**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 9**

|  |
| --- |
| **TITLE:** Study of RISC and CISC Architecture |

**AIM:** Understanding RISC and CISC Architecture

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment: (Mentions the CO/CO’s attained)**

**CO 2: Study of RISC and CISC Architecture**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

**3**. Dr. M. Usha, T. S. Srikanth, “Computer System Architecture and Organization”, First Edition, Wiley-India.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

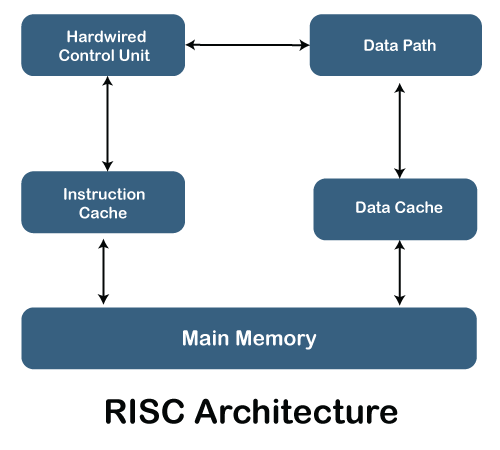
**Reduced Set Instruction Set Architecture (RISC)**The main idea behind is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating and storing operations just like a load command will load data, store command will store the data.

**Complex Instruction Set Architecture (CISC**)   
The main idea is that a single instruction will do all loading, evaluating and storing operations just like a multiplication command will do stuff like loading data, evaluating and storing it, hence it’s complex.Both approaches try to increase the CPU performance

**RISC Architecture**

1. Diagram of RISC Architecture:

Ans.



1. Brief Explanation of each component

Ans. Hardwired Control Unit: It controls instruction execution by providing control signals. In RISC, it is a hardwired unit and not microprogrammed, thus it can execute instructions very rapidly and efficiently.

Data Path: The data path has the arithmetic logic unit, registers and interconnections, which are required for moving and manipulating the data. This is responsible for performing computations and managing the flow of data between registers and memory.

Instruction Cache: The instruction cache stores the instructions in a temporary location to reduce the time needed to access frequently used instructions from main memory. It reduces the delays in accessing the memory and improves overall performance.

Data Cache: Similar to the instruction cache, this cache stores data temporarily to speed up the access time. The data which is accessed frequently from main memory is kept in the memory so that latency can be reduced and performance improved.

It is the actual storage that forms the core of which instruction and data housed. Main memory, in RISC architecture, interacts with both instruction and data caches to streamline the processing and reduce access times.

These altogether enable RISC architectures to perform efficient instruction processing with minimal complexity in instruction design so as to optimize speed and power usage.

1. RISC Processor Instruction Set Examples with explanation (Any 2)

Ans. **LOAD** Instruction

The **LOAD** instruction is used to transfer data from memory to a register. This is a fundamental operation in any computer system, as it allows the processor to access data that is stored in main memory.

In a RISC architecture, only **LOAD** and **STORE** instructions can access memory locations. All data must be loaded into registers before an operation can be performed; the result can then be used for further processing or stored into memory.

For example, the MIPS R4000 instruction:

loads a word from memory into register r2. The address of the word to be loaded is calculated by adding the 32-bit immediate offset #imm to the contents of register r4.3

This instruction could be used to load the value of a variable from memory into a register so that it can be used in a calculation.

**ADD** Instruction

The **ADD** instruction is used to add the values of two registers and store the result in a third register.

For example, the MIPS R4000 instruction:

adds the values of registers r1 and r2 and stores the result in register r3.

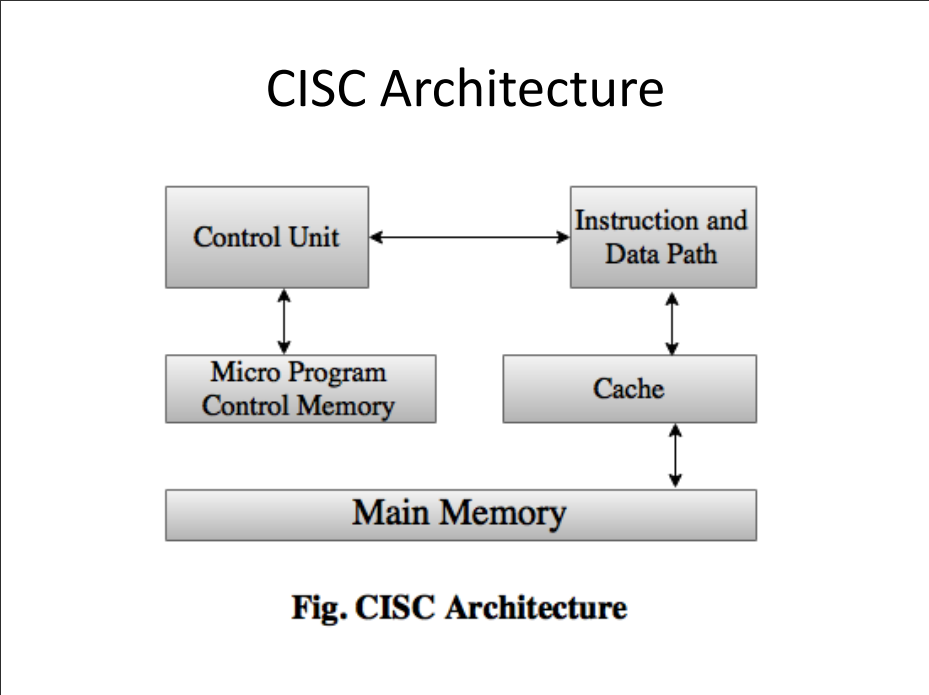
This instruction could be used to perform a simple arithmetic operation, such as adding two numbers together.

Both of these instructions are examples of the simple and regular instructions that are characteristic of RISC architectures. This simplicity allows for efficient pipelining, which can lead to significant performance improvements.

**CISC Architecture**

1. Diagram of CISC Architecture:

Ans.



1. Brief Explanation of each component

Ans.

Control Unit: In CISC, the control unit is what controls instruction execution and steers the flow of data within the processor. It can be implemented with a microprogrammed control system, therefore allowing it to process complicated instructions that might require more than one step.  
  
Micro Program Control Memory: It stores sequences of control signals called microprograms that are used as a guide for the instruction execution process. It basically allows the CISC architecture the interpretation of many different instruction varieties by breaking them up into smaller micro-operations.  
  
Instruction and Data Path: This includes the heart of the processor, such as ALU or arithmetic logic unit, register blocks, and many other structural components. All such processing units include the operations performed on the data, therefore, computational work, transfer of the data between various registers or memory.  
  
It does retain a cache of holding instructions and data for intermediate, temporary storage which also has the effect of delivering relatively low access times for more frequently used information accessed directly from memory itself. This helps in improving performance by removing and or minimizing access delay delays.  
  
Main Memory  
Main memory is used both to store data and programs instructions. The main memory can interface with cache memory and control unit together for obtaining data retrieval as well as execution processes through its pathways.  
  
This results in dealing with complex instructions in easier ways by CISC architectures, which frequently increases the number of instructions per program but reduces individual instruction complexity.

1. CISC Processor Instruction Set Examples with explanation (Any 2)

Ans. **Move Characters (MVC)** Instruction

The **MVC** instruction is a complex instruction that moves a block of bytes from one location in memory to another.

This is a more powerful and complex instruction than the simple **LOAD** and **STORE** instructions found in RISC architectures. It can be used to copy data structures, move strings, or perform other operations that require moving a block of data.

The **MVC** instruction can handle variable block sizes, and it can also be used to perform overlapping moves, where the source and destination blocks overlap in memory. This flexibility makes the **MVC** instruction a valuable tool for programmers, but it also makes it more difficult to implement in hardware.

The format of the **MVC** instruction is as follows:

where:

○ D1 is the destination address.

○ (L,B1) specifies the length and base register for the destination operand.

○ D2 is the source address.

○ (B2) specifies the base register for the source operand.

For example, the instruction:

would move 100 bytes of data from the location starting at the address in register R2 plus 30, to the location starting at the address in register R1 plus 20.

**Multiply (IMUL)** Instruction

The **IMUL** instruction is a complex instruction that multiplies two operands and stores the result.

The **IMUL** instruction can handle different operand sizes, and it can also perform signed or unsigned multiplication. This flexibility makes the **IMUL** instruction a valuable tool for programmers, but it also makes it more difficult to implement in hardware.

The **IMUL** instruction comes in a few different formats, depending on the number of operands and whether an immediate value is used.3

For example, the instruction:

multiplies the value of op2, which can be either a register or a memory location, by the immediate operand value, and places the result in op1, which must be a register.

These instructions, with their variety of addressing modes, operand sizes, and options, are typical examples of CISC instructions that can be used to efficiently perform complex tasks in a single instruction.

**Post Lab Descriptive Questions**

**Write a tabular comparative analysis of RISC v/s CISC**

**Ans.**

|  |  |  |
| --- | --- | --- |
| **Feature** | RISC | CISC |
| **Instruction Set Size** | Small number of instructions | Large number of instructions |
| **Instruction Complexity** | Simple instructions, usually performing a single operation | Complex instructions, often combining multiple operations |
| **Instruction Length** | Fixed length | Variable length |
| **Instruction Format** | Few formats | Many formats |
| **Addressing Modes** | Limited number of addressing modes | Large number of addressing modes |
| **Memory Access** | Only load and store instructions access memory | Many instructions can access memory |
| **Register Usage** | Emphasis on using registers | Less emphasis on using registers |
| **Register File Size** | Large number of general-purpose registers | Small number of general-purpose registers |
| **Control Unit** | Typically hardwired | Typically microprogrammed |
| **Pipelining** | Highly pipelined | Less pipelined |
| **Instruction Decode** | Simple decode logic | Complex decode logic |
| **Compiler Complexity** | Compiler needs to perform more work to generate efficient code | Compiler can take advantage of complex instructions |
| **Code Size** | Larger code size | Smaller code size |
| **Execution Speed** | Faster execution of individual instructions | Slower execution of individual instructions |
| **Typical Applications** | Embedded systems, mobile devices, high-performance computing | Desktop computers, servers, mainframes |

**Conclusion:**

While RISC and CISC architectures once represented distinct approaches to processor design, recent years have witnessed a convergence of the two. As technology advances, RISC designs have incorporated more complex features, and CISC designs have adopted techniques such as increased register counts and sophisticated pipelining, traditionally associated with RISC. Ultimately, the choice between RISC and CISC depends on the specific application, performance goals, and design constraints.

**Date: 30/10/24 Signature of faculty in-charge**