering a good combination of performance and power efficiency.

**Consumer Electronics:** MIPS32 processors have been utilized in various consumer electronics, including set-top boxes, digital TVs, and other home entertainment devices. Individuals involved in the development and design of consumer electronics may find MIPS32 processors suitable for these applications.

**Low-Power Devices:** MIPS32 processors are often chosen for low-power devices where energy efficiency is crucial. This includes applications in IoT (Internet of Things) devices, wearables, and battery-operated gadgets.

VLIW-MIPS processors may find a niche in specialized high-performance computing applications, while MIPS32 processors are well-suited for a broader range of embedded systems, consumer electronics, and low-power devices. The choice depends on the specific requirements of the target applications and the balance between performance, power efficiency, and simplicity needed for the given use case.

1. **CONCLUSION**

In conclusion, the VLIW-MIPS processor represents a fusion of two distinct architectural paradigms: MIPS (Microprocessor without Interlocked Pipeline Stages) and VLIW (Very Long Instruction Word). By integrating elements of VLIW architecture into the established MIPS ISA, the VLIW-MIPS processor combines the familiarity and compatibility of MIPS with the enhanced parallelism and performance advantages offered by VLIW principles. This integration enables efficient execution of multiple operations in parallel within each instruction word while maintaining compatibility with existing MIPS software and tools. Ultimately, the VLIW-MIPS processor offers a compelling balance between performance optimization and architectural familiarity, making it a promising candidate for various computing applications.

* 1. **REFERENCES**

1. ***Springer Link*** –

International Journal of Parallel Programming - Evaluation of Different Processor Architecture Organizations for On-Site Electronics in Harsh Environments – Sven Gesper, Moritz Weißbrich, Tobias Stuckenberg, Pekka Jaaskelainen, Holger Blume, Guillermo Paya-Vaya

*(super good research paper, do check it out!!)*

1. ***Quora-***

<https://www.quora.com/What-are-Interlocked-Pipeline-stages-like-in-MIPS#:~:text=In%20an%20interlocked%20pipeline%2C%20hardware,hazards%20may%20result%20in%20deadlocks>.

1. ***Byjus-***

<https://byjus.com/gate/risc-and-cisc-architecture-notes/#:~:text=RISC%20is%20an%20abbreviation%20for,along%20with%20few%20addressing%20nodes>.

1. ***GeeksForGeeks-***

<https://byjus.com/gate/risc-and-cisc-architecture-notes/#:~:text=RISC%20is%20an%20abbreviation%20for,along%20with%20few%20addressing%20nodes>.

1. ***ChatGPT-***

Link 1: <https://chat.openai.com/share/03af58b0-1fc7-4748-8ead-0301ea984e71>

Link 2: <https://chat.openai.com/share/13e19a49-5358-4dcf-9f03-38165d87ded8>